Context normalization: A new approach for the stability and improvement of neural network performance

Bilal Faye, Hanane Azzag, Mustapha Lebbah, Fangchen Feng

Data & Knowledge Engineering 155 (2025) 102371

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Context normalization: A new approach for the stability and improvement of neural network performance

Bilal Faye [a,*], Hanane Azzag [a], Mustapha Lebbah [b], Fangchen Feng [c]

[a] LIPN, Sorbonne Paris Nord University, Villetaneuse, 93430, France
[b] Laboratoire DAVID, Paris Saclay University, Versailles, 78035, France
[c] L2TI, Sorbonne Paris Nord University, Villetaneuse, 93430, France

ARTICLE INFO

Keywords:
Deep neural network
Activation normalization
Gaussian mixture model

ABSTRACT

Deep neural networks face challenges with distribution shifts across layers, affecting model convergence and performance. While Batch Normalization (BN) addresses these issues, its reliance on a single Gaussian distribution assumption limits adaptability. To overcome this, alternatives like Layer Normalization, Group Normalization, and Mixture Normalization emerged, yet struggle with dynamic activation distributions. We propose "Context Normalization" (CN), introducing contexts constructed from domain knowledge. CN normalizes data within the same context, enabling local representation. During backpropagation, CN learns normalized parameters and model weights for each context, ensuring efficient convergence and superior performance compared to BN and MN. This approach emphasizes context utilization, offering a fresh perspective on activation normalization in neural networks. We release our code at https://github.com/b-faye/Context-Normalization.

1. Introduction

Normalization is a standard data processing operation [1] that equalizes variable amplitudes, aiding single-layer network convergence [2]. In multilayer networks, data distribution changes necessitate various normalization techniques, including activation, weight, and gradient normalization. Batch Normalization (BN) [3] stabilizes multilayer neural network training by standardizing layer activations using batch statistics. While it enables higher learning rates, BN is limited by batch size dependence and the assumption of uniform data distribution. Specialized variants of BN have been proposed to address batch size-related limitations [4]. Mixture Normalization (MN) [5] accommodates data samples from various distributions, enhancing convergence compared to BN, particularly in convolutional neural networks. MN initially employs the Expectation–Maximization (EM) [6] algorithm to estimate parameters for each mixture component and subsequently normalizes samples within the same mixture component using these parameters during deep neural network training. However, the use of the EM algorithm can increase computation time, thus diminishing efficiency.

To address this issues, we introduce a novel normalization method, Context Normalization (CN). CN incorporates prior knowledge structures known as "contexts", grouping similar samples. These contexts, defined by experts or derived from clustering algorithms, can include classes, superclasses, or domains in domain adaptation scenarios. CN operates on the hypothesis that activations follow a Gaussian mixture model, normalizing them during training to estimate parameters for each mixture component. While CN assumes a Gaussian distribution for coherence in comparison, alternative hypotheses can be considered in different scenarios. As an integral

* Corresponding author.
E-mail addresses: faye@lipn.univ-paris13.fr (B. Faye), azzag@univ-paris13.fr (H. Azzag), mustapha.lebbah@uvsq.fr (M. Lebbah), fangchen.feng@univ-paris13.fr (F. Feng).

https://doi.org/10.1016/j.datak.2024.102371
Received 10 April 2024; Received in revised form 19 September 2024; Accepted 22 October 2024
Available online 15 November 2024

layer, CN standardizes activations from the same context using parameters learned through backpropagation. This process enhances data representation's discriminative capacity for the target task. This study introduces three significant contributions:

- We introduce "Context Normalization" (CN), leveraging expert-defined contexts to supervise and estimate normalization parameters. CN, integrated as a neural network layer, estimates parameters for latent Gaussian mixture components associated with these contexts.
- We observe CN's versatility across diverse deep neural network architectures like Transformers and Convolutional Neural Networks. Extensive experiments consistently show its ability to accelerate training and improve generalization.
- We propose CN's application in domain adaptation, addressing the challenge of enhancing model performance across varying data distributions. CN proves adept at tackling this challenge effectively.

2. Related work

To maintain clarity and consistency in our model and enable easy comparison with prior work, we employ the same notations as [5]. Consider $x \in \mathbb{R}^{N\times C\times H\times W}$, a 4-D activation tensor in a convolutional neural network, where N, C, H, and W represent batch size, channels, height, and width respectively.

Batch normalization (BN) [3] operates on the mini-batch $B = \{x_{1:m} : m \in [1, N] \times [1, H] \times [1, W]\}$, with x flattened across all dimensions except channels:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{1}$$

$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ and $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ are the mean and variance of B, where $\epsilon > 0$ is a small value addressing numerical instabilities. Eq. (1) enforces a zero mean and unit variance distribution if mini-batch samples are drawn from the same distribution. This helps stabilize activation distributions, aiding in training. While effective, BN is sensitive to mini-batch size and assumes a single Gaussian distribution. Many variants and alternatives have emerged to tackle these issues: Layer Normalization (LN) [7], Instance Normalization (IN) [8], Group Normalization (GN) [9], and Mixture Normalization (MN) [5].

LN is designed for recurrent neural networks, IN focuses on style removal in individual samples like images, GN divides activations into groups for normalization, and MN utilizes Gaussian Mixture Models (GMM) to handle multi-modal activation distributions.

Consider the vector $i = (i_N, i_C, i_L)$ indexing $x \in \mathbb{R}^{N\times C\times L}$, where the spatial domain is flattened such that $L = H \times W$. The general normalization, $x \to \hat{x}$, as proposed in [5], is expressed as follows:

$$v_i = x_i - \mathbb{E}_{B_i}(x), \quad \hat{x}_i = \frac{v_i}{\sqrt{\mathbb{E}_{B_i}(v^2) + \epsilon}}, \tag{2}$$

where B_i is a set of indices. For instance, in the case of BN, $B_i = \{j : j_N \in [1, N], j_C \in [i_C], j_L \in [1, L]\}$.

In the MN algorithm, each activation x_i is normalized using the mean and standard deviation of its corresponding mixture component. The probability density function p_θ is modeled as a Gaussian Mixture Model (GMM).

For $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$, the density function is given by:

$$p(x) = \sum_{k=1}^{K} \lambda_k p(x|k), \quad \text{s.t. } \forall k : \lambda_k \geq 0, \ \sum_{k=1}^{K} \lambda_k = 1,$$

where

$$p(x|k) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}{2}\right)$$

is the kth component, with μ_k as the mean vector, Σ_k the covariance matrix, and D the dimensionality of x. The probability that x was generated by the kth Gaussian component can be defined as follows:

$$\tau_k(x) = p(k|x) = \frac{\lambda_k p(x|k)}{\sum_{j=1}^{K} \lambda_j p(x|j)}.$$

Based on these assumptions and the general transformation in Eq. (2), the normalization of x_i is defined as follows:

$$\hat{x}_i = \sum_{k=1}^{K} \frac{\tau_k(x_i)}{\sqrt{\lambda_k}} \hat{x}_i^k, \tag{3}$$

with

$$v_i^k = x_i - \mathbb{E}_{B_i}[\hat{\tau}_k(x).x], \ \hat{x}_i^k = \frac{v_i^k}{\sqrt{\mathbb{E}_{B_i}[\hat{\tau}_k(x).(v^k)^2] + \epsilon}}, \tag{4}$$

where $\hat{\tau}_k(x_i) = \frac{\tau_k(x_i)}{\sum_{j\in B_i}\tau_k(x_j)}$, is the normalized contribution of x_i in estimating the statistics of the kth Gaussian component. With this approach, MN can be applied in two steps:

1. Estimation of the mixture model parameters θ using the EM algorithm [6].
2. Normalization of each x_i with respect to the estimated parameters (Eqs. (4),(3))

3. Proposed method: Context Normalization (CN)

This section will be organized as follows: first, we will describe the general concept of CN in Section 3, and then we will provide detailed instructions on implementing our proposed method for activation normalization in Section 3.2.

3.1. Method description

CN effectively models data assuming a mixture of Gaussian components in activation space. In this approach, a "context" is defined as a group of samples sharing similar characteristics. *Contexts can be defined by experts or derived from clustering algorithms, and may include classes, superclasses, or domains in domain adaptation scenarios*. Samples within the same context are grouped and normalized using parameters learned by the neural network during backpropagation, eliminating the need for expensive EM algorithms. Each context is identified by a unique identifier r, and the normalization is handled by trainable parameters μ_r and σ_r. These parameters, learned during training as neural network weights, can be represented collectively as $\theta = \{\mu_r, \sigma_r\}_{r=1}^{T}$, where T denotes the number of contexts. CN follows the formulation in Eq. (2), where $B_i = \{j : j_N \in [i_N], j_C \in [i_C], j_L \in [1, L]\}$.

3.2. Learning parameters for each context

Consider $i = (i_N, i_C, i_L)$ as a vector indexing the tensor of activations $x \in \mathbb{R}^{N \times C \times L}$. Each activation x_i undergoes normalization using the parameters $\theta_{r_i} = \{\mu_{r_i}, \sigma_{r_i}\}$, where r_i denotes the context identifier associated with x_i:

$$\mathrm{CN}_{\theta_{r_i}} : \hat{x}_i \leftarrow \frac{x_i - \mu_{r_i}}{\sqrt{\sigma_{r_i}^2 + \epsilon}} \tag{5}$$

The mean (μ_{r_i}) and standard deviation (σ_{r_i}) are learned as neural network weights through activation normalization resulting from transforming samples of context r_i, as outlined in Algorithm 1.

Algorithm 1: CN- Transformation: Activation Normalization

Input : activation x_i; context r_i associated with x_i
Output: $\hat{x}_i = \mathrm{CN}_{\theta_{r_i}}(x_i)$ where $\theta_{r_i} = \{\mu_{r_i}, \sigma_{r_i}\}$
$\alpha_{r_i} \leftarrow onehot(r_i)$ // *encoding the categorical variable* r_i

$\mu_{r_i} \leftarrow Encoder_{\mu}(\alpha_{r_i})$ // *mean estimation of context* r_i

$\sigma_{r_i}^2 \leftarrow Encoder_{\sigma}(\alpha_{r_i})$ // *variance estimation of context* r_i

$\hat{x}_i = \mathrm{CN}_{\theta_{r_i}}(x_i)$: Normalize x_i using Equation (5)

In Algorithm 1, we employ an affine transformation of the form:

$$\begin{cases} \alpha_{r_i} \leftarrow onehot(r_i) \\ \mu_{r_i} \leftarrow Encoder_{\mu}(\alpha_{r_i}) : W_{\mu}\ \alpha_{r_i} + b_{\mu} \\ \sigma_{r_i}^2 \leftarrow Encoder_{\sigma}(\alpha_{r_i}) : W_{\sigma}\ \alpha_{r_i} + b_{\sigma} \end{cases} \tag{6}$$

In Eq. (6), W and b are the parameters of $Encoder$, and r_i is the context associated with x_i. The function onehot(.) performs one-hot encoding of the categorical variable r_i, yielding α_{r_i}.

When training a neural network with a CN layer, following Algorithm 2, it is vital to propagate the gradient of the loss function ℓ through the transformation. Additionally, computing gradients with respect to the parameters of the CN transform is crucial. This involves applying the chain rule, as depicted in the following expression (before simplification):

$$\frac{\partial \ell}{\partial \mu_{r_i}} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial \mu_{r_i}} = -\frac{\partial \ell}{\partial \hat{x}_i} \cdot (\sigma_{r_i}^2 + \epsilon)^{-1/2}$$

$$\frac{\partial \ell}{\partial \sigma_{r_i}^2} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial \sigma_{r_i}^2} = \frac{\mu_{r_i} + x_i}{2(\sigma_{r_i}^2 + \epsilon)^{3/2}}$$

.

CN normalizes neural network activations, aiding convergence and creating Gaussian mixture components for task-specific representations in a latent space.

Algorithm 2: CN- Training on a activations mini-batch

Input : Deep neural network Net with trainable parameters Θ; subset of activations and its contexts $\{x_i, r_i\}_{i=1}^{m}$, with $r_i \in \{1, ..., T\}$, where T is the number of contexts; randomly initialized mixture components parameters $\theta = \{\mu_r, \sigma_r\}_{r=1}^{T}$

Output: Context Normalized deep neural network, Net_{CN}^{tr}

$Net_{CN}^{tr} = Net$ // *Initialize deep neural network parameters*

for $i \leftarrow 1$ *to* m **do**

- Apply Algorithm 1: $\hat{x}_i = \mathrm{CN}_{\theta_{r_i}}(x_i)$
- Modify each layer in Net_{CN}^{tr} with input x_i to take $\hat{x}_i$ instead

Propagate the gradient through Net_{CN}^{tr} to optimize: $\Theta = \Theta \cup \{\mu_r, \sigma_r\}_{r=1}^{T}$

During inference, we can opt to normalize activations using CN when the context of the input is known, or alternatively, we can consider all contexts collectively when the input context is unknown. This enhancement is referred to as CN+ and is described in Algorithm 3. For any activation x_i, reformulating Eq. (7), CN+ is expressed as follows:

Algorithm 3: CN- Inference

Input : Deep Neural Network Net_{CN}^{tr} with parameters $\Theta \cup \{\mu_r, \sigma_r\}_{r=1}^{T}$ (ref. Algorithm 2); subset of activations and its contexts $\{x_i, r_i\}_{i=1}^{m}$, with $r_i \in \{1, ..., T\}$, where T is the number of contexts; $choice \in$ {CN, CN+};

Output: Deep Neural network with frozen parameters Net_{CN}^{inf}; $\{\hat{x}_i\}$ normalized activations

$Net_{CN}^{inf} \leftarrow Net_{CN}^{tr}$ // *Initialization of the deep neural network for inference with frozen parameters*

if $choice = CN$ **then**

for $i \leftarrow 1$ *à* m **do**

- Retrieve the parameters associated with the context of x_i: θ_{r_i}
- transform $\hat{x}_i = CN_{\theta_{r_i}}(x_i)$ using Algorithm 1

if $choice = CN+$ **then**

for $i \leftarrow 1$ *à* m **do**

- Compute $\hat{x}_i$ using Equation (7) // *Normalization + aggregation*

$$\hat{x}_i = \sqrt{T} \sum_{r=1}^{T} \tau_r(x_i) \hat{x}_i^r, \tag{7}$$

with

$$\tau_r(x_i) = p(r|x_i) = \frac{p(x_i|r)}{T \sum_{j=1}^{T} p(x_i|j)}, \quad \hat{x}_i^r = \frac{x_i - \mu_r}{\sigma_r},$$

assuming that the prior probabilities ($\lambda_r = \frac{1}{T}, r = 1, \ldots, T$) are constant.

4. Experiments

In this section, we compare context normalization (CN) with other methods, including batch normalization (BN) and mixture normalization (MN), across different architectures, including Convolutional Neural Networks (CNNs) (see Sections 4.1 and 4.3) and Vision Transformers (ViT) [10] (see Section 4.2). We evaluate these approaches on various tasks, including classification and domain adaptation.

The experiments in this study are based on several commonly used benchmark datasets in classification. **CIFAR-10** and **CIFAR-100** each contain 50,000 training images and 10,000 test images, with a size of 32 × 32 pixels [11,11]. CIFAR-100, an extension of CIFAR-10, is divided into 100 classes grouped into 20 superclasses. **Tiny ImageNet** is a reduced version of ImageNet with 200 classes [12]. **MNIST digits** contains 70,000 images representing the 10 digits [13]. **SVHN** focuses on digit recognition in natural scenes, totaling over 600,000 images [14]. This consolidation optimizes space while preserving essential information.

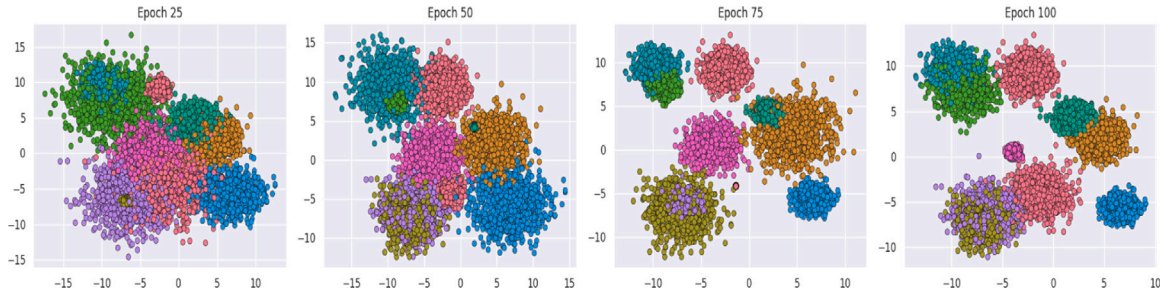


Fig. 1. t-SNE is applied for dimensionality reduction in the latent space after 25, 50, 70, and 100 training epochs, with the first two components visualized. This visualization illustrates the formation and refinement of class-specific clusters throughout the training process with Context Normalization (CN) on CIFAR-10.

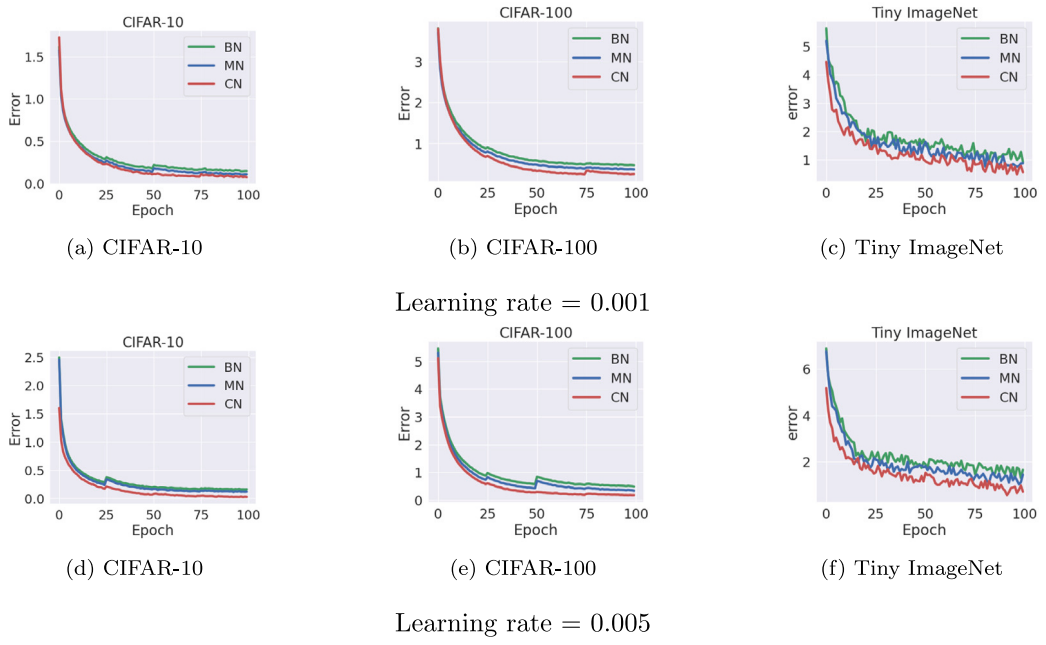


Fig. 2. Test error curves when ConvNet architecture is trained under different learning rate.

4.1. Comparative study: Context normalization vs. Mixture normalization

In this experiment, we use a shallow Convolutional Neural Network (ConvNet) architecture described in the MN paper [5]. The network comprises four convolutional layers with ReLU activation, each followed by a batch normalization layer. We highlight a challenge of batch normalization (BN) related to the use of non-linear functions (e.g., ReLU) after activation normalization. By employing ConvNet, we demonstrate how Context Normalization (CN) tackles this issue, enhancing convergence and overall performance by replacing a BN layer with CN within the ConvNet.

During training on CIFAR-10, CIFAR-100, and Tiny ImageNet datasets, we use the MN method, which estimates a Gaussian mixture model through Maximum Likelihood Estimation (MLE). To facilitate comparison, we utilize the three components discovered by the Expectation–Maximization (EM) algorithm on MN as distinct contexts ($T = 3$) for CN. This enables the normalization of activations within each context. We vary the learning rate from 0.001 to 0.005, use a batch size of 256, and train for 100 epochs with the AdamW optimizer [15,16]. To compare normalization methods, we replace the third BN layer in ConvNet with an MN layer and repeat this process for CN layer.

During CIFAR-10 training, snapshots were taken every 25 epochs using CN normalization. These snapshots were then applied to a random CIFAR-10 batch, and the resulting activations were visualized using t-SNE. The visualizations revealed clusters corresponding to the target classes predicted by the model. Improved clustering was observed with training progression, leading to enhanced performance, as shown in Fig. 1.

Fig. 2 illustrates that CN achieves faster convergence compared to BN and MN. This accelerated convergence leads to improved performance on the validation dataset, with an average increase of **2%** in accuracy on CIFAR-10, **3%** on CIFAR-100, and **4%** on Tiny ImageNet when using CN and CN+ during inference (see Algorithm 3). This positive trend persists across different numbers of

Table 1
Evaluating CIFAR-100 performance with ViT architecture [17] integrating BN, and CN with superclasses as distinct contexts.

Model	Accuracy	Precision	Recall	f1-score
ViT+BN	55.63	8.96	90.09	54.24
ViT+CN	**65.87**	**23.36**	**98.53**	**65.69**

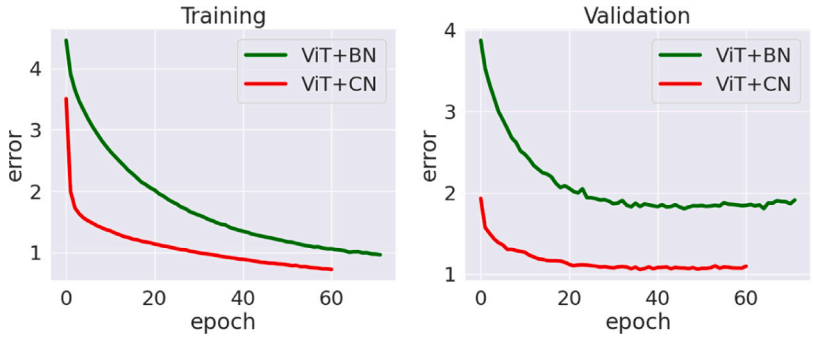


Fig. 3. Comparing training and validation error curves: CN in ViT Architecture on CIFAR-100 show faster convergence and lower validation loss, enhancing learning efficiency and classification compared to BN.

classes (10, 100, 200), even with an increase in the learning rate from 0.001 to 0.005. Increasing the learning rate widens the gap in convergence, demonstrating the effectiveness of our normalization technique in leveraging higher learning rates during training.

The approach in this experiment uses Gaussian components found via MLE as contexts, but it is cumbersome. In the following sections, we will simplify the context definition without resorting to complex algorithms. MN will no longer be used, as it requires preconstruction of Gaussian components (a limitation). Since CN behaves similarly to CN+, it will be favored for inference, being more resource-efficient.

To assess the impact of the number of contexts in CN, we tested $T = \{4, 6, 8\}$ and found similar performance to $T = 3$. This suggests that increasing T is not always necessary and depends more on dataset complexity, task, and neural network architecture. In subsequent experiments, we will demonstrate that increasing T can be beneficial for a more robust model, particularly on the CIFAR-100 dataset.

4.2. Image classification: Leveraging CIFAR-100 superclasses as contexts

The key innovation in the experiment is utilizing CIFAR-100 superclasses as distinct contexts ($T = 20$) for predicting the dataset's 100 classes, particularly with the CN method. We used two models: the base ViT model from Keras [17], which includes a Batch Normalization (BN) layer as its initial component, and a model that replaced the BN layer with a CN layer. Training employed early stopping based on validation performance, and images were pre-processed by normalizing them with respect to the dataset's mean and standard deviation. Data augmentation techniques such as horizontal flipping and random cropping were applied to enhance the dataset. We employed the AdamW optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-4} to prevent overfitting and optimize model parameters [15,16].

Table 1 illustrates significant performance gains with our novel Context Normalization over BN in training the ViT architecture from scratch on CIFAR-100. CN yields an accuracy boost of approximately **10%** compared to BN, showcasing faster convergence and superior performance. This is supported by the training and validation error comparison in Fig. 3, indicating accelerated learning with CN. These results suggest that CN not only stabilizes data distributions and mitigates internal covariate shift but also reduces training time for enhanced results. ViT+CN achieves outstanding performance, surpassing all known ViT models trained from scratch on CIFAR-100.

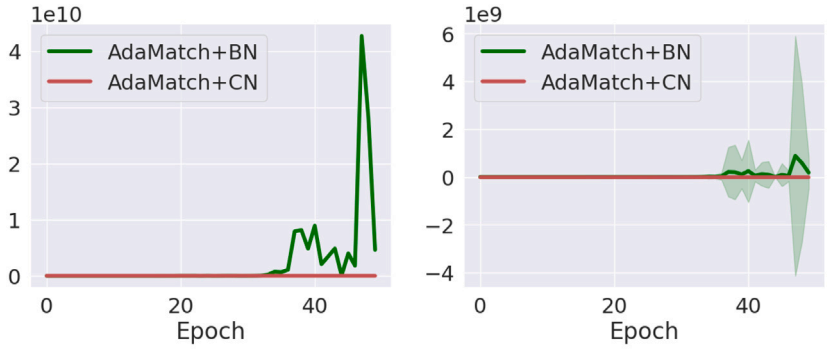
This experiment demonstrates a simple method for creating contexts without algorithms, accelerating convergence and improving neural network performance. However, establishing superclass structure is not always straightforward. The next section presents a typical application of CN, where contexts become evident.

4.3. Domain adaptation: Leveraging distinct domains as contexts

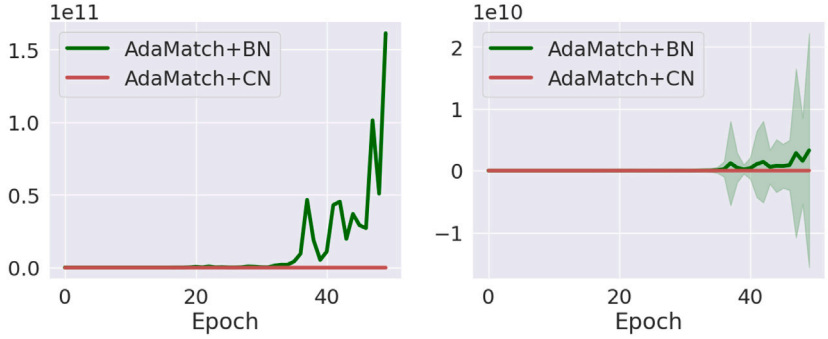
In this experiment, we show that context normalization's ability to enhance local representations can greatly improve domain adaptation. Domain adaptation, as outlined in [18], involves leveraging knowledge from a related domain with sufficient labeled data to improve performance in a target domain with limited labeled data. *Here, we consider two contexts $T = 2$: the "source domain" and the "target domain".*

Table 2
Comparing model performance: AdaMatch+BN vs. AdaMatch+CN on MNIST (source) and SVHN (target) datasets.

MNIST (source domain)				
Model	Accuracy	Precision	Recall	f1-score
AdaMatch+BN	97.36	87.33	79.39	78.09
AdaMatch+CN	**99.26**	**99.20**	**99.32**	**99.26**
SVHN (target domain)				
Model	Accuracy	Precision	Recall	f1-score
AdaMatch+BN	25.08	31.64	20.46	24.73
AdaMatch+CN	**43.10**	**53.83**	**43.10**	**47.46**



(a) Source domain (MNIST)



(b) Target domain (SVHN)

Fig. 4. Evolution of gradient variance on the source (MNIST) and target (SVHN) domains. The left figures show the maximum gradient variance across all layers per epoch, while the right figures display the average gradient variance across all layers per epoch.

To demonstrate this, we utilize CN alongside AdaMatch [19], which combines unsupervised domain adaptation (UDA), semi-supervised learning (SSL), and semi-supervised domain adaptation (SSDA). In UDA, we have labeled data from the source domain and unlabeled data from the target domain, aiming to train a model that can generalize effectively to the target dataset despite distribution variations between the two domains. The objective is to train a model capable of generalizing effectively to the target dataset.

It is important to note that the source and target datasets exhibit variations in distribution. Specifically, we utilize the MNIST dataset as the source domain, while the target domain is SVHN. Both datasets encompass various factors of variation, including texture, viewpoint, appearance, etc., and their domains, or distributions, are distinct from each other.

A reference model (AdaMatch) [20], trained from scratch with wide residual networks [21] using batch normalization on dataset pairs, is employed. The model is trained with the Adam optimizer and a cosine decay schedule to decrease the initial learning rate (initialized at 0.03). CN serves as the initial layer in AdaMatch, incorporating the domain information (source and target) into the image normalization process. Leveraging known labels in the source domain, the model provides a better representation of this domain compared to the target domain, where labels are unknown. This advantage is utilized during inference to enhance the model's performance on the target domain.

In a broader context, indicates a substantial enhancement in validation metrics with context normalization. This is reflected in an accuracy increase of **18.02%** with CN, notably boosting the performance of the AdaMatch model and accelerating convergence during training.

Fig. 4 illustrates the beneficial impact of context normalization in stabilizing the gradient during training, benefiting both the source and target domains. This leads to faster convergence and an overall enhancement in model performance.

5. Limitations and future work

The CN framework requires the number of contexts, denoted by T, to be specified as a hyperparameter. Similar to many clustering algorithms, determining the optimal value for T can be challenging. Our experimental results in Section 4 demonstrate that when the number of contexts is well-defined, such as superclasses in Section 4.2 and domains in Section 4.3, CN exhibits better performance and faster convergence. However, when the number of contexts is not predefined, as in Section 4.1, a clustering algorithm is used to group the data into clusters, treating each cluster as a separate context. In such cases, determining the optimal number of contexts becomes difficult. A common, yet naive, approach is to vary T across a range of values to find the best one.

In future work, we plan to further investigate scenarios where the number of contexts is unknown. We aim to develop methods for more effectively defining and selecting the optimal number of contexts to maintain performance, while avoiding the need to test numerous values of T, which can be computationally expensive.

6. Conclusion

Context Normalization (CN) represents a significant advancement in neural network training. Sharing a similar philosophy with Mixture Normalization (MN), CN eliminates the need for the costly Expectation–Maximization (EM) algorithm estimation used by MN by directly learning the parameters of Gaussian distributions from named data groups called "contexts". Our experiments demonstrated that CN outperformed both Batch Normalization (BN) and Mixture Normalization (MN), offering better convergence and improved generalization performance. Three methods were explored to define contexts: using components from a Gaussian Mixture Model (GMM), identifying superclasses in a dataset, and applying domain adaptation by considering each domain as a distinct context.

In the short term, our aim is to create an unsupervised variant of CN by eliminating the need for context input, opting instead for its estimation during neural network training.

CRediT authorship contribution statement

Bilal Faye: Writing – review & editing, Writing – original draft, Visualization, Validation. **Hanane Azzag:** Writing – review & editing. **Mustapha Lebbah:** Writing – original draft. **Fangchen Feng:** Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Kessy, A. Lewin, K. Strimmer, Optimal whitening and decorrelation, Amer. Statist. 72 (4) (2018) 309–314.
[2] Y. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 2002, pp. 9–50.
[3] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[4] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, L. Shao, Normalization techniques in training dnns: Methodology, analysis and application, 2020, arXiv preprint arXiv:2009.12836.
[5] M.M. Kalayeh, M. Shah, Training faster by separating modes of variation in batch-normalized models, IEEE Trans. Pattern Anal. Mach. Intell. 42 (6) (2019) 1483–1500.
[6] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. Ser. B 39 (1) (1977) 1–38.
[7] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.
[8] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, 2016, arXiv preprint arXiv:1607.08022.
[9] Y. Wu, K. He, Group normalization, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 3–19.
[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929.
[11] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 (Canadian institute for advanced research), 2009, URL: http://www.cs.toronto.edu/~kriz/cifar.html.
[12] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231N 7 (7) (2015) 3.
[13] Y. LeCun, C. Cortes, MNIST handwritten digit database, 2010, URL: http://yann.lecun.com/exdb/mnist/.
[14] P. Sermanet, S. Chintala, Y. LeCun, Convolutional neural networks applied to house numbers digit classification, in: Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), IEEE, 2012, pp. 3288–3291.
[15] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2017, arXiv preprint arXiv:1711.05101.
[16] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[17] K. Salama, Implementing the vision transformer (ViT) model for image classification, 2021, https://github.com/keras-team/keras-io/tree/master.
[18] A. Farahani, S. Voghoei, K. Rasheed, H.R. Arabnia, A brief review of domain adaptation, Adv. Data Sci. Inf. Eng.: Proc. ICDATA 2020 IKE 2020 (2021) 877–894.
[19] D. Berthelot, R. Roelofs, K. Sohn, N. Carlini, A. Kurakin, Adamatch: A unified approach to semi-supervised learning and domain adaptation, 2021, arXiv preprint arXiv:2106.04732.
[20] S. Paul, Unifying semi-supervised learning and unsupervised domain adaptation with AdaMatch, 2019, https://github.com/keras-team/keras-io/tree/master.
[21] S. Zagoruyko, N. Komodakis, Wide residual networks, 2016, arXiv preprint arXiv:1605.07146.