International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume 14 Issue 03 Year 2023

Optimisation of Deep Convolutional Neural Network with the Integrated Batch Normalization and Global pooling

Ms. Priyanga k. k
Department of Computer Science, SRMIST, Kattankulathur, Tamil Nadu
pk9856@srmist.edu.in
Dr. S Sabeen
Department of Computer Science, Christ College (Autonomous), Irinjalakuda, Kerala

Article History

Received: 01 March 2023
Revised: 18 April 2023
Accepted: 17 May 2023

Abstract

Deep convolutional neural networks (DCNN) have made significant progress in a wide range of applications in recent years, which include image identification, audio recognition, and translation of machine information. These tasks assist machine intelligence in a variety of ways. However, because of the large number of parameters, float manipulations and conversion of the machine terminal remains difficult. An optimisation process is initiated to address this issue by adjusting the neural network's convolution characteristics, which minimises information loss and enhances its overall performance. Minimisation of the convolution function addresses the optimisation issues. Initially, batch normalisation is completed, and instead of lowering neighbourhood values, a full feature map is minimised to a single value using the global pooling approach. Traditional convolution is split into depth and pointwise to decrease the model size and calculations. The optimised convolution-based DCNN's performance is evaluated with the assistance of accuracy and occurrence of error. The optimised DCNN is compared with the existing state-of-the-art techniques, and the optimised DCNN outperforms the existing technique.

Keywords: Neural Network, Weight Optimisation, Pooling, Deep Learning, Convolution, Normalisation

1. Introduction

Deep Neural Networks (DNN) is a mathematical function that exploits the functioning of the human brain to think and interpret a suitable mapping. DNN has a wide range of applications in real-life problems [1]. The classification success of DNN is solely based on empirical results, and its theoretical explanation is unknown. Hence the classification accuracy needs to be examined to characterise the ability of DNN. Some studies question the generalisation ability of DNN and experimentally show its tendency to misclassify untrained data. This needs further investigation with a reliable performance measure [2]. The standard measures to evaluate the error in the classification results of DNN are accuracy, mean squared error, mean absolute error and the sum of squared error [3]. But these measures could be more reliable in computing the generalisation error [4].

Deep Neural Networks (DNNs) have shown effectiveness in various challenging applications, including semantic segmentation, object recognition, voice synthesis, and image classification [5], [7]. At a high computational cost, recent neural network models with hundreds of parameters demonstrated human-level ability. DNNs that have a lot of parameters must take a while to train. The

deployment of these enormous networks is equally challenging in embedded environments. Bandwidth and memory become limiting factors when transferring data and weights between Compute Units (CUs) [7]-[10].

Over-parameterisation is a characteristic of neural networks in which extraneous neurons do not improve the accuracy of the results [11]. Usually, this repeat may be cut down with little to no accuracy loss. The three components of network structure are novel elements, network architecture analysis, and knowledge extraction. New aspects include the development of effective building blocks such as inception blocks, residual blocks, and separable convolution [12]. Network components also consist of the many types of connections between levels. Deep neural networks are completely coupled when there are N2 connections between neurons. Only connections in the forward path are considered on feedforward layers, which lowers the number of connections. This reduces the number of linkages overall to N [13].

A neural Network is an information-processing system like the human brain's nervous system [14]. It consists of neurons called nodes that act as the information-processing cells in Artificial Neural Network. The nodes are interconnected with synapses called weights [15]. Like dendrites in the nervous system, the input nodes take the information from different sources and pass it to the neighbouring neurons called nodes through these weighted connections [16]. Each node accumulates these signals and activates based on the threshold connected. The information is processed by adjusting the weights until the neural network recognises a specific pattern [17]. The remainder of the article is emphasised: the related works are detailed in Section 2, the proposed methodology is given in Section 3, acquired results are given in Section 4, and the article is concluded in Section 5.

2. Related Works

The gradient Descent (GD) method, especially the Stochastic Gradient Descent method (SGD), is an effective optimisation algorithm for training DNN [18]. For successful learning, a batch-wise random selection of data should not affect the stability of the learning algorithm in determining the generalisation gap. Since the complexity of the learning function increases the generalisation gap, there is a need for stability analysis in the performance of DNN [19]. Not only the complexity and organisation of data but network structure also plays a vital role in determining the stability and performance of DNN [20].

Arguments on a network's suitable depth and width are widely discussed open problems [21]-[25]. This reveals the necessity of solid mathematical concepts to configure the network structure according to the size and dimension of the input space. Scaling attributes can lead to worse results [26], [27]. Linearly dependent inputs can cause degeneration, slowing the learning process [28], [29]. Convergence is faster for the normalised training set. The input variables should be uncorrelated with approximately the same co-variance for better results. Hence feature scaling in pre-processing has a more significant impact on controlling the stability and generalisation of network performance [30].

A neural network is associated with input/output units, and affiliation options weight gift is quick-witted [31]-[36]. Multilayer Perceptron (MLP) algorithm is the most significant part of extensively engaged and frequent neural networks. MLP is a forward artificial neural network model that maps input data sets to gather output [37], [38]. MLP consists of multiple layers of nodes for a graph, with a consequent one [39]-[43]. The output depends on the present input instance. The network learns in training by changing weights to predict the correct class values. Neural networks have the exceptional capacity to derive that inaccurate data and may well be familiar with extract patterns. A spot trend is that the unit of measurement needs to be more complex to be identified by humans or different computer techniques [44], [45]. Optimised deep-learning techniques can simplify the process of handling the data. The critical contribution of the research is to enhance learning by optimising the layers. The main contribution is to optimise the layer during training, which diversified research areas can utilise for proficient classification.

3. Proposed Methodology - Optimised Deep Convolutional Neural Network

A deep convolutional neural network (Deep CNN) has performed excellently in computer vision tasks. The deep CNN consists of several convolutional (Conv) layers, pooling (POOL) layers, and Fully Connected (FC) layers, each performing a specific task.

3.1 Convolution Layer

The Conv layer was the core of a DCNN, and its output volume held neurons in three dimensions. The first Conv1 layers extracted the local features directly from the noisy raw input signal using the number of sliding filters and feature maps along the frequency and time axis. The spatial size of the output volume in the Conv layers was calculated, and K1×H1×E1 calculated the accepted volume size for inputs. K2=K1-L1+2QS+1 calculate s the weight of the output volume ———. The height of output volume was computed by $H_2 = \frac{H_1 - L + 2Q}{S+1}$ (i.e., height and width were estimated similarly by symmetry), and E2=M calculated output volume depth. Where K1 was the volume size in the input region, L1 is the filter size of the Conv1 Layer neurons, S is the stride where they are incorporated, and q is the quantity of zero padding utilised on the border (right, left, top, and bottom), and M is the count of filters.

The process is crucial to establish a set of generally applicable criteria that can generalise to unknown areas for multi-domain learning. Initially, train the updated ResNet-26 on ImageNet to obtain a decent baseline set of parameters. Whenever a new domain enters the network, there is a new output layer and adjust depth-wise convolutional filters once it has been correctly set up. Various fields use pointwise convolutional filters interchangeably. Additionally, permit domain-specific batch normalisation values because the statistics of the images from other domains may vary. The output of the domain may be determined by stacking the learned depth-wise convolutional filters for all domains, and it is given as

$$\hat{O}_{k,l,m,q} = \sum_{i,j} \hat{K}_{i,j,m,q} \cdot F_{k+i-1,l+j-1,m,q} \quad (1)$$

Modelling cross-channel correlations and spatial correlations may be naturally separated thanks to the use of depthwise separable convolution. The two observations that are the foundation for the suggested approach are model effectiveness and explainability of deep neural network hidden units. The proposed model segregates the filters depthwise and pointwise, as illustrated in Figure 1.

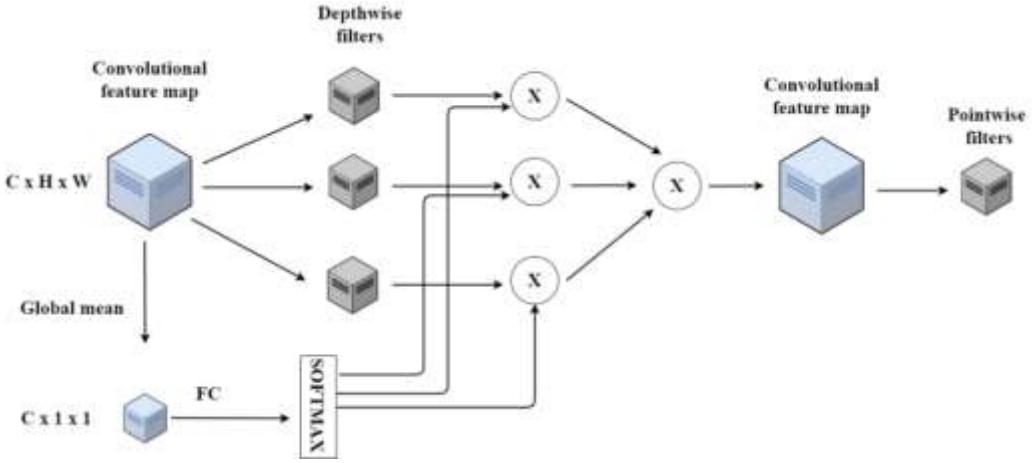

Figure 1. Depthwise and Pointwise Convolution

3.2 ReLU

A ReLU is used as a half-wave rectifier on a single neuron. It computed the function f(x) = max (0, x) with the threshold at zero to activate the network. Instead of a linear activation function, it efficiently added non-linearity to the network without significantly affecting the generalisation accuracy.

3.3 Batch Normalisation

Normalisation is applied to normalise the different sizes of images. The BatchNorm layers are inserted immediately after ReLU to initialise the weights for the outputs. Those normalised images were the inputs for the next layer.

3.4 Sub-sampling and Global Average Pooling

Generally, to remove the noise (smaller case) for each sub-region, it divided the input signal into a collection of non-overlapping rectangles and generated the greatest value. The raw input signal was subjected to independent sub-sampling/max pooling, which used the MAX operation to enlarge the

signal spatially. For the top layers, it decreased both the calculation expense and the size of the input map. The most typical pooling layer configuration with 2x2 filtering. Every slice's depth in the input is increased by two along the frequency and time axes. The accepted input volume of size $K_1 \times H_1 \times E_1$ calculated the size of the pooling layer. Two required hyper-parameters were the spatial extent (filter size) F and the stride S to shift. Finally, the output of a volume of size $K_2 \times H_2 \times E_2$ is produced. Where: K2 = (K1−F)/S+1; H2 = (H1−F)/S+1; E2 = E1. A static input function introduced zero parameters, and the padding was not required to estimate the pooling layer.

One of the main components of the enhanced DCNN-GAP approach is the global average pooling (GAP) layer. Global average pooling is a novel method to address the issue of the fully connected network's excessive number of model parameters in the conventional DCNN. The max-pooling procedure of conventional DCNN and the GAP's fundamental operation is comparable. However, in contrast to the standard max-pooling operation, the GAP layer uses several unique global average pooling filters (also known as pooling windows). The rectangular matrix used by the GAP filter is unique. The global average pooling is expressed as

$$S^{l}_{avrg\_pooling} = \frac{1}{c}\sum_{i=1}^{c} X^{l}_{1:h,\ 1:k,\ i} \quad (2)$$

Where Slavrg_pooling indicates $s_{\text{global average pooling}}$ at the feature map at the lth output, the feature map index is indicated by l, the entire count of the elements in the kernel is indicated by c, and the pooling kernel range is indicated by h from 1st to hth line, and pooling kernel range is denoted by l:k that is with the width direction from 1st to the kth column. Correspondingly, height is h, and width is w, which is indicated in the feature map, and the global average pooling filter is indicated by the $X^{l}_{1:h,\ 1:k,\ i}$ element. Figure 2 illustrates the size of the feature map, the feature map gap and the classification process at the softmax layer.

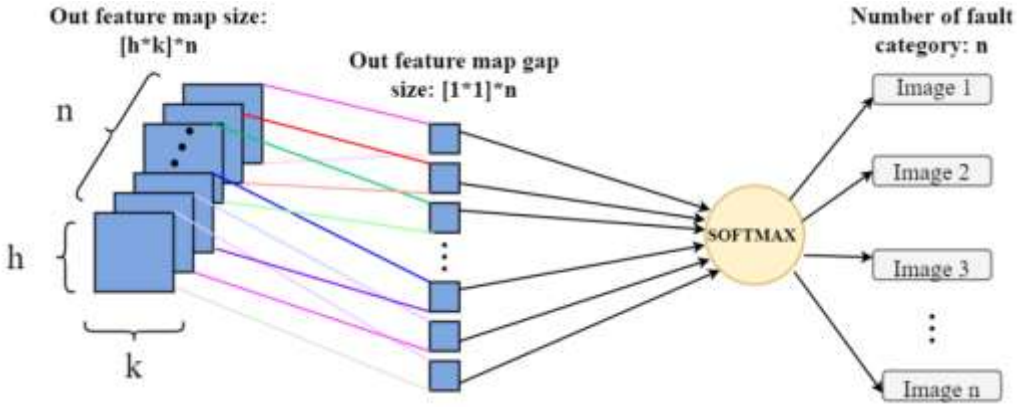

Figure 2. Schema of Global Average Pooling

3.5 Fully Connected Layers

Finally, after several convolutions, max-pooling layers and ReLU, the high-level reasoning was done by the fully connected layers in the NN. Neurons are fully connected to the previous layer as a regular NN. Their activation is computed with a matrix multiplication followed by a bias offset, given as:

$$Y_{out} = \sum_{i=1}^{n} w_i x_i + b_i \quad (3)$$

Where x was the input of the integers i=1, 2,... n; w were the weights given to the inputs; and b were the biases. The words "cons" and "R" stand for the random number between [0, 1] and the constant parameter smaller than 1, respectively. The neural network's optimised nonlinear constrained problem corrects the bias reduction method. In the below equation, the number of neurons in the input layer is N1, those in the hidden layer is N2, those in the output layer is N3 and the weights used to link the st neuron of the output layer to the jth neuron of the hidden layer and the ith neuron of the input layer are w^{O}_{sj} and w^{H}_{ji}. θ^{O}_{s} and θ^{H}_{j} are the bias of the neuron s and j. OO_{es}, OH_{ej}, OI_{ej} is the output of the neuron s,j, I for the eth pattern at the input. The term Iei is the input at ith input layer for the pattern at e. The primary goal is to optimise and reduce the discrepancies between the modelled

and target outputs through training. Figure 3 illustrates the overall schema of the proposed DCNN, whereby the global average pooling and filtering are illustrated.

$$\text{DCNN:}\begin{cases}\text{minimumFn} = \sum_{e=1}^{e}\sum_{s=1}^{N_3}(TO_{es} - OO_{es})^2 \\ \text{subject to } TO_{es} - f\left(\sum_{j=1}^{N_2} w_{sj}^{O} OH_{ej} - \theta_s^{O}\right) = 0 \ (s = 1,2,3,\dots\dots,N_3) \\ OH_{ej} - f\left(\sum_{i=1}^{N_1} w_{ji}^{H} OI_{ej} - \theta_j^{H}\right) \ (j = 1,2,3,\dots\dots,N_2)\end{cases} \quad (4)$$

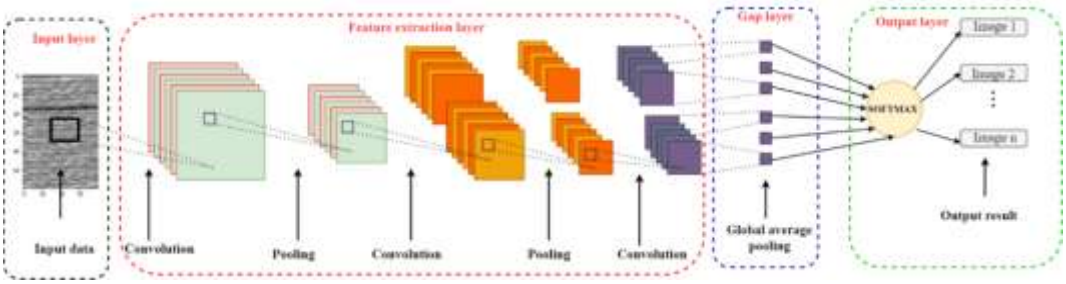

Figure 3. Overall Schema of Optimised DCNN

3.6 Dropout Layer

Since a fully linked layer consumed most of the noise-sensitive signal's characteristics, it was more likely to degrade performance and lead to overfitting. Overfitting and noise were stopped with the introduction of the dropout approach. Individual nodes were either "dropped out" of the net with probability 1-ρ = 0.5 or maintained with probability ρ at each training step. Therefore, a reduced network is left by dropping out the incoming node and the outgoing edges removed. Low LR (0.3) controlled the drastic change in weight while learning the network, i.e., more relevant features found by the DCNN after dropout with low LR. It helped to get more native recognition accuracy and defeated overfitting with clean raw multilingual input signals.

The dropout with max pooling is used for noisy redundant nodes elimination from more sensitive noisy signals (i.e., signals with more noise have less multilingual native speech information). After dropout, the node had the maximum value pooled by the max pool layer to choose the more informative clean native's speech feature. The pre-trained DCNN contained a set of hidden layers. Those were convolution, ReLU, normalisation, pooling, fully connected and dropout layers. At the top of the DCNN layers, a softmax loss layer (dropout) is introduced. Numerous localised features structured as several feature maps for multilingual numerals, words, and short sentences were included in the input. As additional convolutions, pooling, and dropout processes were used, the size (resolution) of the feature maps increased to include more small and precise speech characteristics at the top layers.

On top of the final DCNN layer, one or more fully linked hidden layers are often added to aggregate the features across all frequency bands over time. Sub-sampling with dropout was employed to reduce noise in the pre-trained models' structure since raw pictures were noisy. Following the fully linked layer, the dropout layer is added to prevent overfitting by removing redundant nodes from both networks. The network size was maintained by replacing one ReLU layer after the fully connected layer and adding one dropout layer.

In image net-vgg-f and Alex-net pre-trained models, the total number of Conv layers was five with different sizes of filters. The first Conv1 layer of the Vgg-f model had 64 feature maps, and the Alex-net had 96 feature maps with the same filter size (11*11*3). The second Conv2 layer of Vgg-f and Alex-net had the same number of feature maps, 256 with filter size (5*5*64 and 5*5*48). The third Conv3 layer of Vgg-f had 256, and Alex-net had 384 feature maps of the same size as the filter (3*3*256). The fourth Conv4 layer of Vgg-f had 256, and Alex Net had 384 feature maps with the filter size of (3*3*256 and 3*3*192). The last conv5 layer of Vgg-f was the same as those of C3, C4, and Alex-net had 256 feature maps of the same size as that of C4. In both the pre-trained models, 6 ReLUs were used after ReLU 2 normalisation layers were used for layer-level weight initialisation.

BatchNorm layers (224x224x3) are used immediately after ReLU to initialise the weights for the outputs. Those outputs were the inputs for the next layer. Five pooling and one softmax layer were used to remove noises. In both the pre-trained models, three fully connected layers were used. One dropout layer is introduced after the first fully connected layer to remove the noisy redundant node and defeat the overfitting. The dropout layer dropped out the noisy redundant nodes and chose the

maximum value (having more noise and less nativity information) among the minimum values (less noise affected area having more nativity information) using the max pooling operation.

4. Result and Discussion

The degree to which a specific value closely resembles situations that have been classified is known as accuracy. The testing process is accomplished using 40% of the data, and the training is done with 60% of the data. The representation of systematic errors and statistical bias is accuracy. It is also the recognition of (both TP and TN values) among the count of the assessed classes and the degree to which an estimation matches the actual value. Acquired accuracy over diverse iterations is illustrated in Figure 4. It is determined as

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (5)$$

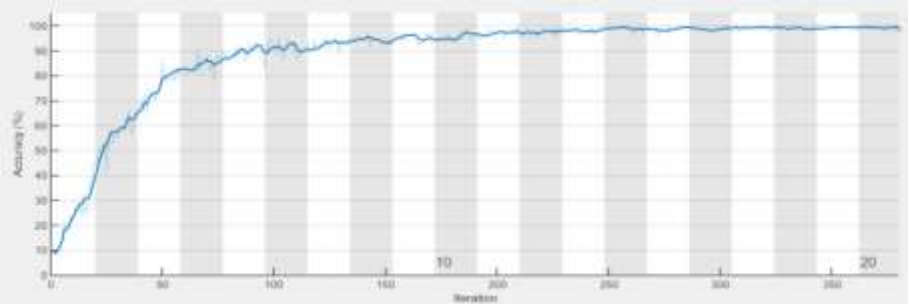

Figure 4. Accuracy of Testing the Image

According to classification accuracy, the proportion of accurate predictions to all predictions, Figure 4 shows how well the classification model performed. Over time, the suggested method becomes more accurate.

Figure 5 shows the loss, and loss occurrence minimises over iterations. To lead the model toward convergence during the training phase, the loss function measures the variation between the system's output data and the appropriate sample data. Eliminating the loss value results in the model fitting the training examples and the lowest test error. Figure 6 illustrates the image acquired from the convolution and batch normalisation process. Figure 7 illustrates the function evaluation process over the count of the minimum objective function.

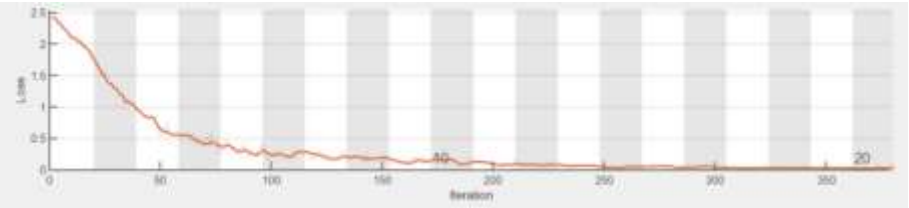

Figure 5. Loss of Testing the Image

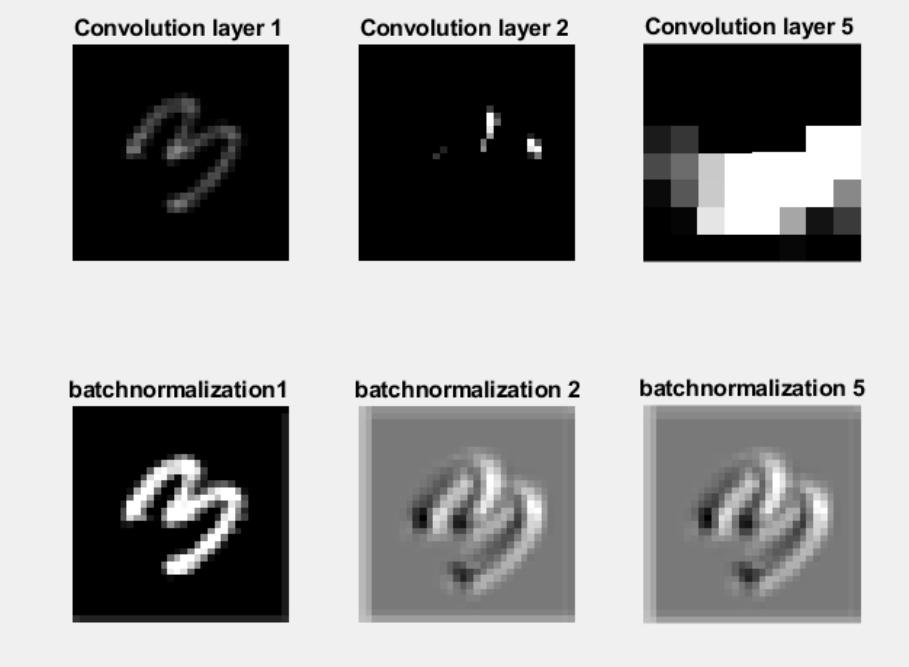

Figure 6. Convolution and Batch Normalisation

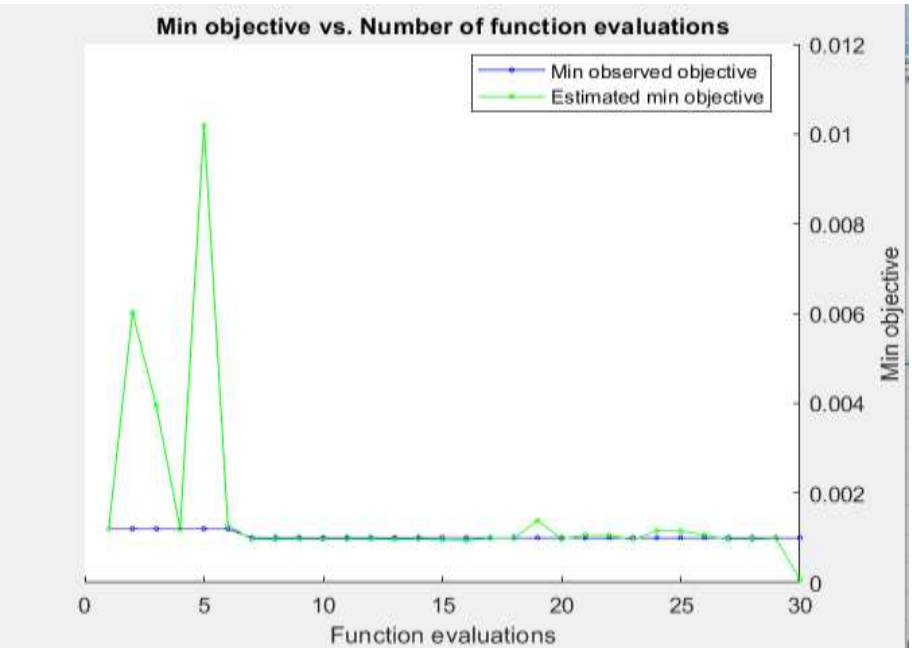

Figure 7. Function Evaluation and Minimum Objective Function

A confusion matrix table, shown in Figure 8, illustrates how effectively a categorisation system operates. In a confusion matrix, the results of a classification algorithm are shown and summarised. The occurrence of an error during optimising the layer is illustrated in Table 1, and the illustration is depicted in Figure 9.

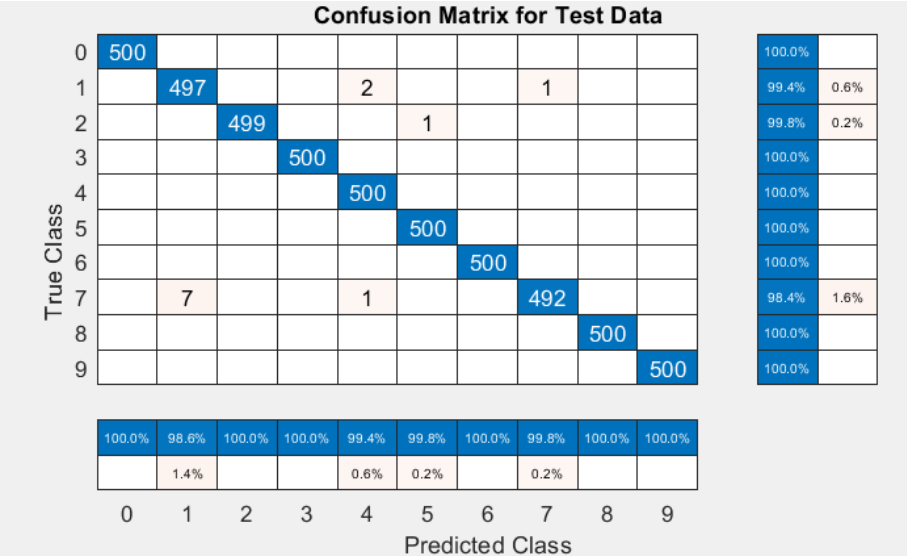

Figure 8. Confusion Matrix for Testing Data

Table 1. Comparison of Error

Algorithm	Error Rate
Stochastic Pooling	4.67
Learned Pooling	4.71
BN Inception Ensemble	4.82
Optimised DCNN	3.92

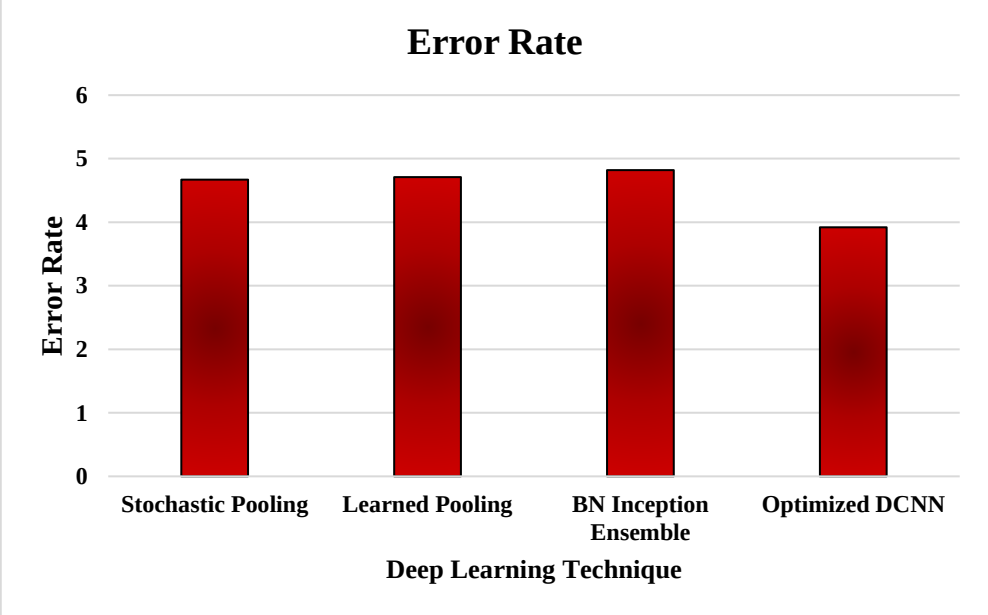

Figure 9. Comparison of Error

The comparison of error rates shows that the proposed approach is practical in terms of minimal error rate and outperforms the existing technique.

5. Conclusion

In recent years, deep convolutional neural networks (DCNN) have achieved significant advancements in various applications, such as image identification, voice recognition, and the translation of machine information. These activities help artificial intelligence in many different ways. However, float operations and terminal machine conversion are still challenging because there are so many factors. To address this problem, optimisation of convolution in the DCNN is started, which

modifies the neural network's properties and minimises information loss while enhancing performance. Convolution function minimisation solves the optimisation problems. After batch normalisation, a whole feature map is first reduced to a single value using the global pooling method rather than decreasing neighbourhood values. Traditional convolution is divided into depth-wise and pointwise to reduce the model size and calculations. Accuracy and error occurrence help assess the performance of the optimised convolution-based DCNN. The proposed approach attained a minimal error rate of 3.92 and higher accuracy, indicating the proposed technique's effectiveness. The proposed approach attains adequate accuracy with minimal error where the occurrence of several layers is a complicated process. In future, it can be addressed by a pruning strategy.

References

[1] Bucilua, C., Caruana, R., Niculescu-Mizil, A., "Model compres- ˇ sion," In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD, vol.6, New York, USA: ACM Press. pp. 535. Avaliable: https://dl.acm.org/DOI/abs/10.1145/1150402.1150464, doi:10.1145/1150402.1150464.

[2] Lebedev, V., Lempitsky, V., "Speeding-up convolutional neural networks: A survey," *BULLETIN OF THE POLISH ACADEMY OF SCIENCES TECHNICAL SCIENCES*, vol.66, 2018. Avaliable: http://www.czasopisma.pan.pl/Content/109869 /PDF/05_799810_00925_Bpast.No.666_31.12.18_K2.pdf?handler=

[3] Mathieu, M., Henaff, M., LeCun, Y., "Fast Training of Convolutional Networks through FFTs." *ArXiv preprint*, 2013. Avaliable: http: //arxiv.org/abs/1312.5851.

[4] Lavin, A., Gray, S., "Fast Algorithms for Convolutional Neural Networks," In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), *IEEE*, 2016, pp. 4013-4021. http://ieeexplore.ieee.org/document/7780804/, http:// arxiv.org/abs/1312.5851, doi:10.1109/CVPR.2016.435.

[5] Zhao, D., Zeng, Y., & Li, Y., "BackEISNN: A deep spiking neural network with adaptive self-feedback and balanced excitatory-inhibitory neurons." *Neural Networks*, 2022.

[6] Chellapilla, K., Puri, S., Simard, P., "High-Performance Convolutional Neural Networks for Document Processing," In Tenth International Workshop on Frontiers in Handwriting Recognition, 2006. Avaliable: URL: https://hal.inria.fr/inria-00112631/, doi:10.1.1.137.482

[7] Blalock, D., Ortiz, J.J.G., Frankle, J., Guttag, J., "What is the State of Neural Network Pruning?," *ArXiv preprint*, 2020. Avaliable: http: //arxiv.org/abs/2003.03033.

[8] Wan, Q., & Choe, Y., "AdjointBackMap: Reconstructing effective decision hypersurfaces from CNN layers using adjoint operators." N*eural Networks*, vol. 154, 2022, pp. 78-98.

[9] Augasta, M.G., Kathirvalavakumar, T., "Pruning algorithms of neural networks - A comparative study." *Open Computer Science*, vol. 3, 2013, pp. 105-115. Avaliable: doi:10.2478/s13537-013-0109-x.

[10] Schmidhuber, J., "Deep learning in neural networks: An overview." *Neural networks*, vol. 61, 2015, pp. 85-117.

[11] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. "Continual lifelong learning with neural networks: A review." *Neural Networks*, vol. 113, 2019, pp. 54-71.

[12] Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., Song, J., "Forward and Backward Information Retention for Accurate Binary Neural Networks," In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), *IEEE*. 2020, pp.2247-2256. Avaliable: https://ieeexplore.ieee.org/document/

[13] 9157443/, doi:10.1109/ CVPR42600.2020.00232

[14] Reed, R., "Pruning Algorithms - A Survey." *IEEE Transactions on Neural Networks*, vol.4, 1993, pp.740-747. Avaliable: http://ieeexplore.ieee.org/document/248452/, doi:10.1109/72.248452.

[15] Alemdar, H., Leroy, V., Prost-Boucle, A., Petrot, F., "Ternary neural networks for resource-efficient AI applications," In 2017 International Joint Conference on Neural Networks (IJCNN), *IEEE*, 2017, pp. 2547-2554. Avaliable: https://ieeexplore.ieee.org/abstract/document/ 7966166/, doi:10.1109/IJCNN.2017.7966166.

[16] Ibtehaz, N., & Rahman, M. S., "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation." *Neural Networks*, vol.121, 2020, pp.74-87.

[17] Bianco, S., Cadene, R., Celona, L., Napoletano, P., "Benchmark analysis of representative deep neural network architectures." *IEEE Access*, vol.6, 2018, pp.64270-64277. Avaliable: doi:10.1109/ACCESS.2018.2877890

[18] Bolukbasi, T., Wang, J., Dekel, O., Saligrama, V., "Adaptive Neural Networks for Efficient Inference." In *Thirty-fourth International Conference on Machine Learning*, 2017. Avaliable: https://arxiv.org/abs/1702. 07811http://arxiv.org/abs/1702.07811.

[19] Gao, X., Zhao, Y., Dudziak, L., Mullins, R., Xu, C.Z., Dudziak, L., Mullins, R., Xu, C.Z., "Dynamic Channel Pruning: Feature Boosting and Suppression," In *International Conference on Learning Representations (ICLR)*, 2019, pp.1-14. Avaliable: http://arxiv.org/abs/1810. 05331.

[20] Lei, W., Chen, H., Wu, Y., "Compressing Deep Convolutional Networks Using K-means Based on Weights Distribution," In *Proceedings of the 2nd International Conference on Intelligent Information Processing - IIP'17*, New York, USA:ACM Press, 2017, pp. 1-6. Avaliable: http://dl.acm.org/citation.cfm?doid=3144789.3144803, doi:10.1145/3144789.3144803.

[21] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P., "Pruning Filters for Efficient ConvNets," In *International Conference on Learning Representations (ICLR)*, 2017. Avaliable: http://arxiv.org/abs/ 1608.08710, doi:10.1029/2009GL038531.

[22] HANSON, S., "Comparing biases for minimal network construction with back-propagation," In *Advances in Neural Information Processing Systems* (NIPS), 1989, pp. 177-185.

[23] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M.A., Dally, W.J., "EIE: Efficient Inference Engine on Compressed Deep Neural Network," In 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), *IEEE*. 2016, pp. 243-254. Avaliable: http://ieeexplore.ieee.org/document/7551397/

[24] Yuan, M., Lin, Y., "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B* (Statistical Methodology), vol.68, 2006, pp.49-67. Avaliable: http://doi.wiley.com/10.1111/j.1467-9868.2005.00532.x, doi:10.1111/j.1467-9868.2005.00532.x.

[25] Lebedev, V., Lempitsky, V., "Fast ConvNets Using Group-Wise Brain Damage," In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), *IEEE*. 2016, pp. 2554-2564. Avaliable: http://openaccess.thecvf.com/content_cvpr_2016/html/Lebedev_Fast_ConvNets_Using_CVPR_2016_paper.htmlhttp://ieeexplore.ieee.org/document/7780649/, doi:10.1109/CVPR.2016.280.

[26] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C., "Learning Efficient Convolutional Networks through Network Slimming," In IEEE International Conference on Computer Vision (ICCV), *IEEE*, 2017, pp.2755-2763. Avaliable: URL: http://ieeexplore.ieee.org/document/ 8237560/, doi:10.1109/ICCV.2017.298.

[27] Huang, Z., Wang, N., "Data-Driven Sparse Structure Selection for Deep Neural Networks," In *Lecture Notes in Computer Science* (including subseries Lecture Notes in Artificial Intelligence and Bioinformatics), vol.11220, LNCS, 2018, pp.317-334. Avaliable: URL: http://link.springer.com/10.1007/978-3-030-01270-0_ 19, doi:10.1007/978-3-030-01270-0{_}19

[28] Han, S., Mao, H., Dally, W. J., "Deep compression: Compressing deep neural networks with pruning, trained quantisation and Huffman coding," In *International Conference on Learning Representations(ICLR)*, 2016, pp.199-203. Avaliable: URL: http://arxiv.org/abs/1510.00149

[29] Ju, W., Luo, X., Ma, Z., Yang, J., Deng, M., & Zhang, M., "GHNN: Graph Harmonic Neural Networks for semi-supervised graph-level classification." *Neural Networks*, vol.151, 2022, pp.70-79.

[30] Choi, B., Lee, J. H., Kim, D. H., "Solving local minima problems with many hidden nodes on two-layered feedforward artificial neural networks." *Neurocomputing*, vol.71, 2008, pp.3640-3643. Avaliable: doi:10.1016/j.neucom.2008.04.004

[31] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever,

I., Amodei, D., "Language Models are Few-Shot Learners." *ArXiv preprint*, 2020, Avaliable: http://arxiv.org/abs/ 2005.14165

[32] Sze, V., Chen, Y. H. H., Yang, T. J. J., Emer, J. S., "Efficient Processing of Deep Neural Networks: A Tutorial and Survey." *Proceedings of the IEEE*, vol.105, 2017, pp.2295-2329. Avaliable: http://ieeexplore. ieee.org/document/8114708/, doi:10.1109/JPROC.2017.2761740.

[33] Liu, X., Zou, X., Ji, Z., Tian, G., Mi, Y., Huang, T., ... & Wu, S., "Neural feedback facilitates rough-to-fine information retrieval." *Neural Networks*, vol.151, 2022, pp.349-364.

[34] Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R., 2018b. backdrop: Dynamic Inference.

[35] "Paths in Residual Networks," In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), *IEEE*. pp. 8817-8826. Avaliable: https://ieeexplore.ieee.org/document/8579017/, doi:10.1109/ CVPR.2018.00919.

[36] Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H., "Learning Structured Sparsity in Deep Neural Networks, in Advances in Neural Information Processing Systems (NIPS)," *IEEE*. 2016, pp. 2074-2082. Avaliable: https://dl.acm.org/doi/abs/10.5555/3157096.3157329, doi:10.1016/j.ccr.2008.06.009

[37] Luo, J.H., Wu, J., "AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference." *Pattern Recognition*, vol.107, 2015, p.107461. Avaliable: https://linkinghub.elsevier.com/retrieve/ pii/S0031320320302648, doi:10.1016/j.patcog.2020.107461.

[38] Ye, J., Lu, X., Lin, Z., Wang, J. Z., "Rethinking the SmallerNorm-Less-Informative Assumption in Channel Pruning of Convolution Layers." *ArXiv preprint*, 2018. Avaliable: http://arxiv.org/abs/1802.00124.

[39] Glossner, J., Blinzer, P., Takala, J., "HSA-enabled DSPs and accelerators." In 2015 IEEE Global Conference on Signal and Information Processing, *GlobalSIP*, 2016, pp. 1407-1411. Avaliable: doi:10.1109/GlobalSIP. 2015.7418430.

[40] Elsken, T., Metzen, J.H., Hutter, F., "Neural Architecture Search." *Journal of Machine Learning Research*, vol.20, 2019, pp.63-77. Avaliable: http://link.springer.com/10.1007/978-3-030-05318-5_3

[41] Rvachev, M. M., "Neuron as a reward-modulated combinatorial switch and a model of learning behaviour." *Neural networks*, vol. 46, 2013, pp. 62-74.

[42] Triche, A., Maida, A. S., & Kumar, A., "Exploration in neo-Hebbian reinforcement learning: Computational approaches to exploring-exploitation balance with bio-inspired neural networks." *Neural Networks*, 2022.

[43] Gou, J., Yu, B., Maybank, S.J., Tao, D., "Knowledge Distillation: A Survey." *ArXiv preprint*, 2020. Avaliable: http://arxiv.org/abs/2006.05525.

[44] Ruffy, F., Chahal, K., "The State of Knowledge Distillation for Classification." ArXiv preprint, 2019. Avaliavle: http://arxiv.org/abs/1912. 10850.

[45] Gale, T., Elsen, E., Hooker, S., "The State of Sparsity in Deep Neural Networks." ArXiv preprint, 2019. Avaliale: http://arxiv.org/abs/1902. 09574.

[46] Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S., "Once-for-All: Train One Network and Specialise it for Efficient Deployment." *ArXiv preprint*, 2019, pp. 1-15. Avaliable: http://arxiv.org/abs/1908.09791.

[47] Filippini, M., Borra, D., Ursino, M., Magosso, E., & Fattori, P., "Decoding sensorimotor information from superior parietal lobule of macaque via Convolutional Neural Networks." *Neural Networks*, vol. 151, 2022, pp. 276-294.