Tiny Image Classification using Four-Block Convolutional Neural Network

Mohsin Sharif[1], Asia Kausar[2], JinHyuck Park[3], Dong Ryeol Shin[4]
Department of Electrical and Computer Engineering
Sungkyunkwan University,
Suwon, South Korea
E-mail: {mohsin[1], asiakausar[2], jjangdol[3], drshin[4]}@skku.edu

***Abstract*—The task of classifying images into predefined classes is a major problem in computer vision and artificial intelligence. Deep neural network such as Convolutional Neural Network (CNN) have shown great success in large-scale dataset of high resolution image classification. Here, it is important to note that most real time images may not have high resolution. With the increasing demand of surveillance camera-based applications, the low resolution images are major problem. To overcome this problem, we propose two Four-Block CNN model; one with four-layers and the other one with three-layers. Our proposed Four-block four-layer CNN model contains four convolution layers, first three layers contains 3×3 kernel size with stride-1 and fourth layer used with stride-2 for dimensionality reduction.The Four-block three-layer has two convolutional layers with stride-1 and third layer with stride-2. We trained our model on CINIC-10 and CIFAR-10 datasets having low-resolution images. For taking average of whole feature map we are using Global Average Pooling layer as a classifier in both models. To reduce training time complexity, we use non-saturating neurons. Overfitting problem has been addressed by dropout and batch normalization methods. On the validation data, we achieved the best accuracy of 81.62%, 92.21% for CINIC-10 and CIFAR-10 respectively, using Four–block four–layer model.**

***Index Terms*—Low-Resolution Images, Convolutional Neural Network, CINIC-10, Multi-class image classication, Batch Normalization**

I. Introduction

In fields such as traffic signs, face recognition and scene classification, classifying images into predefined classes, is a major problem. In real time, images are not always in high resolution. In some applications such as surveillance systems, black box in cars, etc. usually images have low resolutions and have a lot noise. Convolutional Neural Networks (CNNs) use several set of filters to perform the final classification. Therefore, it is difficult to predict the outcome of a low resolution image input in an intuitive way, as the classification accuracy largely depends on the model architecture and the nature of the image.

Because of their automatic feature extraction, deep neural networks have succeeded exceedingly for object detection and image classification in computer vision. Also, due to recent improvement in high performance computing system like graphic processing units (GPU) along with publicly available large scale image datasets, CNN has been used with great interest [1], [2] [3]. The ImageNet Large Scale Visual Recognition challenge (ILSVRC) is the current test-bed for image classification and pattern recognition task [4]. Accuracy can be further improved by increasing the depth of network by varying numbers of layers [5], [6], [7], [8].However, by increasing number of layers in CNN, the number of parameters increase rapidly. Although deeper CNN give us better accuracy but overfitting is severe problem in these networks [9].

Solutions have been provided by researchers to overcome overfitting problem, by replacing fully connected layers with Global Average Poling (GAP) layer [10], by minimizing cross–covariance of hidden activation [11], and using dropout regularization method [12]. If the network stops learning features during training phase, Batch Normalization (BN) may be used to improve the learning process [3]. Also in recent years there has been a shift from traditional (Convolutional layers, pooling layers and fully connected layers) to pure convolutional layers architecture [10], [13].

Considering low-resolution tiny image classification and overfitting problem, we propose a four-block CNN architecture, where each block contains four convolutional layers, first three with stride-1 and last layer of each block has stride-2 for dimensionality reduction. To reduce the complexity, the GAP layer has been used as a classifier. Since features may not be extracted efficiently from tiny images, a network is more likely to over-fit. We use Dropout and BN, to overcome overfitting. Furthermore, to assess the performance of our proposed model we are using newly introduced CINIC-10 dataset [14], which contains 10 classes of different categories. In addition, proficient GPU is being used in our experiment to further increase efficiency. For tiny image classification, according to the best of our knowledge, our proposed model achieves average accuracy of 81.62%, which is 7.1% and 3.3% greater with comparison, to previously proposed ResNet20 v1, ResNet29 v2, respectively[5].

The remainder of this paper is organized as follows. In section II, reviews related work on tiny image classication .In Section III, we present the details of CINIC-10 dataset used in our experiment. We discuss the details of our proposed CNN model in section IV. Section V presents our simulation results. The discussion of the results is given in Section VI.

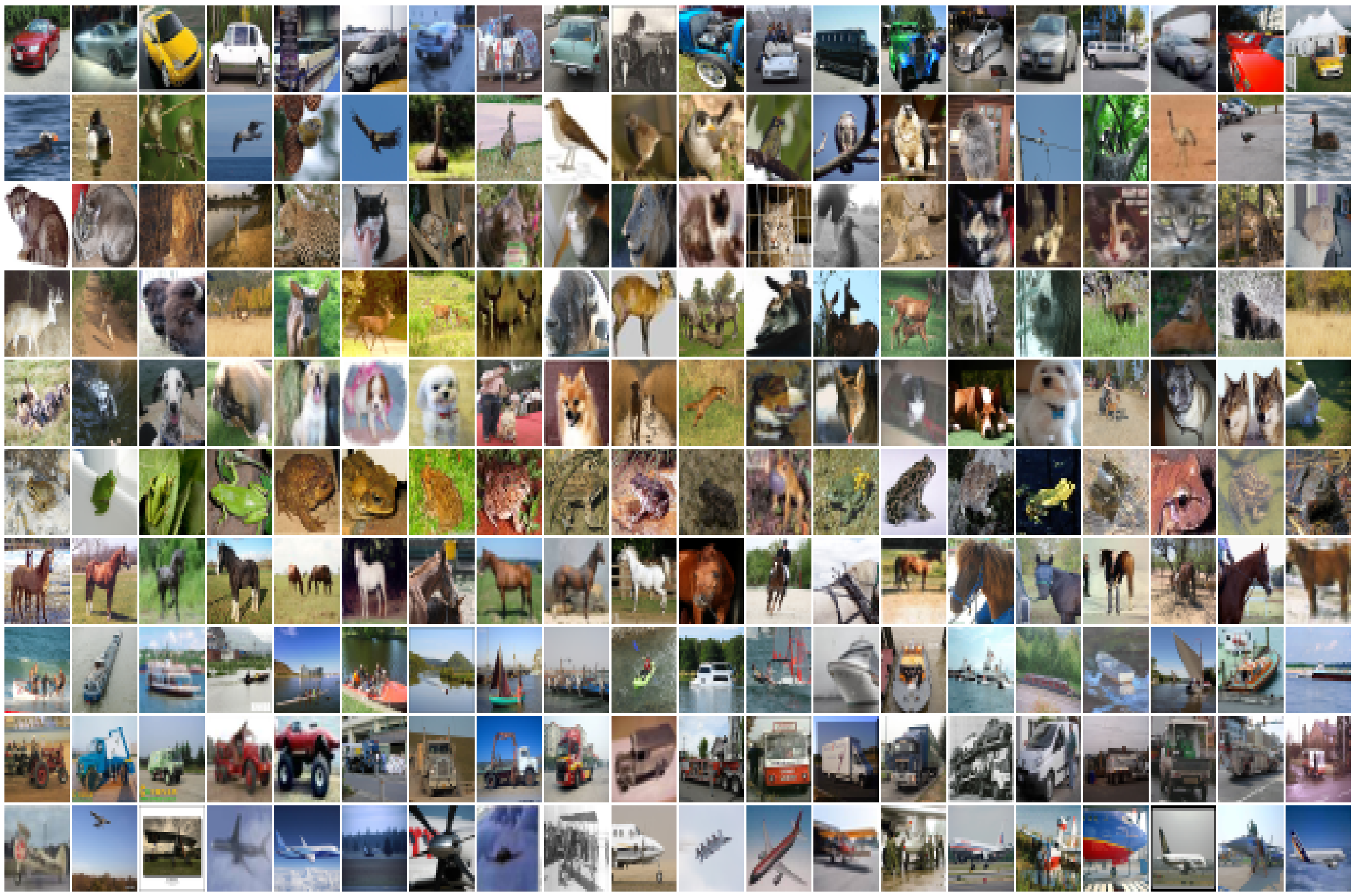

Figure 1: CINIC-10 Dataset Pictorial Representation

II. Related Work

Recently, CNN have been used for wide range of visual perception tasks, such as image classification, object detection and activity recognition etc. Le-Net-5 [3], CNN has a standard architecture (convolutional layers, pooling layers and followed by one or more fully-connected layers) [3], [4], [5], [6]. Variants of this traditional design are dominant in the image classification tasks and have best performance on datasets such as MINIST, CIFAR 10 and CIFAR 100, and ImageNet classification challenge [4], [9], [13], [15].

With a rapid increase in applications related to surveillance, black box etc. [16], [17], low-resolution image classification has gained significance. In [18], CNN architecture has been proposed using fully connected layer as a classifier for tiny images. They down sampled images from ImageNet dataset from 256×256 into 64×64 pixel size, and evaluate performance of their proposed network, achieving test error of 56.7%. To reduce overfitting, they used dropout and $l2$ weight decay. In [19], authors used same sized down-sampled images from ImageNet, and used variants of Inception V3 and VGGNet models and achieves an accuracy of between 40% and 45%. To reduce overfitting they used BN in their work. ResNet proposed by [5], has deeper network with multiple convolutional layers for classification on datasets such as CIFAR-10 and COCO object detection and achieved 21.81% error rate.

III. Dataset: CINIC-10

The dataset we examine is CINIC-10, it is an augmented extension of CIFAR-10 by combining with images down-sampled from ImageNet dataset [14]. The reason behind formation of CINIC dataset to overcome the gap between ImageNet and CIFAR datasets as both of them are widely used by many researchers to train their models. ImageNet dataset has its own flaws as it is a clumsy dataset. The images are large enough to train on Neural Networks as it takes several days on single training [7].

CINIC-10 is introduced to overcome the shortcomes of existing datasets in terms of pixels, image size and information contains by images. At first they down-sampled ImageNet dataset and converted it into 32×32 sized images then by applying data augmentation on CIFAR 10, they increased the number of images in dataset. By combining both datasets images, final version of CINIC developed, which contains $270,000$ images in total. Each subset contains 10 classes same as CIFAR 10 shown in figure 1. For our experiment we used $180,000$ images for training and remaining $90,000$ images for testing data.

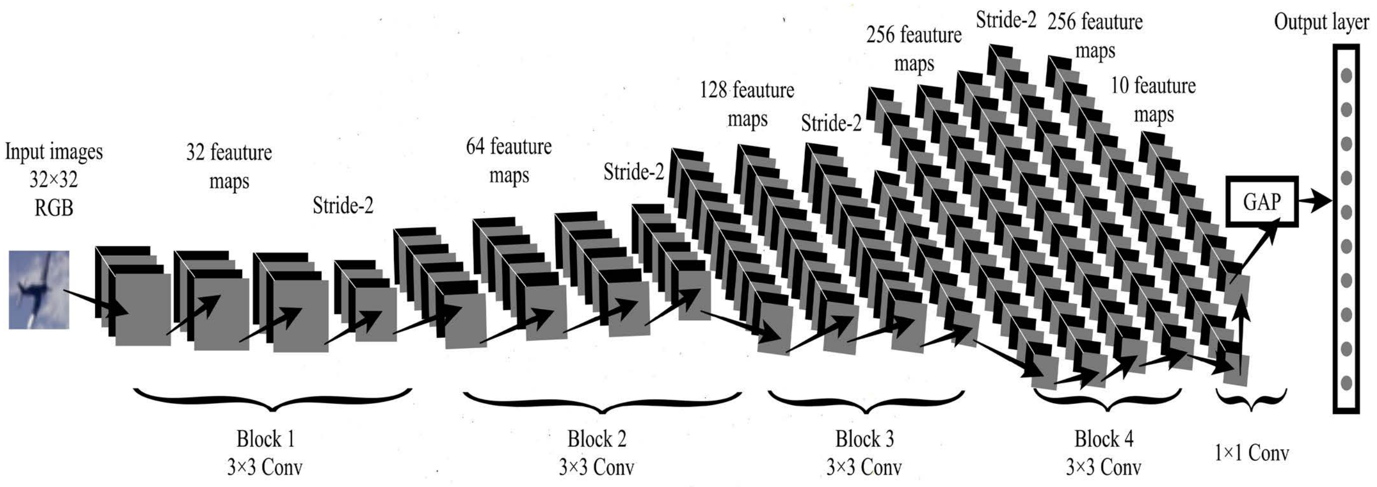

Figure 2: Architecture of Four-Block CNN with Global Average Pooling Layer.

For comparison, we also demonstrate our classification performance on CIFAR-10 [20] dataset contains $60,000$ natural images, these images categories into 10 different classes. It is composed of $50,000$ training images in total, and $10,000$ testing images. Each image is an RGB image of size 32×32. Both datasets contains same sized images 32×32, categorized into 10 different classes but CINIC-10 contains $270,000$ images.

IV. Proposed Model Architecture

A block diagram of the proposed architecture is shown in figure 2. Convolutional network perceives images as volume of three dimensions as height, width and depth. In CINIC-10 images are of size $32 \times 32 \times 3$ (32 wide, 32 high and 3 color channel). Convolutional layer extracts feature maps by linear convolutional filters followed by nonlinear activation function (ReLU, Sigmoid, and Tanh)[21]. For non–linear transformation of input we use ReLU activation function($\sigma(\cdot)$) [1].

We proposed two Four-Block CNN architecture as shown in I, one with four-layers and second one is with three–layer convolutional layers. In Four-block four-layer, Each block consists of four convolutional layers, first three convolutional layers has of 3×3 kernel size with stride-1, But in every fourth layer stride-2 is being used for reducing dimensions. In Four-Block three–layer model, each block consists of three convolutional layers, first two with stride-1 and the last layer of every block contains stride-2. Stride is the size of steps the convolutional filters move each time, as the kernel is sliding the input [13]. To reduce the complexity, Global Average pooling (GAP) layer has been used as classifier. The idea is to reduce the filter size by taking average of whole feature map is introduced in [10], where dimension from $h \times w \times d$ are reduced to $1 \times 1 \times d$. The output layer uses softmax activation function $(\sigma(z)j)$[2].

[1] $\sigma(\cdot) = ReLU(x) = max(0, x)$

[2] $\sigma(z)j = \frac{e^{zj}}{\sum_{j=1}^{k} e^{zk}} \; for \; j = 1, ..., k$

Dropping out the nodes in the network is a simple and effective regularization method. In our model we used dropout at the end of each block[12]. First block, second and third block , and fourth block have a dropout rate of 0.3, 0.6 and 0.3, respectively. During training, using dropout in our experiments, we found that network stopped learning features. We used BN [15], to overcome this problem by normalizing the input. *It is our finding, using BN before or after activation function in each block, does not have a significant impact.*

Additionally, we have used Adaptive Moment Estimation (Adam) optimizer to update network weights. Adam is proved to be more computationally efficient and well suitable for large scale data [22].

V. Performance Evaluation

To train our model Tensor flow and Keras is being used in this experiment. Tensor Flow [23] has become a preferred deep learning library at Uber for a variety of reasons. To start, the framework is one of the most widely used open source frameworks for deep learning, which makes it easy to on-board new users. It also combines high performance with an ability to tinker with low-level model detailsfor instance, we can use both high-level APIs, such as Keras [24], and implement our own custom operators using NVIDIAs CUDA toolkit. Additionally, Tensor flow has end-to-end support for a wide variety of deep learning use cases, from conducting exploratory research to deploying models in development of many computer vision and image processing tasks such as image classification, robotics, traffic sign detection and many more.

On the other hand, Keras [24] is an open source library, written in python and can run on Tensor flow and Theano. It allows easy and fast prototyping as it is user friendly and more extensible. It gives support for both, Convolutional neural network and recurrent neural network as well as for combination of both. It can run on CPU and GPU but in our

Table I
Four-Block Convolutional Neural Network (CNN) with four and three Convolutional layer architecture.

	CNN with 4 Convolutional Layer		**CNN with 3 Convolutional Layer**	
	Layer Type	Dimensions	Layer Type	Dimensions
Block 1	Convolutional Layer	$32 \times 3 \times 3$	Convolutional Layer	$48 \times 3 \times 3$
	Convolutional Layer	$32 \times 3 \times 3$	Convolutional Layer	$48 \times 3 \times 3$
	Convolutional Layer	$32 \times 3 \times 3$		
	Convolutional Layer Stride 2	$32 \times 3 \times 3$	Convolutional Layer Stride 2	$48 \times 3 \times 3$
Block 2	Convolutional Layer	$64 \times 3 \times 3$	Convolutional Layer	$96 \times 3 \times 3$
	Convolutional Layer	$64 \times 3 \times 3$	Convolutional Layer	$96 \times 3 \times 3$
	Convolutional Layer	$64 \times 3 \times 3$		
	Convolutional Layer Stride 2	$64 \times 3 \times 3$	Convolutional Layer Stride 2	$96 \times 3 \times 3$
Block 3	Convolutional Layer	$128 \times 3 \times 3$	Convolutional Layer	$192 \times 3 \times 3$
	Convolutional Layer	$128 \times 3 \times 3$	Convolutional Layer	$192 \times 3 \times 3$
	Convolutional Layer	$128 \times 3 \times 3$		
	Convolutional Layer Stride 2	$128 \times 3 \times 3$	Convolutional Layer Stride 2	$192 \times 3 \times 3$
Block 4	Convolutional Layer	$256 \times 3 \times 3$	Convolutional Layer	$384 \times 3 \times 3$
	Convolutional Layer	$256 \times 3 \times 3$	Convolutional Layer	$384 \times 3 \times 3$
	Convolutional Layer	$256 \times 3 \times 3$		
	Convolutional Layer Stride 2	$256 \times 3 \times 3$	Convolutional Layer Stride 2	$384 \times 3 \times 3$
	Convolutional Layer	$256 \times 1 \times 1$	Convolutional Layer	$384 \times 1 \times 1$
	Convolutional Layer	$10 \times 1 \times 1$	Convolutional Layer	$10 \times 1 \times 1$
	Global Average Pooling			
	Softmax (Activation)	10	Softmax (Activation)	10

experiment we are using GPU for training our model over CINIC-10 and CIFAR-10 dataset.

A. Experimental setup

Anticipated for experiments on CINIC-10 and CIFAR-10. We first train our proposed Four-Block CNN with GAP layer as explained before. And to make comparison we train same model with less number of convolutional layers. Our comparison model contains three convolutional layers in each block, with increased number of feature maps. Model description of both architectures shown in table I.

B. Classification Results

We have made different attempts by varying the number of layers to check the performance of our proposed CNN architecture as shown in table 1. Through experiments we found that by increasing the depth of network with less number of feature maps, we only cannot get better accuracy but also able to reduce the computational complexity of network. The classification results shows that CNN model with less number of convolutional layers does not perform very well, as the computational complexity increased from 2.8 to 4.5 million parameters and accuracy reduced from 81.62% to 80.77% for CINIC-10 dataset and for CIFAR-10 dataset accuracy reduced from 92.21% to 88.03% . Classification results, along with training, validation accuracy and difference between total number parameters of both models, for both CINIC-10 and CIFAR-10 datasets can be observed in table II.

It is clear that both architecture performs equally well in terms of training and validation accuracy as well as training and validation loss of both models on both datasets. In addition, to estimate the error each time the model weights are updated, we first set learning 0.001, after 80 and120 epochs we divided learning rate by 10. As can be observed in figure 3, 6 and 4, 7, after 120 epochs we have reached at remarkable accuracy of 81.62% and 92.21% for both CINIC-10 and CIFAR-10 respectively. We didn't applied any preprocessing on dataset as our main purpose was to classify the low-resolution images with partially connected layers.

We have made another comparison using ResNet version-1 and version-2 [5] which gives us slightly low validation accuracy as compared to our proposed models. For training we have used CINIC-10 dataset and got 74.6% and 78.28% validation accuracy for ResNet Version-1 and version-2 respectively, shown in Table II and figure 5,8.

Table II
Classification results on CINIC-10 and CIFAR-10.

	Data Set	Training Accuracy	Validation Accuracy	Total no of Parameters
CNN with 4 Convolutional Layer	CINIC-10	86.74%	**81.62%**	2,817,042
	CIFAR-10	97.48%	**92.21%**	
CNN with 3 Convolutional Layer	CINIC-10	87.07%	80.77%	4,559,242
	CIFAR-10	98.25%	88.03%	
ResNet20 V1	CINIC-10	99.83%	74.91%	274,442
ResNet29 V2	CINIC-10	**99.97%**	78.28%	849,002

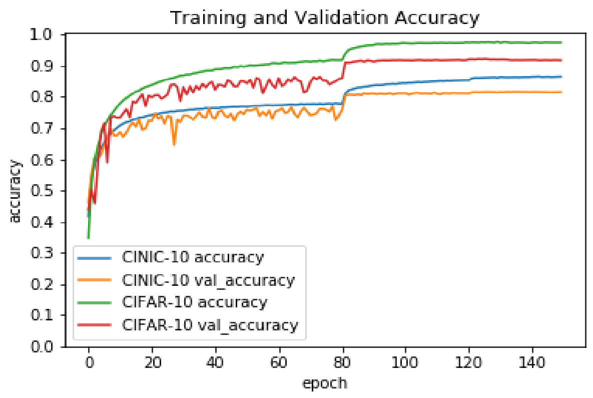

Figure 3: Accuracy at each epoch of four layer architecture

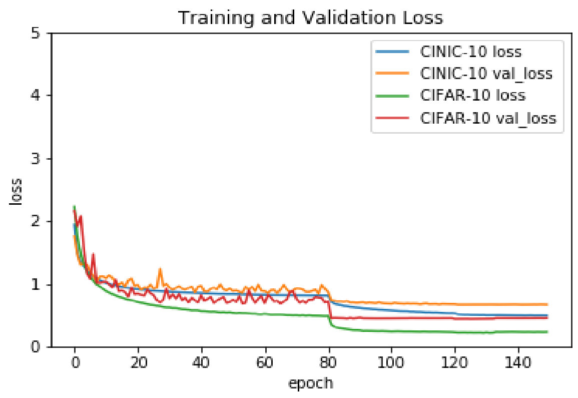

Figure 6: Loss at each epoch of four layer architecture

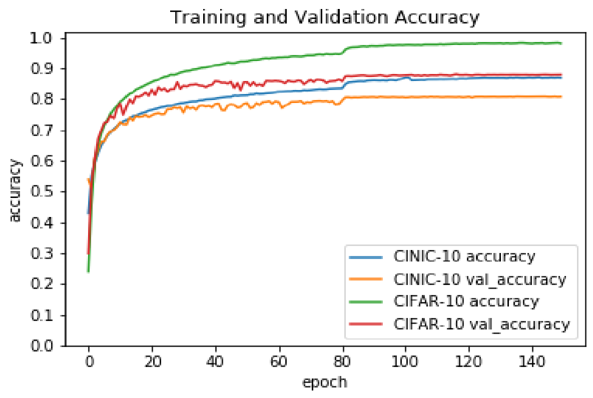

Figure 4: Accuracy at each epoch of three layer architecture

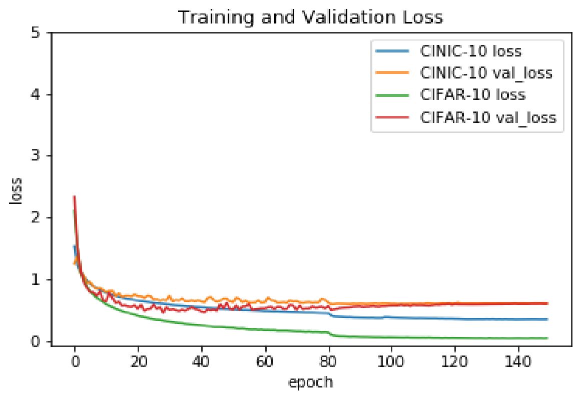

Figure 7: Loss at each epoch of three layer architecture

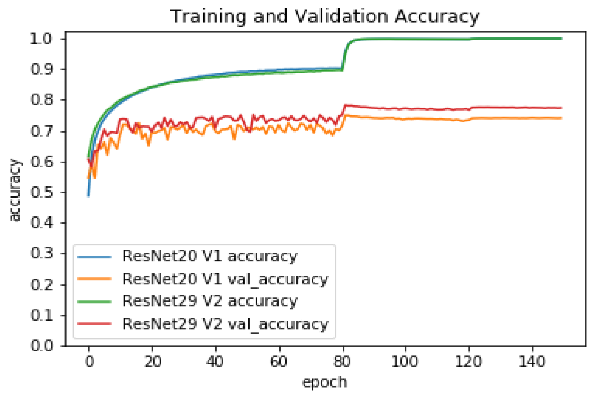

Figure 5: Accuracy at each epoch of ResNet20 V1 and ResNet29 V2 architecture

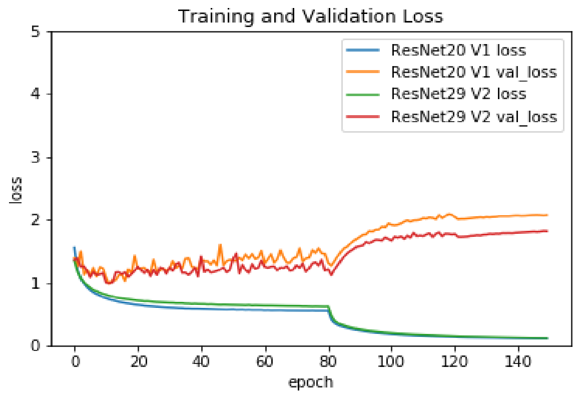

Figure 8: Loss at each epoch of ResNet20 V1 and ResNet29 V2 architecture

VI. Conclusion

In this paper, we proposed two method to classify low resolution images using Four-block convolutional neural network; one with four-layers and the other one with three-layers. Partially connected convolutional layers have been implemented for feature extraction. For dimensionality reduction, some convolutional layers with stride-2 are used. Additionally, to overcome overfitting problem dropout and batch normalization was used. Global Average pooling layer (GAP) is implemented as a classifier. On the validation data, we achieved the best accuracy of 81.62%, 92.21% for CINIC-10 and CIFAR-10 respectively, using Four–block four–layer model. In the future, we shall apply different types of noise in low resolution images to further improve the classification performance.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03032855).

References

[1] Henggang Cui, Hao Zhang, Gregory R Ganger, Phillip B Gibbons, and Eric P Xing. Geeps: Scalable deep learning on distributed gpus with a gpu-specialized parameter server. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 4. ACM, 2016.

[2] Jeff A Stuart and John D Owens. Multi-gpu mapreduce on gpu clusters. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 1068–1079. IEEE, 2011.

[3] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[9] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[10] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[11] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[13] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[14] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Shinichi Tamura, Hideo Kawai, and Hiroshi Mitsumoto. Male/female identification from 8×6 very low resolution face images by neural network. *Pattern recognition*, 29(2):331–335, 1996.

[17] Rizwan Ahmed Khan, Alexandre Meyer, Hubert Konik, and Saida Bouakaz. Framework for reliable, real-time facial expression recognition for low resolution images. *Pattern Recognition Letters*, 34(10):1159–1168, 2013.

[18] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2015.

[19] Alexei Bastidas. Tiny imagenet image classification, 2017.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[24] https://keras.io/.