© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

Improving Convolutional Neural Network (CNN) Architecture (miniVGGNet) with Batch Normalization and Learning Rate Decay Factor for Image Classification

Asmida Ismail[1,3]*, Siti Anom Ahmad[1], Azura Che Soh[1], Khair Hassan[1], Hazreen Haizi Harith[2]

[1]Department of Electrical and Electronic Engineering, Faculty of Engineering, UPM, Serdang, 43400, MALAYSIA

[2]Department of Biological and Agricultural Engineering, Faculty of Engineering, UPM, Serdang, 43400, MALAYSIA

[3]Department of Engineering Technology, Faculty of Technical Vocational, UPSI, Tanjung Malim, 35900 MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2019.11.04.006
Received 25 April 2019; Accepted 8 July 2019; Available online 5 September 2019

Abstract: The image classification is a classical problem of image processing, computer vision, and machine learning. This paper presents an analysis of the performance using Convolutional Neural Network (CNN) for image classifying using deep learning. MiniVGGNet is CNN architecture used in this paper to train a network for image classification, and CIFAR-10 is selected dataset used for this purpose. The performance of the network was improved by hyper parameter tuning techniques using batch normalization and learning rate decay factor. This paper compares the performance of the trained network by adding batch normalization layer and adjusting the value of learning rate decay factor for the network architecture. Based on the experimental results, adding batch normalization layer allow the networks to improve classification accuracy from 80% to 82%. Applying learning rate decay factor will improve classification accuracy to 83% and reduce the effects of overfitting in learning plot. Performance analysis shows that applying hyper parameter tuning can improve the performance of the network and increasing the ability of the model to generalize.

Keywords: Convolutional Neural Network, deep learning, MiniVGGNet, hyper parameter.

1. Introduction

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision [1].

In traditional classification approaches, image features are normally carefully hand-crafted to maximize distinction capability. A lot of hand-crafted designed features have been explored before (such as Local Binary Pattern, LBPs [2], histogram of gradient oriented, HOG [3], SIFT and SURF [4]) and achieved excellent success in computer vision tasks. However, these hand-designed features are not learned from the nature of data and subjective to the perception of the designers. Sometimes there is not the optimal feature required for a given task [5]. Besides, classifiers (such as the k-

*Corresponding author: asmida@ftv.upsi.edu.my, sanom@upm.edu.my

nearest neighbor [6] and SVM [7]) are generic and not robust to the different varieties of the data. These traditional approaches represent a shallow architecture, which is severely challenged by the non-linear complexity of the features.

Recently, deep learning researchers have proposed to learn feature representations in a hierarchy from pixels to classifiers through multiple layers (deep) architecture [8, 9]. The study of image classification using deep learning was explored in this paper. Deep learning is a subdivision of machine learning algorithms, which are excellent in identifying patterns, and usually, require more data.

The most popular technique used to improve the accuracy of image classification is the Convolutional Neural Network (CNN) [10]. CNN is an efficient and effective recognition, identification and classification algorithm which is globally used in image processing and pattern recognition [11].

The deep learning method can learn to extract the feature from large-scale dataset automatically compared to hand-crafted feature extraction based method. The deep architecture allows the system to learn to represent features by itself based on the nature of the data, rather than the subjective nature of human perception [5]. And it achieves excellent performance in image classification and recognition in recent years AlexNet [12] proposed in 2012, have eight trainable layers demonstrates extraordinary performance on ImageNet dataset and can classify up to1000 object. Since then, CNN architecture was used because of its excellent performance and showed the trend to become more profound. The VGGNet [13] proposed in 2014 has up to 19 trainable layers. Deeper architectures are proven to improve classification performance. In this paper, a smaller version of VGGNet called miniVGGNet was used as architecture to train the network for image classification. Hyper parameter tuning method is used to improve the performance of the network. Adding batch normalization layer to the network and tuning the value of learning rate decay factor will improve the performance of the network by increasing the value of classification accuracy and reduce the effect of the overfitting in the learning plot.

This paper is organized as follows: Section II explains the overall methodology to build image classification model using deep learning convolutional neural network. Section III explains on results and discussions, and the last section IV concludes the paper.

2. Methods

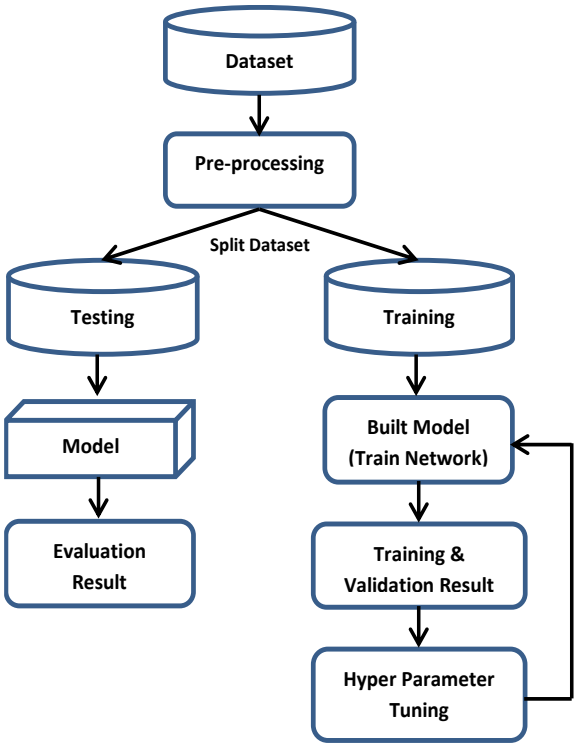


Fig. 1: Image classification algorithm – process to build image classification model using deep learning convolutional neural network.

Fig. 1 show the process to build an image classification model using deep learning CNN. The processes consist of several steps that are:

2.1 Dataset

The first step of building a deep learning network is to gather an initial dataset. The images need to be labelled and linked with each image. These labels should come from a finite set of classes. The number of images for each category should be approximately uniform (i.e., the same number of examples per class). In this paper, CIFAR-10 dataset is selected for the classification purpose.

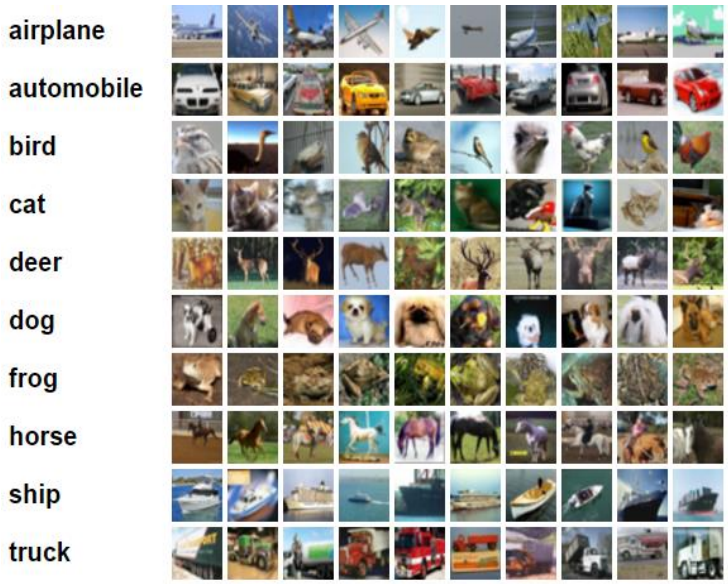


Fig. 2: Classes in the CIFAR-10 dataset

Fig. 2 shows the image classes in CIFAR-10 dataset as well as ten random images from each class. CIFAR-10 is standard benchmark dataset for image classification in the computer vision and machine learning literature. CIFAR-10 consists of 60,000 32 x 32 x 3 (RGB) images resulting in a feature vector dimensionality of 3072. As the name suggests, CIFAR-10 consists of 10 classes, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset consists of 50000 training images and 10000 test images. The test images are randomly selected from each class.

2.2 Pre-processing

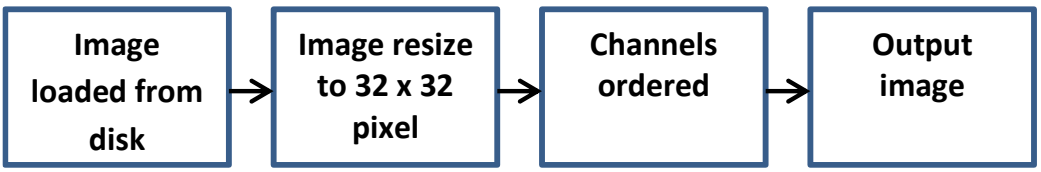


Fig. 3: Image pre-processing pipeline

There are some pre-processing steps that might carry out before training deep learning model. Fig. 3 shows the image pre-processing pipeline. The image will load from the disk and resize to 32 x 32 pixels. This step is to ensure that the images have the same size and aspect ratio. To start constructing the model, the images presented to the input layer should be square. After the image is resized, the next step is applying channel ordering. For inputs to the CNN, the depth is the number of channels in the image or the number of filters in a layer. The dimension ordering specified the shape of the input and used to learn multi-level features in the image.

2.3 Split Dataset

Initial dataset needs to split into two parts: a training set and a testing set. A training set is used by the classifier to "learn" what each class looks like by making predictions on the input data and then correct the data by itself when predictions are wrong. A testing set is used to evaluate the performing of classifier after the classifier has been trained [15].

It's extremely important that the training set and testing set are independent and not overlap each other.

2.4 Train the Network

From the training set of images, the network will be trained. From the training process, the network will learn how to recognize each of the categories in labeled data. When the model makes a mistake, it learns from this mistake and improves itself. In this paper, CNN architecture used to build a classification model is miniVGGNet architecture. The network architecture detailed is described in the next section below.

2.4.1 CNN Architecture

VGGNet, (sometimes referred to as simply VGG), was first introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Learning Convolutional Neural Networks for Large-Scale Image Recognition [13]. VGGNet is unique in that it uses 3 x 3 kernels throughout the entire architecture. The use of these small kernels is arguably what helps VGGNet generalize to classification problems outside what the network was originally trained on.

In VGGNet, multiple CONV => ACT layers were arranging before applying a single POOL layer. This multiple CONV => ACT layer allows the network to extract more features from the image before reducing the size of an input volume via the POOL operation.

In general, miniVGGNet consists of two sets of CONV => ACT => CONV => ACT => POOL layers, followed by a set of FC => ACT => FC => SOFTMAX layers. 32 filters with each of size 3 x 3 will learn on the first two CONV layers. The following CONV layers will learn 64 filters, with each of size 3 x 3. POOL layers will perform max pooling over a 2 x 2 window with a 2 x 2 stride and activation function (ACT) used in this architecture is RELU [15].

The network architecture is detailed in Table 1, where the initial input image size is assumed to be 32 x 32 x 3 and will be training on CIFAR-10 dataset.

Table 1: Summary of miniVGGNet architecture. Output volume sizes are included for each layer, along with convolutional filter size/pool size.

Layer Type	Output Size	Filter Size/ Stride
INPUT IMAGE	32 x 32 x 3	
CONV	32 x 32 x 32	3 x 3, K=32
ACT	32 x 32 x 32	
CONV	32 x 32 x 32	3 x 3, K=32
ACT	32 x 32 x 32	
POOL	16 x 16 x 32	2 x 2
DROPOUT	16 x 16 x 32	
CONV	16 x 16 x 64	3 x 3, K=64
ACT	16 x 16 x 64	
CONV	16 x 16 x 64	3 x 3, K=64
ACT	16 x 16 x 64	
POOL	8 x 8 x 64	2 x 2
DROPOUT	8 x 8 x 64	
FC	512	
ACT	512	
DROPOUT	512	
FC	10	
SOFTMAX	10	

3. Results and Discussion

The result for the training and validation will be shown in the graph of training loss and accuracy and validation loss and accuracy. The percentage of evaluation accuracy will be shown from the python terminal. The network was trained on miniVGGNet architecture on CIFAR-10 dataset and the result shown in Fig.4.

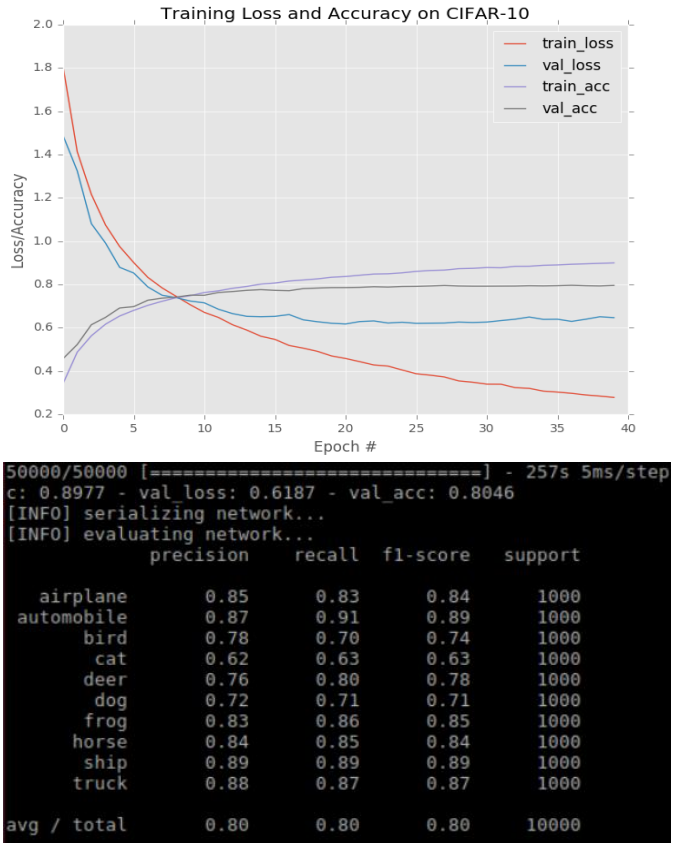


Fig. 4: miniVGGNet trained on CIFAR-10

Fig. 4 shows miniVGGNet network architecture trained on CIFAR-10 data. The value of the classification accuracy is 80%, and the loss accuracy plot shows that the loss starts to increase past epoch 30, indicating that the network is overfitting to the training data. There is show the validation accuracy has become quite saturated by epoch 25.

The performance of the network can be improved using hyper parameter tuning technique. This technique was discussed in the next section of this paper.

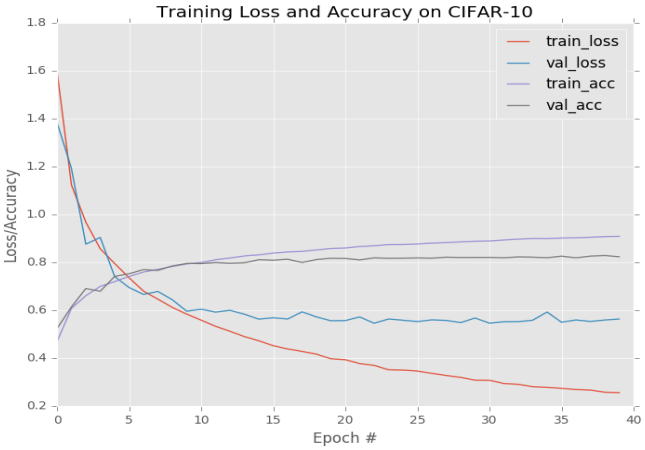
3.1. Hyper Parameter Tuning

The machine learning model can require different constraints, learning rates or weights to extrapolate different data patterns. These measures are called hyper parameters, and these parameters need to be tuned in order to improve the model accuracy and optimally solve the machine learning problem. Hyper parameter tuning finds an ordered list of elements of hyper parameter - the optimal model which minimizes a predefined loss function on given independent data. In this paper, the value of the learning rate decay factor was adjusted, and batch normalization layers were applied to the networks to compares the performance of the trained network.

3.1.1. Batch Normalization

Introduced by Ioffe and Szgedy in their 2015 paper, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [14], batch normalization layers (BN), are used to normalize the activations of a given input volume before passing it into the next layer in the network. BN layer usually inserted after an activation layer.

BN is extremely effective at reducing the number of epochs it takes to train a neural network. It also has the added benefit of helping “stabilize” training, allowing for a larger variety of learning rates and regularization strengths.




```
50000/50000 [==============================] - 456s 9ms/step
c: 0.9032 - val_loss: 0.5626 - val_acc: 0.8223
[INFO] serializing network...
[INFO] evaluating network...
              precision    recall  f1-score   support

    airplane       0.86      0.83      0.85      1000
  automobile       0.93      0.89      0.91      1000
        bird       0.82      0.68      0.75      1000
         cat       0.68      0.65      0.66      1000
        deer       0.76      0.84      0.80      1000
         dog       0.71      0.77      0.74      1000
        frog       0.85      0.88      0.86      1000
       horse       0.87      0.86      0.86      1000
        ship       0.90      0.90      0.90      1000
       truck       0.86      0.91      0.88      1000

avg / total       0.82      0.82      0.82     10000
```

Fig. 5: miniVGGNet trained on CIFAR-10 with batch normalization

Based on the result shown in Fig. 5, the miniVGGNet implementation with batch normalization is more stable while both loss and accuracy start to flatline past epoch 35. After training complete, the classification accuracy increase by 2% from 80% without batch normalization (Fig. 4) to 82% with batch normalization implementation.

3.1.2. Learning Rate Decay Factor

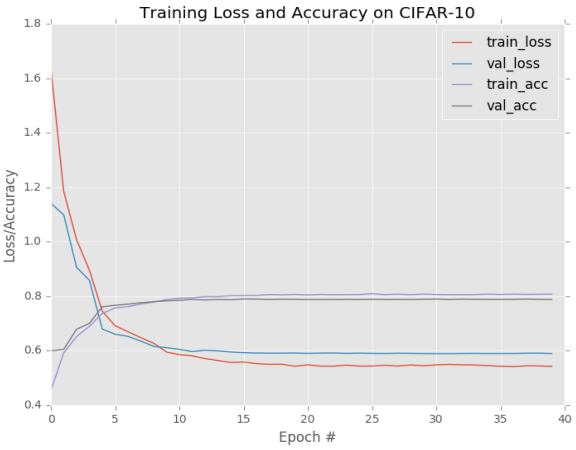
Adding decay to the learning rate will help alleviate the effects of overfitting when training the network. By adjusting the learning rate value on an epoch-to-epoch basis, the loss will reduce; accuracy will increase, and the total amount of time takes to train a network will decrease. Learning rate α controls the "step" make along the gradient. Larger values of α imply that it's taking bigger steps while smaller values of α will make a tiny step.

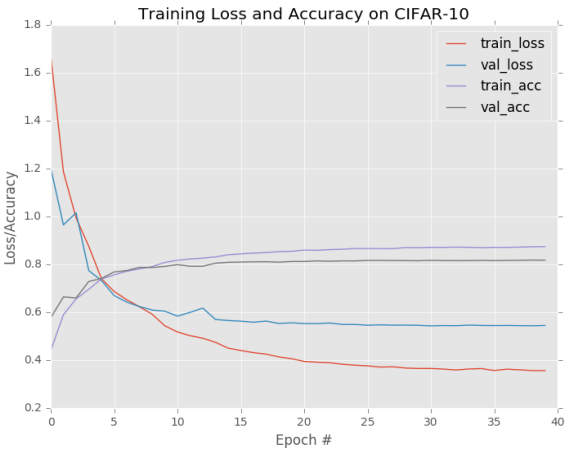
The Keras library provides LearningRateScheduler class to define a custom learning rate function and then have it automatically applied during the training process. This function should take the epoch number as an argument and then compute the desired learning rate based on a define function. In this paper, a piecewise function that will drop the learning rate by a certain factor F after every D epoch was defined. The equation will look as in (1):

$$\alpha_{E+1} = \alpha_1 \times F^{(1+E)/D} \quad (1)$$

Where α_1 is the initial learning rate, F is the factor value controlling the rate in which the learning rate drops, D is the "drop every" epochs value, and E is the current epoch. The larger the value of factor F is, the slower the learning rate will decay. Conversely, the smaller the factor F is, the faster the learning rate will decrease.

To evaluate the effect of the factor has on learning rate scheduling and overall network classification accuracy, three drop factors value: 0.25, 0.5 and 0.75, will be evaluating respectively.






```
50000/50000 [==============================] - 2345s - lo
0 - val_loss: 0.6078 - val_acc: 0.7894
[INFO] serializing network...
[INFO] evaluating network...
             precision    recall  f1-score   support

   airplane       0.84      0.80      0.82      1000
 automobile       0.90      0.89      0.90      1000
       bird       0.72      0.65      0.69      1000
        cat       0.63      0.55      0.59      1000
       deer       0.74      0.78      0.76      1000
        dog       0.68      0.72      0.70      1000
       frog       0.79      0.89      0.84      1000
      horse       0.84      0.83      0.84      1000
       ship       0.89      0.90      0.89      1000
      truck       0.84      0.88      0.86      1000

avg / total       0.79      0.79      0.79     10000
```

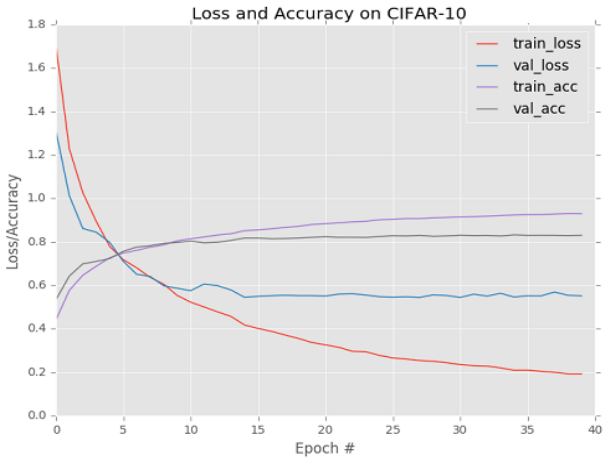
(a)

```
50000/50000 [==============================] - 1496s - loss
5 - val_loss: 0.5640 - val_acc: 0.8108
[INFO] serializing network...
[INFO] evaluating network...
             precision    recall  f1-score   support

   airplane       0.84      0.83      0.84      1000
 automobile       0.91      0.90      0.90      1000
       bird       0.75      0.70      0.72      1000
        cat       0.67      0.63      0.65      1000
       deer       0.75      0.81      0.78      1000
        dog       0.73      0.72      0.73      1000
       frog       0.81      0.89      0.85      1000
      horse       0.86      0.84      0.85      1000
       ship       0.90      0.90      0.90      1000
      truck       0.87      0.89      0.88      1000

avg / total       0.81      0.81      0.81     10000
```

(b)




```
50000/50000 [==============================] - 451s 9ms/step
ss: 0.5523 - val_acc: 0.8308
[INFO] evaluating network...
             precision    recall  f1-score   support

   airplane       0.86      0.85      0.85      1000
 automobile       0.93      0.90      0.91      1000
       bird       0.80      0.73      0.76      1000
        cat       0.68      0.69      0.69      1000
       deer       0.78      0.81      0.80      1000
        dog       0.76      0.76      0.76      1000
       frog       0.83      0.90      0.86      1000
      horse       0.88      0.86      0.87      1000
       ship       0.91      0.91      0.91      1000
      truck       0.89      0.90      0.89      1000

avg / total       0.83      0.83      0.83     10000
```

(c)

Fig. 6: (a) miniVGGNet on CIFAR-10 with learning rate decay (factor=0.25), (b) miniVGGNet on CIFAR-10 with learning rate decay (factor=0.5), (c) miniVGGNet on CIFAR-10 with learning rate decay (factor=0.75)

Based on the result shown in Fig. 6 (a), the network obtains only 79% classification accuracy with learning rate decay factor 0.25. The learning rate is dropping quite aggressively after epoch 15, meaning that the network is taking a tiny step along the loss landscape. Notice the accuracy/loss plotting of the network using a faster learning rate drop with factor 0.25 stagnate past epoch 15 as the learning rate is too small.

As the value factor increase to 0.5 and 0.75, the learning rate will decay slowly, the classification accuracy increase from 81% to 83%. Notice the accuracy/loss plotting of the network continues to learn after epoch 25-30 until stagnation occurs.

Table 2: Summary of the result showing the classification accuracy with and without batch normalization

	Without BN	**With BN**
Classification Accuracy	80%	82%

Table 3: Summary of the result showing the classification accuracy by applying learning rate decay factor

	Learning Rate Decay Factor		
	0.25	**0.5**	**0.75**
Classification Accuracy	79%	81%	83%

Table 2 shows the result by applying batch normalization to the network. The classification accuracy will increase by 2% when applying BN to the network compared without BN. Table 3 shows the results when applying the learning rate decay factor. The larger factor of learning rate decay will increase the value of classification accuracy.

4. Conclusion

In this paper, the analysis of the performance of convolutional neural network (CNN) for image classifying using miniVGGNet architecture on CIFAR-10 dataset was presented. By applying batch normalization layer to the network, the classification accuracy of the models increased. Applying learning rate decay schedulers to the network architectures will help to prevent overfitting and allows the model to obtain significantly higher classification accuracy.

Acknowledgement

The author would like to express their deep gratitude toward the Control and Signal Processing group Faculty of Engineering University Putra Malaysia for their support and encouragement during this work. This research is supported by Putra Graduate Initiative (IPS) grant.

References

[1] M Manoj krishna1, M Neelima, M Harshali, M Venu Gopala Rao," Image classification using Deep learning", International Journal of Engineering & Technology, 7 (2.7) (2018), pp.614-617. https://www.researchgate.net/publication/325116934

[2] T. Ojala, M. Pietikainen, and T. Maenpaa. "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.7 (2002), pp: 971–987

[3] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Washington, DC, USA: IEEE Computer Society, - Vol.1 – Vol.01 (2005), pp: 886–893

[4] Ethan Rublee et al. "ORB: An Efficient Alternative to SIFT or SURF", Proceedings of the 2011 International Conference on Computer Vision. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp:2564–2571.

[5] Nguyen, K, Fookes, C., & Sridharan, S. Improving deep convolutional neural networks with unsupervised feature learning. Proceedings - International Conference on Image Processing, ICIP, (2015), https://doi.org/10.1109/ICIP.2015.7351206

[6] Guo G., Wang H., Bell D., Bi Y., Greer K., "KNN Model-Based Approach in Classification", Meersman R., Tari Z., Schmidt D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Vol.2888 (2003). pp: 986-996, https://doi.org/10.1007/978-3-540-39964-3_62

[7] Anagnostopoulos, G.C. "SVM-Based Target Recognition From Synthetic Aperture Radar Images using Target Region Outline Descriptors". Nonlinear Analysis: Theory, Methods & Applications, Vol.71, Issue.12, (2009), pp:2934–2939, https://doi.org/10.1016/j.na.2009.07.030
[8] Yoshua Bengio, "Learning Deep Architectures for AI", Foundations and Trends® in Machine Learning, Vol.2, (2009), pp: 1-127. http://dx.doi.org/10.1561/2200000006
[9] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, Vol. 61, (2015), pp. 85 – 117, https://doi.org/10.1016/j.neunet.2014.09.003
[10] D. P. Sudharshan, S Raj, "Object Recognition in Images using Convolutional Neural Network", 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp: 718-722, https://doi.org/10.1109/ICISC.2018.8398893
[11] Safiyah, R. D., Rahim, Z. A., Syafiq, S., Ibrahim, Z., & Sabri, N, "Performance Evaluation for Vision-Based Vehicle Classification Using Convolutional Neural Network ",International Journal of Engineering and Technology(UAE), Vol.7, (2018), pp: 86-90. https://doi.org/10.14419/ijet.v7i3.15.17507
[12] Krizhevsky, A.; Sutskever, I.; Hinton, G.E, "Imagenet Classification with Deep Convolutional Neural Networks", Proceedings of the Neural Information Processing System (NIPS), Harrahs and Harveys,Lake Tahoe, NV, USA, Vol.2 (2012), pp: 1097-1105.
[13] Simonyan, K., Zisserman, A, "Very Deep Convolutional Networks for Large-Scale Image Recognition", Conference paper at ICLR 2015, arXiv:1409.1556, https://arxiv.org/pdf/1409.1556.pdf
[14] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: CoRR abs/1502.03167 (2015), pp: 189-193 URL: http://arxiv.org/abs/1502.03167
[15] Adrian Rosebrock, "Deep Learning for Computer Vision with Python", PyimageSearch, (2017), pp: 17-319