IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 6, December 2026, pp. 5279~5288
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i6.pp5279-5288

Impact of normalization techniques in convolutional neural networks for CIFAR-10 image classification

Abderrahmane Bouidi[1], Youssef Taki[2], Hmad Zennou[3], Lahcen Oughdir[1]
[1]Laboratory of Engineering, Systems, and Applications, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[2]Higher School of Technology of Khenifra (EST Khénifra), Sultan Moulay Slimane University, Khénifra, Morocco
[3]Information Processing and Decision Support Laboratory (TIAD), Sultan Moulay Slimane University, Beni Mellal, Morocco

Article Info

Article history:

Received Aug 4, 2025
Revised Aug 5, 2026
Accepted Aug 31, 2026

Keywords:

Convergence
Deep learning
Generalization accuracy
Image classification
Normalization techniques
Training stability

ABSTRACT

Normalization techniques are essential in deep neural networks (DNNs) that could deal with problems arising from unstable gradients, poor convergence, and weak generalization. This article looks at the effect of normalization on image classification tasks by choosing four famous techniques: batch normalization (BN), layer normalization (LN), instance normalization (IN), and group normalization (GN). It compares the four normalization techniques in terms of measuring their effects on training stability, convergence rate, and generalization accuracy within convolutional neural networks (CNNs). The experiment highlighted how each normalization method interacted with the training dynamics and generalization capabilities of the models, thus explaining their advantages and disadvantages. This indicates that the performance of high-dimensional image classification tasks involving deep networks depends critically on the correct selection of normalization techniques. The research has guided the selection of appropriate normalization techniques for specific contexts, helping researchers and practitioners build robust deep learning models.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Abderrahmane Bouidi
Laboratory of Engineering, Systems, and Applications, National School of Applied Sciences
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: Abderrahmane.bouidi@usmba.ac.ma

1. INTRODUCTION

Deep neural networks (DNNs) have revolutionized object detection and image classification, enabling automated systems to identify and classify complex visual patterns with unprecedented accuracy [1], [2]. Their multi-layered architecture enables DNNs to learn features from raw images hierarchically, ranging from simple edges to complex textures and shapes [3]. This hierarchical representation learning has accelerated progress in several applications, such as facial recognition [4], autonomous driving [5], medical diagnosis [6], and security monitoring [7].

Despite superior performance, image classification training using deep models still suffers from some problems [8]. Of particular concern is the vanishing gradient problem, which significantly hinders convergence, where the backpropagated gradients to the deep layers become exponentially small, disrupting weight updates [9]. Additionally, unstable training dynamics, expressed as oscillating convergence loss or failure, often arise due to improper weight initialization, inappropriate learning rates, or uncontrolled hyperparameters [10], [11]. These combined effects prevent effective optimization and impair model generalization. To address these difficulties, several approaches have been proposed, e.g., data augmentation [12], dropout

regularization [13], and normalization methods [14]. Among these approaches, batch normalization (BN) stands out for its ability to minimize the shift of internal covariates, thus stabilizing training dynamics and accelerating convergence [15]. BN normalizes intermediate-layer inputs to zero mean and unit variance in small batches, improving gradient flow and enabling deeper and more complex networks [16].

As a result, BN has become a popular building block in state-of-the-art architectures such as ResNet [17], DenseNet [18], and EfficientNet [18], facilitating improved generalization, optimization efficiency, and training stability. Furthermore, a wide range of normalization variants have been proposed to overcome the limitations of BN or extend normalization to custom settings [13], [19]. For instance, spectral normalization enhances the training stability of generative adversarial networks (GANs) [20], [21] by constraining the spectral norm of weight matrices [16], [22]. Layer normalization (LN), crucial to transformer models, normalizes across features rather than batch instances, enabling stable training of sequence models in natural language processing (NLP) [23]–[25].

Despite extensive development and widespread adoption, the normalization literature suffers from fragmentation, lacking a unified framework for systematic comparison and evaluation [26]. This gap impedes a comprehensive understanding of normalization techniques' relative strengths, limitations, and suitability for specific tasks. Particularly in image classification, where training complexity is heightened by deep architectures and diverse datasets, clarifying normalization's role remains vital.

This paper aims to bridge this gap by providing a thorough review and comparative analysis of prominent normalization methods applied to deep networks in image classification contexts. The proposed contributions are as follows:

- A unified perspective that categorizes normalization techniques based on their operational principles, normalization domains, and intended benefits.
- Extensive experiments on benchmark image datasets, systematically evaluating the impact of various normalization methods on training stability, convergence speed, and classification accuracy.
- The interplay between normalization approaches and network architectures, highlighting scenarios where specific techniques excel or underperform.
- Practical guidelines for selecting normalization strategies tailored to the characteristics of the datasets, network depth, and application requirements.

The remainder of this paper is organized as follows: section 2 reviews the normalization methods, section 3 describes the datasets and practical experience, section 4 presents comparative results and discussions, and section 5 concludes with insights and future research directions.

2. RELATED WORK

DNN training is a difficult operation. Researchers have developed a variety of techniques to speed up and stabilize the learning process over time. One method that has shown to be highly successful in achieving this is normalization. Deep learning begins with data preprocessing, a crucial step to prepare data for training. This process involves organizing and refining the data to ensure it is consistent and usable by the model. By addressing issues in the data and structuring it effectively, preprocessing helps the model learn patterns more efficiently and improves its overall performance. It sets the stage for successful training and accurate predictions. Data preprocessing [7], [27], [28] data mining, and other fields make extensive use of normalization. Depending on the subject, normalization may mean several things. In this study, normalization is defined as a general transformation that guarantees specific statistical properties in the transformed data. For greater precision, the following formal definition is provided.

Given a dataset (D), normalization can be defined as a function: D=$\{X^i\}_i^N$, the normalization operation is a function $\mathcal{N}$: X→ $\hat{X}$, which ensures that the transformed data $\widehat{D}$=$\{\hat{X}^i\}_i^N$ has certain statistical properties. It transforms the original data into a representation with specific statistical properties. Depending on the objective, this transformation may involve adjusting the data distribution to facilitate subsequent processing and improve the stability of learning algorithms. The five primary normalization processes are centering, scaling, decorrelating, standardizing, and whitening, as illustrated in Figure 1 [29].

Over the years, several normalization strategies have been developed to improve the stability and efficiency of deep learning models. Each strategy is characterized by its mathematical formulation, normalization axis, and specific application, depending on the network architecture and training requirements. Among the most widely used techniques are BN, LN, group normalization (GN), and instance normalization (IN). These techniques differ in the way they compute statistical quantities, such as the mean and variance, and in the dimensions over which normalization is performed. Figure 2 provides a comparative illustration of these four normalization methods, highlighting their respective normalization dimensions and the differences in how statistical information is calculated and applied. Understanding these differences is essential for selecting an appropriate normalization strategy for a given deep learning architecture.

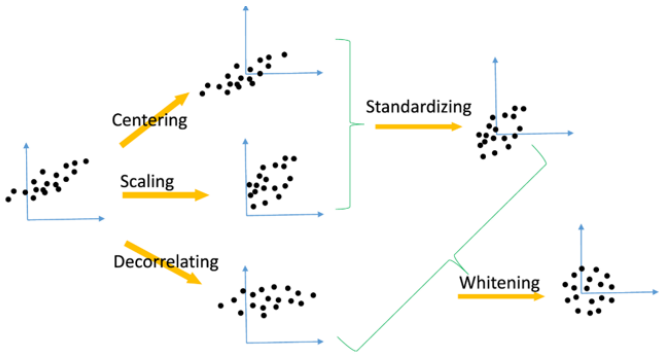


Figure 1. The main normalization operations

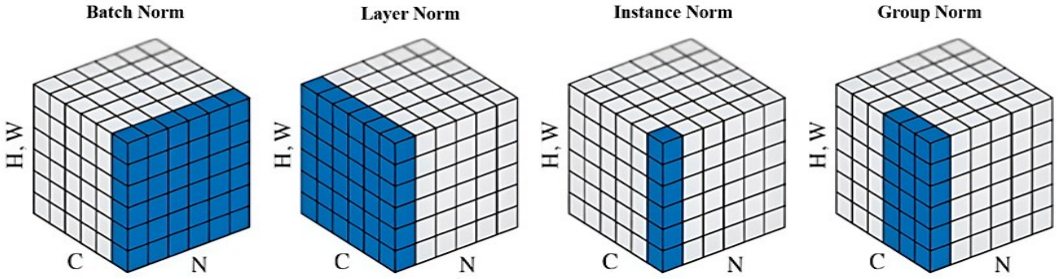


Figure 2. Comparative illustration of four normalization methods: BN, LN, IN, and GN

2.1. Batch normalization

BN is a technique designed to improve the training stability and efficiency of DNNs. This technique was introduced by Baldi and Sadowski [30]. It works by normalizing the activations of a layer for each mini-batch during training, ensuring that the output distribution remains consistent. Specifically, BN normalizes activations by subtracting the batch mean and dividing by the batch standard deviation, maintaining a zero mean and unit variance. To preserve the networks capacity to model complex relationships, BN introduces two learnable parameters: a scaling factor γ and a shift parameter β. These parameters allow the network to "denormalize" activations when necessary, enhancing its flexibility. By stabilizing activations, BN mitigates issues such as exploding or vanishing gradients, speeds up convergence, and enables the use of larger learning rates, ultimately improving both the training process and the model's generalization capabilities [30]. The steps involved in the BN process are summarized in Algorithm 1.

Algorithm 1. BN process
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1, \ldots, x_m\}$; parameters to be learned: γ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$
1: Compute mini-batch mean:

$$\mu_\beta \leftarrow (1/m)\sum_{i=1}^{m} x_i$$

2: Compute mini-batch variance

$$\sigma_\beta{}^2 \leftarrow (1/m)\sum_{i=1}^{m}(x_i - \mu_\beta)^2$$

3: Normalize:

$$\hat{x_i} \leftarrow (x_i - \mu_\beta)/\sqrt{\sigma_\beta{}^2 + \varepsilon}$$

4: Scale and shift:

$$y_i \leftarrow \gamma\hat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$$

2.2. Layer normalization

LN, as proposed by Ba *et al.* [13], computes the mean and variance for each instance independently, across all features, and operates over the feature dimension. It is frequently utilized in recurrent models where BN fails because, in contrast to BN, it is independent of batch size. Unlike BN, which blends throughout the batch, it concentrates on each data point separately. For every data point, it calculates the mean and variance among the features in a layer. This guarantees that every sample is handled separately, which makes it particularly useful in models with variable input sequence lengths, such as transformers or RNNs. The LN formula is defined as (1).

$$\hat{x}_i = \frac{x_i + \mu_{LN}}{\sigma_{LN}} + \beta \tag{1}$$

Where x_i represents the input activation while μ_{LN}, μ_{LN}, and σ_{LN} denote the mean and standard deviation computed across all features of a single input sample, respectively. The normalized activation is then scaled by the learnable parameter γ and shifted by the learnable bias β, producing the normalized output $\hat{x}_i$.

2.3. Group normalization

GN, a compromise strategy that splits the channels into smaller groups, was implemented by Wu and He [15]. In contrast to LN, which applies normalization along the feature direction, GN separates the features into distinct groups and normalizes each group independently. GN whose parameter num-groups is adjusted as a hyperparameter, outperforms LN in practice. The formula for GN is defined as (2).

$$\hat{x}_g = \frac{x_g + \mu_g}{\sigma_g} \cdot \gamma_g + \beta_g \tag{2}$$

This is a basic formula in which: x_g refers to the input, μ_g and σ_g are the mean and standard deviation of the group, and γ_g and β_g are learnable parameters.

2.4. Instance normalization

IN is a normalization technique applied to each instance in a batch independently. Unlike BN, which computes a single mean and standard deviation across the batch, IN calculates separate mean and standard deviation for each instance in the batch. Specifically, for a feature map of shape (C, H, and W), the normalization is applied across the spatial dimensions (H and W) for each channel C. The normalized output for each instance is given by (3).

$$\hat{x}_{(i,c,h,w)} = \frac{x_{i,c,h,w} + \mu_{i,c}}{\sigma_{i,c}} \cdot \gamma_c + \beta_c \tag{3}$$

Where $\mu_{i,c}$ is the mean of the instance, $\sigma_{i,c}$ is the standard deviation of the instance γ_c and β_c are learnable parameters for scaling and shifting.

3. RESULTS AND DISCUSSION

Here, a detailed analysis of the training results of DNNs using various approaches is presented. The analysis is organized to address key aspects of model performance and training dynamics, providing a comprehensive evaluation of each method. First, training stability is examined by analyzing the convergence behavior of the different approaches, since initialization strategies, optimization methods, and normalization techniques can substantially influence the training dynamics of DNNs [11], [31]. Second, model performance is compared using accuracy and loss curves to assess learning effectiveness and convergence quality. Third, the generalization capabilities of the models are evaluated based on their performance on unseen data, taking into account the potential differences between training and test performance [32]. Batch-size sensitivity is also investigated, as batch size can influence optimization behavior and generalization performance. Finally, the computational and memory costs of the different approaches are compared to assess their practical applicability, given the importance of computational efficiency in deep learning [33]. This systematic analysis provides insights into the strengths and limitations of the techniques under consideration.

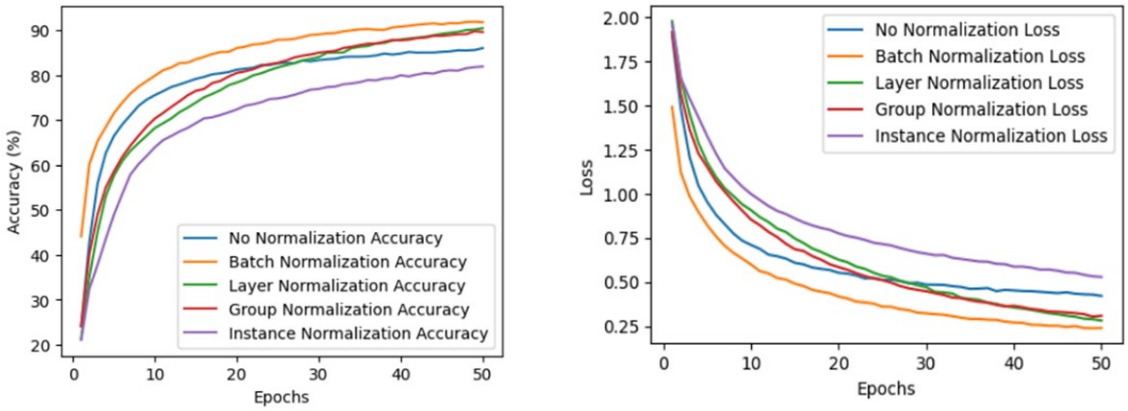
3.1. Experimental setup

To establish a comparison between the impact of normalization methods on image classification, experiments on the CIFAR-10 and CIFAR-100 datasets. CIFAR-10, consisting of 60,000 images over 10 classes, features a simple classification problem, while CIFAR-100, consisting of 100 classes, adds

complexity. A basic convolutional neural network (CNN) was utilized with two convolutional layers, a max-pooling layer, and a fully connected output layer. Normalized and non-normalized models were both trained, including BN, LN, and GN. Hyperparameters like a batch size of 64, a learning rate of 0.001, and 10 training epochs were used to evaluate performance, as well as metrics like accuracy, convergence rate, and training stability. All implementations took advantage of PyTorch flexibility and robust model discovery capabilities. For comparison of how varying normalizing methods affect model performance, the results and comparisons in the next section.

3.2. Training stability

Training stability determines how well a model can maintain its learning dynamics without significant drift or fluctuations during optimization. Figure 3 illustrates the training accuracy and loss curves under different normalization strategies, providing an overview of the model's learning dynamics and stability during optimization. Figure 3(a) presents the accuracy evolution over epochs and Figure 3(b) presents the corresponding loss behavior. Together, these curves demonstrate the model's learning efficiency and convergence stability during training.



(a) (b)

Figure 3. Comparison of training performance using different normalization methods: (a) training accuracy and (b) training loss

The trend in training dynamics provides some insight into convergence characteristics and stability, although it fails to quantify as much about each normalization techniques overall performance as can be done through final classification accuracy, which is shown in Figure 4. The final classification accuracy test provides a side-by-side comparison to measure how model generalization is affected by each of the five normalization methods used in this study. This comparison enables a direct evaluation of the effectiveness of these normalization techniques in developing a well-generalized model on the CIFAR-10 dataset.

By analyzing the training loss and accuracy curves, it is clear that normalization techniques play a pivotal role in improving model performance, significantly outperforming unnormalized training. BN has been proven to be the most effective method, showing the fastest convergence of the loss and achieving the highest final accuracy of over 90%. This demonstrates BN's ability to stabilize gradients during backpropagation, prevent vanishing or exploding gradients, and accelerate the learning process by standardizing intermediate activations within each mini-batch. This results in faster optimization and improved generalization. LN and GN also demonstrate significant improvements over unnormalization. LN achieves smoother and more stable convergence across timescales, effectively normalizing data along feature dimensions, which is particularly useful for tasks involving small and variable batch sizes. Similarly, GN, which partitions features into smaller batches for normalization, offers similar performance to GN, demonstrating its strength in scenarios with limited batch sizes or where data distribution varies. In contrast, training without normalization results in slower convergence rates, higher residual losses, and lower final accuracy, reaching a plateau of around 80%. This reflects optimization challenges due to unstable gradients and the difficulty of aligning weights effectively, which often leads to inefficient learning and poor generalization to unseen data. These results underscore the critical role of normalization techniques in deep learning. Choosing the right normalization method is critical not only for improving

training efficiency but also for ensuring model stability and optimal performance, especially on complex, high-dimensional tasks.

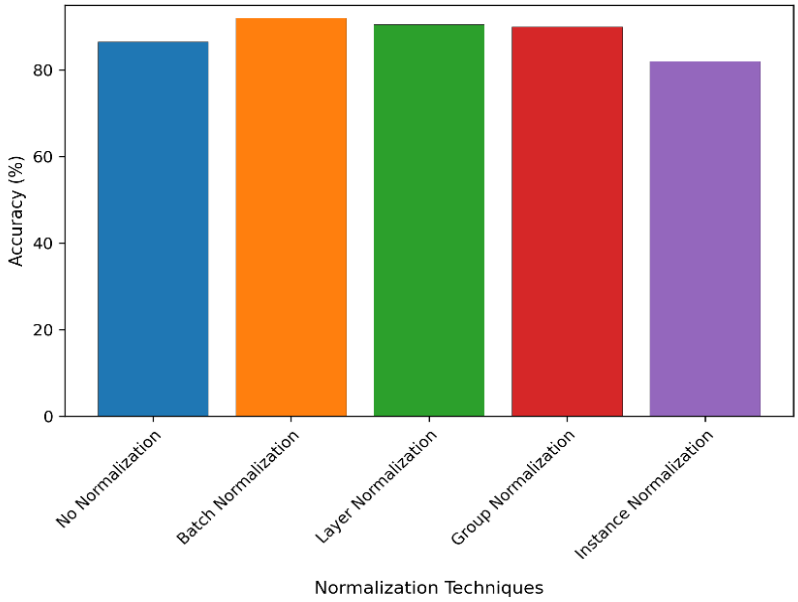


Figure 4. Final accuracy of five normalization techniques on CIFAR-10

3.3. Batch size sensitivity

Batch size sensitivity refers to the extent to which the performance and behavior of a machine learning model or training technique depend on the choice of batch size. The batch size determines the number of training samples processed before the model parameters are updated during the learning process. Different batch sizes can affect the model's convergence speed and training stability, particularly during optimization. Furthermore, batch size can influence the model's generalization ability, as small and large batches may lead to different learning dynamics and performance on unseen data.

This section investigates the sensitivity of four commonly used normalization techniques—BN, LN, GN, and IN—to different batch sizes. It examines how variations in batch size affect the performance and training stability of each normalization technique. Furthermore, the analysis considers the potential impact of batch size on the generalization ability of the resulting models. Understanding these relationships is essential for selecting the most appropriate normalization technique for different batch size configurations and improving the effectiveness of deep learning models across various tasks.

In the image classification task, Table 1 presents the training and testing accuracies obtained using different normalization techniques and batch sizes. The evaluated batch sizes are 8, 16, 32, and 64, allowing the influence of batch size on model performance to be examined. The comparison includes five configurations: no normalization, BN, LN, GN, and IN. These results provide a basis for comparing the performance of the different normalization techniques under varying batch size configurations.

Table 1. Comparative analysis of normalization techniques on CIFAR-10 across different batch sizes

Batch size	8		16		32		64	
	Train accuracy	Test accuracy	Train accuracy	Test accuracy	Train accuracy	Test accuracy	Train accuracy	Test accuracy
No normalization	57.08	62.32	73.8	72.85	81.20	80.95	86	81.68
BN	87.09	85.56	90.06	86.57	91.82	86.21	91.77	86.92
LN	85.34	77.56	91.74	81.14	92.61	81.32	90.44	81.27
GN	87.8	84.03	84.96	90.36	91.23	84.75	89.61	85.53
IN	60.58	66.77	72.56%	74.11	85.10	72.49	81.9	73.4

As the batch size grows, no normalization demonstrates a notable improvement in both training and testing accuracy. Both the testing and training accuracy increase from 62.32% to 81.68% and 57.08% to

86.00%, respectively, suggesting that higher batch sizes aid in stabilizing training and enhancing performance without normalization. All batch sizes benefit from BN's excellent training and testing accuracy, which ranges from 87.09% to 91.82% for training and from 85.56% to 86.92% for testing. This implies that, independent of batch size, Batch Norm efficiently stabilizes training and improves generalization.

While testing accuracy is still lower, at about 81%, LN achieves excellent training accuracy, peaking at 92.61% with a batch size of 32. This suggests that although LN stabilizes training well, it might not generalize as well as BN. With testing accuracy ranging from 84.03% to 90.36% and training accuracy between 84.96% and 91.23%, GN performs admirably. Strong generalization is demonstrated by this method, especially at a batch size of 16, when testing accuracy approaches 90.36%, indicating that it can manage lesser batch sizes.

At small batch sizes, IN results in poorer training and testing accuracies (60.58% and 66.77%, respectively). However, as batch sizes increase, these metrics improve dramatically, reaching 85.10% for training and 73.40% for testing. IN still performs worse overall than alternative normalizing techniques, though. Overall, BN and GN work well over a range of batch sizes, with BN showing somewhat better stability and generalization. LN works well in training but poorly in testing, while IN often performs poorly, especially when batch sizes are smaller.

3.4. Results analysis

Table 2 compares several normalization methods under the image classification task. With moderate learning stability and low batch size sensitivity, BN achieves the best test accuracy, indicating strong generalization for this task. Class normalization exhibits excellent learning stability and moderate batch size sensitivity, making it reliable for stable learning even with moderate test accuracy. Moreover, ensemble normalization is effective, showing low batch size sensitivity while maintaining excellent accuracy and stability. In contrast, IN exhibits greater batch size sensitivity and lower test accuracy and stability, suggesting that it may be less useful for image classification. It is important to remember that these results are specific to image classification and may not apply to other tasks, such as NLP, where different normalization methods may exhibit distinct behaviors due to the diverse data types and model architecture used in these domains.

Table 2. Performance evaluation of normalization techniques for CIFAR-10 image classification

Technique	Testing accuracy	Training stability	Batch size sensitivity
No normalization	Low	Moderate	High
BN	High	Moderate	Low
LN	Moderate	High	Moderate
GN	High	High	Moderate
IN	Low	Low	High

3.5. Critical discussion and implications

Comparative analysis of the normalization techniques in CIFAR-10 involves critical trade-offs between stability, batch sensitivity, and accuracy, a master set of determinants of deep learning model resilience and efficacy. BN performed excellently across all, with the highest accuracy, quickest convergence rate, and consistent training patterns, confirming its foundational role in CNN-based image recognition. But BN's batch-statistic dependency limits its application in the case of small or dynamic batch sizes, say edge computing or real-time computing. GN then came forth as a strong contender, particularly under low-batch scenarios, since it is batch-independent in nature and thus naturally memory- or latency-constrained compatible.

LN showed very stable training but lagged in end accuracy, indicating its better suitability to sequential or NLP tasks rather than spatial feature learning. IN worked worst for image classification because of its batch sensitivity and lower generalization but is yet useful in style transfer and generative modeling. All in all, these results point towards the need to use normalization methods tailored to the task, architecture, and deployment scenario.

4. CONCLUSION

The results of this research indicate that the technique used for normalizing inputs during the training of a deep convolutional neural nets (DCNN) for image classification has a significant impact on the degree of training stability, speed of convergence, and overall effectiveness of the model. While BN produced the fastest training speed and largest final accuracy, layer and GN produced improved training

stability as well as better final accuracies. IN was effective with small minibatch sizes but was limited in its ability to perform to the same standards on a larger scale. These results support the importance of considering the requirements of a specific task, the size of the training minibatch, and the desired stability of the training when selecting a normalization method for use with a deep learning model. These findings can be beneficial when deploying a DCNN-based deep learning model in an applied setting, such as mobile applications or embedded systems, in which training is necessary to be both efficient and stable. As future work, hybrid or adaptive normalization techniques could be investigated, as well as their impacts on the newer transformer-based vision networks, and a further comparison could be made between the performance and cost of resource constrained DCNN models.

ACKNOWLEDGMENTS

The authors are thankful to National School of Applied Sciences, University of Sidi Mohamed Ben Abdellah University for APC support.

FUNDING INFORMATION

The authors declare that no funding was received for the research, authorship, or publication of this article.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Abderrahmane Bouidi	✓	✓	✓		✓	✓	✓		✓	✓			✓	✓
Youssef Taki		✓		✓				✓	✓					
Hmad Zennou						✓		✓		✓	✓			
Lahcen Oughdir		✓		✓	✓					✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known financial, personal, or professional conflicts of interest that could have influenced the work reported in this manuscript. The authors further affirm that this study was conducted with academic integrity and objectivity, and that the findings presented are based solely on the research conducted.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

Ethical approval was not required for this study because it involved only publicly available image datasets and did not involve human participants, animals, or personal identifiable information.

DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author, [AB], upon reasonable request.

REFERENCES

[1] J. Brownlee, *Deep learning for computer vision: image classification, object detection, and face recognition in Python*. Vermont, Victoria, Australia: Machine Learning Mastery, 2019.

[2] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, Apr. 1998, doi: 10.1142/S0218488598000094.

[3] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, 1998, pp. 9–50, doi: 10.1007/3-540-49430-8_2.

[4] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, Technical Report, University of Toronto, Toronto, Canada, 2009.

[5] S. Sharma and K. Guleria, "Deep learning models for image classification: comparison and applications," in *2022 2nd International Conference on Advanced Computing and Innovative Technologies in Engineering (ICACITE)*, Apr. 2022, pp. 1733–1738, doi: 10.1109/ICACITE53722.2022.9823516.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[7] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[8] X. Zhong, O. Gong, W. Huang, L. Li, and H. Xia, "Squeeze-and-excitation wide residual networks in image classification," *Proceedings - International Conference on Image Processing, ICIP*, pp. 395–399, 2019, doi: 10.1109/ICIP.2019.8803000.

[9] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[10] M. Tan and Q. V. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," *Proceedings of Machine Learning Research*, vol. 97, pp. 6105–6114, 2019.

[11] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.

[12] S. Ioffe, "Batch renormalization: towards reducing minibatch dependence in batch-normalized models," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1945–1953, 2017.

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[14] T. Salimans and D. P. Kingma, "Weight normalization: a simple reparameterization to accelerate training of deep neural networks," in *Proc. 30th Int. Conf. Neural Information Processing Systems (NIPS)*, 2016, pp. 901–909, doi: 10.5555/3157096.3157197.

[15] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19, doi: 10.1007/978-3-030-01261-8_1.

[16] Z. Lin, V. Sekar, and G. Fanti, "Why spectral normalization stabilizes GANs: analysis and improvements," *Advances in Neural Information Processing Systems*, vol. 12, pp. 9625–9638, 2021.

[17] L. Huang *et al.*, "Controllable orthogonalization in training DNNs," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 6428–6437, doi: 10.1109/CVPR42600.2020.00646.

[18] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training DNNs: methodology, analysis and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, 2023, doi: 10.1109/TPAMI.2023.3250241.

[19] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[20] T.-Y. Lin *et al.*, "Microsoft COCO: common objects in context," in *Computer Vision–ECCV 2014*, Cham: Springer, 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[21] X. Chang *et al*., "ShapeNet: an information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[22] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2014, pp. 2672–2680.

[23] B. Saha, "Resource-aware GAN training on cloud infrastructure for large-scale image and video synthesis," *International Journal of Progressive Research in Engineering Management and Science*, vol. 5, no. 4, 2025, doi: 10.58257/ijprems39830.

[24] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly, "A large-scale study on regularization and normalization in GANs," *Proceedings of Machine Learning Research*, vol. 97, pp. 3581–3590, 2019.

[25] A. Vaswani *et al*., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[26] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, 2019, pp. 4171–4186, doi: 10.18653/v1/n19-1423.

[27] T. Wolf *et al.*, "Transformers: state-of-the-art natural language processing," in *EMNLP 2020 - Conference on Empirical Methods in Natural Language Processing, Proceedings of Systems Demonstrations*, 2020, pp. 38–45, doi: 10.18653/v1/2020.emnlp-demos.6.

[28] A. Mumuni and F. Mumuni, "Data augmentation: a comprehensive survey of modern approaches," *Array*, vol. 16, Dec. 2022, doi: 10.1016/j.array.2022.100258.

[29] A. Kessy, A. Lewin, and K. Strimmer, "Optimal whitening and decorrelation," *The American Statistician*, vol. 72, no. 4, pp. 309–314, Oct. 2018, doi: 10.1080/00031305.2016.1277159.

[30] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2013, pp. 2814–2822.

[31] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning*, vol. 28, no. 3, 2013, pp. 1139–1147.

[32] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *5th International Conference on Learning Representations*, 2017, pp. 1–16.

[33] B. R. Bartoldson, B. Kailkhura, and D. Blalock, "Compute-efficient deep learning: algorithmic trends and opportunities," *The Journal of Machine Learning Research*, vol. 24, no. 1, pp. 5465–5541, 2023, doi: 10.5555/3648699.3648821.

BIOGRAPHIES OF AUTHORS



Abderrahmane Bouidi obtained his master's degree in Decision Support Information Systems and Imaging from the Faculty of Science and Technology of Errachidia, Moulay Ismail University of Meknes, Morocco, in September 2022. He also earned a bachelor's degree in Computer Science, specializing in software engineering, from the same faculty in February 2000. Currently, he is a Ph.D. student at the National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco. His research focuses on artificial intelligence (AI) and deep learning, particularly on the development of a functional architecture for object classification based on deep learning techniques. He also holds a diploma as a Secondary Education Inspector in Computer Science from the Inspector Training Center for Education in Rabat, Morocco. He can be contacted at email: abderrahmane.bouidi@usmba.ac.ma.



Youssef Taki received the bachelor's degree in Applied Mathematics in 2017 and the master's degree in Modeling and Scientific Computing for Mathematical Engineering in 2019. He holds a Ph.D. degree from Sultan Moulay Slimane University, Morocco. His research focuses on advanced driver assistance systems (ADAS), intelligent transportation systems, road safety, and computer vision. He can be contacted at email: y.taki@usms.ma.



Hmad Zennou was awarded his master's degree in Software Engineering in 2021 from the Faculty of Sciences in Kenitra, Ibnou Tofail University, Morocco. Prior to this, in 2020, he obtained a diploma as a Secondary Education Inspector in Computer Science from the Inspector Training Center for Education in Rabat, Morocco. Currently, he is a fourth-year doctoral candidate at the Faculty of Science and Technology (FST) in Beni Mellal, Sultan Moulay Slimane University, Morocco. His research focuses on artificial intelligence (AI) and deep learning, with a particular emphasis on emotion recognition. He has conducted work on emotion recognition via audio signals, utilizing deep learning algorithms such as LSTM and Bi-LSTM, among others. Furthermore, his research extends to online course recommendation systems based on user emotions. He can be contacted at email: hmannou2007@gmail.com.



Lahcen Oughdir is a professor at the National School of Applied Sciences (ENSA) in Fez and a mission officer at the Presidency of Sidi Mohamed Ben Abdellah University of Fez. Specializing in mathematics and computer science, he focuses on machine learning and deep learning problems. He has authored over 37 articles indexed in Scopus and supervised more than 9 successfully defended Ph.D. students, in addition to being a member of the jury for over 30 Ph.D. theses. He is deeply involved in academic research and the development of the next generation of experts in his field, contributing to both teaching and the advancement of technology at the university level. He can be contacted at email: lahcen.oughdir@usmba.ac.ma.