International Journal for Modern Trends in Science and Technology, 6(12): 402-406, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST061276
Available online at: http://www.ijmtst.com/vol6issue12.html

Comparative Study of Various Convolutional Neural Networks on Cifar-10

Tushar Goyal

[1]Department of Information Technology, Delhi, India

To Cite this Article
Tushar Goyal, "Comparative Study of Various Convolutional Neural Networks on Cifar-10", *International Journal for Modern Trends in Science and Technology*, 6(12): 402-406, 2020.

Article Info
Received on 16-November-2020, Revised on 09-December-2020, Accepted on 12-December-2020, Published on 18-December-2020.

ABSTRACT

Image recognition plays a foundational role in the field of computer vision and there has been extensive research to develop state-of-the-art techniques especially using Convolutional Neural Network (CNN). This paper aims to study some CNNs, heavily inspired by highly popular state-of-the-art CNNs, designed from scratch specifically for the Cifar-10 dataset and present a fair comparison between them.

Keywords *- Convolutional Neural Networks, Sequential Network, Inception Network, Residual Network, Dense Network, Wide Residual Network.*

1. INTRODUCTION

Cifar-10 is one of the most popular datasets in the field of computer vision due to the small number of classes and fairly high complexity of the images. The dataset consists of 60,000 32*32 RGB images evenly distributed in 10 classes. Training and testing sets are having 50,000 and 10,000 images respectively with each class having 5,000 and 1,000 images in respective sets.

In this paper, we design four Convolutional Neural Networks (CNN) based on already researched highly successful techniques such as sequential network, inception network, dense network, and wide residual network. We compare the performance of these networks on the accuracy, recall, f1-score, training loss, training accuracy, training time, and the number of parameters.

Section 2 reviews previous studies on the comparison of CNNs on the Cifar-10 dataset. Section 3 provides summary and architecture of various CNNs that are to be compared. In section 4, we provide the details on how the networks are trained. In section 5, we thoroughly present the results and compare the networks on various parameters. Section 6 provides some conclusions from the findings of our work.

2. RELATED WORK

There has been extensive work on developing new networks that need to be compared by previous networks but there has not been much standalone work regarding the comparisons of the CNNs on the un-augmented Cifar-10 dataset.

[1] discusses the comparison of deep neural networks and humans on the Cifar-10 dataset. It provides a good insight on how some networks can even exceed the human accuracy of 93.91%. [2] presents the accuracy achieved by various conventional machine learning algorithms such as Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and a combination of these algorithms with Principal Component Analysis (PCA). It also aims to study CNNs and has achieved an accuracy of 94.03% using an ensemble

of four CNN classifiers and one KNN classifier along with data augmentation. The author in [3] studied many CNNs using vanilla, ensemble, and co-learning and has achieved better results for co-learning.

3. CONVOLUTIONAL NEURAL NETWORKS

CNNs are one of the most effective deep learning techniques used for image recognition and classification. The core of a CNN is to perform convolution operation to learn features of the input images. A CNN may consist of convolutional layer, activation layer, batch normalization layer, pooling layer, dropout layer, and fully-connected dense layer. A loss function is used to calculate loss between output of the network and the ground truth which is then minimized using optimizer by doing back propagation for a number of times, called epochs. Initialization of weights, learning rate of optimizer, number of epochs, loss function, dropout rate, activation function, kernel size, number of filters, etc. are some hyperparameters that can be adjusted to achieve better results.

3.1 SequentialNet

Figure 1: Sequential Block

Block/Layer	Output Size	Remarks
Sequential Block	16 * 16 * 64	filters = 64 in Convs
Sequential Block	8 * 8 * 128	filters = 128 in Convs
Sequential Block	4 * 4 * 256	filter = 256 in Convs
Flatten	4096	
Dense	1024	units = 1024 activation = ReLU
Dropout	1024	rate = 0.5
Dense	1024	units = 1024 activation = ReLU
Dropout	1024	rate = 0.5
Dense	1024	units = 1024 activation = ReLU
Dropout	1024	rate = 0.7
Dense	1024	units = 1024 activation = ReLU
Dropout	1024	rate = 0.8
Dense	10	units = 10 activation = Softmax

Figure 2: Architecture of SequentialNet

When the layers of a network are connected sequentially, the network can be termed as a sequential network. VGG, as described in [4] is essentially a sequential network with kernels of size 3*3 across the network. Our SequentialNet is inspired by the architecture presented in [2] and [4]. The architecture of our network is described in Figure 2. Each convolutional layer has a kernel size of 3*3, strides of 1*1, and 'same' padding. Downsampling is performed using MaxPooling layer, present inside the Sequential Block with a pool size of 2*2 which divides the dimension by 2. The Dense layers near the end of our network have high dropout rate to avoid overfitting.

3.2 InceptionNet

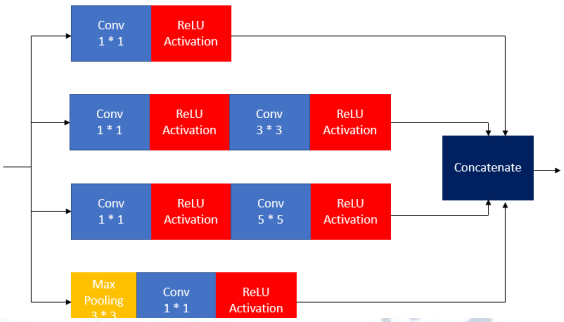

Figure 3: Inception Block

Block/Layer	Output Size	Remarks
Conv	32 * 32 * 64	filters = 64 kernel_size = 7 * 7
Inception Block	32 * 32 * 128	filters = 32 in Convs
Inception Block	32 * 32 * 128	filters = 32 in Convs
MaxPooling	16 * 16 * 128	pool_size = 2 * 2
Inception Block	16 * 16 * 256	filters = 64 in Convs
Inception Block	16 * 16 * 256	filters = 64 in Convs
MaxPooling	8 * 8 * 256	pool_size = 2 * 2
Inception Block	8 * 8 * 512	filters = 128 in Convs
Inception Block	8 * 8 * 512	filters = 128 in Convs
MaxPooling	4 * 4 * 512	pool_size = 2 * 2
Inception Block	4 * 4 * 1024	filters = 256 in Convs
Inception Block	4 * 4 * 1024	filters = 256 in Convs
MaxPooling	2 * 2 * 1024	pool_size = 2 * 2
Inception Block	2 * 2 * 2048	filters = 512 in Convs
Inception Block	2 * 2 * 2048	filters = 512 in Convs
AveragePooling	1 * 1 * 2048	pool_size = 2 * 2
Flatten	2048	
Dense	10	units = 10 activation = Softmax

Figure 4: Architecture of InceptionNet

[5] suggests the use of Inception modules having variable size kernels to capture the image features effectively. Inception networks are sparsely connected thus allow us to increase the depth. Our Inception Block in Figure 3 is exactly same as the 'Inception module with dimension reduction' of [5]. The output of previous layer goes through four paths with 1*1 Conv, 3*3 Conv, 5*5 Conv, and MaxPooling and is concatenated to produce output. Additional 1*1 Convs are used to limit the computational requirements. The convolutional

layers use strides of 1*1 and 'same' padding. Downsampling is performed using MaxPooling layer with a pool size of 2*2 after every two Inception Blocks.

3.3 DenseNet

Figure 5: Conv Block

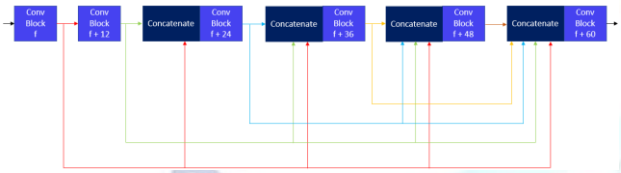

Figure 6: Dense Block

Figure 7: Transition Block

Block/Layer	Output Size	Remarks
Conv	32 * 32 * 24	filters = 24 kernel_size = 3 * 3
Dense Block	32 * 32 * 84	f = 24
Transition Block	16 * 16 * 84	filters = 84 in Convs
Dense Block	16 * 16 * 144	f = 84
Transition Block	8 * 8 * 144	filters = 144 in Convs
Dense Block	8 * 8 * 204	f = 144
Transition Block	4 * 4 * 204	filters = 204 in Convs
Dense Block	4 * 4 * 264	f = 204
AveragePooling	1 * 1 * 264	pool_size = 4 * 4
Flatten	264	
Dense	10	units = 10 activation = Softmax

Figure 8: Architecture of DenseNet

Residual networks (ResNet) were introduced in [6] to allow the training of very deep networks by introduction of 'skip connections' where output of a layer is added to some farther layer in network. This resulted in huge success and paved the way for more architectures based on this technique. DenseNet of [7] is fundamentally a ResNet which connects each layer to every other layer in a feed-forward fashion. Figure 6 shows a Dense Block having dense connection between the Conv Blocks. The 'f' in the Conv Blocks of the Dense Block is the number of filters used in the convolutional layers. The Dense Block adds up 60 more channels in the input provided to it. Transition blocks are used to downsample the input. We use BatchNormalization-ReLU-Conv sequence in the Conv Block as shown in Figure 5. The architecture of our DenseNet in Figure 8 is very similar to the architecture discussed in [7].

3.4 WideResNet

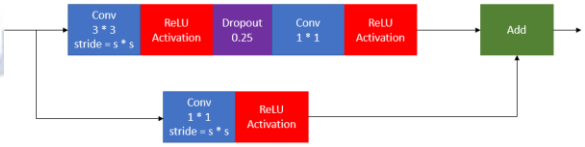

Figure 9: WideRes Block

Block/Layer	Output Size	Remarks
Conv	32 * 32 * 16	filters = 16 kernel_size = 3 * 3
Activation	32 * 32 * 16	activation = ReLU
Batch Normalization	32 * 32 * 16	
WideRes Block	32 * 32 * 128	filters = 128 in Convs s = 1
WideRes Block	32 * 32 * 128	filters = 128 in Convs s = 1
WideRes Block	32 * 32 * 128	filters = 128 in Convs s = 1
WideRes Block	16 * 16 * 256	filters = 256 in Convs s = 2
WideRes Block	16 * 16 * 256	filters = 256 in Convs s = 1
WideRes Block	16 * 16 * 256	filters = 256 in Convs s = 1
WideRes Block	8 * 8 * 512	filters = 512 in Convs s = 2
WideRes Block	8 * 8 * 512	filters = 512 in Convs s = 1
WideRes Block	8 * 8 * 512	filters = 512 in Convs s = 1
AveragePooling	1 * 1 * 512	pool_size = 8 * 8
Flatten	512	
Dense	10	units = 10 activation = Softmax

Figure 10: Architecture of WideResNet

Wide residual network (WideResNet) presented in [8] are also a type of ResNet. The WideResNet aims to increase the width of the network rather than depth to tackle the common problems of deep networks such as high computational requirement and diminishing feature reuse. The WideRes Block in Figure 9 is itself responsible for downsampling when the 's' (stride) is greater than one. We use 'same' padding and stride of 1*1 in all convolutional layers unless specifically stated. We use three groups each having three WideRes Blocks in our WideResNet architecture. The first WideRes Block of the groups usually performs downsampling as shown in Figure 10.

4. TRAINING

We train the networks on 45,000 images of the training set having 50,000 images and use the remaining 5,000 images for validation. All the

networks are trained with a batch size of 100 thus making exactly 450 iterations in each epoch. We train the network for 120 epochs so that they reach the saturation point for training accuracy. We use Adam optimizer with a learning rate of 0.0001 and categorical cross entropy as the loss function. We use Keras implementation of all the layers, optimizer, etc. and Google Colab with GPU runtime as the platform.

5. RESULTS

Model	Accuracy	Number of Parameters (in millions)	Training Time (for 1 epoch) (in seconds)
SequentialNet	86.82%	9.28	26
InceptionNet	78.64%	32.16	78
DenseNet	79.87%	7.22	104
WideResNet	84.34%	17.84	64

Figure 11: Comparison Table

In Figure 11, we can see that simpler networks such as SequentialNet and WideResNet can perform better than InceptionNet and DenseNet. The reason behind the lesser accuracy of these networks could be attributed to the small size of our dataset.

The confusion matrices in Figure 12 illustrate that all the networks are confusing between 'cat' and 'dog' images. It is to be noted that there is significant confusion between 'bird' and other classes.

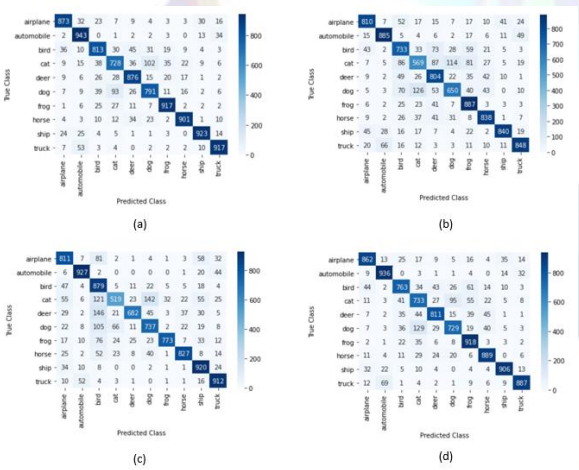

Figure 12: Confusion Matrix for (a) SequentialNet; (b) InceptionNet; (c) DenseNet; (d) WideResNet

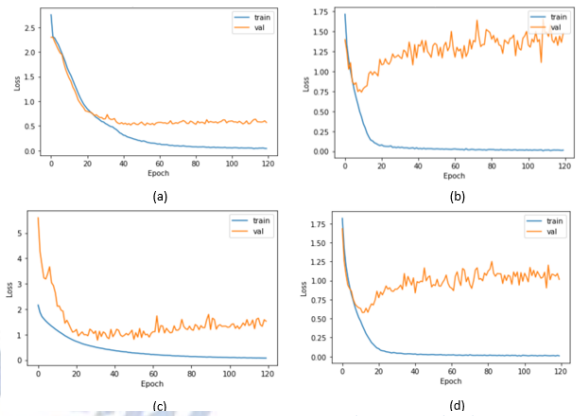

Figure 13: Loss vs Epoch graph for (a) SequentialNet; (b) InceptionNet; (c) DenseNet; (d) WideResNet

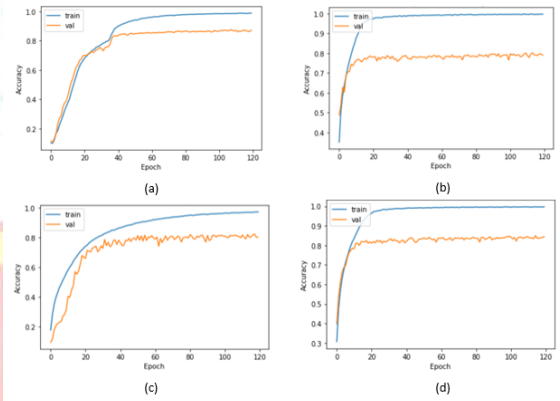

Figure 14: Accuracy vs Epoch graph for (a) SequentialNet; (b) InceptionNet; (c) DenseNet; (d) WideResNet

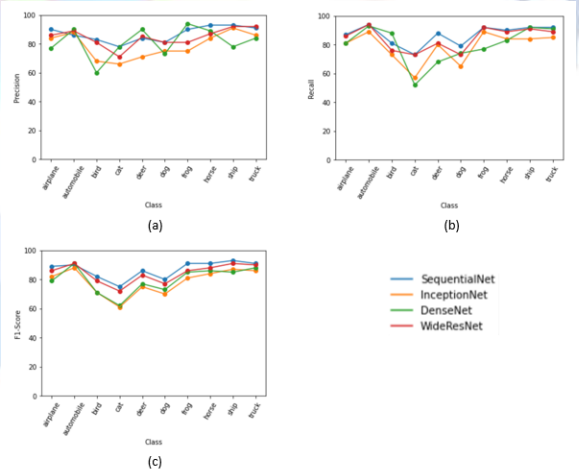

Figure 15: Class wise plot for (a) Precision; (b) Recall; (c) F1-Score

The graphs in Figure 13 and Figure 14 show that all the networks except SequentialNet are highly fluctuating around a particular value of loss and accuracy. They also show that the networks can achieve the saturation point much before 120 epochs. There is a considerably large gap between the training metrics (loss and accuracy) and validation metrics which signifies overfitting of the

models besides using many regularization techniques.

Study in [1] reveals that the humans are also falling short in classifying the 'cat' images which is also a consistent problem in all of our networks as shown in Figure 15. It suggests that the 'cat' images are harder to learn and generalize.

6. CONCLUSION

Though our networks are not the exact copy of the state-of-the-art networks but they still have the same principle behind them and in some cases the architecture is also very similar. In this paper, we evaluated the performance of various CNNs on un-augmented Cifar-10 dataset. We found that simpler networks can outperform complex networks. It can be concluded that going deeper may not always work even after using regularization. Our work provides a direction to the researchers that neural networks have tremendous scope in generalization and learning of complex feature.

REFERENCES

[1]Tien Ho-Phuoc, "CIFAR10 to Compare Visual Recognition Performance between Deep Neural Networks and Humans", arXiv:1811.07270v2 [cs.CV], 2019

[2]Y. Abouelnaga, O. S. Ali, H. Rady and M. Moustafa, "CIFAR-10: KNN-Based Ensemble of Classifiers," 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2016, pp. 1192-1195, doi: 10.1109/CSCI.2016.0225.

[3]Kele Xu, Haibo Mi, Dawei Feng, Huaimin Wang, Chuan Chen, Zibin Zheng, Xu Lan, "Collaborative Deep Learning Across Multiple Data Centers", arXiv:1810.06877v1 [cs.LG], 2018

[4] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large Scale Image Recognition", arXiv:1409.1556v6 [cs.CV], 2015

[5]C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[7]G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[8]Sergey Zagoruyko, Nikos Komodakis, "Wide Residual Networks", arXiv:1605.07146v4 [cs.CV], 2017