ECS795P - Deep Learning & Computer Vision: Deeper Networks for Image Classification

Jake Barrett
190722595
Queen Mary University of London
School of Electronic Engineering and Computer Science

***Abstract*—Deep Learning has been at the forefront of image classification tasks, dominating the field. VGG and GoogLeNet are two models which come to mind when approaching these problems of Computer Vision. In this report, both models are critically discussed. Next, the models are tested on the CIFAR-10 and MNIST datasets, first the original architectures are replicated (as far as possible) bearing in mind the fact that VGG and GoogLeNet were both originally designed for classification of the ILSVRC dataset which contains many more samples and larger images. It is shown that MNIST keeps stability whereas CIFAR-10 often suffers from the vanishing gradient problem. After, modifications are made to both models, where we see the benefits of batch normalization with CIFAR-10 on instability. Other modifications are implemented such as smaller batch and stride sizes which prove to be important in the improvement of GoogLeNet on the mentioned datasets.**

Keywords—Deep Learning, Computer Vision, Image Classification, VGG, GoogLeNet, MNIST, CIFAR-10

I. Introduction

Convolutional Neural Networks (CNN) and Image Classification are two terminologies which go hand in hand. Deep Learning has been pushing the boundaries and coming up with new ideas to solve problems in this area and improve on the existing state-of-the-art solutions. Understanding when “deeper is better” [4] is a typical question people ask when looking at deep networks. Two deep network designs which have been trying to answer this question are the VGG [11] and the GoogLeNet [12] models. They both achieve outstanding results with Simonyan *et al.* earning second place in ILSVRC 2014 in classification for their VGG model and Szegedy *et al.* winning with GoogLeNet, but this does not belittle VGG’s achievement as they significantly improve upon ZFNet [14] which won the competition the year before.

An in-depth discussion of both VGG and GoogLeNet will be presented in this report as well as an analysis of the strengths and shortcomings of each in relation to competing models and an introduction of the methodology used in both models. Next, their performance will be tested on two popular datasets, MNIST and CIFAR-10. The models will be recreated so that they can have these datasets inputted in them and so that they can be classified. Following this, certain modifications will be put in place in order to seek improvement in the model’s performances. This is done applying techniques, such as batch normalization as discussed in [5], kernel regularization and adjusting the learning rate amongst others, along with changes which were not necessarily implemented in the original architectures, such as reducing stride and batch sizes.

II. Critical Analysis / Related Work

A. *VGG*

For the VGG model configurations A to E which are experimented by Simonyan *et al.* [11], the number of parameters used range from 133 million to 144 million. This enabled them to secure first and second place in the localization and classification tracks respectively for the ImageNet Challenge 2014 and they claim that the depth was extremely beneficial to the classification accuracy. However, the major drawback of having such a high number of parameters in a network as Khan *et al.* identify [6] is that it will naturally increase the computational time due to the complexity of the model increasing and it would require much more powerful GPUs. Fortunately for Simonyan *et al.*, they had available four NVIDIA Titan Black GPUs to reap the benefits of data parallelism and even this took them up to 3 weeks to train a single network.

On the other hand, the following year He *et al.* won first place in the ILSVRC 2015 classification task [3] by utilizing a 152-layer deep network known as ResNet, this being 8 times deeper than the VGG network. Unsurprisingly, with such a deep architecture, training becomes a lot more challenging and models suffer from degradation [9], so He *et al.* combat this by introducing the residual building block. This may suggest that further improvement to the VGG network could have been achieved by building an even deeper model with more clever architectural changes such as in the ResNet model.

Another reason for the success of the VGG model was the utilization of smaller receptive fields of size 3×3 used throughout all the VGG configurations, with there even being three 1×1 receptive fields applied to configuration C. This is significantly smaller than the 11×11 receptive field which the rival CNN model of Krizhevsky *et al.* [7] has implemented. The use of smaller receptive fields results in the reduction of computational complexity in the VGG model [6], perhaps counteracting the complexity gained from the networks depth. This, nonetheless, begs the question of whether using a larger receptive field could lead to further improvement in performance, albeit at the sacrifice of computational simplicity.

B. *GoogLeNet*

Szegedy *et al.* in their GoogLeNet model say that their success didn’t come from simply building bigger and deeper networks but from the combination of architectures [12]. The aim was to obtain a model with state-of-the-art performance without compromising the computational efficiency. This is backed up by the fact that the construction of their model is based on the Hebbian principle and the incarnation of the Inception architecture which is heavily used in their network. Additionally, their model uses 12 times fewer parameters compared to the aforementioned model of Krizhevsky *et al.* [7]. They adopted an approach which was similar to that used by Lin *et al.*, where they replace a generalized linear model with a micro network [8], but Szegedy *et al.* replaced it with the Inception blocks.

Hanyue He in [2] argues that GoogLeNet is designed to deal more specifically with classification type tasks rather than any direct implementation on detection and segmentation problems. One example in lung cancer detection [1] is that Almas *et al.* found GoogLeNet having 22 layers may enhance accuracy, but has a weakness being that it increases the likelihood of over-fitting. However, it is recognized that this can be reduced by using techniques such as data augmentation and adjusting parameters such as the learning rate and training time.

The previously mentioned Inception blocks use small filters of size 1×1, 3×3 and 5×5 which tackle the computational bottleneck, particularly when inserting a 1×1 filter for dimensionality reduction before the larger sized kernels [12]. A consequence of this is a representational bottleneck due to the smaller filters reducing the feature space in the subsequent layers which could potentially cause loss of information [6].

Szegedy *et al.* used auxiliary classifiers to first counter the vanishing gradient issue by bettering the models training convergence [13]. Unfortunately, they did not improve convergence during the early training period meaning that the network would have to be trained for a longer time for these to finally have an effect on the accuracy. To avoid this issue, they introduced batch normalization as described by Ioffe *et al.* [5] into the auxiliary classifiers to act as a regularizer. The performance of the main classifier does increase with the introduction of batch normalization, but they admit that this is only weak evidence for it acting as a regularizer [13].

III. Method / Model Description

A. VGG

Architecture. [1] Simonyan *et al.* consider six ConvNet configurations from A to E. They all consist of a stack of convolutional layers with 3×3 kernels (other than in configuration C where 1×1 kernels are utilized in some layers) which have stride 1. They all use the ReLU rectification non-linearity activation function. Max pooling layers follow some of the convolution layers with 2×2-pixel windows with stride 2. This stack is followed by 3 Fully Connected (FC) layers with the first two composed of 4096 channels each both using a ReLU activation. The last FC layer has 1000 channels so there is one channel per class for the ILSVRC dataset and is equipped with a soft-max activation function.

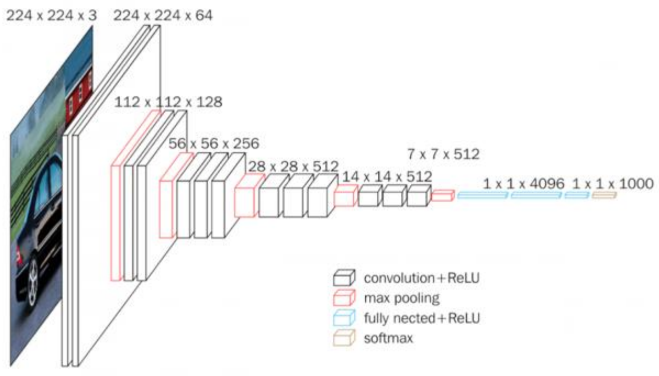
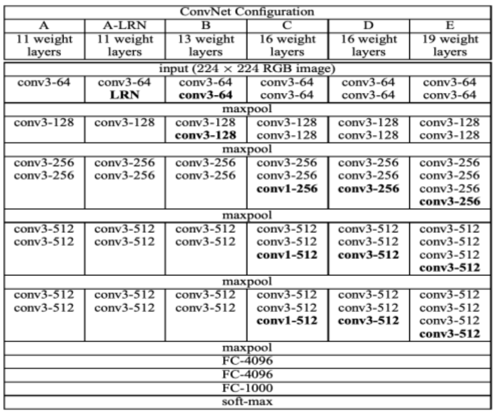


Fig. 1. VGG16 model architecture proprosed by Simonyan *et al.* for the ILSVRC dataset.

TABLE I. VGG CONFIGURATIONS AS ILLUSTRATED IN [11].

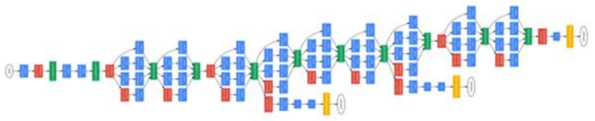


ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

These architectures are recreated in this report (other than A-LRN as Simonyan *et al.* mention that Local Response Normalization doesn't improve the performance on the ILSVRC dataset) with the mentioned datasets considered. This means that the last FC layer will have 10 channels so there is one channel per class for the CIFAR-10 and MNIST datasets. Adaptations to and additions to this architecture are also explored in Section IV.

B. GoogLeNet

Architecture. [2] Szegedy *et al.* describe a 22-layer deep neural network. Though it is deep, GoogLeNet introduces blocks of Inception modules to reduce the number of parameters and so one can see a drastic decrease to that of the VGG networks as seen in Table II. The Inception module achieves this by utilizing a 1x1 convolution layer which reduces the computational complexity. Max pooling layers are added to the starting convolution layers as well as into the fourth tower of the Inception blocks, each with 3×3-pixel windows with stride 2. Rather than using FC final layers, GoogLeNet uses average pooling before the classifier as in [4] and a dropout layer of ratio 0.4 is followed after. Finally, there are two dense layers, the first and second with linear and SoftMax activation functions respectively. As with VGG, this structure will be recreated in this report but with the use of CIFAR-10 and MNIST meaning that the last two dense layers will be 10 units being equal to the number of classes of each of the datasets. However, the auxiliary classifier is not incorporated into the recreated GoogLeNet in this paper since Szegedy *et al.* comment on how it did not improve convergence in early training and since the datasets used are far smaller than that of ILSVRC and the training time is shorter, it seems unnecessary to use. Adaptations to and additions to this architecture are also explored in Section IV.



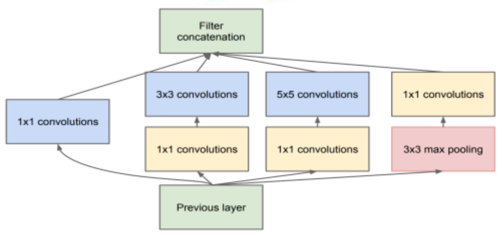


Fig. 2. (Top) The 22-layer deep GoogLeNet architecture [12].

Fig. 3. (Bottom) The implemented Inception block [12].

TABLE II. NUMBER OF PARAMETERS FOR THE SPECIFIED DATASETS FOR THE VGG CONFIGURATIONS AND GOOGLENET.

	Network	VGG-A	VGG-B	VGG-C	VGG-D	VGG-E	GoogLeNet
Number of parameters (in millions)	**CIFAR-10**	28	28	29	34	39	6
	MNIST	28	28	29	34	39	6
	ILSVRC	133	133	134	138	144	57

Implementation. The VGG and GoogLeNet models were built using the Keras API with a TensorFlow backend. Google's cloud servers were utilized for the training of the models to take advantage of free NVIDIA Tesla GPUs they offer. Although CPUs can carry out the task of image classification, GPUs were used to accelerate the training of the models and so that more progress on the investigation can be made. Images tend to have high dimensionality and for a CPU to train a model on the CIFAR-10 and MNIST datasets, it could take several hours or even days to train a single model. When using GPUs, there is a significant increase in training speed and therefore the training time is decreased by several orders of magnitude, with training taking no more than an hour on average.

IV. EXPERIMENTS

Datasets. In the CIFAR-10 dataset there are 60,000 images which are divided into 50,000 training and 10,000 test samples. There are 10 classes with 6,000 images per class and there is no overlap between these classes. Each image is of fixed size 32×32 and has 3 color channels, RGB. The dataset has been downloaded from the University of Toronto, https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz.

Fig. 4. Five labelled sample images from the CIFAR-10 dataset.

The MNIST dataset contains 70,000 images and is divided into 60,000 training and 10,000 test samples. Again, there are 10 classes but with 7,000 images per class and there is no overlap between these classes. Each image is of fixed size 28×28 with a single grayscale channel. The dataset has been downloaded from the collection of TensorFlow datasets, https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz.

Fig. 5. Five labelled sample images from the MNIST dataset.

For the images in both datasets, the pixel values were normalized to between 0 and 1. However, the MNIST dataset needed to be reshaped to incorporate the single channel so it can be fed through the networks.

A. *Initial Experimentation*

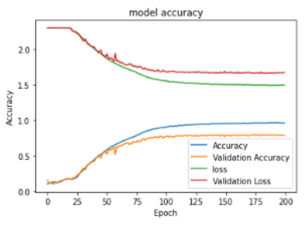
To commence proceedings, the recreated configurations of the VGG models and GoogLeNet are trained over 200 epochs, using a batch size of 256 on the MNIST and CIFAR-10 datasets. The optimizer used is the stochastic gradient descent optimizer (SGD) with the learning rate (LR) as the hyperparameter to be tuned, experimenting with LRs 0.005 and 0.01. The LR decay and momentum are fixed parameters taking values of 10^{-6} and 0.9 respectively. LR decay is the decrease of the LR over each epoch of training and the momentum helps accelerate the convergence of the loss function in place being the sparse categorical cross entropy. The results are shown in Table III.

TABLE III. TRAIN AND TEST ACCURACIES FOR VGG AND GOOGLENET ON MNIST AND CIFAR-10 WITH LR AS THE HYPERPARAMETER.

	CIFAR-10				**MNIST**			
Epochs 200	**LR 0.005**		**LR 0.01**		**LR 0.005**		**LR 0.01**	
Network	**Train***	**Test***	**Train***	**Test***	**Train***	**Test***	**Train***	**Test***
VGG-A	0.9548	0.7328	0.9677	0.7663	0.9986	0.9914	0.9988	0.9919
VGG-B	0.9376	0.6342	0.9559$_{+}$	0.6964$_{+}$	0.9987	0.9922	0.9990	0.9921
VGG-C	0.9472	0.7653	0.9712	0.7796	0.9991	0.9926	0.9991	**0.9940**
VGG-D	0.9465	**0.7757**	0.9618	**0.7844**	0.9986	0.9915	0.9990	0.9930
VGG-E	0.8252	0.7332	0.5714$_{++}$	0.4978$_{++}$	0.9977	0.**9937**	0.9989	0.9920
GoogLeNet	0.8368	0.6774	0.1000$_{+++}$	0.1000$_{+++}$	0.9984	0.9925	0.9990	0.9936

* The result with best validation accuracy out of 3 runs was considered.
$_{+}$ The no. of + symbols represent the no. of times the model suffered from vanishing gradient.

CIFAR-10. VGG network D has the highest test accuracy for both LR 0.005 and 0.01 over all considered networks, achieving an accuracy of 0.7757 and 0.7844 respectively. The accuracy drastically decreases at VGG-E with LR 0.01, perhaps meaning that the model is too complex for the dataset and becomes unstable, as indicated by the decrease in training accuracy. GoogLeNet with 0.6774 accuracy only performs better than the VGG-B network at LR 0.005, maybe suggesting that the VGG architecture compared to GoogLeNet is better suited to the CIFAR-10 dataset. Looking at Fig. 7, the training does appear to be unstable from epochs 0 to 75.



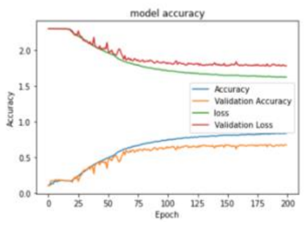
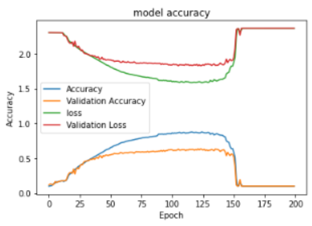


Fig. 6. (Left) VGG-D accuracy and loss plot achieving 0.7844 validation accuracy.

Fig. 7. (Right) GoogLeNet accuracy and loss plot achieving 0.6774 validation accuracy.

Configurations of VGG-B and VGG-E both suffer from vanishing gradient with LR 0.01 during three training runs, once for VGG-B and twice for VGG-E. Vanishing gradient also occurs for all 3 GoogLeNet runs with LR 0.01. This suggests that the LR is too high causing the training to become unstable since with LR 0.005, this problem does not arise and the training is relatively stable.



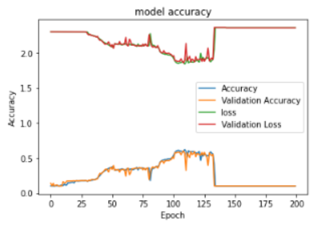


Fig. 8. (Left) VGG-B suffering from vanishing gradient.

Fig. 9. (Right) VGG-E suffering from vanishing gradient.

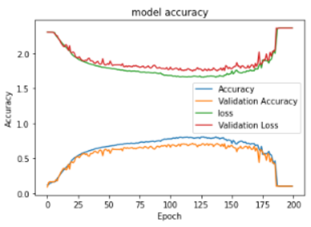
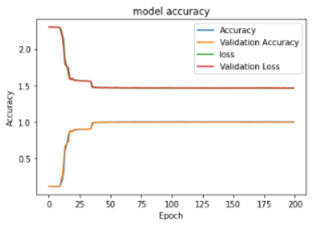


Fig. 10. GoogLeNet suffering from vanishing gradient.

MNIST. Though there is little difference in the performance of all models, VGG-C obtains the best test accuracy of 0.9940 at LR 0.01 and VGG-E achieves the best test accuracy of 0.9937 at LR 0.005, as seen in Table III. In general, it can be seen that the increase in LR improves the test accuracy. There is no concrete explanation to why VGG-C performs best on MNIST, one might say VGG-C utilizes its 1×1 kernels and has an ideal depth so that it is not over/under-fitting. The GoogLeNet model certainly competes with the VGG-C network with a test accuracy of 0.9936 at LR 0.01.



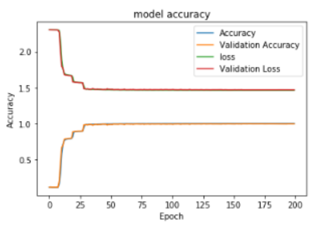


Fig. 11. (Left) VGG-C accuracy and loss plot achieving 0.9940 validation accuracy on MNIST.

Fig. 12. (Right) GoogLeNet accuracy and loss plot achieving 0.9936 validation accuracy on MNIST.

Fig. 11 and 12 illustrate the stability of VGG-C and GoogLeNet on the MNIST dataset. After approximately epoch 30, the models both reach convergence with very little change in accuracy. Furthermore, there is no case where the model on the MNIST dataset suffers from the vanishing gradient problem.

Comparison. Stability is a key difference of the training on the MNIST and CIFAR-10 datasets with MNIST exceeding in this domain. Moreover, the models hit convergence much earlier for MNIST along with a major increase in the performance. This is more than likely due to the shape of the datasets being fed into the models. MNIST contains images of shape 28×28×1 representing a grayscale image whereas CIFAR-10 images have shape 32×32×3 representing an RGB color image and this makes the classification task of CIFAR-10 that much more complex. It is also a bonus that MNIST has 10,000 more images for training to further improve the model performance.

B. *Model Amendments and Additions*

Since VGG-D was the best performing model of all the VGG configurations on CIFAR-10, it will be the only model taken for further investigation alongside GoogLeNet. It will also be the only VGG configuration used on MNIST as the performance and stability of VGG models deviate only by a small magnitude and because it achieved the second highest test accuracy of the VGG models.

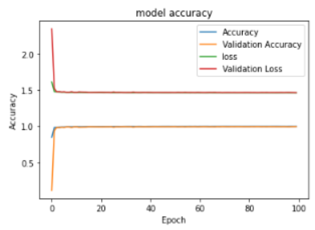
To combat the vanishing gradient issue and instability of the VGG and GoogLeNet models on the CIFAR-10 dataset, still training over 200 epochs, batch normalization [5] layers will be added to the architectures in the hope of regularizing the training process. As stated by Ioffe *et al.*, the benefit of batch normalization is that higher LRs can be used without the need to be overly concerned about potential instability. For the VGG architecture, batch normalization layers will be inserted after each convolution layer. For GoogLeNet, these layers will be inserted before each max pooling layer not contained in the inception blocks. Since MNIST proved more stable, larger LRs 0.01 and 0.05 will be used over 100 epochs but the LRs will be kept at 0.005 and 0.01 for training CIFAR-10.

The previously described architectural modification can be viewed as follows:

Modification 1.	VGG-D: Batch normalization layers added after every convolution layer.
	GoogLeNet: Batch normalization layers inserted before each max pooling layer not contained in the inception blocks.

TABLE IV. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 1** ON MNIST.

	MNIST			
Epochs 100	**LR 0.01**		**LR 0.05**	
Network	**Train***	**Test***	**Train***	**Test***
VGG-D	0.9997	0.9949	0.9992	**0.9965**
GoogLeNet	0.9989	0.9927	0.1124+++	0.1135+++



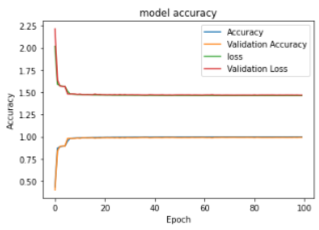


Fig. 13. (Left) VGG-D accuracy and loss plot achieving 0.9965 validation accuracy at LR 0.05 on MNIST.

Fig. 14. (Right) GoogLeNet accuracy and loss plot achieving 0.9927 validation accuracy at LR 0.01 on MNIST.

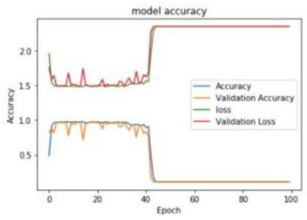
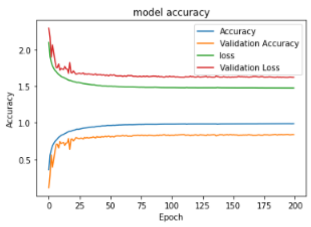


Fig. 15. GoogLeNet suffering from vanishing gradient at LR 0.05 on MNIST.

TABLE V. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 1** ON CIFAR-10.

	CIFAR-10			
Epochs 200	**LR 0.005**		**LR 0.01**	
Network	**Train***	**Test***	**Train***	**Test***
VGG-D	0.9871	0.8183	0.9870	**0.8384**
GoogLeNet	0.9439	0.7230	0.8899	0.6974



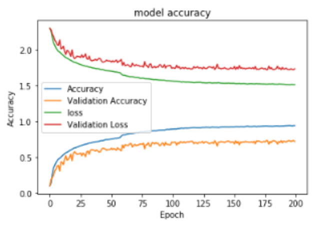


Fig. 16. (Left) VGG-D accuracy and loss plot achieving 0.8384 validation accuracy at LR 0.01 on CIFAR-10.

Fig. 17. (Right) GoogLeNet accuracy and loss plot achieving 0.7230 validation accuracy at LR 0.005 on CIFAR-10.

From Tables IV and V, it appears that batch normalization has a positive effect on the performance on the VGG-D network with the test accuracy increasing to 0.9965 and 0.8384 on MNIST and CIFAR respectively with a higher LR being favorable. As seen in Fig. 16, VGG-D stabilizes on MNIST as soon as epoch 2 or 3 which is to no surprise seeing how it performed prior to modification. Batch normalization clearly impacts the CIFAR-10 training since it stabilizes at around epoch 25. We do see however that GoogLeNet suffers from the vanishing gradient problem at LR 0.05 even with batch normalization layers incorporated and we do see a deterioration with LR 0.01 achieving an accuracy of 0.9927 from the GoogLeNet architecture without batch normalization.

C. *CIFAR-10: Further Evaluation & Improvements*

GoogLeNet. Since GoogLeNet was unable to improve with batch normalization when the batch normalizations were not in the inception blocks, the following modifications will be tested:

Modification 2.	GoogLeNet: Batch normalization layers inserted also in the inception blocks before the max pooling layers.
Modification 3.	GoogLeNet: Kernel regularization on all convolution layers to penalize the model weights for training stability with weight decay set to 0.0002.

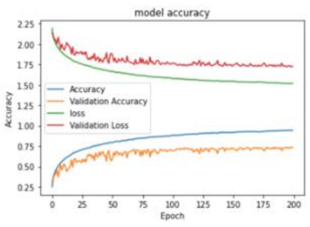
A LR of 0.005 will be used throughout as it proved to be more effective. Modification 2 is applied with different batch sizes of 128, 256 and 512 and Modification 3 will be applied to the batch size which performs best to seek further performance increase. Modification 3 is then dropped and replaced with the following:

Modification 4.	GoogLeNet: Batch normalization layers as in Modification 1 with batch size 128 and stride size in first convolution layer reduced to 1.

Modification 4 was used whilst changing the kernel sizes, 5×5 and the original 7×7.

TABLE VI. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 2, 3,4** ON CIFAR-10.

LR 0.005	**CIFAR-10 (200 epochs)**	
GoogLeNet	**Train***	**Test***
Modification 2 Batch Size 128	0.9427	0.7315
Modification 2 Batch Size 256	0.9551	0.7089
Modification 2 Batch Size 512	0.9391	0.6654
Modification 3 Batch Size 128	0.7876	0.6800
Modification 4 5×5 Kernel	0.9867	**0.8473**
Modification 4 7×7 Kernel	0.9748	0.8095



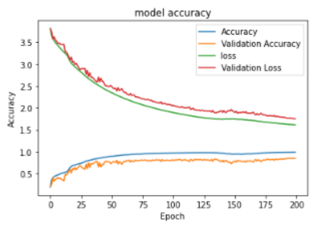


Fig. 18. (Left) GoogLeNet accuracy and loss plot achieving 0.7315 validation accuracy at LR 0.005 and batch size 128 on CIFAR-10.

Fig. 19. (Right) GoogLeNet accuracy and loss plot achieving 0.8473 validation accuracy at LR 0.005 and batch size 128 on CIFAR-10.

When Modification 2 is applied, its seen that the batch size hyperparameter performs best at size 128 with accuracy 0.7315. Shirish Keskar *et al.* state how larger batches, in practice, can lead to depreciation in model quality [10]. For this reason, a smaller batch size was used and tested with weight regularization as per Modification 3. This was not successful due to the reduced test accuracy of 0.6800 being obtained and perhaps having both batch normalization and kernel dampening over-regularizes training.

Then, introducing a smaller stride size of 1 significantly increases the accuracy. This suggests that using bigger strides may lead to loss of information. It may have been necessary to use a larger stride size on images on the ILSVRC dataset since the image size is much larger than that of CIFAR-10. The smaller kernel size 5×5 performed best yielding a score of 0.8473, favoring a finer grained feature extraction.

Lastly, since the training has been much more stable compared to the initial experiments, different LRs will be tested over 300 epochs whilst applying Modification 4 with a 5×5 kernel to seek further improvement.

TABLE VII. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 4** ON CIFAR-10.

	CIFAR-10 (300 epochs)					
	LR 0.001		**LR 0.005**		**LR 0.01**	
GoogLeNet	**Train***	**Test***	**Train***	**Test***	**Train***	**Test***
Modification 4 5×5 Kernel	0.9628	0.7629	0.9877	**0.8560**	0.9312	0.7933

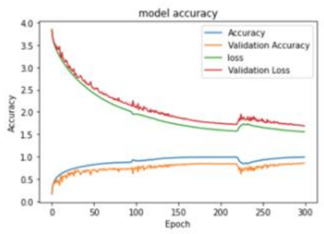


Fig. 20. GoogLeNet accuracy and loss plot achieving 0.8560 validation accuracy at LR 0.005 and batch size 128 over 300 epochs.

The unchanged LR 0.005 performs best over 300 epochs with an accuracy of 0.8560, increasing from that of 200 epochs. In Fig. 20, at around epoch 225 it's seen that the accuracy dips slightly but recovers and looks as if with more epochs it could potentially perform even better. For the LRs, 0.01 is too high making it difficult to converge whereas at LR 0.001, the LR appears to be too low causing the progress of training to not be too slow.

VGG-D. Before, VGG attained an accuracy of 0.8384 which is relatively close to the improved score of GoogLeNet being 0.8560. In the attempt to boost the performance of VGG-D further, the following modification will be implemented:

Modification 5.	VGG-D: Kernel regularization on all convolution layers to penalize the model weights for training stability with weight decay set to 0.0002.

TABLE VIII. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 5** ON CIFAR-10.

LR 0.01	**CIFAR-10 (200 epochs)**	
VGG-D	**Train***	**Test***
Modification 5	0.8112	**0.6929**

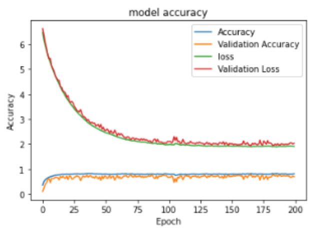


Fig. 21. VGG-D accuracy and loss plot achieving 0.6929 validation accuracy at LR 0.01 and batch size 256 over 200 epochs.

As with GoogLeNet, weight dampening did not aid the performance of the model with it reducing to 0.6929 and again, this could be due to batch normalization and kernel dampening over-regularizing the training process.

The best VGG-D accuracy was obtained whilst Modification 1 was in place. Therefore, Modification 1 will be reimplemented with higher LRs of 0.02 and 0.03 in order to see whether the previous best score can be beaten and if the training remains stable.

TABLE IX. TRAIN AND TEST ACCURACIES FOR ARCHITECTURAL CHANGES SPECIFIED IN **MODIFICATION 1** ON CIFAR-10.

	CIFAR-10 (200 epochs)			
	LR 0.02		**LR 0.03**	
VGG-D	**Train***	**Test***	**Train***	**Test***
Modification 1	0.9853	**0.8505**	0.9801	0.8484

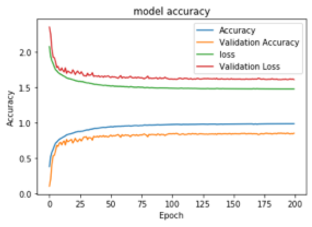


Fig. 22. VGG-D accuracy and loss plot achieving 0.8505 validation accuracy at LR 0.02 and batch size 256 over 200 epochs.

The results in Table IX show that when we increase the LR to 0.02, the test accuracy increases to 0.8505 but increasing the LR to 0.03, though it is an improvement from LR 0.01, the accuracy decreases to 0.8484. Looking at Fig. 22, even with the relatively high LR of 0.02, the training on CIFAR-10 remains stable and maybe, if more epochs were used, further improvements could be witnessed, even if they were small.

V. CONCLUSION

Many modifications of the original architectures of GoogLeNet and VGG have been discussed and experimented with to determine how to best approach the classification tasks on both the CIFAR-10 and MNIST datasets.

It appears that training on MNIST remains stable under most of the modifications made for both VGG-D and GoogLeNet. However, VGG-D favors the introduction of batch normalization, obtaining its top accuracy of 0.9965 whereas GoogLeNet doesn't gain any improvement from it, achieving its highest accuracy of 0.9936. The GoogLeNet architecture may have been too complex for the task of classifying MNIST compared to VGG-D, perhaps with too many unnecessary layers, enabling VGG-D to perform better.

Stability was kept, likely due to the shape of MNIST being a 28×28 single channel image and making it a simpler classification task whilst achieving higher test accuracy.

On the other hand, stability was a major issue for training on CIFAR-10 with the initial architectures of VGG and GoogLeNet, with the vanishing gradient issue often occurring. It was found that batch normalization was a significant aid to combatting this problem, with it acting as a regularizer [5]. In the GoogLeNet model, performance was enhanced through reducing the stride size and the filter size to 1 and 5×5 respectively into the first convolution layer, probably due to the fact that information was previously lost using larger filters and stride sizes, which may have been necessary for Szegedy *et al*. when training on the much larger ILSVRC dataset. However, a limitation of this reduction was more intensive computation which would require more powerful GPUs for shorter training times. Also, reducing the batch size fed into the GoogLeNet model for training helped improve performance, but again, training time would have increased. Therefore, if time permitted, more epochs would have been used testing with smaller filter sizes and smaller batches to search for better performance in GoogLeNet. Also, Szegedy *et al.* used an auxiliary classifier in their model which may have helped with the convergence of the GoogLeNet model in this paper.

VGG-D which was taken forward out of all the VGG models for testing performed slightly worse than GoogLeNet on CIFAR-10. Since CIFAR-10 consists of images of size 32×32 with 3 color channels which is more complex than MNIST, the deeper architecture of GoogLeNet is favored, achieving test accuracy of 0.8560. If more time could be spent on this task or presented with powerful enough computation resources, VGG-D could have been trained over many more epochs to see if improvement could have been made.

The ILSVRC dataset which Szegedy *et al.* and Simonyan *et al.* had available to them is comprised of over 1 million images which helps vastly with training, thus improving test accuracy. Perhaps the use of data augmentation to generate image data mimicking the much smaller MNIST and CIFAR-10 datasets could have been cleverly incorporated into the training processes of the GoogLeNet and VGG-D models discussed to help boost their performances.

REFERENCES

[1] B. Almas, K. Sathesh, and S. Rajasekaran, “A Deep Analysis of Google Net and Alex Net for Lung Cancer Detection,” *International Journal of Engineering and Advanced Technology*, vol. 9, no. 2, pp. 395–399, Dec. 2019.

[2] H. He, “A Deep Research in Classic Classification Network,” *IOP Conference Series: Materials Science and Engineering*, vol. 740, p. 012152, Mar. 2020.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015.

[4] E. Malach and S. Shalev-Shwartz, “Is Deeper Better only when Shallow is Good?,” Mar. 2019.

[5] S. Ioffe, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015.

[6] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial Intelligence Review*, Apr. 2020.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[8] M. Lin, Q. Chen, and S. Yan, “Network In Network,” 2014.

[9] W. Rawat and Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.

[10] N. Shirish Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. Tak, and P. Tang, “ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA,” Feb. 2017.

[11] K. Simonyan and A. Zisserman, “Published as a conference paper at ICLR 2015 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” Apr. 2015.

[12] C. Szegedy et al., “Going deeper with convolutions,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *arXiv.org*, 2015. [Online]. Available: https://arxiv.org/abs/1512.00567.

[14] M. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” Nov. 2013.