COMPARATIVE STUDY & ENSEMBLE OF VARIOUS CONVOLUTIONAL NEURAL NETWORKS ON CIFAR-

IJSREM Journal

IJSREM Journal

COMPARATIVE STUDY & ENSEMBLE OF VARIOUS CONVOLUTIONAL NEURAL NETWORKS ON CIFAR-10

Tushar Goyal

Maharaja Agrasen Institute of Technology

Delhi, India

Abstract - Image classification plays an important role in the field of computer-vision and there has been extensive research to develop state-of-the-art neural networks. This paper aims to study some CNNs, inspired by some highly popular state-of-the-art CNNs, designed from scratch specifically for the Cifar-10 dataset and present a comparison between them. The paper also provides a new model based on ensembling of the models developed in the paper to classify Cifar-10 images.

Keywords - Convolutional Neural Networks, Data Augmentation, Sequential Network, Residual Network, Squeeze Network, Depthwise Convolutional Network, Ensembling.

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, residual network, squeeze network, and depthwise convolutional network. We use augmentation of images to enhance the accuracy of models. We compare the performance of these networks on the accuracy, recall, f1-score, training loss, training accuracy, training time, and the number of parameters. We then design ensemble model of the CNNs developed in this paper. For ensembling, we use various voting techniques to get better results.

Section 2 reviews previous studies on the comparison and ensemble of CNNs on the Cifar-10 dataset. The Section 3 discusses the data augmentation techniques and benefits. Section 4 provides summary and architecture of various CNNs that are to be compared. In section 5, we provide the details on how the networks are trained. In section 6, we thoroughly compare the networks on various parameters. Section 7 presents the ensemble models and the results are being presented in Section 8. Section 9 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 CNN classifiers on the Cifar-10 dataset. There is hardly any detailed research work in the domain of application of ensemble learning on Cifar-10 dataset.

[1] discusses the comparison of CNN classifiers on Cifar-10 dataset in details. It discusses four types of CNNs such as Sequential Network, Inception Network, Dense Network and Wide Residual Network. The paper found the Sequential Network, among all models, best suited for the dataset with an accuracy of 86.82%. It doesn’t use augmentation technique. The study in [2] compares many machine learning models and uses ensemble of four sequential CNN classifiers and a PCA-KNN classifier for final classification which achieves an accuracy of 94.03%. It also uses image preprocessing and data augmentation which might be one of the reasons for excellent results. [3] gives an interesting insight by comparing CNNs with human vision. The CNNs can far exceed human accuracy of 93.91%.

3. Data Augmentation

CNNs works exceptionally well on image datasets but they often tend to overfit the training data. Overfitting can be reduced to a great extent with a large training dataset but researchers usually don’t have access to big datasets. In order to overcome this shortcoming, data augmentation technique is used to enhance the size and quality of training data. [4] describes the effectiveness of data augmentation in image classification. The survey of [5] presents detailed information of various data augmentation techniques out of which flipping, translation, rotation, shearing and zooming have been used in the paper for every CNN model.

4. 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.

4.1 SequentialNet + Augmentation

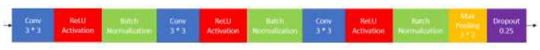
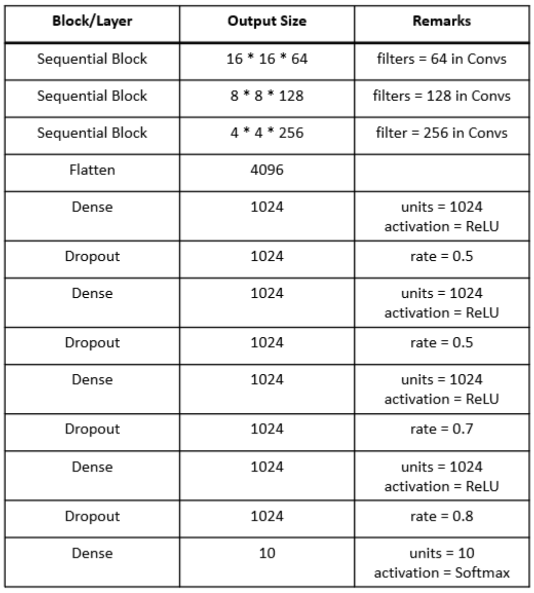


Figure 1: Sequential Block

When the layers of a network are connected sequentially, the network can be termed as a Sequential Network. VGG, as described in [6] is essentially a sequential network with kernels of size 3*3 across the network. Our SequentialNet is inspired by the architecture presented in [1] and [2] and [6]. The architecture of a sequential block has been described in Figure 1. The architecture of our network formed using sequential blocks is described in Figure 2.



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

Each convolutional layer has a kernel size of 3*3, strides of 1*1, and 'same' padding. Downsampling is performed using max pooling 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.

4.2 ResidualNet + Augmentation

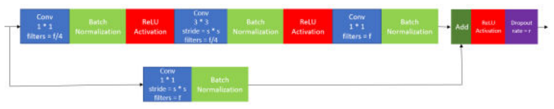


Figure 3: Residual Block

Subsequent increase of the depth of a sequential network leads to the problem of vanishing/exploding gradient. This problem was solved in [7] using Residual Network. In this network we use a technique called 'skip connections' which skips training from a few layers and connects directly to the output. The advantage of adding this type of skip connection is that if any layer hurt the performance of architecture, then it will be skipped by regularization.

We present a residual block in Figure 3, which uses a convolution layer with 1*1 kernel size in skip connection to match the dimensions before adding up it to the sequential flow. The 'f' in convolution layers represents number of filters, 's' represents strides and 'r' represents the dropout rate.

Block/Layer	Output Size	Remarks
Conv	32 * 32 * 64	filters = 64, kernel_size = 5 * 5
Activation	32 * 32 * 64	activation = ReLU
Batch Normalization	32 * 32 * 64	
Residual Block	16 * 16 * 128	filters = 128, stride = 2 * 2, dropout = 0.15
Residual Block	16 * 16 * 128	filters = 128, stride = 1 * 1, dropout = 0.25
Residual Block	8 * 8 * 256	filters = 256, stride = 2 * 2, dropout = 0.15
Residual Block	8 * 8 * 256	filters = 256, stride = 1 * 1, dropout = 0.25
Residual Block	4 * 4 * 512	filters = 512, stride = 2 * 2, dropout = 0.15
Residual Block	4 * 4 * 512	filters = 512, stride = 1 * 1, dropout = 0.25
Residual Block	2 * 2 * 1024	filters = 1024, stride = 2 * 2, dropout = 0.15
Residual Block	2 * 2 * 1024	filters = 1024, stride = 1 * 1, dropout = 0.25
Residual Block	1 * 1 * 2048	filters = 2048, stride = 2 * 2, dropout = 0.15
Residual Block	1 * 1 * 2048	filters = 2048, stride = 1 * 1, dropout = 0.00
Flatten	2048	
Dense	1024	units = 1024, activation = ReLU
Dense	512	units = 512, activation = ReLU
Dense	10	units = 10, activation = Softmax

Figure 4: Architecture of ResidualNet

The architecture of Residual Network has been described in Figure 4. We use five pairs of residual blocks in which the first block of the pair always performs downsampling by using a stride of 2*2.

4.3 SqueezeNet + Augmentation

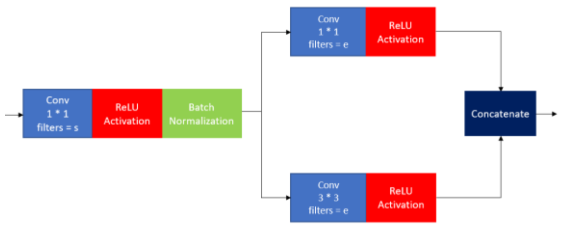


Figure 5: Fire Block

Squeeze Network is essentially a small and efficient convolutional network that provides equivalent accuracy as that of deep convolution networks. As described in [8], it uses three principal strategies to reduce the size of network such as replacement of 3*3 filters with 1*1 filters, decreasing the number of input channels to 3*3 filters and late downsampling.

The fire block in Figure 5 is comprised of two layers namely squeeze layer and expand layer. The squeeze layer consists of 1*1 convolutional kernels and the expand layer consists of 1*1 convolution kernels and 3*3 convolutional kernels which are later concatenated. The 's' represents number of filters in squeeze layer and 'e' represents number of filters in expand layer.

Block/Layer	Output Size	Remarks
Conv	16 * 16 * 96	filters = 96, kernel_size = 3 * 3, strides = 2 * 2
Activation	16 * 16 * 96	activation = ReLU
Max Pooling	8 * 8 * 96	pool_size = 2 * 2, strides = 2 * 2
Batch Normalization	8 * 8 * 96	
Fire Block	8 * 8 * 256	s = 32, e = 128
Max Pooling	3 * 3 * 256	pool_size = 3 * 3, strides = 2 * 2
Fire Block	3 * 3 * 384	s = 48, e = 192
Fire Block	3 * 3 * 384	s = 48, e = 192
Fire Block	3 * 3 * 384	s = 48, e = 192
Max Pooling	1 * 1 * 384	pool_size = 3 * 3, strides = 2 * 2
Dropout	1 * 1 * 384	rate = 0.5
Conv	10	filters = 10, , kernel_size = 1 * 1
Activation	10	activation = ReLU
Batch Normalization	10	
Average Pooling	10	
Activation	10	activation = Softmax

Figure 6: Architecture of SqueezeNet

The architecture of Squeeze Network as described in Figure 6 is similar to that in [8]. It performs downsampling using max pooling layers embedded between certain fire blocks.

4.4 DepthwiseConvNet + Augmentation

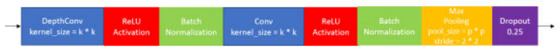


Figure 7: DepthwiseConv Block

Block/Layer	Output Size	Remarks
Conv	32 * 32 * 64	filters = 64 in Convs, kernel_size = 3 * 3
Activation	32 * 32 * 64	activation = ReLU
Batch Normalization	32 * 32 * 64	
DepthwiseConv Block	16 * 16 * 128	filters = 128 in Convs, kernel_size = 3 * 3, pool_size = 2 * 2
DepthwiseConv Block	8 * 8 * 256	filters = 256 in Convs, kernel_size = 3 * 3, pool_size = 2 * 2
DepthwiseConv Block	4 * 4 * 512	filters = 512 in Convs, kernel_size = 2 * 2, pool_size = 2 * 2
DepthwiseConv Block	1 * 1 * 1024	filters = 1024 in Convs, kernel_size = 2 * 2, pool_size = 4 * 4
Flatten	1024	

Dense	10	units = 10
		activation = Softmax

Figure 8: Architecture of DepthwiseConvNet

In depthwise convolution layer, we use one filter for each input channel. For example, if the number of input channels are three, we use three different filters thus producing an output with exactly same number of channels. The depthwise network was proposed in [9] with meticulous details. The technique of depthwise convolution is widely used in Mobile Network [10] and Shuffle Network [11].

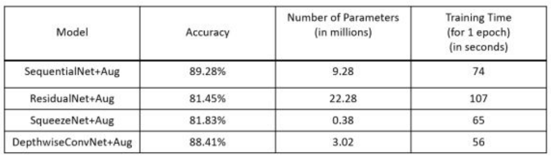
The Figure 7 presents the depthwise convolutional block consisting depth convolution, ReLU activation, batch normalization, convolution, ReLU activation, batch normalization, max pooling for downsampling and a dropout of 0.25.

Instead of using a 1*1 pointwise convolution, we have used 3*3 and 2*2 convolutions as described in Figure 8. The architecture is largely linear in nature, one of the many arrangements described in [9].

5. 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. We train the network for 200 epochs so that they reach the saturation point for training accuracy. We use Adam optimizer with a learning rate of 0.0001 (except that in SqueezeNet which uses a learning rate of 0.001) 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.

6. Comparison



Model	Accuracy	Number of Parameters (in millions)	Training Time (for 1 epoch) (in seconds)
SequentialNet+Aug	89.28%	9.28	74
ResidualNet+Aug	81.45%	22.28	107
SqueezeNet+Aug	81.83%	0.38	65
DepthwiseConvNet+Aug	88.41%	3.02	56

Figure 9: Comparison Table of CNNs

In Figure 9, we can see that the networks such as SequentialNet+Aug and DepthwiseConvNet+Aug can perform better than ResidualNet+Aug and SqueezeNet+Aug.

The confusion matrices in Figure 10 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.

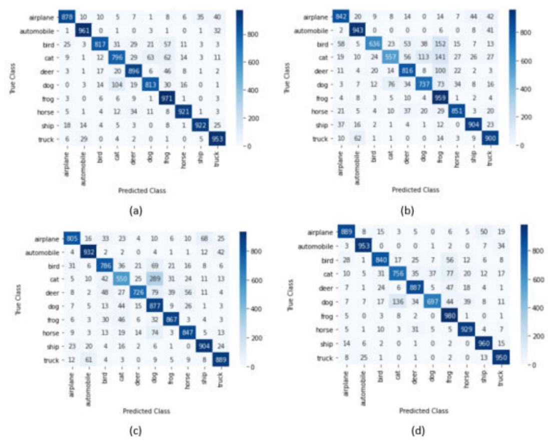


Figure 10: Confusion Matrix for (a) SequentialNet+Aug; (b) ResidualNet+Aug; (c) SqueezeNet+Aug; (d) DepthwiseConvNet+Aug

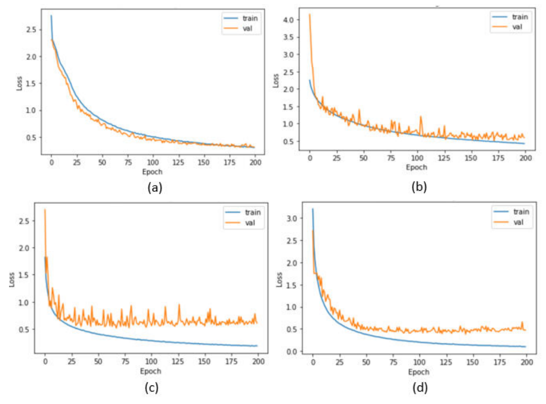


Figure 11: Loss vs Epoch graph for (a) SequentialNet+Aug; (b) ResidualNet+Aug; (c) SqueezeNet+Aug; (d) DepthwiseConvNet+Aug

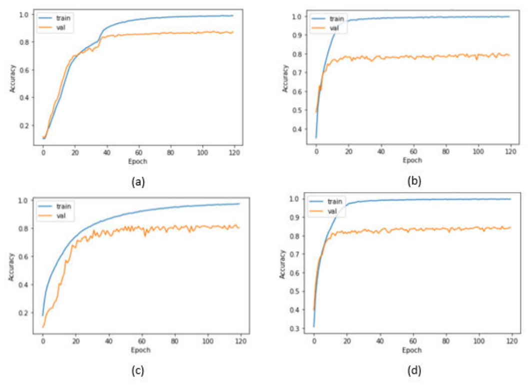


Figure 12: Accuracy vs Epoch graph (a) SequentialNet+Aug; (b) ResidualNet+Aug; (c) SqueezeNet+Aug; (d) DepthwiseConvNet+Aug

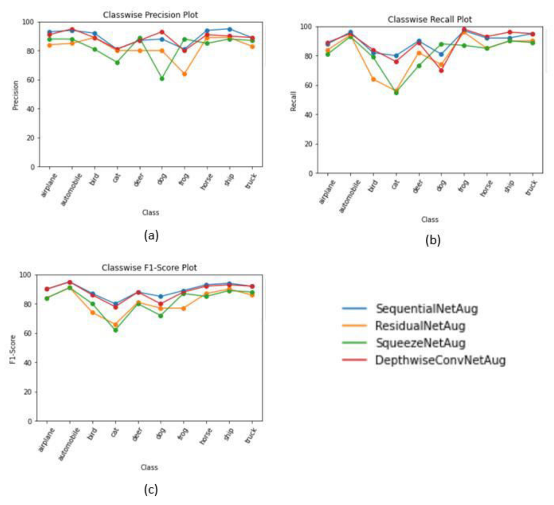


Figure 13: Class wise plot for (a) Precision; (b) Recall; (c) F1-Score

The graphs in Figure 11 and Figure 12 show that the ResidualNet+Aug and SqueezeNet+Aug are highly fluctuating around a particular value of loss and accuracy. They also show that the SqueezeNet+Aug and DepthwiseConvNet+Aug have reached saturation point at 75 epochs and don't show much improvement in validation metrics after it. 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 and data augmentation.

Figure 13 suggests that the 'cat' images are harder to learn and generalize by all the models.

7. Ensembling

Ensemble is a technique of combining results of multiple independent machine learning models to produce better result than any of the individual model used in the process. As described in [12], it helps in reducing the three general problems of networks that are statistical problem, computational problem and representation problem. [13] presents a variety of voting mechanisms to develop an ensemble model.

7.1 Four Model Max Voting Ensemble

In maximum voting, the resultant class is the one which has maximum number of votes. The four models developed in this paper are used to develop ensemble model.

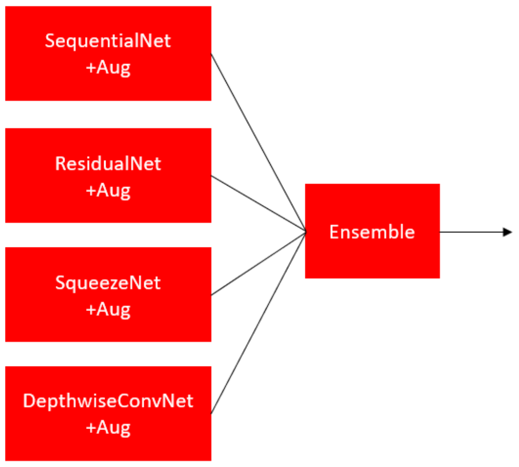


Figure 14: 4-Model Max Voting Ensemble

7.2 Four Model Weighted Average Voting Ensemble

In weighted average voting, each model is assigned certain weight. The predictions of models are multiplied by their respective weights and the resultant is average of them. The four models developed in this paper are used to develop this ensemble model.

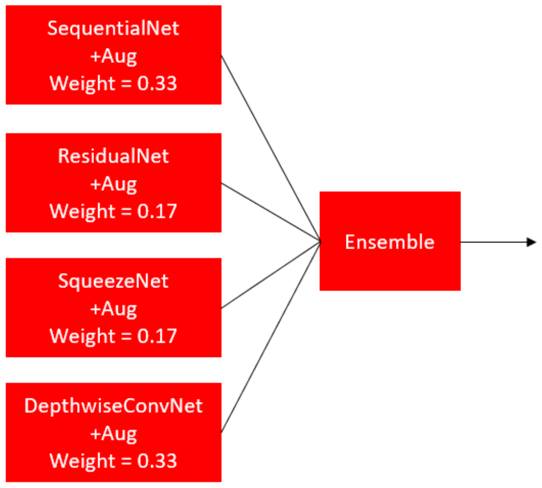

SequentialNet +Aug Weight = 0.33
ResidualNet +Aug Weight = 0.17
SqueezeNet +Aug Weight = 0.17
DepthwiseConvNet +Aug Weight = 0.33
Ensemble

Figure 15: 4-Model Weighted Average Voting Ensemble

7.3 Eight Model Max Voting Ensemble

The eight models include the four models developed in this paper and four models developed in [1]. These eight models are combined using max voting to develop this ensemble model.

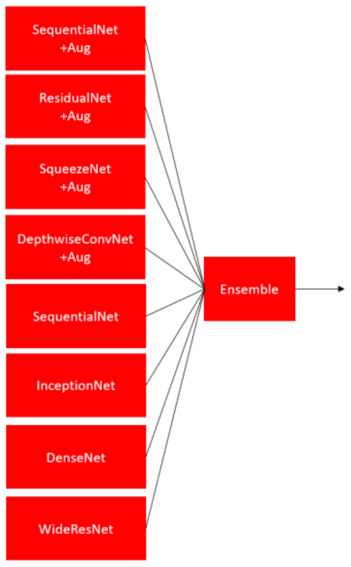


Figure 16: 8-Model Max Voting Ensemble

7.4 Eight Model Weighted Average Voting Ensemble

The eight models include the four models developed in this paper and four models developed in [1]. These eight models are combined using weighted average voting to develop this ensemble model.

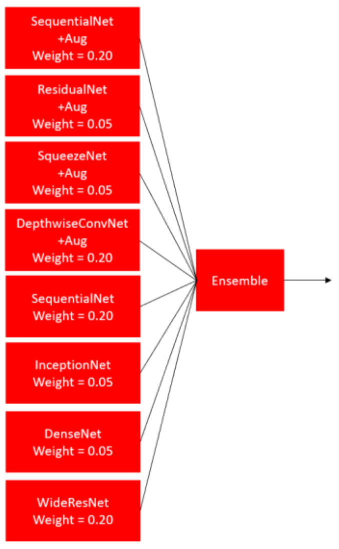

SequentialNet +Aug Weight = 0.20
ResidualNet +Aug Weight = 0.05
SqueezeNet +Aug Weight = 0.05
DepthwiseConvNet +Aug Weight = 0.20
SequentialNet Weight = 0.20
InceptionNet Weight = 0.05
DenseNet Weight = 0.05
WideResNet Weight = 0.20
Ensemble

Figure 17: 8-Model Weighted Average Voting Ensemble

8. Results

The Figure 18 presents a comparison of accuracy and Figure 19 presents the confusion matrices of various ensembles. It can be interpreted that the increase in number of models in the ensembling actually results in increased accuracy. The weighted average voting performs better than the max voting probably due to the large weightage to high accuracy models.

Ensemble	Accuracy
4-Model Max Voting	89.98%
4-Model Weighted Average Voting	90.97%
8-Model Max Voting	91.07%
8-Model Weighted Average Voting	91.55%

Figure 18: Comparison Table of Ensembles

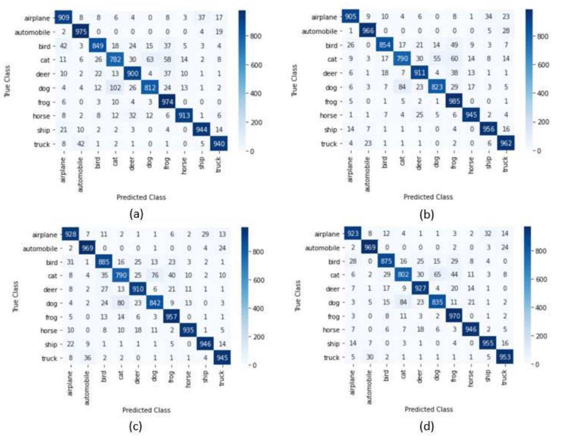


Figure 19: Confusion Matrix for (a) 4-Model Max Voting Ensemble; (b) 4-Model Weighted Average Voting Ensemble; (a) 8-Model Max Voting Ensemble; (b) 8-Model Weighted Average Voting Ensemble

9. Conclusion

In this paper, we evaluated the performance of various CNNs on augmented Cifar-10 dataset. We found that the data augmentation not necessarily reduce the overfitting significantly though minor improvement is visible. The ensemble technique can drastically improve the performance using many CNNs. The ensembling works better in case of weighted-average voting by providing larger weights to high accuracy models. Our work provides a direction to the researchers that neural networks have tremendous scope in generalization and learning of complex feature.

References

[1] 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.

[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), 2016, pp. 1192-1195, doi: 10.1109/CSCI.2016.0225.

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

[4] Jason Wang, Luis Perez, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", arXiv:1712.04621v1 [cs.CV], 2017.

[5] Shorten, Connor & Khoshgoftaar, Taghi. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data. 6. 10.1186/s40537-019-0197-0.

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

[7] 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), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[8] Iandola, Forrest & Han, Song & Moskewicz, Matthew & Ashraf, Khalid & Dally, William & Keutzer, Kurt. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size.

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

[10] Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[11] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848-6856, doi: 10.1109/CVPR.2018.00716.

[12] Dietterich T.G. (2000) Ensemble Methods in Machine Learning. In: Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol 1857. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45014-9_1

[13] R. K. Shahzad and N. Lavesson, 'Comparative Analysis of Voting Schemes for Ensemble-based Malware Detection', Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 4, no. 1, pp. 98–117, 2013.