Deeper Networks for Image Classification

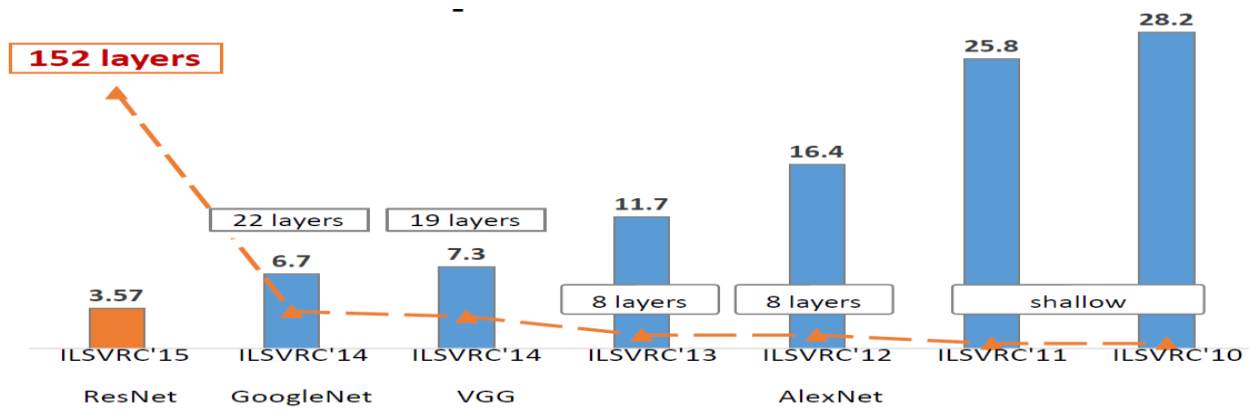
1. Introduction

Deep convolutional neural networks [2] have led to a series of breakthroughs for image classification. In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014). Convolutional neural networks (CNNs) [1] consist of alternating convolutional layers and pooling layers. Deep networks naturally integrate low/mid/high-level features and classifiers in an end-to-end multilayer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth).Convolution layers take inner product of the linear filter and the underlying receptive field followed by a nonlinear activation function at every local portion of the input. The resulting outputs are called feature maps. Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition [1,2,3] which has become possible due to the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012).

2. Critical Analysis / Related Work

Starting with LeNet-5 [1], convolutional neural networks (CNN) have typically had a standard structure –stacked convolutional layers (optionally followed by contrast normalization and max-pooling) are followed by one or more fully connected layers. Variants of this basic design are prevalent in the image classification literature and have yielded the best results to-date on MNIST, CIFAR and most notably on the ImageNet classification challenge [2]. For larger datasets such as ImageNet, the recent trend has been to increase the number of layers and layer size, while using dropout to address the problem of overfitting. Despite concerns that max-pooling layers result in loss of accurate spatial information, the same convolutional network architecture as [3] has also been successfully employed for localization [3], object detection [4] and human pose estimation [5] like in VGG. Furthermore, Inception layers are repeated many times, leading to a 22-layer deep model in the case of the GoogLeNet model. The current state of the art for object detection is the Regions with Convolutional Neural Networks (R-CNN) method by Girshick et al. [6]. R-CNN decomposes the overall detection problem into two subproblems: utilizing low level cues such as color and texture in order to generate object location proposals in a category-agnostic fashion and using CNN classifiers to identify object categories at those locations.

After the celebrated victory of AlexNet [1] at the LSVRC2012 classification contest, deep Residual Network [2] was arguably the most ground-breaking work in the computer vision/deep learning community in the last few years. ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance. As ResNet gains more and more popularity in the research community, its architecture is getting studied heavily. ResNeXt, Densely Connected CNN and Deep Network with Stochastic Path were new small network variants over ResNet to tackle the problems of previous models.



3. Method / Model Description

Three models have been used to accomplish the task VGG16, GoogleNet and ResNet. All the models have been made according to description in the paper published. The model has been trained and tested according to the input size of the data. No pre trained weights are used on any of the network.

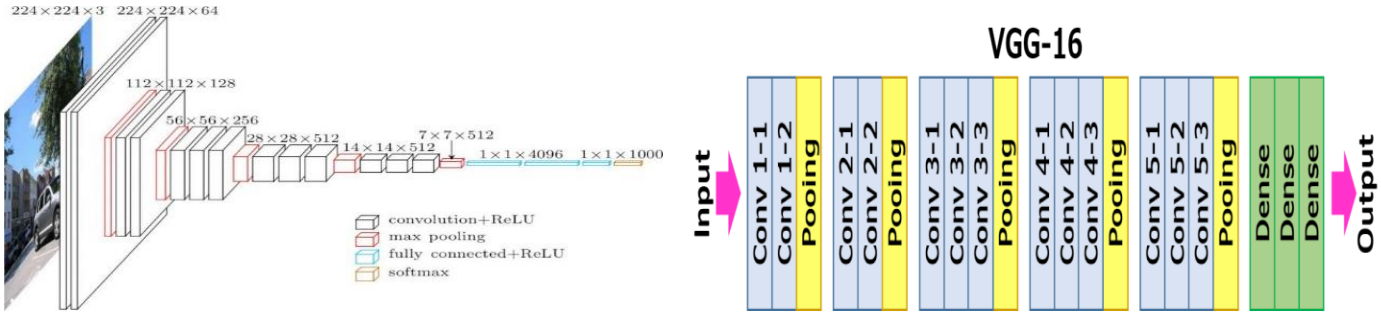
3.1 Model Architecture

(I) VGG-16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". Its name VGG-16 comes from the fact that it has 16 layers. Its layers consist of Convolutional layers, Max Pooling layers, Activation layers, Fully connected layers.

Architectural Details:

There are 13 convolutional layers, 5 Max Pooling layers and 3 Dense layers which sums up to 21 layers but only 16 weight layers. During training, the input to VGG ConvNets is a fixed-size 32 × 32 RGB image. The only pre-processing that is done is subtracting the mean RGB value, computed on the training set, from each pixel.The image is passed through a stack of convolutional (conv.) layers, where VGG use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations model also utilise 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolution layers Max-pooling is performed over a 2 × 2-pixel window, with stride 2. 3 Dense layers is used at the end to add the non-linearity property which helps to model any mathematical function.

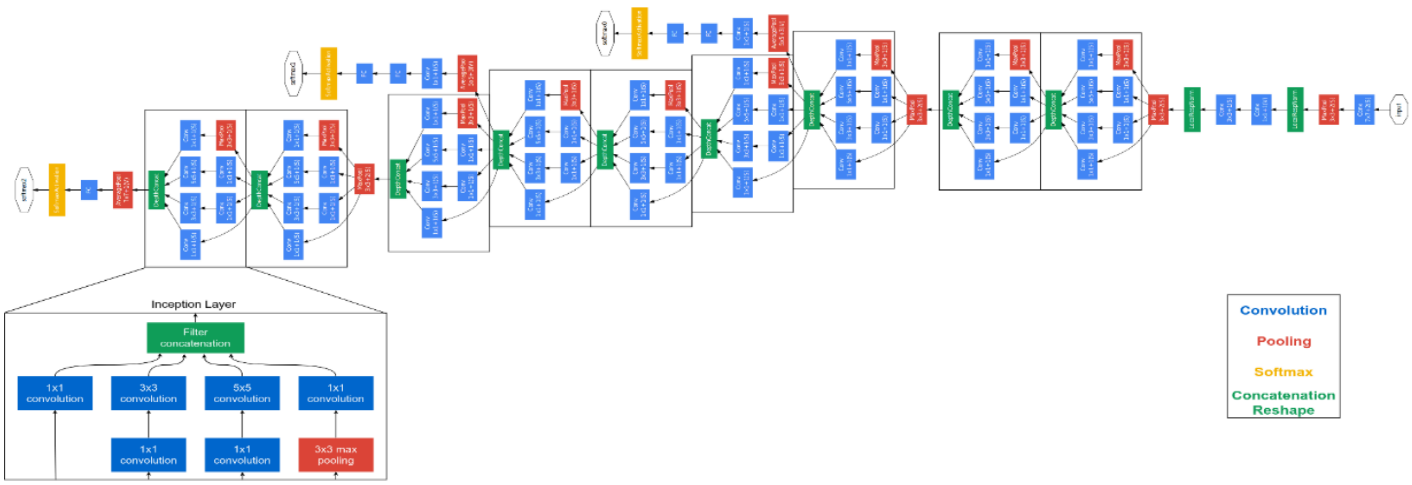


(II) GoogLeNet

ILSVRC 2014 competition is one of the largest and the most challenging computer vision challenge. This challenge is held annually and each year it attracts top machine learning and computer vision researchers. The winner of the competition was GoogleNet from Google. It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was rather hard to do and required some human training in order to beat GoogLeNet's accuracy. GoogleNet has 22 layers, and almost 12x less parameters (So faster and less then Alexnet and much more accurate).The GoogLeNet builds on the idea that most of the activations in a deep network are either unnecessary (value of zero) or redundant because of correlations between them. Therefore, the most efficient architecture of a deep network will have a sparse connection between the activations, which implies that all 512 output channels will not have a connection with all the 512 input channels. There are techniques to prune out such connections which would result in a sparse weight/connection.The Inception network was an important milestone in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance. The idea of the inception layer is to cover a bigger area, but also keep a fine resolution for small information on the images. So, the idea is to convolve in parallel different sizes from the most accurate detailing (1x1) to a bigger one (5x5).

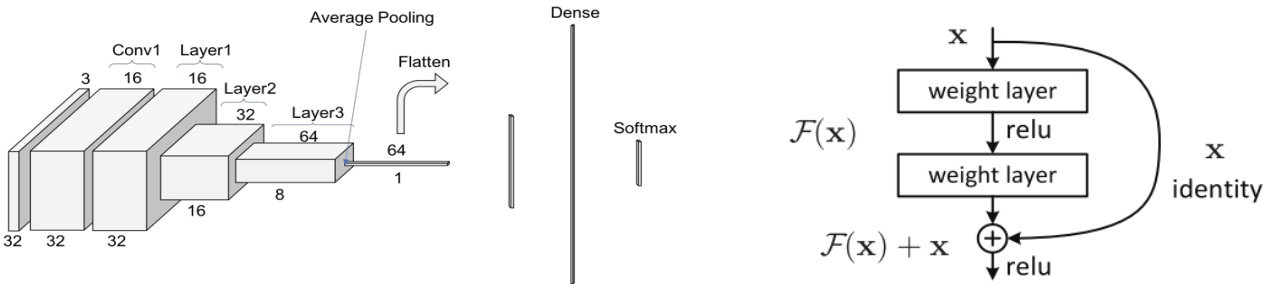
Architectural Details:

The first inception module of GoogLeNet as an example which has 192 channels as input. It has just 128 filters of 3X3 kernel size and 32 filters of 5X5 size. The order of computation for 5X5 filters is 25X32X192 which can blow up as we go deeper into the network when the width of the network and the number of 5X5 filter further increases. In order to avoid this, the inception module uses 1X1 convolutions before applying larger sized kernels to reduce the dimension of the input channels, before feeding into those convolutions. So, in the first inception module, the input to the module is first fed into 1X1 convolutions with just 16 filters before it is fed into 5X5 convolutions. This reduces the computations to 16X192 + 25X32X16. All these changes allow the network to have a large width and depth.



(III) ResNet20v1

Results from recent studies has come to conclusions that network degradation is not caused by overfitting and adding more layers to a suitably deep model leads to higher training error, as reported in and thoroughly verified by experiments done on different citations. Here I have used ResNet20 model to perform the image classification task. The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in the below figure.



Architectural Details:

Here I have used 18-layer network ,the convolutional layers mostly have 3×3 filters and follow two simple design rules i.e. for the same output feature map size, the layers have the same number of filters and if the feature map size is halved, the number of filters is doubled to preserve the time complexity per layer. The model performs down sampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully connected layer with softmax.

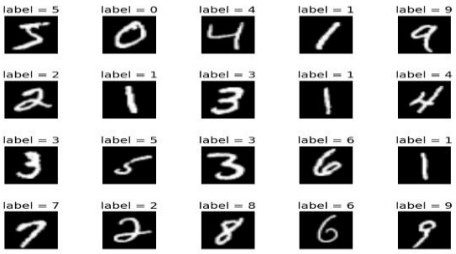
4. Experiments

In this deeper network task, I have used MNIST dataset to evaluate the deeper networks and CIFAR10 to further evaluate the effectiveness of deeper CNN models for image classification.

4.1 Datasets

a. MNIST Dataset
The MNIST database [1] of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been sized normalized and centred in a fixed-size image.

b. CIFAR-10 Dataset
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.



Input Shape and Optimizer:

I. **VGG16:** The MNIST data has been fed to the model by reshaping from 28*28*1 to 32*32*3, stacking extra 3 channels. The CIFAR10 data has been fed directly in the shape 32*32*3.Stochastic gradient descent(**SGD**) optimizer has been used to train the model with parameters Includes support for momentum, **learning rate** decay, and **Nesterov** momentum which helps to accelerates SGD in the relevant direction and dampens oscillations and **clipvalue** which has been used to control gradient clipping.

II. **GoogleNet:** The MNIST data has been fed in shape 28*28*3 and 32*32*3 for CIFAR10 data. **Adadelta** optimizer is used which adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done. In the original version of Adadelta you do not have to set an initial learning rate.

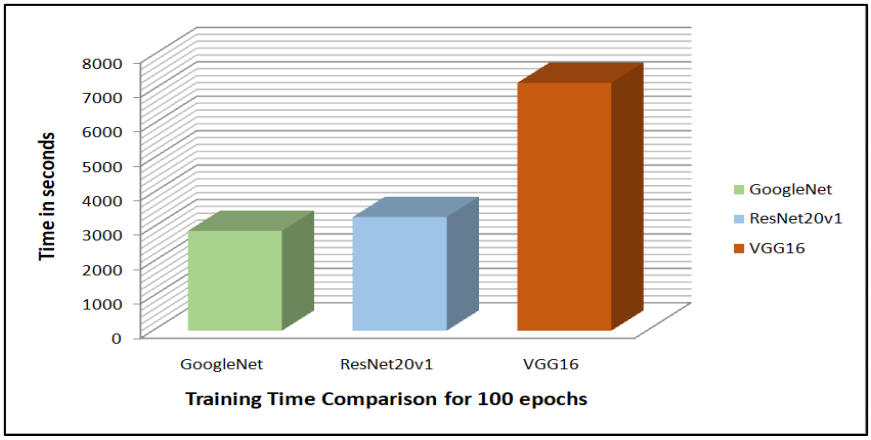
III. **ResNet20v1:** The model takes the input shape of 32*32*3 and both the dataset have been fed in the similar format. The MNIST 1 channel has been converted to 3 channels. **Adam** optimizer is used to train the model with parameters include support for learning rate.

All the three networks have been trained from scratch on GPU and RAM provided over Google Colab. The models have been evaluated on different learning rate and optimizer. For the ResNet network the learning rate has been fed in the loop which changes with the number of epochs.

4.2 Testing Results

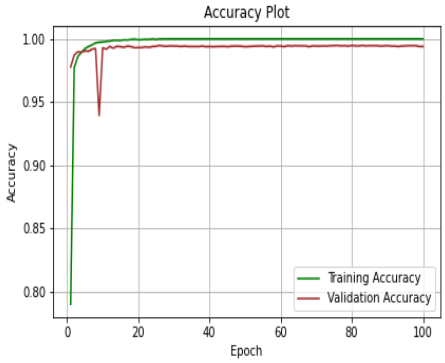
4.2.1 Training Time

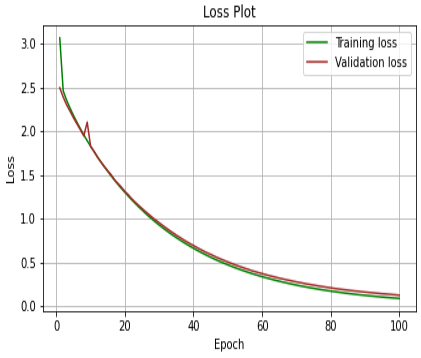
The training time has been found out for all the models. The Initial Learning Rate for all the model is kept at 0.01 and it is clear from below figure that VGG takes longer time to train the network and is quite slow when compared to GoogleNet and ResNet. The VGG model takes around 7200 sec which is more than double the time what GoogleNet and ResNet take.

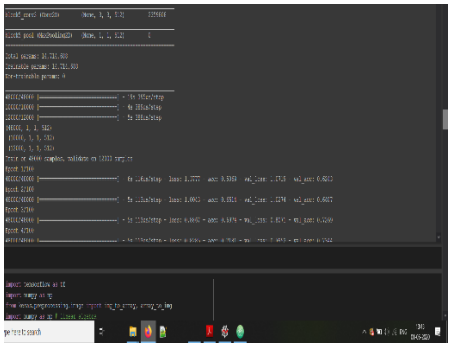


4.2.2 Training Accuracy and Loss for MNIST Dataset:

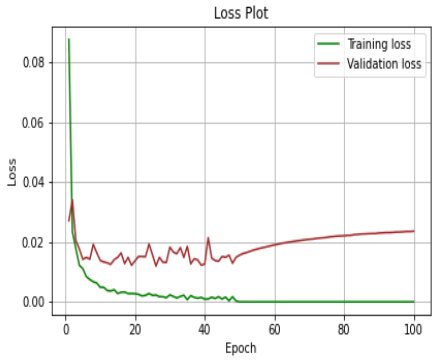
I. **VGG16:** The model performs very well for MNIST dataset reaching maximum test accuracy of 99.42% with learning rate of 0.001. The model is time costing but provides good results overall on the dataset. The model learns easily and even at 10 epochs gives around 99% accuracy.

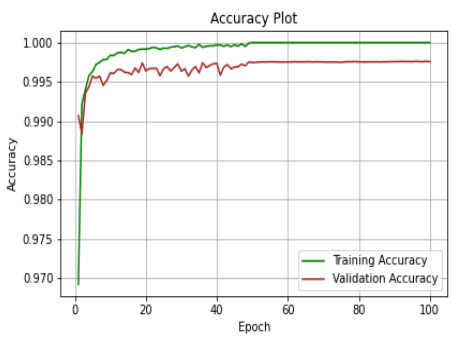


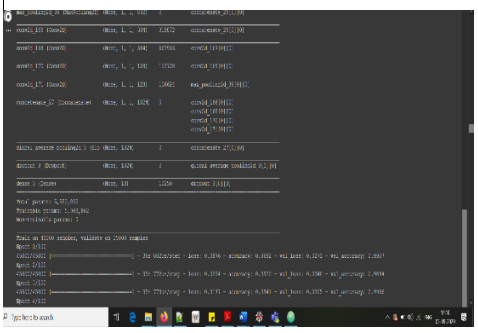




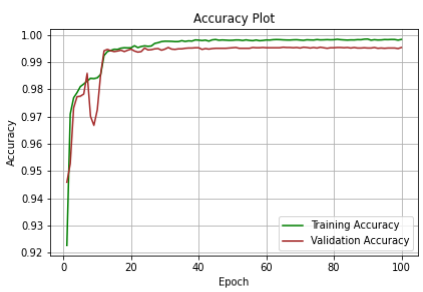
II. **GoogleNet:** This model is complex and has the maximum layers amongst the model tested but model does excellent on the MNIST dataset and reaches maximum accuracy of 99.83% with learning rate of 0.001. The model at 10 epochs gives around 99.5% accuracy, increases further but stabilizes after 50 epochs.

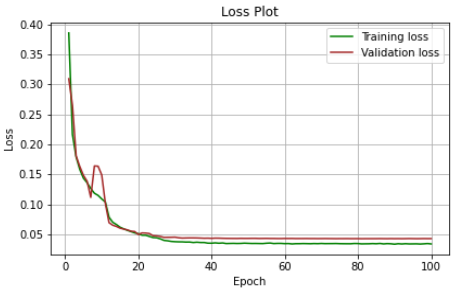


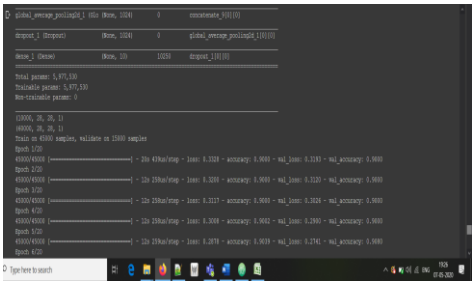




III. **ResNet20v1:** The model reaches accuracy of 99.54% with learning rate of 0.001 and perfoms better than the VGG16 and is quite comparitive with the googleNet results. The model reaches significantly well accuracy after 10 epochs.



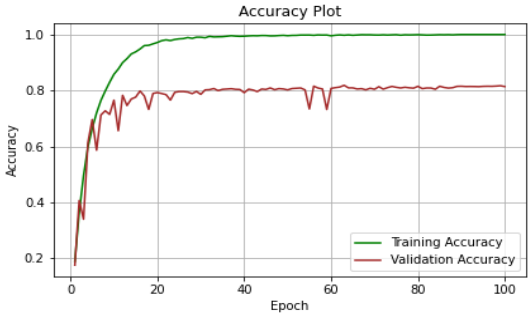


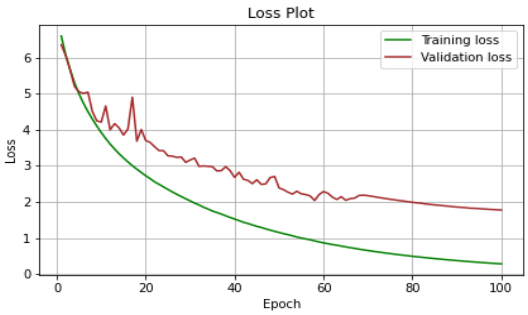


4.3 Further Evaluation

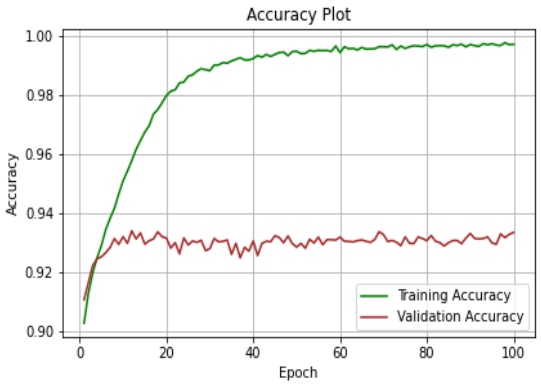
Training Accuracy and Loss for CIFAR-10 Dataset:

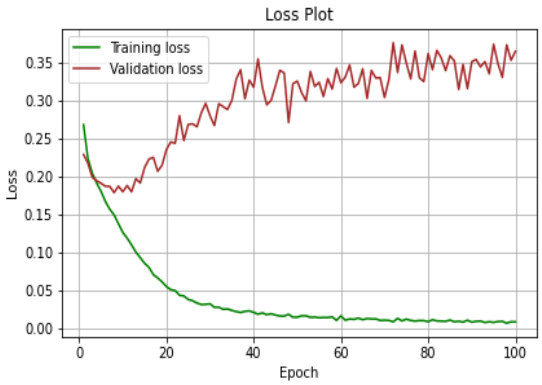
I. VGG16: The model performs very well for CIFAR dataset reaching maximum test accuracy of 81.16% with learning rate of 0.001. The model is time costing but provides good results overall on the dataset. The model learns slowly and provides stable results after 20 epochs.



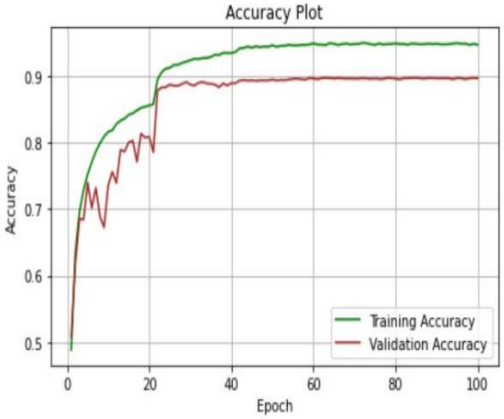


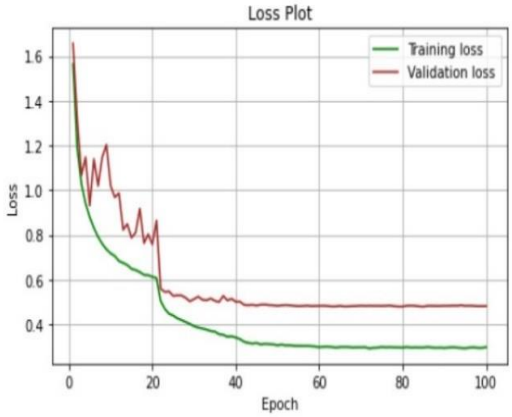
II. GoogleNet: The model does excellent on the CIFAR dataset as well and reaches maximum accuracy of 92.664% with learning rate of 0.001. The model learns slowly and after 40 epochs gives around 92% accuracy, increases further but stabilizes after 50 epochs. But the Validation loss increases after 20 epochs and shows clear signs of overfitting due to more layers.





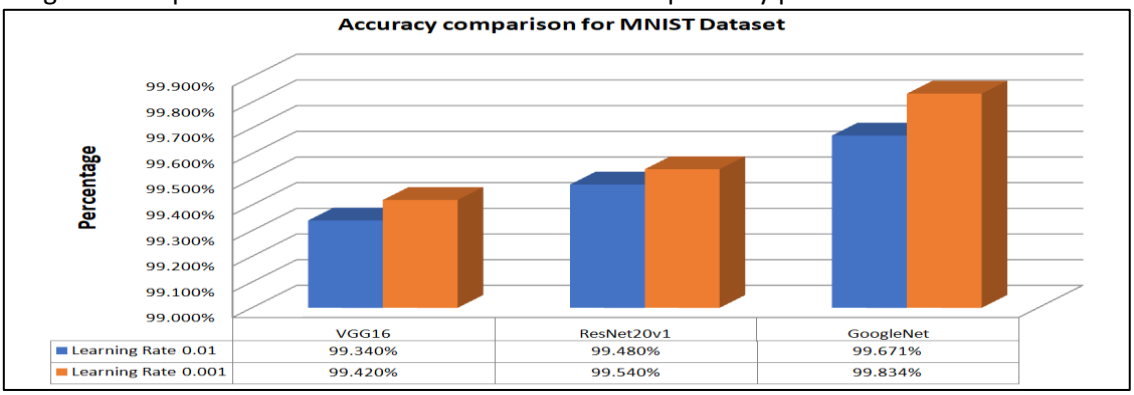
III. ResNet20v1: The model reaches accuracy of 89.31% with learning rate of 0.001 and perfoms better than the VGG16 and is quite comparitive with the googleNet results. The model reaches significantly well accuracy after 30 epochs. The loss also decreases well till 40 epochs and stablizes thereafter.



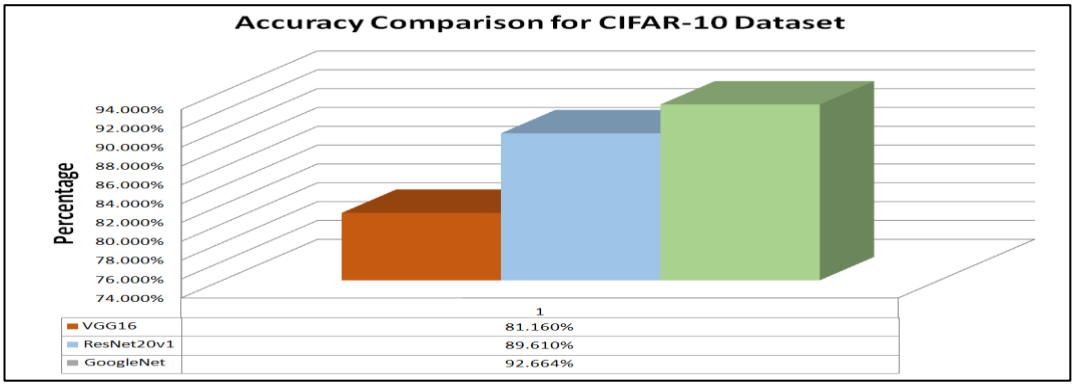


4.4 Comparison between models

I. Below is the comparison chart for all the models with different learning rates.All the models has performed really well with more than 99% accuracy result on test data. It is clearly evident that the Googlenet has perfomed best on all and VGG16 has comparetivly performed least of all.



II. Below is the comparison on models for CIFAR dataset. The values are compared over learning rate of 0.001. The clear performer is again GoogleNet here due to complexity and dense layers.



4.5 Additional Work

On the VGG16 model I have tried to add an extra dense layer to the model to add an interesting non-linearity property. The activation function used is ReLU with dropout of 0.5. The training and testing have been done on CIFAR-10 dataset. The model accuracy increases slightly with LR of 0.001 but decreases if the LR is 0.01. So, it is evident that if the LR is small, accuracy will be more but at the cost of time which is not fruitful for bigger datasets.

	Test Accuracy	
Model/LR	0.01	0.001
VGG16	0.81279999	0.809899986
VGG16 with extra dense layer	0.806900024	0.811600029

5. Conclusion

In this work I have evaluated very deep convolutional networks for largescale image classification. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth. While VGG achieves a phenomenal accuracy on MNIST dataset, its deployment on even the most modest sized GPUs is a problem because of huge computational requirements, both in terms of memory and time. It becomes inefficient due to large width of convolutional layers. GoogleNet has 22 layers which increases the accuracy, but it can increase the overfitting problem. Likewise, Number of layers and ScaleRange increases the accuracy of the GoogleNet and the result above speak for themselves. GoogleNet has performed very well on both the datasets. The ResNet model with 18 layers has performed well on both the datasets as well. ResNet with the idea of 'identity shortcut connection' helps us to understand that increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. ResNet with 18 layers has performed significantly better than the VGG and comparable to GoogleNet.

I have also shown that models generalise well to a wide range of tasks and datasets, matching or outperforming more complex recognition pipelines built around less deep image representations. The results have yet again confirmed the importance of depth in visual representations.

6. References

1. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E.Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition.Neural Comput., 1(4):541–551, Dec. 1989.
2. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. J. Fleet, T. Pajdla,B. Schiele, and T. Tuytelaars, editors, ECCV,volume 8689 of Lecture Notes in Computer Science,pages 818–833. Springer, 2014.
3. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1106–1114, 2012.
4. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition,localization and detection using convolutional networks. CoRR, abs/1312.6229, 2013.
5. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. CoRR, abs/1312.4659, 2013.