RUTGERS

THE STATE UNIVERSITY OF NEW JERSEY

DEPARTMENT OF STATISTICS
&
DATA SCIENCE

STAT 588 - FINANCIAL DATA MINING - FALL 2023

An Experimental Study of What Makes Deep Neural Networks Generalize Well - The Role of Regularization

FINAL PROJECT REPORT

Ganesh Raj Kyatham (grk62)

Manisha Gayatri Damera (md1723)

Neha Thonta (nt446)

Contents

1. Introduction

1.1 Background

Deep neural networks are machine learning models capable of learning and processing large volumes of data, making them a popular choice in various applications, including image recognition, natural language processing, and speech recognition. One of the critical strengths of deep neural networks is their ability to memorize large amounts of data, which allows them to identify complex patterns and relationships within the data. However, this strength can also lead to overfitting, where the model becomes too specialized to the training data and fails to generalize to new, unseen data. This occurs when the model becomes overly complex and starts to fit the noise in the training data instead of the underlying patterns.

Generalization error is measured by comparing the model's performance on the training data to its performance on new, unseen data. If the model performs well on the training data but poorly on new, unseen data, it has a high generalization error, indicating overfitting. On the other hand, if the model performs well on both the training data and new, unseen data, it has a low generalization error, indicating that it can generalize what it has learned to recent examples. Controlling generalization error is one of the critical challenges in machine learning, especially in deep neural networks with large parameters, and can easily overfit the training data.

The observation that deep neural networks can achieve 0 training error even with meaningless labels is an example of overfitting. Overfitting occurs when a model becomes too specialized to the training data and performs poorly on new, unseen data. This happens when the model is too complex, with too many parameters relative to the amount of training data available, and can memorize the training data instead of learning the underlying patterns.

To prevent overfitting, various techniques are used, including L2 regularization, dropout, and etc. L2 Regularization involves adding a penalty term to the loss function, which discourages the model from assigning too much importance to any one feature or parameter. Dropout is a technique where random nodes in the neural networks are temporarily dropped out during training, which encourages the network to learn more robust representations of the data.

The paper's observation that deep neural networks can exhibit low differences between training and test performance, despite their large size, highlights the importance of controlling the generalization error in these models. Although deep neural networks are capable of memorizing large amounts of data, their ability to generalize to new, unseen data is critical to their performance in real-world applications. The paper suggests that explicit forms of regularization, such as L2 regularization or dropout, may not always be necessary for achieving good performance in deep neural networks. Instead, controlling the capacity of the model by adjusting the number of parameters and layers can be an effective way to prevent overfitting and improve generalization performance.

To analyze the generalization performance of deep neural networks, we conducted image classification experiments using various models. By evaluating the performance of these models on new, unseen data, and were able to gain deeper insights into the factors that influence generalization performance and identify effective strategies for controlling the generalization error in deep neural networks.

1.2 Problem Description

In the present work, the effect of explicit regularization methods like weight decay and dropout is investigated by performing image classification on the Fashion MNIST datasets, and implicit procedures like SGD are theoretically explored. The problem is to determine the experimental framework for learning regularization. Experiments are performed with three different neural network models, and the effect on the generalization error is explored. The impact of model architectures and regularization techniques are inspected in the framework built, and the work looks at how well they generalize. The generalization ability of the models with and without regularization is evaluated.

2. Dataset Description

To perform the image classification using the deep neural network models, the Fashion MNIST dataset is considered. This dataset is chosen due to its benchmark for deep learning models. The Fashion MNIST dataset is a collection of 70,000

grayscale images of fashion items, including ten different categories, such as t-shirts, dresses, sneakers, and sandals. This dataset is easily accessible and challenging due to low resolution, a lot of noise, and variability, making the ability of models to learn to recognize the objects despite variations in shape, texture, and lighting.

3. Theorems

3.1 Vapnik - Chervonenkis theory (VC Dimension)

A classification model f with some parameter vector θ is said to shatter a set of datapoints (x1, x2, x3,.....xn) if, for all assignments of labels for these points, there exists a θ such that the model f makes no errors while evaluating the set of data points.

The VC dimension of a model is the maximum number of datapoints that can be arranged so that f shatters them.

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

This theory gives a probabilistic upper bound on test error.

It is valid only when D << N. (where D is the number of parameters in the model and N is the number of datapoints)

This fails for neural networks since D becomes greater than N.

This gives us a hint that neural networks possess the potential to memorize any kind of dataset fed to them. And there is no definite upper bound to it.

However, that is not the case. A lot of neural networks fail to perform well on unseen dataset despite having very high capability to memorize patterns in the data.

4. Regularization Techniques

Regularization is a deep learning strategy that enhances the model's ability to generalize to new data and prevents overfitting on the training set of data. Overfitting is a condition in which a model performs badly on fresh data because it has grown too complicated and has begun to incorporate noise and irrelevant information from the training set. Regulating the model's parameters during training

helps regularization techniques avoid this. In the present work to understand the effect of regularization on generalization, regularization techniques like weight decay and dropout are implemented on the models.

4.1 L2 Regularization/ Weight Decay

L2 Regularization also known as weight decay works at reducing overfitting. A penalty term is added to the model's loss function using L2 regularization. The sum of squares of the model's parameters is added by L2 regularization. As a result, the model is encouraged to have lower parameter values, which lowers its complexity and prevents overfitting.

4.2 Dropout

During training, certain neurons in the network are randomly removed using the dropout technique. By lowering the interdependence between neurons, this helps prevent overfitting by driving the network to learn more robust features.

5. Methodology

5.1 Deep Neural Network Models

Deep neural network models are often used to test generalization because of their capability to learn complex patterns and relationships in data. When training a neural network model, the goal is to find the optimal set of weights that accurately map the given task's inputs to outputs. In this study, three deep neural network models are built on the Fashion MNIST dataset. The neural networks used in the framework are MLP, CNN, and VGG16. The models are implemented in python using Tensorflow’s Keras.

5.1.1 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a type of feedforward neural network consisting of an input layer, a hidden layer, and an output layer. It takes in the input signal and performs computations on it in the hidden layers to transform it into a more useful representation for solving classification problems. The output layer is responsible for tasks like prediction and classification. The backpropagation learning algorithm is used to train the neurons in the MLP. MLPs

can approximate any continuous function and are capable of solving non-linearly separable problems.

For the MNIST dataset, this specific MLP model has two hidden layers, each with an unspecified number of units. The ReLU activation function is applied to each hidden layer to introduce non-linearity, and dropout regularization is used to prevent overfitting. L2 regularization is also applied to the model to reduce the model's complexity and prevent overfitting further.

The MLP model is enhanced with L2 Regularization, which imposes penalties on the weights of the model during optimization to avoid overfitting. The L2 norm of the weights is added to the loss function to optimize the model. In Keras, L2 regularization can be added to any Dense layer by setting the kernel_regularizer parameter to an appropriate value. In addition, the base model is also augmented with dropout regularization, where a certain percentage of nodes are randomly dropped during training to prevent overfitting. To build a model with both L2 Regularization and Dropout, both regularization techniques can be added to the model during training. The model can be fine-tuned by adjusting hyperparameters such as the number of hidden layers, the learning rate, and the batch size to achieve better accuracy.

In combination with dropout, L2 regularization can help to improve the model's performance by reducing the complexity of the model and preventing overfitting. The output layer of the model uses the softmax activation function to distribute probability over the classes, and the model is trained using the categorical cross-entropy loss function and the Adam optimizer. This model achieves a test accuracy of almost 85%.

5.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are widely used for image classification tasks; CNN takes pixel input, combining the edges to get the object. The images are preprocessed to make the training more efficient. This may include normalizing pixel values, reshaping the images to a standardized size, and converting the labels to one-hot encoded vectors. On the MNIST dataset, CNN is developed to classify the images using Keras. First, a base model is built on the

preprocessed data after shuffling the datasets to create train and test data, cross-validation, and test sets.

The experimental framework implementing CNN considers fifteen layers with six Convolutional layers and remaining hidden layers, with additional max pooling and dense layers. The first layer uses 64 filters and ReLU activation with padding. Batch Normalization is also performed on the model. The convolutional layers extract features from the input images, the pooling layers reduce the spatial dimensions of the feature maps, and the fully connected layers perform the classification. Then the parameters are fine-tuned, i.e., If the performance of the model is not satisfactory, the model can be fine-tuned by adjusting the hyperparameters such as the number of layers, the learning rate, and the batch size and weights are calculated and processed using gradients to minimize the loss function. The model resulted in a test accuracy of 92.8 %.

The CNN model is added with L2 Regularization. An L2 regularizer imposes penalties on layer parameters during optimization (= L2 Norm of weights). The loss function that the network optimizes includes these penalties. We end up with simpler models that overfit less because higher weights/parameters generally tend to be penalized more. L2 Regularization can be added to any Keras Conv2D or Dense layer by setting the kernel_regularizer parameter to an appropriate value. L2 regularization is applied if you set use_l2_reg=True in the build_model() function. The base model is also added with dropout regularization. Dropout regularization is applied to a layer by adding a Dropout() layer afterward. To this layer, a certain percentage of nodes must be randomly dropped *during training*. As a final test, the experiment is also built by developing both L2 Regularization and Dropout to the base model.

5.1.3 Visual Geometry Group- VGG16

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is an annual event to showcase and challenge computer vision models. In the 2014 Karen Simonyan & Andrew Zisserman from Visual Geometry Group a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date and has won the ImageNet contest. The model has a depth of 16-19 layers and approximately 138 trainable parameters.

VGG16 is object detection and classification algorithm which is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

6. Results of Empirical Evaluation of Models

After the models are built, they are empirically evaluated testing the model performance. This helps in understanding the working and performance of different model architectures along with the role of regularization. Below are the evaluation metrics used for measuring our model performance. The model performance is evaluated on a *cross-validation* data on each epoch of training

Log loss function: The log loss function calculates how far the predicted probability is from the true value using a formula.

$$LogLoss = -1 * (Y * log(p) + (1-Y) * log(1-p))$$

where Y is the true value of the predicted class , and p is the probability of the prediction function.

Train/ Test accuracy: Accuracy is simply the number of correct predictions divided by total number of datapoints. The same metric measured on a test dataset is test accuracy.

Generalization error: It is the difference between the training error and test error. This tells us whether a model is overfitting or not.

6.1 MLP

6.1.1 MLP without regularization

Without using regularization approaches, the base MLP model was trained. The model's reported accuracy rates show how effectively it classified both the input data it was trained on and brand-new, untainted data.

The training accuracy of 96% suggests that the model is overfitting to the training data, which means it may not perform well when presented with new data. This is further supported by the lower testing accuracy of 88%, which is a more reliable indicator of the model's performance on new data. The output shows that the

model's training loss decreased and its training accuracy increased as the number of epochs increased. At the same time, the validation loss decreased and the validation accuracy increased, indicating that the model was able to learn and generalize well to the data.

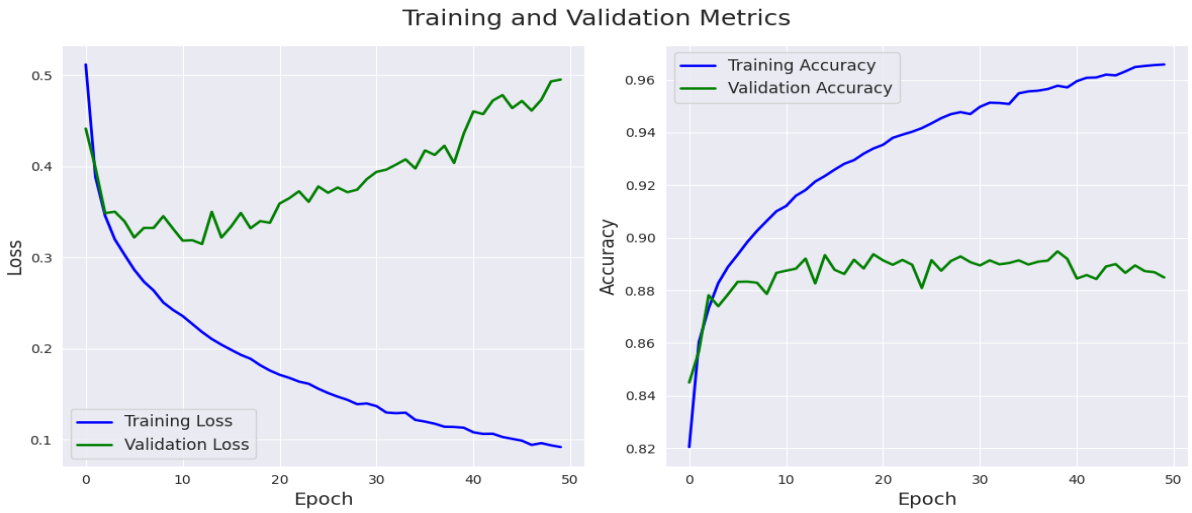

Fig. 1 Model fit using MLP without any regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs.

6.1.2 MLP Base Model + L2 Regularization

After applying L2 regularization to the MLP model, the training accuracy of 88% suggests that the model is able to fit the training data reasonably well. This means that the model is able to learn the patterns in the training data and make accurate predictions. However, the testing accuracy of 85% is lower than the training accuracy, indicating that the model may not be able to generalize well to new, unseen data. This means that the model may be overfitting to the training data and not capturing the true underlying patterns in the data.

To further investigate the performance of the model, we can look at the training and validation loss. The training loss is around 0.39, and the validation loss is around 0.43, which is relatively low, indicating that the model is learning to make accurate predictions. However, there is a slight difference between the training and validation loss, which may suggest that the model is overfitting.

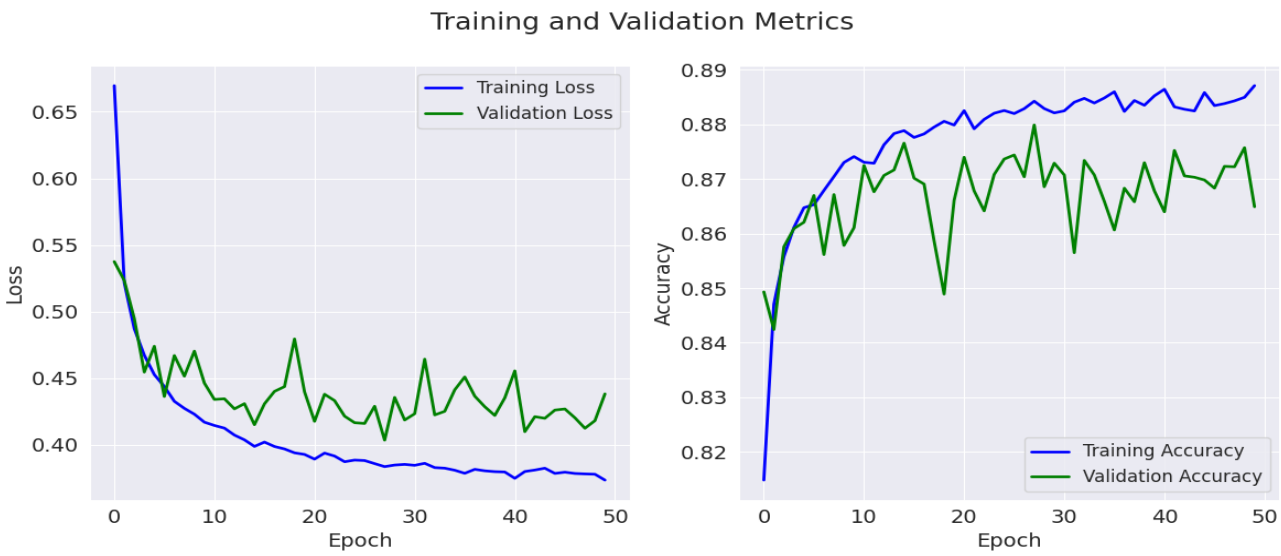

Fig. 2 Model fit using MLP with L2 regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.1.3 MLP Base Model + Dropout

With the addition of dropout, we have managed to improve the test accuracy while maintaining a high train accuracy, which is great! The model is now less likely to overfit to the training data. The train accuracy of 0.8455 indicates that the model was able to correctly predict the target class for 84.55% of the training examples, while the test accuracy of 0.8595 indicates that the model was able to correctly predict the target class for 85.95% of the test examples. The slight difference between the train accuracy and test accuracy suggests that the model may be slightly overfitting the training data, but overall, the performance is good and the model is generalizing well to the test data. The use of dropout regularization is likely helping to reduce overfitting and improve the model's generalization performance.

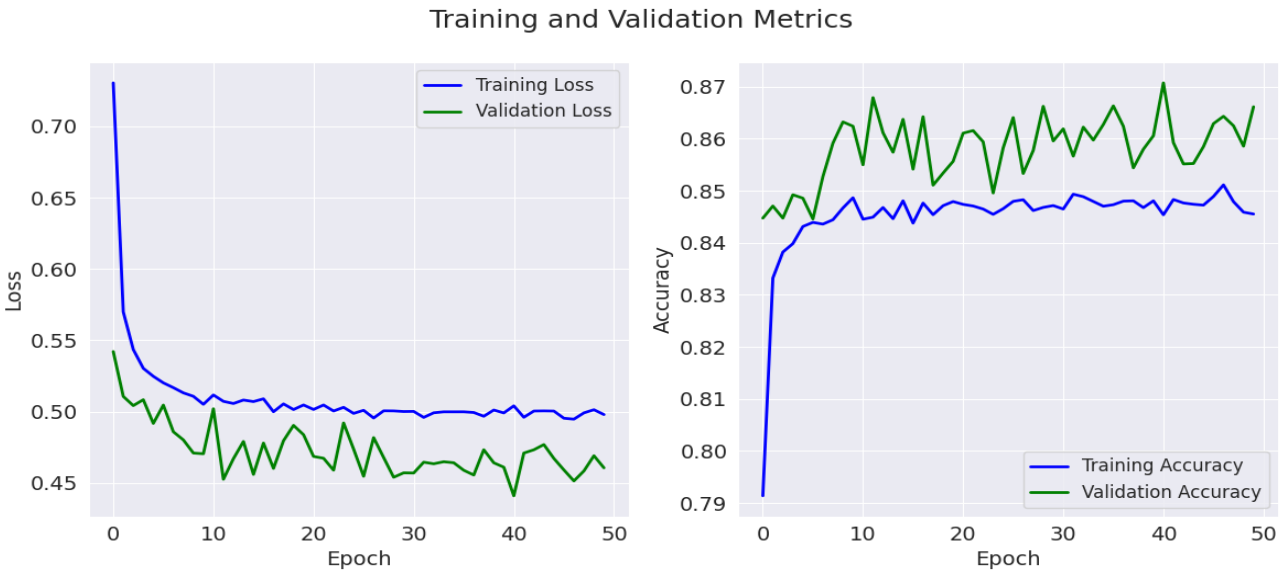

Fig. 3 Model fit using MLP with Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.1.4 MLP Base Model + L2 Regularization + Dropout

The model with both L2 regularization and dropout is performing better than the model with only L2 regularization. The training accuracy has decreased slightly, but the test accuracy has increased from 84.14% to 84.94%. This suggests that the addition of dropout has helped to reduce overfitting, leading to better generalization performance on the test set.

The loss and accuracy curves also show that the model with both L2 regularization and dropout is performing better. The validation loss and validation accuracy curves are smoother, indicating that the model is not overfitting as much. The training loss and training accuracy curves are also smoother, which suggests that the addition of dropout is helping the model to converge more smoothly during training. Overall, it seems that the combination of L2 regularization and dropout has improved the performance of the model, both in terms of reducing overfitting and improving generalization performance on the test set.

Fig. 4 Model fit using MLP with both L2 Regularization and Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.2 CNN

6.2.1 CNN Without Regularization

From training the base model, i.e., without applying any regularization, it is observed that the model is overfitting. A significant difference between training and cross-validation accuracies is noticed. Fig.2 shows the loss and accuracy for training and validation plotted for each epoch. After an epoch of 10, the difference between training and validation accuracy increases, i.e., leading to overfitting. The final training and test accuracies are 100 % and 92.8 %.

Fig. 5 Model fit using CNN without any regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs.

6.2.2 CNN + Base Model + L2 regularization

After the L2 regularizer is applied to the base CNN model, there is a significant reduction in overfitting. The training accuracy has reached 96 - 98 % from 99 - 100% of the base model. Cross-validation and test accuracies are at the same values as the base model - 92-93%. From the plots Fig. 5, we perceive that many more epochs delay overfitting - the cross-val loss & accuracy plots follow the training plots for many more epochs before digressing.

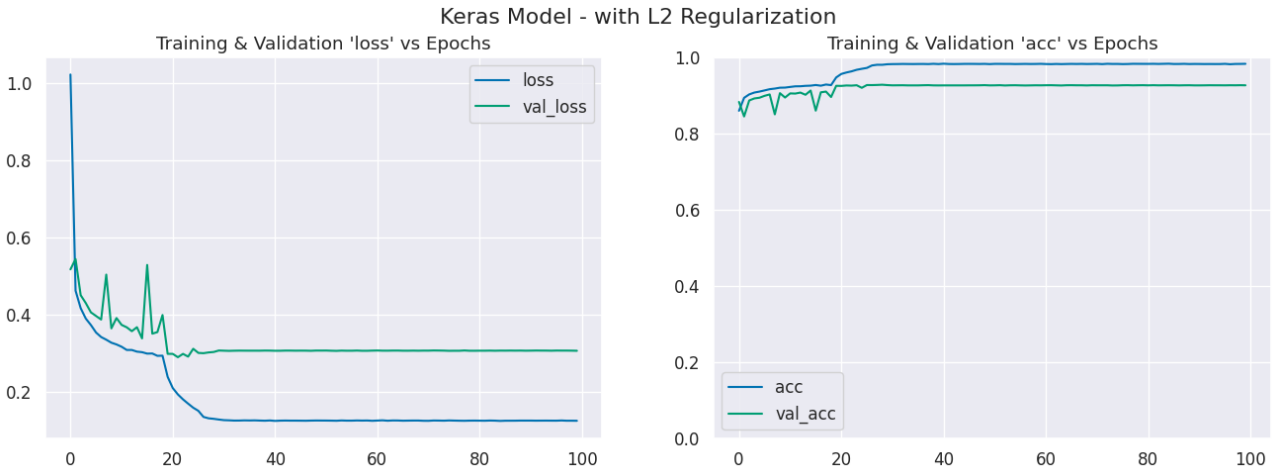

Fig. 6 Model fit using CNN with L2 regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.2.3 CNN Base Model + L2 Regularization

Dropout regularization is added to the CNN base model as the model is being developed. Similar to how L2 regularization reduced overfitting, adding dropout regularization had the same effect. Training accuracy decreased from 99 to 97 % with the base model to 96 to 97 % after adding dropout regularization—a minor improvement over L2 regularization. The accuracy of cross-validation and tests is marginally higher than that of the base model and the model with L2 Regularization. The cross-val loss & accuracy plots follow the training plots for many more epochs; Fig. 8's charts demonstrate that many more epochs postpone overfitting. It is challenging to choose between the two models—one with L2 regularization and one with batch normalization—and declare which is superior. Based on the results, the model with Dropout Regularization can be chosen purely based on exam accuracy.

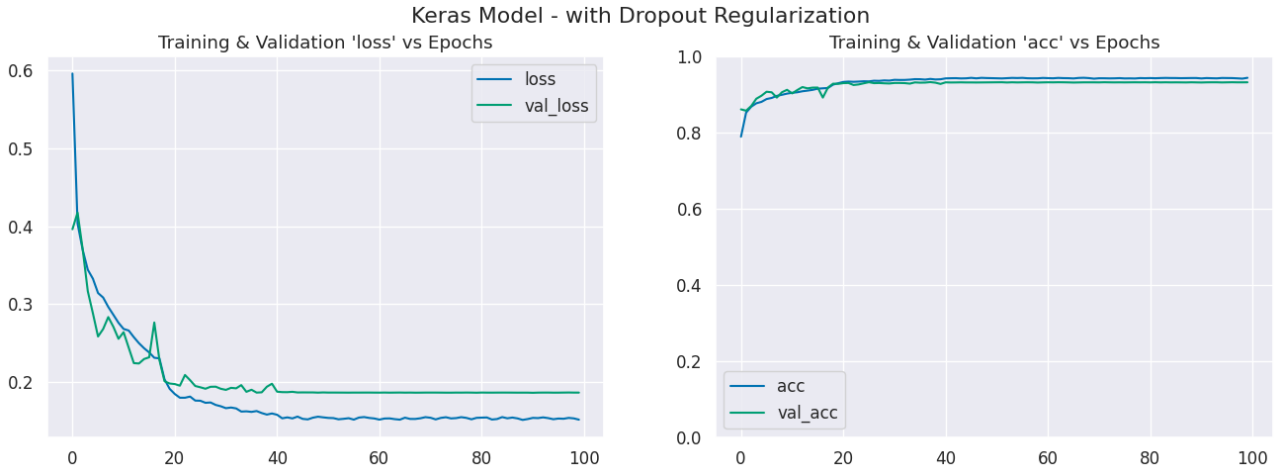

Fig. 7 Model fit using CNN with Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.2.4 CNN Base Model + L2 regularization + dropout

The effect of regularization on the model architectures is also tested by adding L2 Regularization and Dropout to the CNN model. Using both L2 regularization & Dropout regularization gives worse performance than using just L2 or just dropout regularization. Our combination of L2 regularization & dropout regularization was *too aggressive* for this model's architecture. We can *simplify* by reducing the values used. Fig. 11 shows that though the generalization error for the model is less, the accuracy for both training and validation reduces significantly. The work concludes that with this model architecture, the best performance can be achieved using

Dropout Regularization - changing model architecture may give different performance metrics.

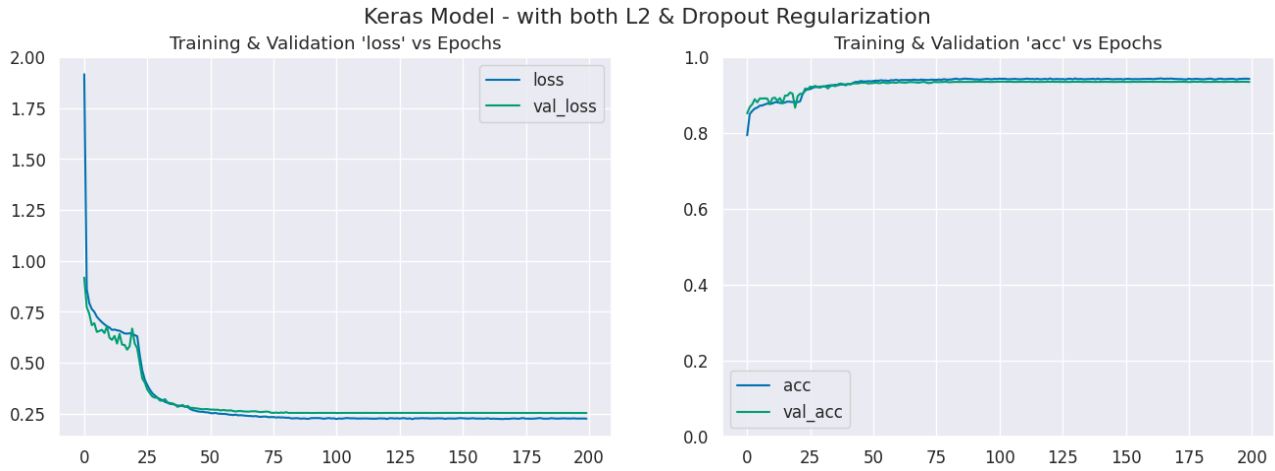

Fig. 8 Model fit using CNN with both L2 Regularization and Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

6.3 VGG-16

6.3.1 VGG16 Without Regularization

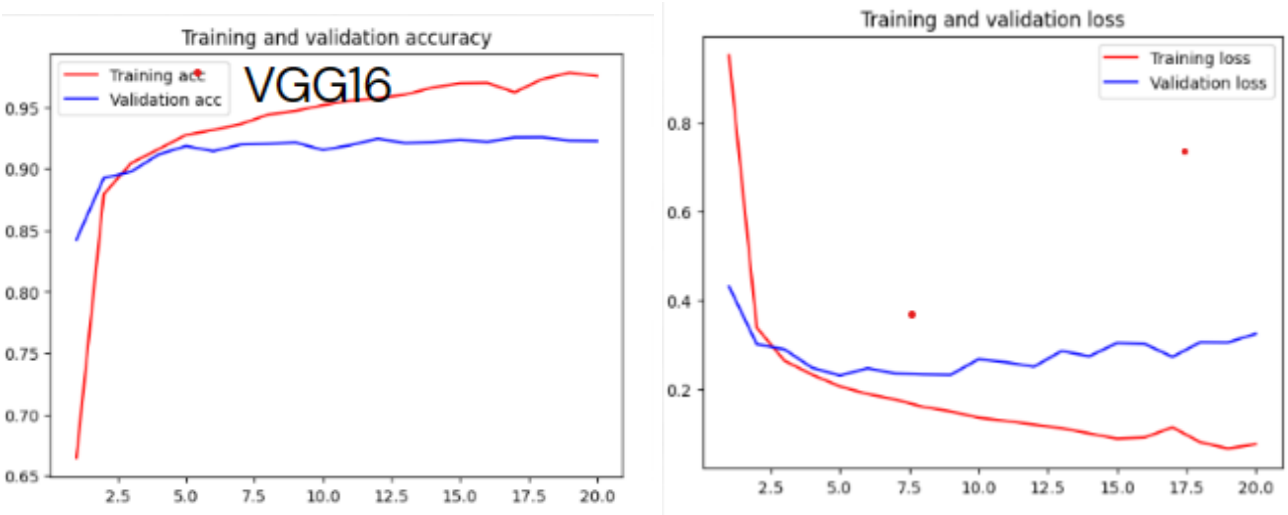

Fig. 9 Model fit using VGG16 without any regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs.

The base version of the VGG16 model has 14 Convolution layers and 5 Max Pooling layers. And without any regularization the training accuracy was 97.5% and the test accuracy 92.2%. Which clearly shows us the presence of some overfit. Looking at the plots, we can see there is a major gap between training and validation loss. And the validation loss doesn't change much after a point while the training loss keeps declining.

6.3.2 VGG-16 Base Model + L2 Regularization

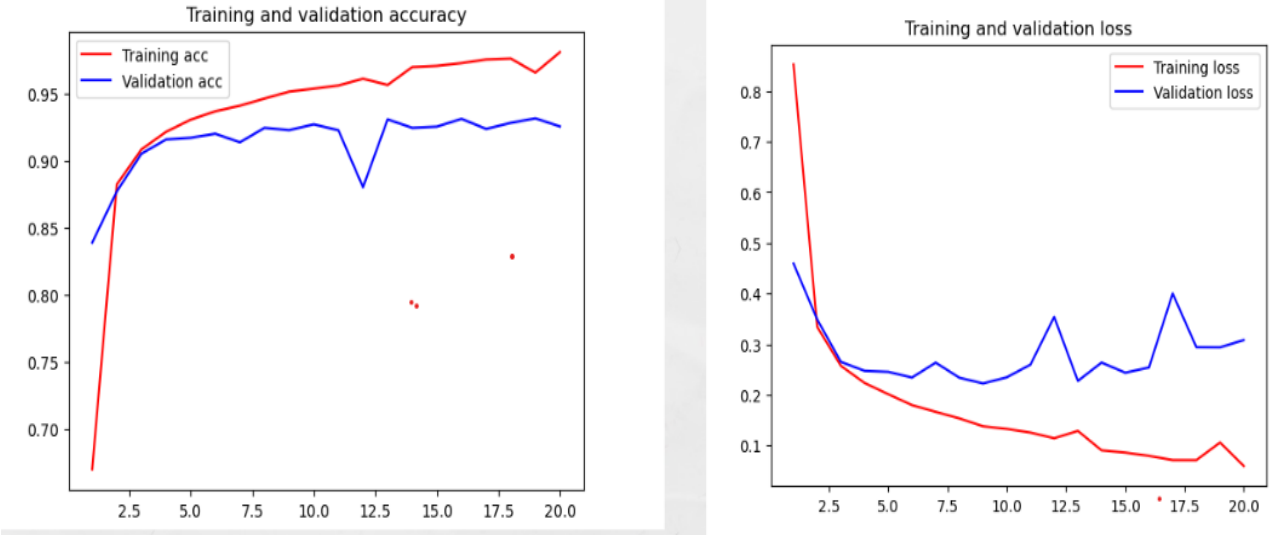

Fig. 10 Model fit using VGG16 with L2 regularization: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

For L2 regularization, a dense layer has been added before the final layer. And the regularization parameter of that layer is set to L2 or weight decay. The value of λ has been set to 0.01 for this evaluation. The output of this regularized layer are fed to the next layer in the network.The L2 method yielded a training accuracy of 98.1 and a test accuracy of 92.5. Using L2 regularization slightly improved the test accuracy but overall did not improve generalization by much.

6.3.3 VGG-16 Base Model + Dropout

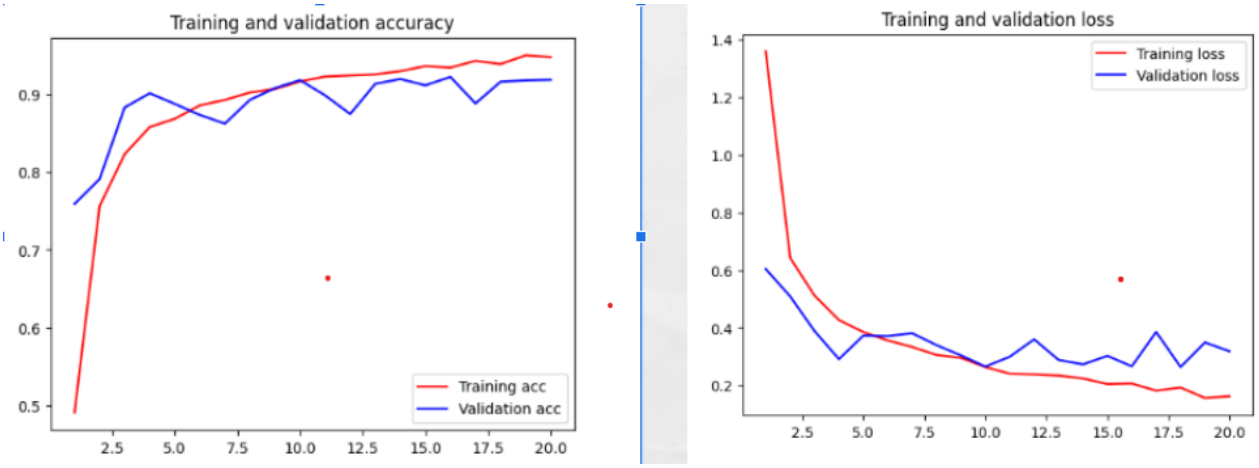

Fig. 11 Model fit using VGG16 with Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

For the dropout method a dense layer was added before the last layer and then a dropout layer with a dropout probability of 0.5. The output of this dropout layer is fed to the next layer. The training accuracy for this method was 94.7 and the test accuracy 91.8.

The dropout method improved generalization and also reduced overfitting. We can observe that the training loss doesn't decrease like it did in the models without dropout.

6.3.4 VGG16 Base Model + L2 regularization + Dropout

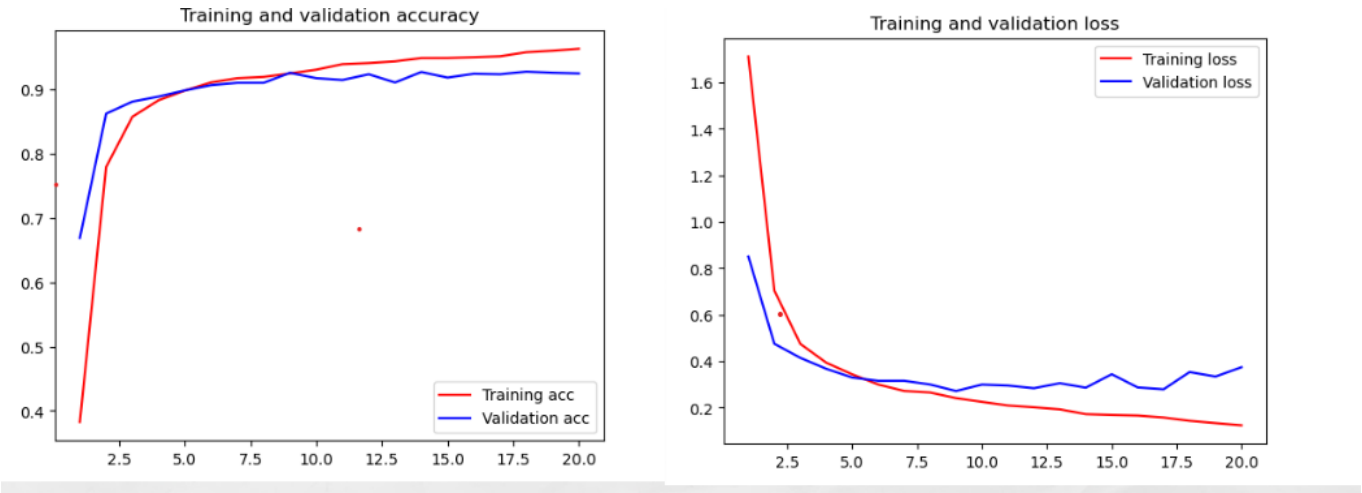

Fig. 12 Model fit using VGG16 with both L2 Regularization and Dropout: Training and Validation loss vs epochs and Training and Validation accuracy vs epochs

To implement both types of regularization, a dense layer with L2 regularization is added first and then a dropout layer after that. The train accuracy was 96.3 with this model and the test accuracy 92.5. Although the test accuracy is similar to what we saw in the L2 method, we can say that this model generalized slightly better than the previous ones.

Model comparison and Observations:

Model	L2 Regularization	Dro-pout	Train Accuracy	Test Accuracy
MLP	No	No	96.6	88.5
	Yes	No	88.7	85.8
	No	Yes	84.5	85.3
	yes	Yes	82.4	84.5
CNN	No	No	100	92.8
	Yes	No	98.4	92.6
	No	Yes	96.2	93.2
	Yes	Yes	96.3	93.4
VGG16	No	No	97.5	92.2
	Yes	No	98.1	92.5
	No	Yes	94.7	91.8
	Yes	Yes	96.3	92.5

After evaluating the performance of these models with several variations, we observe a lot of things. The CNN model fits extremely well giving a 100% accuracy when no form of regularization is used. But it doesn't perform the same on unseen data, which clearly tells us there is some overfitting. The same goes with VGG16 as well, which is a more complex CNN model. In our observations dropout regularization gave the best results among the techniques we used. Using regularization techniques did control the amount of overfit for all the models, but showed little to no improvement in test accuracy.

In conclusion, the study implies that there are several techniques of addressing overfitting. No best way works for all models - several options must be explored to get the best results. As seen in the case of CNN, regularization can help get better results and generalize well, but the model also performs well without regularization. Hence, regularization is not the cause for generalization.

7. Implicit Regularization

Implicit regularization is when the algorithm implicitly regularizes the solution to converge to certain minima while avoiding others. Implicit regularization is significant when not all models that match the training data well generalize, even when explicit regularizers like dropout and weight decay may not be necessary. We always choose our model when stochastic gradient descent is used in neural networks. We examine how SGD functions as an implicit regularizer using linear models.

Stochastic Gradient Descent (SGD):
A gradient-based optimization technique called SGD is used in machine learning and other domains to modify a model's parameters to reduce a specified cost function. It is an iterative process in which, to lower the cost of the sample, a random sample is taken from the dataset at each iteration, and the model's parameters are updated. The following is the SGD weight update formula:

$$W_{t+1} = W_t - \eta_t e_t x_i$$

e is the partial derivative of the loss and x – linearly independent data samples. Training by labeling is achieved with flawless precision. Global minima are obtained by solving the global equation. So, we get a unique global minimum. Additionally, we have a kernel matrix that we refine to receive the lowest l2 norm. However, the minimal norm says nothing about regularization.

8. Advantages and Disadvantages of Methods

I. CNN: These are distinguished by their capacity to automatically learn and extract useful features from raw input data without the requirement for explicit feature engineering. They can also handle input of multiple sizes and scales as well as other image adjustments, like rotation and scaling. CNNs are also computationally efficient because of weight sharing and pooling procedures, which limit the amount of parameters that must be learned.

II. MLP: It is capable of learning complex nonlinear correlations between input and output data, making it useful for a variety of applications. With the help of scalable optimization methods like stochastic gradient descent (SGD), it can effectively manage a variety of input and output types. Furthermore, MLP is interpretable and partially illustrable, allowing one to comprehend how the network derives its predictions. Moreover, MLP is capable of handling both regression and classification tasks.

III. VGG16:

VGG16 provides a high accuracy on a wide range of image recognition tasks. VGG16 can be used for transfer learning, which allows for the rapid development of deep learning models.VGG16 is a very deep and complex network, which means it takes a long time to train and requires a lot of computational resources.VGG16 does not scale well to larger images, thus requiring more training data to achieve good performance.It is also very prone to overfitting.

9. Conclusion

The study demonstrates the effect of explicit regularization methods like weight decay, and dropout performed on the MNIST datasets and shows the importance of model architectures. The architecture of a deep learning model can significantly affect its generalization performance, which refers to how well the model performs on new, unseen data. Though CNN and VGG16 are both convolutional, the number of layers determines the model's generalization. And CNN performs accurately resulting in better model accuracy and less generalization error than VGG16. In this work, the adequate capacity of neural networks is identified. Their capacity to train on the 70000 images, resulting in a high accuracy, is astonishing. Explicit forms of regularization, such as weight decay and dropout, may improve generalization performance; they are only sometimes necessary for good performance. In conclusion, our explicit and implicit regularizer observations consistently imply that regularizers could enhance generalization performance when appropriately tuned. The networks continue to function well even with all the regularizers removed; hence, it is doubtful that the regularizers are the primary cause of generalization.

This work defined and explained the effective capacity of deep learning models using a straightforward experimental approach. The studies we carried out highlight that the effective capacity of several successful neural network architectures can successfully shatter the training data. As a result, these models are, in theory, complex enough to remember the practice data. This circumstance presents a conceptual challenge to statistical learning theory since a massive artificial neural network's generalization ability is difficult to explain using typical measurements of model complexity. A precise formal measure under which these huge models are simple has yet to be found. Another insight from our experiments is that optimization remains empirically accessible even if the resulting model does not generalize. This shows that the reasons optimization is empirically easy must differ from the true cause of generalization.

References:

[1] The paper under analysis - Understanding Deep Learning requires rethinking Generalization

[2] VGG16 model by Karen Simonyan, Andrew Zisserman

[3] understanding weight decay in MLP

[4] Dropout regularization to handle overfitting

[5] A Survey on Regularization Methods for Convolutional Neural Networks