Deeper Networks for Image Classification

Author - Madhuri Ramesh Patil
Department of EECS,Queen Mary University of London
ec19584@ac.uk

1. Introduction

Convolutional networks (ConvNets) is achieving a great success in computer vision field Mainly in image retrieval and classification domain. For instance it is applied to image processing tasks, human action recognition[1][3] .One encouraging news is that most of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures.
The aim of this work is to analyze the Performance and evaluate image classification with deeper networks. The few models of CNN namely, VGG-16 and Googlenet are used for this task. Dataset used for training are mnist and CIFAR. A training process is coded with KERAS and TENSORFLOW. In the competition held in 2014 in Image classification task[2] Google net acquired first place and VGG acquired 2nd place. Google net achieved this by reducing computational time with high accuracy and VGG explored more in depth and got equally high accuracy but at the expense of more computational time. The computational time of VGG16 is less than VGG19 so this model is chosen for classification task in this paper.

2. Related Work

Convolutional Neural Networks have been applied to a variety of tasks with state-of-the-art performance in several applications. The most used architecture follows that of (LeCun et al. 1998)[4], where a CNN was first applied to recognize hand-written digits.
The network has been improving since its creation with the development of new layers [5][6] and the addition of classic computer vision techniques. CNN is often used in the ImageNet Challenge with many different variations. 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)[5], which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings[6] (the winner of ILSVRC-2011) to deep ConvNets[7] (the winner of ILSVRC-2012).
[8] used a series of fixed Gabor filters of different sizes to handle multiple scales. A similar strategy was used further. However, contrary to the fixed 2-layer deep model of [8], all filters in the Inception architecture are learned. Furthermore, Inception layers are repeated many times, leading to a 22-layer deep model in the case of the GoogLeNet model. The best-performing submissions to the ILSVRC2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014)[1]

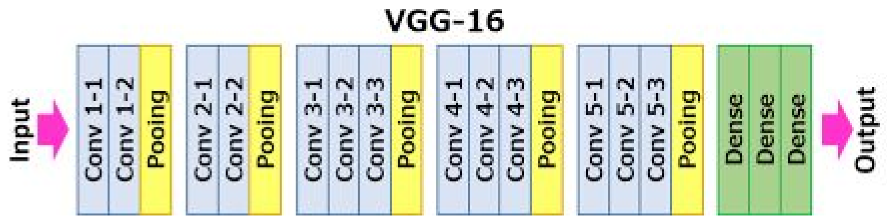
3.Model Description

In this paper, I use various deeper networks for evaluating the effectiveness of deeper CNN models for image classification on MNIST and CIFAR.

3.1 Model Architecture

(I) VGG-16

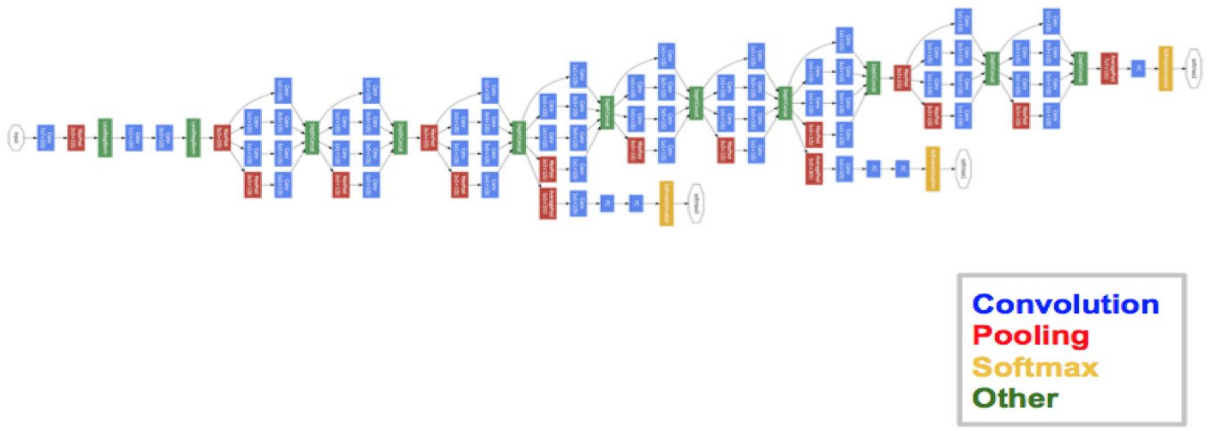
During training, the input to ConvNets is a fixed-size 224 × 224 RGB image. The preprocessing is done by subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations 1 × 1 convolution filters are utilized, 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 conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third contains 10 channels as it contains 10 labels in both MNIST and CIFAR with softmax activation. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. Local Response Normalisation (LRN) (Krizhevsky et al., 2012), when included with VGG does not improve the performance on the dataset, but leads to increased memory consumption and computation time.



(II) GoogLeNet

It consists of 22 layers. Mainly it contains repeated stack of Inception
GoogLeNet architecture is also known as Inception Module. It goes deeper in parallel paths with different receptive field sizes.This architecture consists of 22 layer in deep. It reduces the number of parameters from 60 million (AlexNet) to 4 million.Hence it has high computational speed, lower memory use and lower power use too.
Adding more layers increases the number of parameters and it is likely that the network overfits. There will be more computation, because a linear increase in filters results in a quadratic increase in computation. So, the use of the inception module and GAP is done. The fully

connected layer at the end of the network is replaced with a GAP layer because fully connected layers are generally prone to overfitting. GAP has no parameters to learn or optimize.



3.2 Training of VGG:

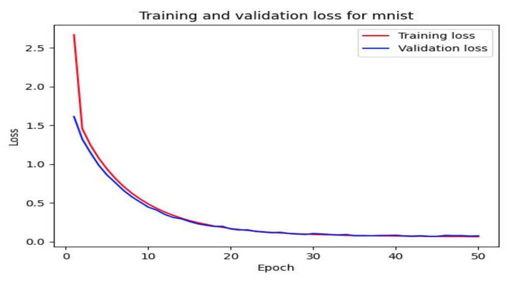
VGG:

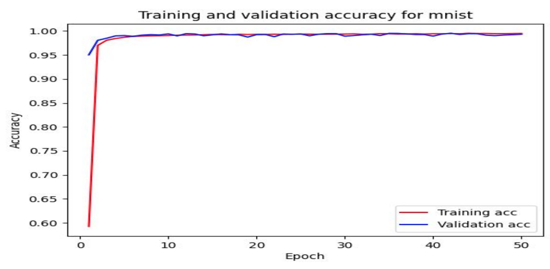
The input dataset given to the model is MNIST and CIFAR. MNIST dataset is of size 28X28,which is resized to 224X224.For CIFAR dataset the its size 32X32 is resized to 224 X224.The ConvNet training procedure generally follows Krizhevsky et al. (2012). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation (LeCun et al., 1989)) with momentum. The batch size was set to 32, momentum to 0.9. The training was regularised by weight decay and dropout regularisation for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate is set to 10−2 , and the learning was stopped after 40epochs. We conjecture that in spite of the larger number of parameters and the greater depth of our nets compared to (Krizhevsky et al., 2012), the nets required less epochs to converge due to (a) implicit regularisation imposed by greater depth and smaller conv. filter sizes; (b) pre-initialisation of certain layers.

To obtain the fixed-size 224×224 ConvNet input images, they were randomly cropped from rescaled training images (one crop per image per SGD iteration). To further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift (Krizhevsky et al., 2012).

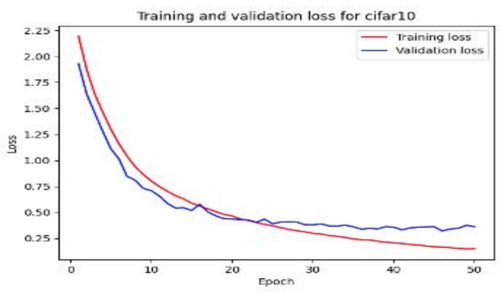
Following are the plots for accuracy and loss during training and testing process

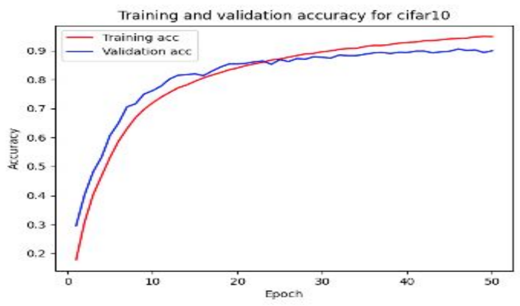
For MNIST:





For CIFAR-10:



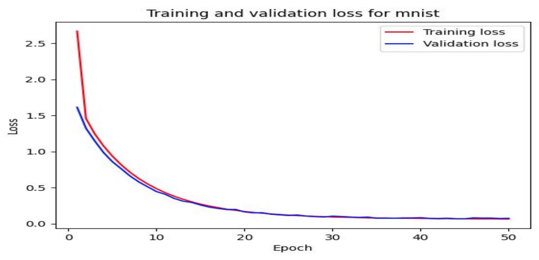


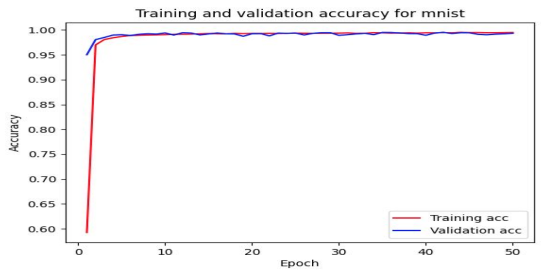
3.3 Training of Google net:

The model was trained on the MNIST(70000 images, spread over 10 different classes) and CIFAR 10 training set (60000images, spread over 10 different classes) . Each RGB image was preprocessed by resizing the smallest dimension to 256, cropping the center 256x256 region, subtracting the per-pixel mean (across all images) and then using 10 different sub-crops of size 224x224 (corners + center with(out) horizontal flips). Stochastic gradient descent with a mini-batch size of 32 was used to update the parameters, starting with a learning rate of 10−2, in conjunction with a momentum term of 0.9.The learning rate is set to 10−3 for CIFAR and 10-2 for MNIST, and the learning was stopped after 40epochs.Computational time for google net was much lesser than that of VGG model

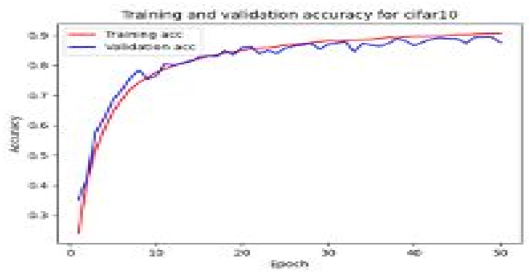
Following are the plots for accuracy and loss during training and testing process

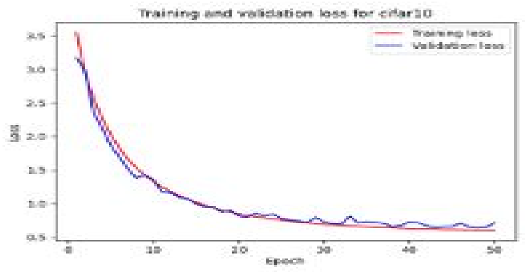
For MNIST:





For CIFAR-10: Train and validation(Test) Plots:





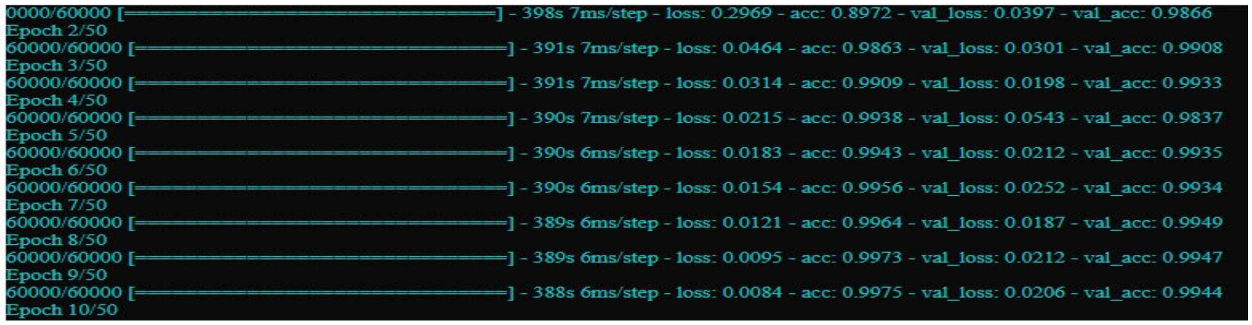
4. Experiments

4.1 Datasets

The MNISTdatabase [1] of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples of size 28X28 grayscale images and 10classes. It is a subset of a larger set available from NIST. The digits have been size normalized and centred in a fixed-size image.
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.

4.2 Testing Results

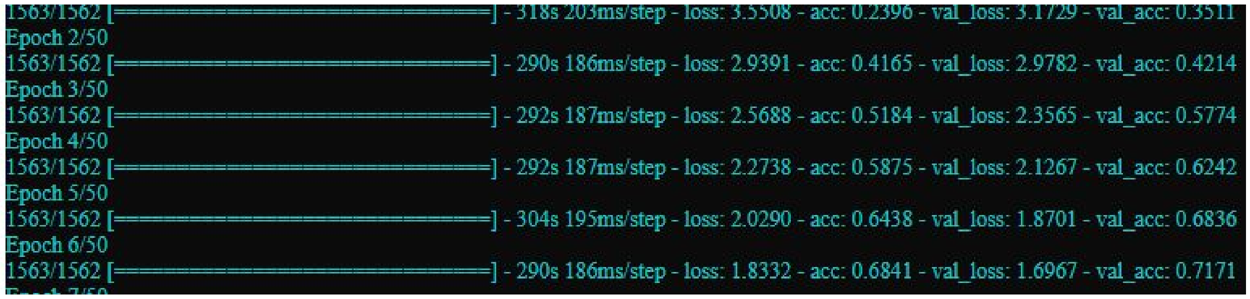
Google NETmodel:
For MNIST dataset in initial epoches the accuracy is(learning rate = 0.01):



```
0000/60000 [==============================] - 398s 7ms/step - loss: 0.2969 - acc: 0.8972 - val_loss: 0.0397 - val_acc: 0.9866
Epoch 2/50
60000/60000 [==============================] - 391s 7ms/step - loss: 0.0464 - acc: 0.9863 - val_loss: 0.0301 - val_acc: 0.9908
Epoch 3/50
60000/60000 [==============================] - 391s 7ms/step - loss: 0.0314 - acc: 0.9909 - val_loss: 0.0198 - val_acc: 0.9933
Epoch 4/50
60000/60000 [==============================] - 390s 7ms/step - loss: 0.0215 - acc: 0.9938 - val_loss: 0.0543 - val_acc: 0.9837
Epoch 5/50
60000/60000 [==============================] - 390s 6ms/step - loss: 0.0183 - acc: 0.9943 - val_loss: 0.0212 - val_acc: 0.9935
Epoch 6/50
60000/60000 [==============================] - 390s 6ms/step - loss: 0.0154 - acc: 0.9956 - val_loss: 0.0252 - val_acc: 0.9934
Epoch 7/50
60000/60000 [==============================] - 389s 6ms/step - loss: 0.0121 - acc: 0.9964 - val_loss: 0.0187 - val_acc: 0.9949
Epoch 8/50
60000/60000 [==============================] - 389s 6ms/step - loss: 0.0095 - acc: 0.9973 - val_loss: 0.0212 - val_acc: 0.9947
Epoch 9/50
60000/60000 [==============================] - 388s 6ms/step - loss: 0.0084 - acc: 0.9975 - val_loss: 0.0206 - val_acc: 0.9944
Epoch 10/50
```

As we can in first epoch the accuracy was 98.66 percent In 10 epoches the validity accuracy is 99.44%After 50 epoches Test loss is : 0.0249 and Test accuracy is : 99.52

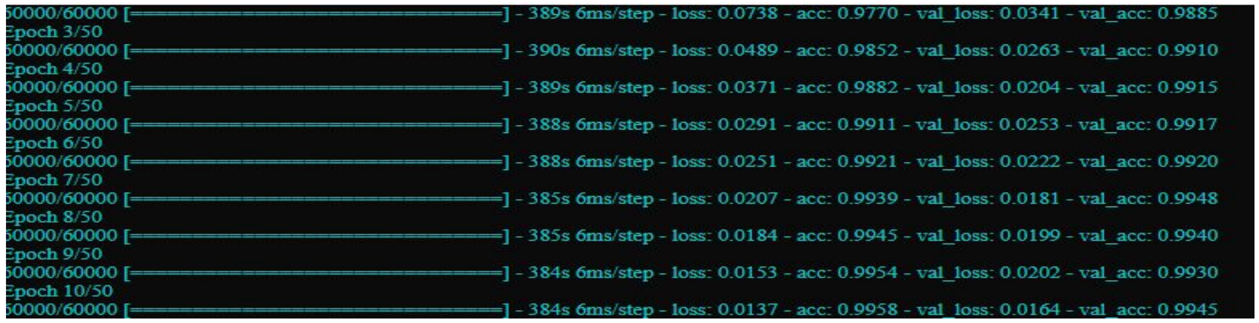
Google NET with augmentation on CIFAR10 with learning rate 0.001:
Accuracy and loss in initial epoches:



```
1563/1562 [==============================] - 318s 203ms/step - loss: 3.5508 - acc: 0.2396 - val_loss: 3.1729 - val_acc: 0.3511
Epoch 2/50
1563/1562 [==============================] - 290s 186ms/step - loss: 2.9391 - acc: 0.4165 - val_loss: 2.9782 - val_acc: 0.4214
Epoch 3/50
1563/1562 [==============================] - 292s 187ms/step - loss: 2.5688 - acc: 0.5184 - val_loss: 2.3565 - val_acc: 0.5774
Epoch 4/50
1563/1562 [==============================] - 292s 187ms/step - loss: 2.2738 - acc: 0.5875 - val_loss: 2.1267 - val_acc: 0.6242
Epoch 5/50
1563/1562 [==============================] - 304s 195ms/step - loss: 2.0290 - acc: 0.6438 - val_loss: 1.8701 - val_acc: 0.6836
Epoch 6/50
1563/1562 [==============================] - 290s 186ms/step - loss: 1.8332 - acc: 0.6841 - val_loss: 1.6967 - val_acc: 0.7171
```

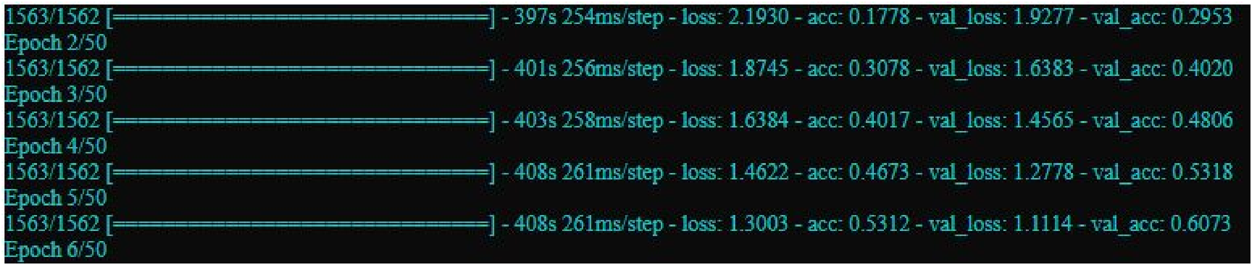
As we can see in the first epoch the test accuracy was 35.11 percent In.After 45 epoches the test accuracy :89.44 percent and loss :0.6531 and it remained constant till 50 epoches as its further learning was stopped.

VGG16 model:
For MNIST dataset (learning rate = 0.01):As from the below figure we can see its initial epoch the test accuracy: 98.85 percent and then it gradually increases and stops learning after 40th epoches with testing accuracy 99.88 percent and loss 1.6



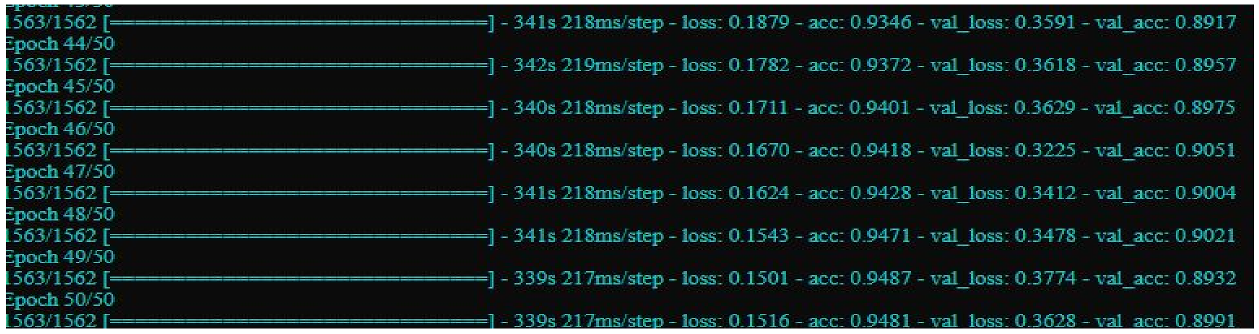
```
60000/60000 [==============================] - 389s 6ms/step - loss: 0.0738 - acc: 0.9770 - val_loss: 0.0341 - val_acc: 0.9885
Epoch 3/50
60000/60000 [==============================] - 390s 6ms/step - loss: 0.0489 - acc: 0.9852 - val_loss: 0.0263 - val_acc: 0.9910
Epoch 4/50
60000/60000 [==============================] - 389s 6ms/step - loss: 0.0371 - acc: 0.9882 - val_loss: 0.0204 - val_acc: 0.9915
Epoch 5/50
60000/60000 [==============================] - 388s 6ms/step - loss: 0.0291 - acc: 0.9911 - val_loss: 0.0253 - val_acc: 0.9917
Epoch 6/50
60000/60000 [==============================] - 388s 6ms/step - loss: 0.0251 - acc: 0.9921 - val_loss: 0.0222 - val_acc: 0.9920
Epoch 7/50
60000/60000 [==============================] - 385s 6ms/step - loss: 0.0207 - acc: 0.9939 - val_loss: 0.0181 - val_acc: 0.9948
Epoch 8/50
60000/60000 [==============================] - 385s 6ms/step - loss: 0.0184 - acc: 0.9945 - val_loss: 0.0199 - val_acc: 0.9940
Epoch 9/50
60000/60000 [==============================] - 384s 6ms/step - loss: 0.0153 - acc: 0.9954 - val_loss: 0.0202 - val_acc: 0.9930
Epoch 10/50
60000/60000 [==============================] - 384s 6ms/step - loss: 0.0137 - acc: 0.9958 - val_loss: 0.0164 - val_acc: 0.9945
```

VGG with augmentation on CIFAR10 with learning rate 0.001:
Accuracy and loss in initial epoches:



```
1563/1562 [==============================] - 397s 254ms/step - loss: 2.1930 - acc: 0.1778 - val_loss: 1.9277 - val_acc: 0.2953
Epoch 2/50
1563/1562 [==============================] - 401s 256ms/step - loss: 1.8745 - acc: 0.3078 - val_loss: 1.6383 - val_acc: 0.4020
Epoch 3/50
1563/1562 [==============================] - 403s 258ms/step - loss: 1.6384 - acc: 0.4017 - val_loss: 1.4565 - val_acc: 0.4806
Epoch 4/50
1563/1562 [==============================] - 408s 261ms/step - loss: 1.4622 - acc: 0.4673 - val_loss: 1.2778 - val_acc: 0.5318
Epoch 5/50
1563/1562 [==============================] - 408s 261ms/step - loss: 1.3003 - acc: 0.5312 - val_loss: 1.1114 - val_acc: 0.6073
Epoch 6/50
```

In last 5 epochs the results are as follow:



```
1563/1562 [==============================] - 341s 218ms/step - loss: 0.1879 - acc: 0.9346 - val_loss: 0.3591 - val_acc: 0.8917
Epoch 44/50
1563/1562 [==============================] - 342s 219ms/step - loss: 0.1782 - acc: 0.9372 - val_loss: 0.3618 - val_acc: 0.8957
Epoch 45/50
1563/1562 [==============================] - 340s 218ms/step - loss: 0.1711 - acc: 0.9401 - val_loss: 0.3629 - val_acc: 0.8975
Epoch 46/50
1563/1562 [==============================] - 340s 218ms/step - loss: 0.1670 - acc: 0.9418 - val_loss: 0.3225 - val_acc: 0.9051
Epoch 47/50
1563/1562 [==============================] - 341s 218ms/step - loss: 0.1624 - acc: 0.9428 - val_loss: 0.3412 - val_acc: 0.9004
Epoch 48/50
1563/1562 [==============================] - 341s 218ms/step - loss: 0.1543 - acc: 0.9471 - val_loss: 0.3478 - val_acc: 0.9021
Epoch 49/50
1563/1562 [==============================] - 339s 217ms/step - loss: 0.1501 - acc: 0.9487 - val_loss: 0.3774 - val_acc: 0.8932
Epoch 50/50
1563/1562 [==============================] - 339s 217ms/step - loss: 0.1516 - acc: 0.9481 - val_loss: 0.3628 - val_acc: 0.8991
```

As we can see In first epoch the test accuracy is 29.53 percent and then it gradually increases and stops learning after 40th epoches with testing accuracy 89.91
Appendix I gives a table of the results with variation of preprocessing done during training.It shows how the different methods of preprocessing affects the output of the model.I used VGG with subtracting the mean RGB value, VGG with just data augmentation, VGG with both subtracting the mean RGB value and data augmentation and VGG without any of this preprocessing. (this results are for 50 epoches)

5. Conclusion:

In this paper we evaluated very deep convolutional networks namely VGG16 and GoogleNet (of 16 and 22 weight layers respectively) for large scale image classification. It was demonstrated that the representation depth is beneficial for classification accuracy, and that state-of-the-art performance on the dataset can be achieved using a conventional ConvNet architecture.It is observed that the architecture of VGG16 is much simplest then google net but the computational time of Google net is way better than VGG16.From results it can be stated that applying augmentation to model improves the generalisation of model and so the accuracy is improved.

References:

1. Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. CoRR, abs/1404.5997, 2014. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.
2. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.
3. 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.
4. LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324.
5. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.
6. Perronnin, F., S´anchez, J., and Mensink, T. Improving the Fisher kernel for large-scale image classification. In Proc. ECCV, 2010
7. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proc. ICLR, 2014.
8. T. Serre, L. Wolf, S. M. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. IEEE Trans. Pattern Anal. Mach. Intell., 29(3):411–426, 2007

Appendix 1 :

Preprocessing methods	VGG		Googlenet	
	Accuracy	Loss	Accuracy	Loss
With Augmentation	89.91	0.3628	89.44	0.6531
With Subtracting the mean RGB value	79.44	1.2376	80.67	1.20
With both	90.40	0.3148	90.41	0.3012
Without both	78.96	0.734	79.71	0.56

Note:Both denotes Augmentation and Subtracting the mean RGB value