Deeper Networks for Image Classification

Temitope Damilola Joloko
220432052
t.d.joloko@se22.qmul.ac.uk

***Abstract*— Deep Convolutional Neural Networks have revolutionized the performance of image classification and other computer vision tasks. Though they have limitations and challenges, they have gained widespread recognition due to their ability to learn feature representations automatically coinciding with the availability of large image datasets and more powerful computing infrastructures. This paper analyses and evaluates the performance of two notable Deep Convolutional Neural Networks for image classification – GoogLeNet and VGG. The evaluation was done using the MNIST and CIFAR-10 datasets. Both models achieved approximately 99% accuracy on MNIST, while VGG achieved 91.1% percent accuracy on CIFAR-10 dataset, 0.2% more than the accuracy of GoogLeNet on the same dataset.**

I. Introduction

Image Classification is a well-researched area of Computer Vision that involves assigning a label to an image from a list of known classes. Because of the dimensionality and complex structure contained in images, rather than passing in an image pixel-wise directly to a classification algorithm, a computer vision pipeline usually includes a feature representation extraction stage where the features that encapsulate the image structure are extracted and then passed to the classification algorithm. Good feature representations are important for the success of classification tasks as they allow for robust recognition in images. Traditional feature representation methods like Histogram of Oriented Gradients (HOG) [1] and Scale Invariant Feature Transform (SIFT) [2] descriptors are hand-crafted and need to be manually adapted to the task at hand [3].

The development of Convolutional Neural Networks (CNNs) [4] has resulted in remarkable advancements in computer vision. CNNs automate the process of feature extraction and execute an end-to-end learning process for image classification and other computer vision tasks like object detection, semantic segmentation etc. Though it has been said that, at the root of other unique computer vision tasks, is an image classification task [5]. The usage of CNNs gained popularity for image classification after the CNN architecture AlexNet [6] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It has been shown that an increase in network depth results in an increase in performance. [7, 9]. The deeper the network, the more complex relationship it can capture resulting in improved performance. However, as the network depth increases, the performance might also become saturated where there is no more improvement in performance. Deeper networks also come with other set of challenges, such as overfitting, vanishing gradients [7] and computational complexity due to increase in parameters [9]. Various solutions have been proposed to handle these challenges which will be explored in the next section. CNNs also have some limitations where they are only suited to specific tasks and require labelled data for training [10]. To tackle the former and enhance reusability, research is being done in transfer learning to adapt trained networks outside of the tasks they were created for.

This report analyses and evaluates the performance of two notable deep CNNs for image classification — GoogLeNet [8] and VGG [9]. GoogLeNet [8] won first place in the ILSVRC in 2014 while VGG won second place in the same year. The performance of these networks were evaluated on the MNIST and CIFAR-10 datasets which contain only 10 classes compared to the 1000 ImageNet classes used in the competition.

The subsequent sections of this report will be as follows: Section II explores related works, providing insight into the key ideas and problems solved by some notable CNNs for image classification. Section III dives deeper into the architecture, analysis, and training configuration of the two models being evaluated. Section IV presents the results of the experiments performed, and finally in Section V, a concluding summary.

II. Related Work

CNNs in comparison with regular feed-forward networks of similar size have fewer parameters and connections [6], hence why their performance can be improved by increasing the depth. General CNN architecture includes convolution layers, pooling layers and fully connected layers [8]. The AlexNet [6] architecture had 8 layers, with 5 convolutional layers and 3 fully connected layers. To prevent overfitting, a dropout mechanism was used in the fully connected layers where the output of some neurons are not propagated to the next layer. This is done by setting the output of the neuron to zero with some fixed probability [6]. This is a form of regularization that allows more robust features to be learnt without over adapting to any representation. This method has been utilized by subsequent architectures that have been developed [Dropout here]. Another trend that emerged was the use of smaller convolutional filter sizes. While the first convolutional layer of AlexNet had a filter of size 11x11, newer networks tend to use filter with size 1x1, 3x3, and 5x5 across the whole network. This allows the networks to learn finer grained representation of image features, which contributes to the performance gains.

With the goal of evaluating depth increase for CNNs, VGG [9] was developed. This network pushed the depth to 16-19 layers with higher performance than AlexNet, but with increased computational cost, having about 144 million learnable parameters. This network uses 1x1 and 3x3 filter sizes to increase discrimination within the network by adding more non-linearity rather than one large filter size. Two or three convolutional layers of the same shape are stacked before a max pooling layer. Some other types of network architecture are based on blocks where some or all the layers of the network are micro-networks [8, 11]. GoogLeNet [8] is one of such methods, where the micro-networks are called “Inception modules”. These modules contain multiple convolution branches and a max-pooling branch such that the output of these operations inside each module are concatenated and then used as input to the next layer. Based on the Hebbian principle —closer neurons are stronger

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					

Fig.1. VGG Network Architecture, with the implemented variant highlighted.

together when activated simultaneously, and the Network in Network (NIN) [11] approach of using 1x1 filters to achieve greater representation, [8] uses 1x1 filters to reduce dimensionality. The local sparse structure of the network is optimized using these inception modules as dense components. While being 22 layers deep, this approach has about 20 times less parameters than VGG, while also being more computationally efficient and faster due to the use of these modules [8].

In 2016, an entirely new approach to network architecture was introduced. Because deeper networks are prone to the degradation problem where the accuracy plateaus before beginning to decline, ResNet [7] uses the approach of residual learning to overcome this. A residual function is the mapping from a few stacked layers added to the input to the start of this stack of layers which becomes the output of the stacked layers. This residual mapping is fitted by using skip connections between layers. The depth limit was pushed to 152 layers and even 1202 layers in ResNet [7]. This residual mapping also solves the vanishing gradient problem due to the input addition being done by the residual mappings. This network has even greater accuracy and less computational complexity. Some other newer networks that have produced great results are [12] which combines residual mapping, inception modules and similar size stacked convolution layers together from VGG in a multi-branch architecture, using a summation unlike concatenation as done in GoogLeNet's inception modules. DenseNet [13] uses dense blocks where each output of a feature map is used as an input to all subsequent layers inside the dense block via concatenation. Efficient Net [14] uses a compound scaling method to balance the width, depth and resolution to create an optimal network architecture.

III. Model Description

To evaluate the effectiveness of deeper networks for image classification, the VGG-19 and GoogLeNet networks are used in this report. The model architectures and training configuration are detailed below.

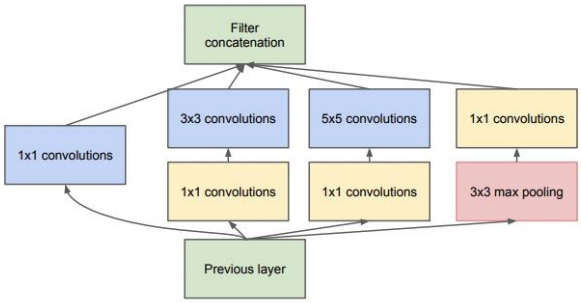

Fig. 2. Inception Module.

A. Model Architecture

1) *VGG-19*

The variation of VGG used in this evaluation is the VGG-19 with 19 layers consisting of 16 convolutional layers and 3 fully connected layers [9]. A max-pooling layer with kernel size 2x2 and stride of 2 exists between two to three stacks of convolutional layers. The convolutional layers all have 3x3 filters with a stride of 1 and a padding of 1 to maintain spatial resolution. The first 2 fully connected layers have an out channel of 4096, while the last fully connected layer that does the classification has 10 channels representing the number of possible class predictions. A softmax layer is added after the last fully connected layer. Both the convolutional and fully connected layers use ReLU [7] as the activation function for non-linearity.

While not in the original implementation, the version used in this evaluation uses batch normalization [17] in the convolution layers. This is a method that reduces internal covariance shift by transforming the input of each layer to have zero mean and unit variance [17]. This results in faster network convergence.

2) *GoogLeNet*

This model has 22 layers including 8 inception module layers, a few convolution layers at the start of the network, some pooling layers and a single fully connected layer. The structure of an inception module is shown in Fig. 2. Using inception modules allows for network depth increase with improved performance and less parameters making it more computationally efficient. The usage of 1x1 convolutions before the 3x3 and 5x5 filters in the inception blocks allows for dimensionality reduction hence less parameters and acts an added non-linearity activation. The structure of this network begins with the convolution layers, followed by the inception blocks, a global average pooling which was shown to capture more parameters than strided average pooling and a fully connected layer with 10 channels with a softmax activation. Max-pooling layers are interspersed between the convolution layers and inception blocks. The implementation for this report also uses batch normalization [17] and ReLU activation.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fig 3. GoogLeNet Architecture.

B. *Training Configuration*

1) ***VGG-19***

The input to this network is a 224x224 RGB image with zero mean. As in the original network, the batch size was set to 256, and momentum to 0.9. The regularization procedures are also the same, using a dropout ratio of 0.5 in the fully connected layers and weight decay of 0.005 as an L2 penalty. Experimenting with a different optimizer known to be efficient — Adam optimizer, while using the same hyperparameters resulted in slower convergence so the Stochastic Gradient Descent is retained as the optimizer for the cross-entropy loss. As recommended in the paper [7], weight initialization was done using the method proposed by [17], providing a good starting point for optimization. Because the original sizes of the images used for this evaluation are quite small compared to those used for training the network, multi-scale training approach was not utilized. The learning rate was scheduled starting with 0.01 with a decay of 0.1 after some epochs. The data was augmented using a random horizontal flip with a 0.5 probability.

2) ***GoogLeNet***

The input to this network is also a 224x224 RGB image with zero mean. Based on my experimentation, the most effective batch size was 64, with a scheduled learning rate as VGG above. Momentum was set to 0.9 and a dropout ratio of 0.4 as per the original training methodology [8]. To improve performance, I used L2 regularization with a weight decay of 0.005. Controlled weights initialization was utilized, while the cross-entropy loss was optimized using Stochastic Gradient Descent. Attempting the Adam optimizer for this network also did not provide good results. The same data augmentation technique mentioned above was also applied here.

IV. EXPERIMENTS

Different epochs were used based on the dataset being trained. Training using MNIST dataset achieved good results under 15 epochs, while training using CIFAR needed more epochs.

A. *Datasets*

1) *MNIST*

MNIST [15] is a dataset of handwritten digits with a training set of 60,000 examples and a test set of 10,000 examples. Each digit is size normalized and centered on a 28x28 image. The images are grayscale and have only one channel. The labels are values from 0 to 9. For this experiment, this dataset was preprocessed by expanding to 3 channels using values replicated from the first channel, normalizing to zero mean and then resizing to 224x224 before passing as input to both models.

2) *CIFAR-10*

The CIFAR-10 [16] dataset consists of 50,000 training images and 10,000 test images. The images are 32x32 colored images. There are 10 classes of objects in the datasets which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. This dataset was also preprocessed for this experiment by normalizing to zero mean and resizing to 224x224 before using input to both models.

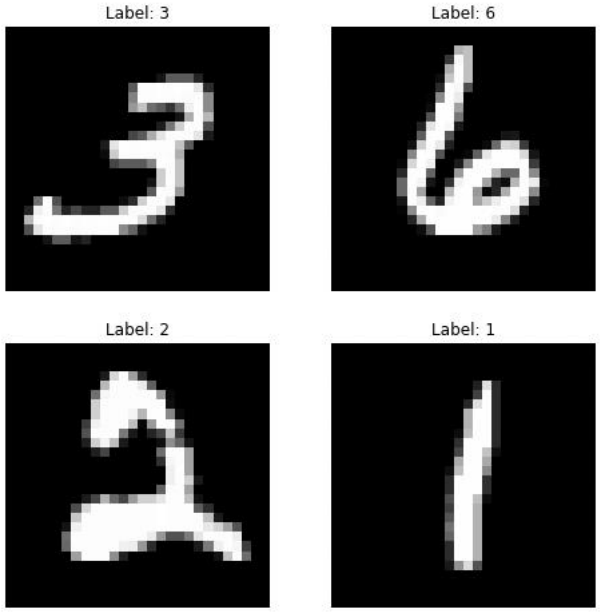

Fig. 4. Samples from the MNIST dataset

Fig. 5. Samples from the CIFAR-10 dataset

B. *Training Process*

During the training process, different adjustments were made to the hyperparameters to find the best performance. Setting weight initialization was one of the factors that improved network performance and allowed the networks to reach convergence faster. Fig. 6 – Fig 9 show the value of the loss function per epoch. The MNIST datasets were trained using 15 epochs as they reach convergence earlier, while the CIFAR-10 datasets were trained until 30 epochs.

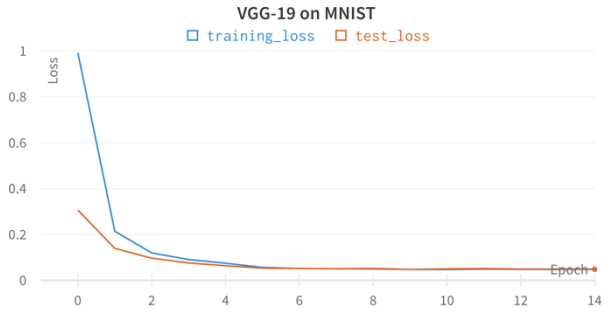

Fig. 6. Loss function values from training and test dataset while training the VGG-19 network using the MNIST dataset. The initial low loss on the test dataset may be attributed to the size of the test samples.

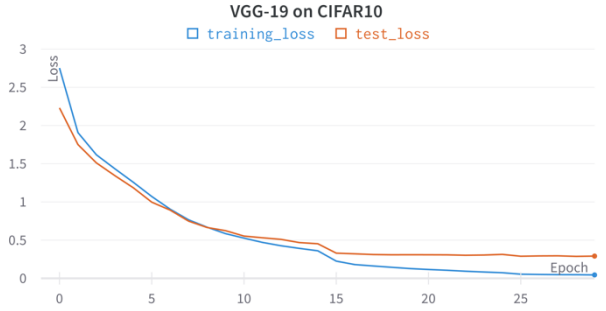

Fig. 7. Loss function values from training and test dataset while training the VGG-19 network using the CIFAR10 dataset.

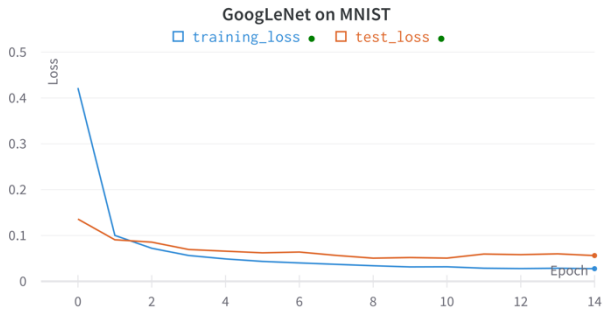

Fig. 8. Loss function values from training and test dataset while training the GoogLeNet network using the MNIST dataset.

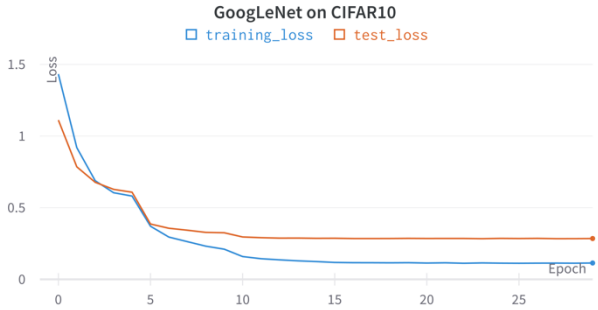

Fig. 9. Loss function values from training and test dataset while training the GoogLeNet network using the CIFAR10 dataset. The initial low loss on the test dataset may be attributed to the size of the test samples.

Also noteworthy is that VGG has a higher starting loss value on both datasets, while GoogLeNet has starts with about half that value, but eventually both networks converge. This might be due to the fact that GoogLeNet has a lot less parameters than VGG.

C. *Testing Results*

Table I. shows the results of training the VGG-19 and GoogLeNet models on the MNIST and CIFAR-10 datasets. The testing dataset contained 10,000 samples for both datasets. Their performances are evaluated using the top-1 accuracy and top-5 accuracy. The top-1 accuracy measures the percentage of correct predictions, while the top-5 accuracy measures the number of samples with correct labels in the top-5 predicted labels. The reported results are the highest accuracy gotten from the best forming configuration.

Table I. Performance results on both networks using the MNIST and CIFAR-10 datasets.

	MNIST		**CIFAR-10**	
	Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy
VGG-19	**98.6%**	100%	**91.1%**	99.7%
GoogLeNet	98.5%	100%	90.9%	**99.8%**

D. *Further Evaluation*

Inspecting the results from these models, Fig. 10. and Fig. 11. show some of the correctly classified and incorrectly classifies samples from the CIFAR-10 dataset. As this dataset is very complex, it is worth visually this. For the incorrectly classified samples, the top-3 predictions are displayed and for those examples, the correct labels are within the top-3 predictions.

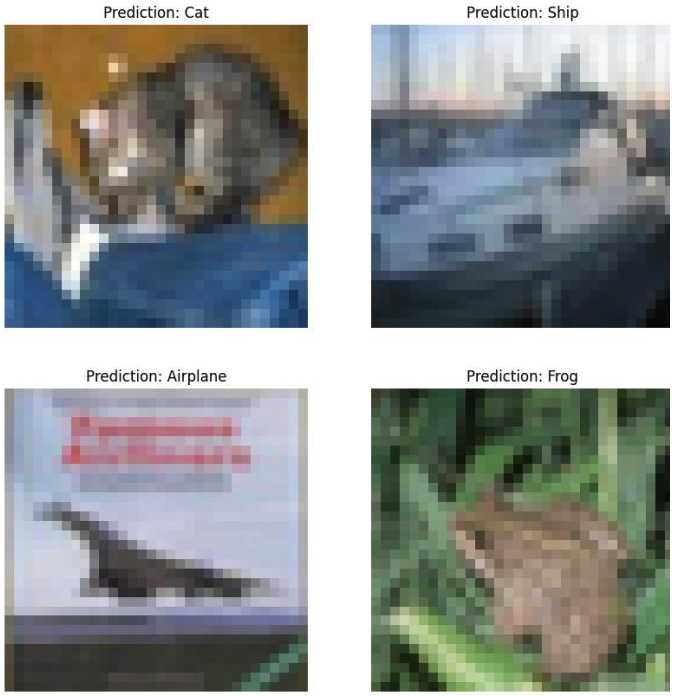

Fig. 10. Some correctly classified samples from the CIFAR10 training dataset.

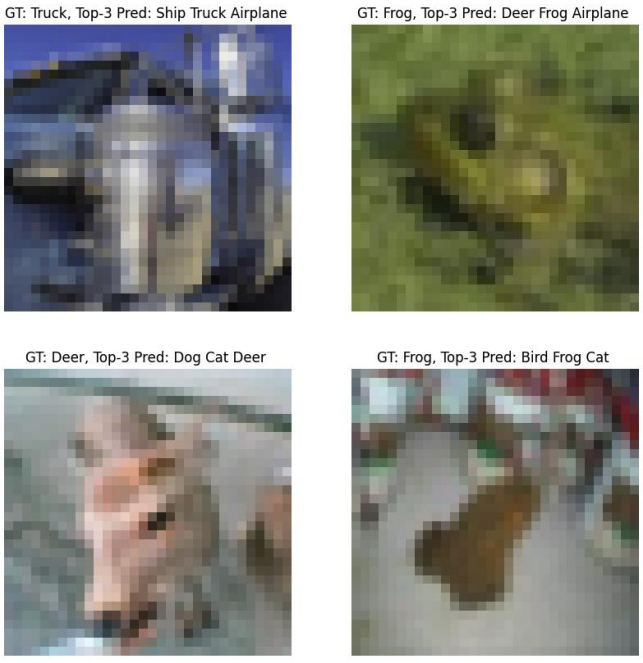

Fig. 11. Some incorrectly classified samples from the CIFAR10 training dataset, along with the top-3 predictions.

V. Conclusion

This report evaluated the performance of two of the most notable deep convolutional neural networks for image classification – VGG and GoogleNet. Both networks were evaluated on the MNIST and CIFAR-10 datasets. The results show similar performance on both datasets showing that indeed deep networks provide improved accuracy. Further improvements to this evaluation can be done by applying other model training strategy and using these trained networks for other learning tasks apart from classification. Some other learning tasks could include object detection, visual question answering, and segmentation. Another could be to use larger datasets for this comparative evaluation.

References

[1] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) 2005 Jun 20 (Vol. 1, pp. 886-893). Ieee.

[2] Lowe DG. Distinctive image features from scale-invariant keypoints. International journal of computer vision. 2004 Nov;60:91-110.

[3] Al-Saffar AA, Tao H, Talab MA. Review of deep convolution neural network in image classification. In2017 International conference on radar, antenna, microwave, electronics, and telecommunications (ICRAMET) 2017 Oct 23 (pp. 26-31). IEEE.

[4] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989 Dec;1(4):541-51.

[5] Stanford University CS231n: Deep Learning for Computer Vision [Internet] [cited 2023 May 11]; Available from https://cs231n.github.io/classification/

[6] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems. 2012: 25:1106–1114.

[7] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[8] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1-9).

[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[10] Yuille AL, Liu C. Deep nets: What have they ever done for vision?. International Journal of Computer Vision. 2021 Mar;129:781-802.

[11] Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:1312.4400. 2013 Dec 16.

[12] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 1492-1500).

[13] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4700-4708).

[14] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. InInternational conference on machine learning 2019 May 24 (pp. 6105-6114). PMLR.

[15] MNIST handwritten digit database [Internet] [cited 2023 May 11]; Available from http://yann.lecun.com/exdb/mnist/

[16] CIFAR-10 and CIFAR-100 datasets [Internet] [cited 2023 May 11]; Available from https://www.cs.toronto.edu/~kriz/cifar.html

[17] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. InProceedings of the thirteenth international conference on artificial intelligence and statistics 2010 Mar 31 (pp. 249-256). JMLR Workshop and Conference Proceedings.