Defence Science Journal, Vol. 74, No. 6, November 2024, pp. 878-884, DOI : 10.14429/dsj.74.19475

Accelerating Deep Network Training for Radar Identification Using Batch Normalisation

Preeti Gupta[#,*], Pooja Jain[$] and O.G. Kakde[$]

[#]*Indian Institute of Information Technology, Nagpur - 441 108, India*
[$]*DRDO-Defence Electronics Research Laboratory, Hyderabad - 500 005, India*
[*]*E-mail: preetigupta.dlrl@gov.in*

ABSTRACT

Deep learning techniques have shown remarkable success in radar identification. However, deep neural network training can be time and resource intensive. Batch normalisation is a popular approach for quickening deep feed-forward neural network training. The training of deep neural networks is accelerated by minimising the internal covariate shift and stabilizing the training process by normalising the intermediate activations within each mini-batch. In this research, the convergence behavior of networks with and without batch normalisation is compared. Batch normalisation standardizes the input to a layer for each mini-batch applied to either the activations of a prior layer or inputs directly. Our experiments indicate that batch normalisation is effective in improving a variety of neural network properties. However, the primary objective of the study is to accelerate convergence during training and also improve the accuracy of the model in identifying radar. Batch normalisation contributes to better generalization, reduces overfitting, and provides consistent learning which can improve the accuracy of the model on unseen data. The results show that batch-normalized models have higher test and validation accuracies across all datasets, which we attribute to their regularizing impact and more steady gradient propagation. This research also examines the impact of several parameters, such as batch size, momentum, and beta and gamma parameters, on the effectiveness of DNNs with batch normalisation. The radar dataset used for training is the fused emitter set obtained after feature level fusion of the tracks intercepted by ESM (Electronic Support) and ELINT (Electronic Intelligence) systems.

Keywords: Batch normalisation; Momentum; Beta and gamma parameters; Batch size

1. INTRODUCTION

Radar emitter identification plays a crucial role in Electronic Warfare Systems and its identification is one of the key purposes of Electronic Intelligence (ELINT) and Electronic Support Measure (ESM) equipment in modern warfare. ELINT-ESM sensors are installed on platforms for the recognition of emitters in the area of responsibility. The tracks sent from ELINT-ESM sensors contain the features of the intercepted emitter signal, the azimuth, and the identification information among other things. It helps the war fighters in making decisions by providing timely information about the emitters.

Deep Neural Networks (DNNs) have become a key artificial intelligence technology for specific radar emitter identification. It is challenging to train densely layered deep neural networks because these networks might be sensitive to the learning algorithm's configuration parameters and initial random weights. The techniques employed to initialize the weights before training can have an impact on deep neural networks. Batch Normalisation (BN)[1] increases the stability of deep network training by reducing the sensitivity to the weight initialization approach. The paper focuses on accelerating the training of deep networks for radar identification using batch normalisation. The present study is done by fusing the track measurements received from the ESM and ELINT system at the feature level and training the network with the fused track. Applied to a Multi-Layer Perceptron model, Batch Normalisation achieves the same accuracy with less number of epochs and increases the classification accuracy by 13 %.

Received : 11 September 2023, Revised : 13 January 2024
Accepted : 15 January 2024, Online published : 25 November 2024

1.2 Batch Normalisation and Related Work

Batch normalisation[2] expedites the training of deep neural networks by incorporating data standardization into the design of the network. The inputs to a layer are normalized by batch normalisation. Most network types including convolution neural networks, multilayer perceptrons, and recurrent neural networks can benefit by using BN. It may be employed on the inputs to the layer before or after the activation function in the preceding layer. For s-shaped functions like the logistic and hyperbolic tangent, it provides good results after the activation function. The activation functions that lead to non-Gaussian distribution such as rectified linear activation function, it is preferred to place it before the activation function. This might need to use learning rates that are significantly higher than usual, which would speed up learning even further. Batch normalisation reduces generalisation error and may even eliminate the need for dropout regularization in some situations.

As a result of the adjustment of network parameters during training, a phenomenon known as the internal covariate shift occurs which alters the network activations distribution across layers because of the variation in the parameters during network training. In the machine learning community, the covariate shift is a well-known issue that commonly appears in real-world problems[3]. The ideal transformation of each layer into a space would maintain the functional relationship while maintaining the same distribution. Every input feature is normalized across each layer and minibatch such that it has a zero mean and standard deviation of one to prevent expensive computation of covariance matrices to remove correlation and whiten the data.

Taking into account a fully connected deep neural network, the m[th] hidden layer of the network is defined by:

$$h^{(m)} = g(W^{(m)}h^{(m-1)} + b^{(m)}), \quad (1)$$

where, $h^{(0)}$ represents the network input x, g represents an element-wise nonlinear function, and $h^{(m)} \in \mathbb{R}^{n_m}$ represents the m^{th} hidden variable. $W^{(m)}$ and $b^{(m)}$ are the associated weight and bias. During training, consider $\{x_1, x_1, \ldots, x_p\}$ed a mini-batch comprising of P examples and $H = \left[h_1^{(m-1)}, h_2^{(m-1)}, \ldots h_P^{(m-1)}\right]^T$ is the matrix of hidden variables related to the layer m-1. Batch normalisation replaces the m[th] hidden layer by

$$h^{(m)} = g(W^{(m)}\mathrm{B}_{\beta,\gamma}(h^{(m-1)}) + b^{(m)});$$
$$\mathrm{B}_{\beta,\gamma}(h^{(m-1)}) = \gamma\frac{h^{(m-1)} - \mu_H}{\sigma_H} + \beta \quad (2)$$

where, σ_H and μ_H are the standard deviation and mean vectors of $\left\{h_i^{(m-1)}\right\}$ and γ, β are there-scaling and re-centering parameter vectors respectively which are trainable. The BN transformation operator $B_{\beta,\gamma}(\cdot)$ first normalizes the activation $h^{(m-1)}$ and then re-scales with γ parameter and re-centers with β parameter. In practice, the denominator in (2) is the square root of the variance vector plus a small ε>0 to avoid division by zero[4]. σ_H and μ_H depends upon mini-batch input and BN network (2) varies with different mini-batches during training.

There have been plenty of works that examine various aspects of Batch Normalisation. Essentially, batch normalisation is the same as normalisation preprocessing, with the exception that it takes place after data has already been fed into the network. It enhances the convergence and generalization of neural network training; the phenomena are understood theoretically[5]. The Decorrelated Batch Normalisation (DBN) process has been proposed[6] which centers, scales, and also whitens activations. The experiments were conducted on multilayer perceptrons & convolution neural networks and have shown improved performance.

Normalisation layers are crucial in deep networks with leaky, parametric, and ReLU-like piecewise linear activation functions, according to Liao et al[7]. They discovered through their research on the CIFAR10, CIFAR100, MNIST, and SVHN datasets that introducing BN is crucial for networks with saturating nonlinearities and piecewise linear activation functions. They also demonstrated that for the network training to proceed quickly and accurately, batch normalisation must be added before the nonlinear activation functions.

After applying batch normalisation, the zero mean and unit variance properties of all intermediate layers will have been restored. Since the activations of the other layers are continuously renormalized, their weights essentially do not influence the majority of the cases, and learning this model has become much easier[8]. The authors have performed an empirical investigation[9] to examine the impact of batch normalisation and dropout on deep learning model training on the CNN model and shown that dropout and batch normalisation can be utilized when short on time to experiment. Experimental evidence provided by Laurent[10], *et al.* demonstrates the difficulty of extending the BN method for deep feed forward networks to Recurrent Neural Networks (RNN). It was discovered that batch normalisation does not speed up convergence for hidden-to-hidden transitions in recurrent networks. Although batch normalisation can speed up the training criterion's convergence when used for input-to-hidden transitions, it does not show to enhance generalization performance on either language modeling or audio recognition tasks. Adaptive batch normalisation[11] to improve deep neural network generalisation capabilities is suggested through the efficient method of domain adaptation using batch-normalized neural networks.

The effectiveness of batch normalisation decreases drastically when applied to small mini-batch sizes. Cross-iteration batch normalisation technique[12] to enhance estimation quality which compensates for the changes in network weight over multiple iterations is suggested. The issues of mini-batches in batch normalisation are also resolved by restructuring the batch normalisation layer by splitting it into two sub-layers on the CNN model[13]. The image classification by the CNN network can be improved by adding a batch normalisation layer with hyperparametertunings[14]. Experiment[15] has been done on various CNN and GAN networks and demonstrated that L1BN is equivalent in performance to L2BN with high computational efficiency. To improve the instance-specific representation, a feature calibration scheme along with the advantage of batch normalisation is suggested[16].

2. BACKGROUND

ELINT-ESM sensors are deployed as part of an electronic recognition system on various platforms in the area of responsibility. ELINT-ESM sensors comprise passive receivers and direction finders for intercepting emitter signals from various directions. The signals intercepted by the sensors are processed to extract information about the emitter which includes feature parameters, azimuth direction, and identity information. Both sensors work asynchronously for emitter identification. The emitter information collected by ELINT systems is used for strategic purposes and those collected by ESM systems are for tactical purposes. In conclusion, ELINT sensors identify the opponent's capabilities, while ESM sensors identify which enemy radar is active at any given time and where it is located. For the study, the tracks from ESM and ELINT systems are fused at the feature level in this paper.

3. METHODOLOGY

In the present study, the sensor tracks intercepted from ESM and ELINT systems are fused in the Multi-Sensor Data

Fusion (MSDF) block using feature-level fusion. In feature-level fusion as depicted in Figure 1 below, the representative features from the ESM and ELINT sensor data are extracted and combined to form a concatenated feature vector of the target entity. This feature vector is given as input to the deep network for training for identity declaration. Data alignment and data association need to be performed before concatenating the features of both sensors.

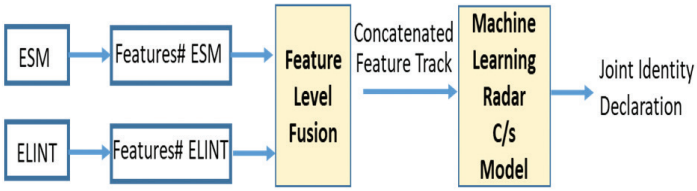

Figure 1. Feature level fusion.

The fused track file forms part of an integrated radar database as shown in Fig. 2. The verified fused track datasets train the deep neural network to predict the identity of unknown intercepts. The study emphasizes accelerating the training of deep neural networks using batch normalisation methodology. To choose an acceptable approach for our studies, it is essential to keep in mind the purported impact of batch normalisation, which may be broadly classified as improvements in convergence rate and generalisation performance. Here we compared the model with batch normalisation to the identical model without batch normalisation to show how the batch normalisation technique alone changes the training behavior. The studies described in the next sections are compared for training speed, convergence curve, performance, generalization analysis, and parameter sensitivity to show the effect of batch normalisation.

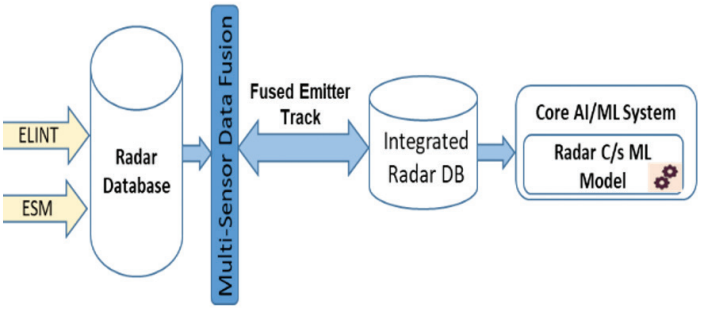

Figure 2. Operational flow diagram.

4. EXPERIMENTAL SETUP

As our problem is to classify radar into one of the classes, it is a multiclass classification problem. A multilayer perceptron model is developed with 14 input features in the visible layer. The model contains a single hidden layer with 150 nodes, which are randomly selected. The batch normalisation layer is added to normalize the inputs of the layer. The model uses the Rectified Linear Activation (ReLU) function and He initialisation is used as a random weight initialization technique. ReLU function introduces non-linearity in the model, allowing one to learn complex patterns in the data and it is computationally efficient and converges faster than other activation functions. The issue of vanishing or exploding gradients can be mitigated by using He initialization. By initializing weights with values that take the non-linearity of the activation functions like ReLU into account, the gradients propagate effectively. He initialization is particularly suitable with the ReLU activation function. The output layer has 175 nodes, each of which corresponds to one class. The activation function chosen at the output layer is the softmax activation function as it is a multi-class classification problem. The model is trained using an Adam optimization technique, and the optimization is directed by a sparse categorical cross-entropy loss function. In this study, we aim to illustrate the properties of batch normalisation using different parameters. First, we will divide the samples into an 80:20 train and test dataset. This will give the model a sample set large enough to learn from and allow for a balanced evaluation of how well it performs. After some trial and error, it was found that the model is suitable with 1500 epochs. The datasets used for the experiments are radar datasets. The radar dataset contains the parametric detail of emitters including the emitter name and their mode parameters. The unclassified radar data sets are used in the experiment. There is limited availability of labeled radar emitter datasets due to data privacy and security. In this experiment, although it was challenging, diverse and representative emitter data sets covering a range of emitters were obtained from various sources.

Table 1. Characteristics of radar dataset

Dataset	Classes	Trng set	Test set	Input features
Radar dataset	175	10764	2692	14

5. RESULTS

Batch normalisation can be added to the model presented in the preceding section. The use of batch normalisation is projected to speed up training and give the model classification accuracy that is comparable to or better in fewer training epochs. According to reports, batch normalisation also offers a moderate kind of regularization, which may potentially result in a slight decrease in generalization error, as shown by a slight improvement in accuracy in classifying the validation set. Individual layers may be subject to batch normalisation, or all of them may. The model may incorporate an additional batch normalisation layer before the output layer and after the hidden layer.

5.1 Training Speed Comparison

The time required to reach convergence with the training dataset is compared between the DNN models with and without batch normalisation. The results show the reduction in training time achieved by incorporating batch normalisation, highlighting its ability to accelerate the training process.

The results in Table 2 display the convergence time with and without batch normalisation after the activation function.

Table 2. Convergence time

Convergence time	Training time (s)	Epoch (No)
Without BN	15191.1	516
With BN	351.3	404

5.2 Convergence Analysis

For analysis of the DNN models' convergence behavior, the training and validation losses can be shown over iterations. It is possible to demonstrate how batch normalisation enhances the stability and consistency of the training process by comparing the convergence curves, which results in convergence that is both quicker and more dependable.

5.3 Performance Comparison

The accuracy of the performance measures is evaluated for the DNN models trained with and without batch-normalisation. The findings show that batch normalisation affects the models' overall ability to identify objects. By adding a batch normalisation layer before and after the prior layer's activation function, the model's accuracy is assessed.

The results in Table 3 display the classification accuracy with & without BN after the activation function.

Table 3. Classification accuracy

Classification accuracy	Training (%)	Test (%)
Without BN	77.5	75.7
With BN	90.8	89.2

The results of the classification accuracy of the MLP radar model without batch normalisation and with batch normalisation after activation function are shown in Table 3 and the graph showing line plots of the classification accuracy and cross-entropy loss on the train (dashed) and test (solid) datasets is shown in Fig. 3. As per experimental results, when batch normalisation is applied to the network, as opposed to when it is not, the classification accuracy increases by 13 %.

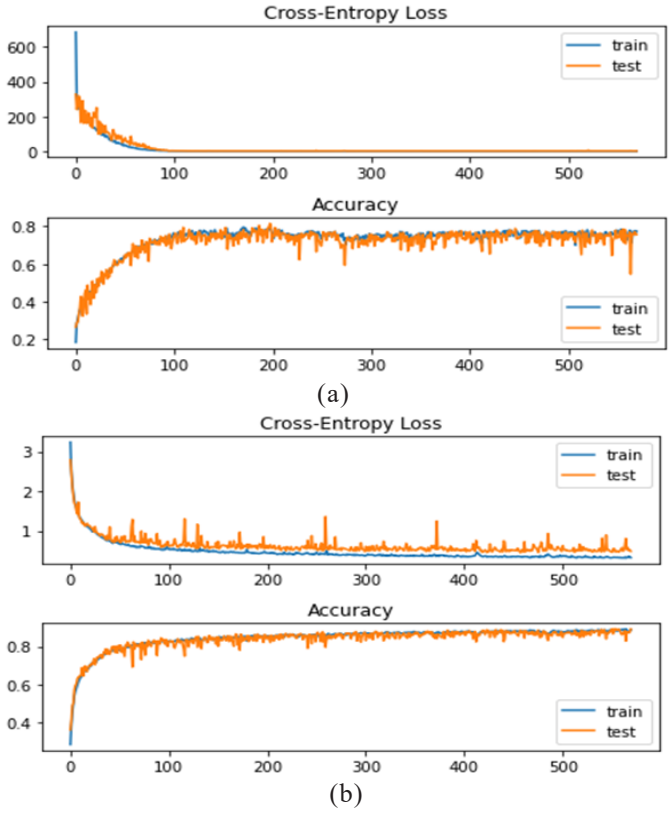

Figure 3. (a) Cross-entropy loss & classification accuracy without BN after activation function, and (b) Cross-entropy loss & classification accuracy with BN after activation function.

5.4 Effect of Batch Normalisation After and Before the Activation Function

By using batch normalisation either before or after the activation function of the preceding layers, the inputs to the layers can be standardized. The properties of the dataset, the model architecture, and the specific task at hand determine whether to apply batch normalisation before or after the activation function. By adding a batch normalisation layer before and after the prior layer's activation function, the model's accuracy is assessed in the present study. It shows that it is best when applied after the activation function.

The model was modified so that batch normalisation is applied before, rather than after, the hidden layer's activation function. The result of applying batch normalisation before the activation function is shown in Table 4. The graph showing the plots of classification accuracy and cross-entropy loss is depicted in Fig. 4. It appears that BN is effective after the rectified linear activation function, at least for this model setup on this particular dataset when compared with results in Table 3, where BN is applied after the activation function.

The results in Table 4 display the classification accuracy with BN before the activation function.

Table 4. Classification accuracy with BN

Classification Accuracy	With BN before the activation function		
	Training	Test	Time
	85 %	83.1 %	546.95

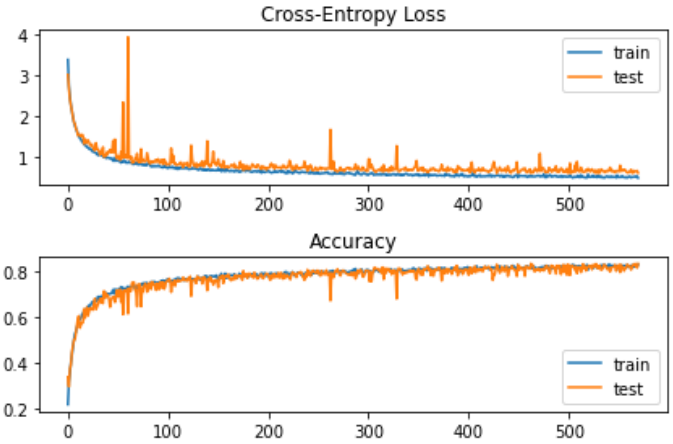

Figure 4. Classification accuracy with BN before activation function.

5.5 Effect of Batch Normalisation at the Input Layer

The results of implementing batch normalisation at the input layer are depicted in Table 5, but the model's performance on the training dataset is worse, with an accuracy range of 71 % to 75 %. It is not suitable for this problem.The graph showing the plots of classification accuracy and cross-entropy loss is depicted in Fig. 5.

The results in Table 5 display the classification accuracy with BN at the input layer.

Table 5. classification accuracy

Classification accuracy	With BN at the input layer		
	Training	Test	Time
	74.5 %	71.9 %	472.1

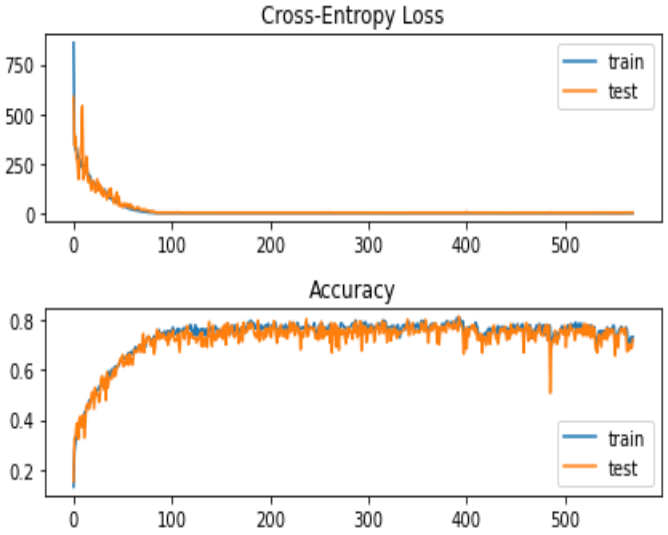

Figure 5. Classification accuracy with BN at the input layer.

5.6 Generalisation Analysis

It is important to assess the generalization capability of the DNN models. This is done by evaluating the model's performance on a separate validation set. The comparison of the performance of the models with and without batch normalisation on unseen data is done to determine whether batch normalisation boosts the generalisation ability of the models. Table 6 depicts the results of the experiment done with batch normalisation on the training and validation data set which shows improvement in the generalization error when compared to the result without batch normalisation.

5.7 Hyper Parameter Sensitivity

The sensitivity of the DNN models to hyperparameter choices is investigated. The study assesses the impact of combining batch normalisation with different batch sizes, momentum, beta, and gamma parameters.

5.8 Batch Size

The performance of the model is examined in an experiment using various batch sizes to determine the most suitable choice for this specific problem. All other hyperparameters are left unchanged, except the batch size utilised for training is modified. The batch sizes selected are 32,64,128,256.

The results in Table 6 display the classification accuracy with BN at the input layer.

Table 6. Classification accuracy with BN at input layer

Batch size	Training accuracy	Test accuracy	Training time(s)
32	92.1 %	87.4 %	1009.5
64	90 %	87.3 %	486.9
128	91.8 %	87.9 %	361.6
256	85.6	82.2	107.8

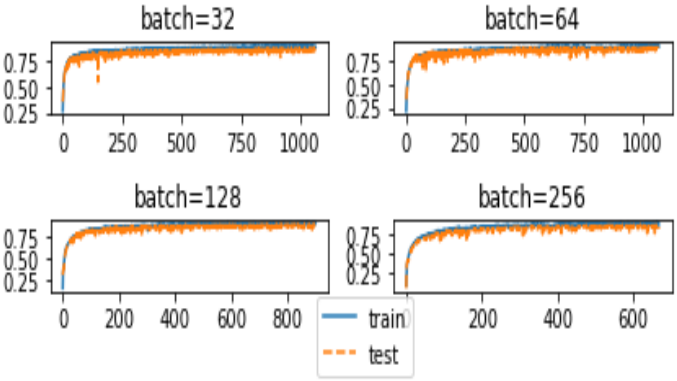

Figure 6. Classification accuracy with different batch sizes-32,64,128,256.

The results in Table 6 show the classification accuracy and training time with different batch sizes.The graph showing the plots of classification accuracy and cross-entropy loss for different batch sizes is depicted in Fig. 6.

From the results, it can be demonstrated that for smaller batches (32,64) the overall training time is longer due to the increased number of iterations required to cover the entire dataset. For larger batch sizes (128,256) the overall training time is shorter since fewer iterations are needed to process the entire dataset. Mini-batch statistics estimates are more accurate when the batch sizes are larger, resulting in better normalisation. This can contribute to improved accuracy, especially when the dataset is large and diverse. However, using larger batch sizes might also reduce the randomization effect and potentially lead to overfitting, especially if the model is complex or the dataset is relatively small. In the present study, the model's performance is good with batch size 128. The validation accuracy is 87.9 % with a training time of 361.6 sec.

5.9 Momentum

The effect of using momentum is that it smoothens out the estimates of mean and variance over time, making them more stable and reliable. It improves the convergence speed and improves the generalisation performance of the model. Since the batch size determines the regularization strength of BN, a small batch size could cause an under-fitting problem, which would make the model less useful. The noise level in the training process can be automatically controlled using momentum parameters. As a result, the model performs better with small batch sizes. Table 7 displays the effect of applying the momentum in the batch normalisation layer while training a small batch size of 16, the training time is reduced when compared without momentum. The graph showing the plots of classification accuracy and cross-entropy loss is depicted with and without momentum with batch size 16 is shown in Fig. 7.

The results in Table 7 display the classification accuracy using batch normalisation after activation without momentum and with batch size 16.

Table 7. Classification accuracy using batch normalisation

Classification accuracy with batch Size =16	Trng (%)	Test (%)	Epoch (No)	Trng Time(s)
Without momentum	87.8	85.8	743	1376.2
With momentum	89	86.5	519	835.87

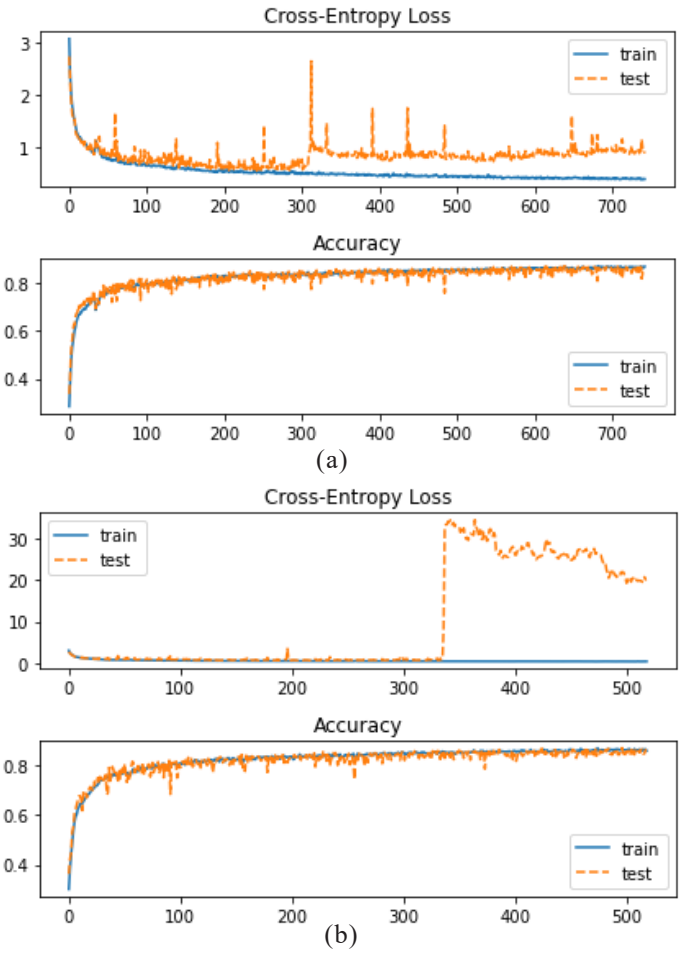

Figure 7. (a) Classification accuracy With BN after activation function with batch size 16, and (b) Classification accuracy with BN after activation function with batch size 16 using momentum.

6. BETA AND GAMMA PARAMETERS

Beta and Gamma parameters aim to restore the representation power of the networks. Since transforming input data to a 0-mean unit-variance distribution on its own, may affect what the layer represents, Table 8 shows that the beta and gamma parameters which are learnable parameters in batch normalisation provide the model with additional flexibility to adjust the mean and variance of the output features. They allow the model to learn biases and scaling factors that optimize the normalisation process and improve the model's performance. They provide automatic scaling and shifting of the standardized layer inputs.The graph showing the plots of classification accuracy and cross-entropy loss is depicted in Figure 8.

The results in Table 8 displays the classification accuracy using batch normalisation after activation with beta and gamma parameters

Table 8. Classification accuracy using batch normalisation after activation with beta and gamma parameters

Classification accuracy with beta and gamma parameters	
Training accuracy (%)	**Test accuracy (%)**
91.6	88.6

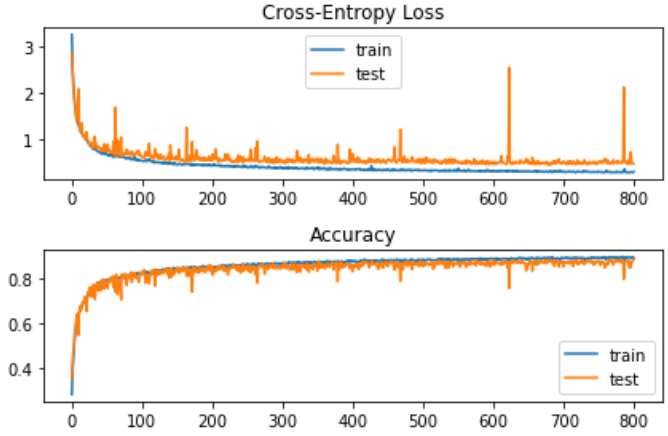

Figure 8. Cross-entropy & classification accuracy with BN using beta and gamma parameters.

6. CONCLUSION

This research paper explored the use of batch normalisation as a technique to quicken the training of deep neural networks for radar emitter identification. The experimental results demonstrated that batch normalisation significantly reduces the training time while maintaining or even improving the accuracy of radar emitter identification. The findings highlight the benefits of batch normalisation for training efficiency, convergence, and overall performance. The use of BN in radar emitter identification tasks has practical implications for real-world applications. The accelerated training process allows for faster model development, enabling quicker response times and improved system efficiency. The improved convergence and stability of the models contribute to better overall performance and robustness. There were some challenges faced during the training process as the batch normalisation process can be sensitive to the choice of learning rates. Using too high or too low learning rates might affect the convergence and stability of the training. However, some limitations with batch normalisation are there as batch normalisation introduces additional computations during both training and inference, which can increase the overall computational cost. While batch normalisation is generally beneficial, there are cases where it might not lead to improvements depending on the architecture and characteristics of the task or dataset.

REFERENCES

1. Bjorck, N.; Gomes, P.C.; Selman, B. & Weinberger, K.Q. Understanding batch normalisation. *Adv. Neural Inf. Process. Syst.*, 2018, **31**.
2. Ioffe, S. & Szegedy, S. Batch normalisation: Accelerating deep network training by reducing internal covariate shift. CoRR, 2015.
3. Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Planning and Inference*, 2000, **90**(2), 227 – 244. doi: 10.1016/S0378-3758(00)00115-4
4. Lange, S.; Helfrich, K. & Ye, Q. Batch normalisation preconditioning for neural network training. *J. Machine Learn. Res.,* 2022, **23**, 1-41
5. Luo, P.; Wang, X.; Shao, W. & Peng, Z. Towards

understanding regularisation in batch normalisation, 2018. (arXiv.org)
doi: 10.48550/arXiv.1809.00846

6. Huang, L.; Yang, D.; Lang, B. & Deng, J. Decorrelated batch normalisation. *In* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 791-800
doi: 10.1109/CVPR.2018.00089
7. Liao, Z. & Carneiro, G. On the Importance of normalisation layers in deep learning with piecewise linear activation units. CoRR, 2015.
doi: 10.1109/WACV.2016.7477624
8. Goodfellow; Bengio, Y. & Courville, A. Deep learning. Book in preparation for MIT Press, 2016.
doi: 10.1007/s10710-017-9314-z
9. Garbin, C.; Zhu, X. & Marques, O. Dropout vs. batch normalisation: An empirical study of their impact on deep learning. *Multimedia Tools and Appl.*, 2020, **79**, 12777-12815.
doi: 10.1007/s11042-019-08453-9
10. Laurent, C.; Pereyra, G.; Brakel, P.; Zhang, Y. & Bengio. Y. Batch normalized recurrent neural networks. *In* IEEE International Conference on Acoustics, 2016, pp. 2657-2661.
doi:10.1109/icassp.2016.7472159
11. Li, Y.; Wang, N.; Shi, J.; Hou, X. & Liu, J. Adaptive batch normalisation for practical domain adaptation. *Pattern Recognition*, 2018, **80,** 109-117.
doi:10.1016/j.patcog.2018.03.005
12. Yao, Z.; Cao, Y.; Zheng, S.; Huang, G. & Lin, S. Cross-iteration batch normalisation. *In* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.
doi:10.1109/cvpr46437.2021.01215
13. Jung, W.; Jung, D.; Kim, B.; Lee, S.; Rhee, W. & Ahn, J.H. Restructuring batch normalisation to accelerate CNN training. *In* Proceedings of the 2nd SysML Conference, 2019.
doi:10.48550/arXiv.1807.01702
14. Ismail, A.; Ahmad, S.A.; Soh, A.C.; Hasan, K. & Harith, H.H. Improving Convolutional Neural Network (CNN) Architecture (miniVGGNet) with batch normalisation and learning rate decay factor for image classification. *Int. J. Integrated Engin.*, 2019, **11,**51-59.
doi:10.30880/ijie.2019.11.04.006
15. Wu, S.; Li, G.; Deng, L.; Liu, L.; Wu, D; Xie, Y. & Shi, L. *L1*-Norm batch normalisation for efficient training of deep neural networks. *IEEE Transact. Neural Networks and Learn. Syst.*, 2018, **30**, 2043-2051.
doi: 10.1109/tnnls.2018.2876179
16. Gao, S.; Han, Q.; Li, D.; Cheng, M. & Peng, P. Representative batch normalisation with feature calibration. *In* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8669-8679.
doi:10.1109/cvpr46437.2021.00856
17. Yong, H.; Huang, J.; Meng, D.; Hua, X. & Zhang, L. Momentum batch normalisation for deep learning with small batch size. Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII, pp. 224–240.
doi:10.1007/978-3-030-58610-2_14

CONTRIBUTORS

Ms Preeti Gupta obtained her M.E. (Digital Systems) from NIT, Allahabad. She is working as a Scientist at DRDO-DLRL, Hyderabad. Her areas of research include: Design and development of an electronic warfare operational support system. Her contribution in the current study: She undertook a retrospective qualitative study to accelerate deep learning techniques for radar emitter identification.

Dr Pooja Jain obtained her PhD from JUIT, Solan. She is presently employed by the Department of Computer Science and Engineering at IIIT, Nagpur. Her area of interest include: Machine learning algorithms.

Dr O.G. Kakde obtained his PhD from Visvesvaraya National Institute of Technology (VNIT), Nagpur. He is currently working as a Director at IIIT, Nagpur. His area of interest include: Machine learning algorithms.
His contribution in the current study: He has given insightful suggestions and provided guidance and support as part of his contribution to the current study.