ADAPTATIVE CONTEXT NORMALIZATION: A BOOST FOR DEEP LEARNING IN IMAGE PROCESSING

Bilal FAYE [1], Hanane AZZAG [1], Mustapha LEBBAH [2], Djamel BOUCHAFFRA [3]

[1] LIPN, UMR CNRS 7030, Sorbonne Paris Nord University, Villetaneuse, France
[2] Paris-Saclay University, UVSQ, David Lab, 78035, Versailles, France.
[3] Center for Development of Advanced Technologies, Algiers, Algeria

ABSTRACT

Deep Neural network learning for image processing faces major challenges related to changes in distribution across layers, which disrupt model convergence and performance. Activation normalization methods, such as Batch Normalization (BN), have revolutionized this field, but they rely on the simplified assumption that data distribution can be modelled by a single Gaussian distribution. To overcome these limitations, Mixture Normalization (MN) introduced an approach based on a Gaussian Mixture Model (GMM), assuming multiple components to model the data. However, this method entails substantial computational requirements associated with the use of Expectation-Maximization algorithm to estimate parameters of each Gaussian components. To address this issue, we introduce Adaptative Context Normalization (ACN), a novel supervised approach that introduces the concept of "context", which groups together a set of data with similar characteristics. Data belonging to the same context are normalized using the same parameters, enabling local representation based on contexts. For each context, the normalized parameters, as the model weights are learned during the backpropagation phase. ACN not only ensures speed, convergence, and superior performance compared to BN and MN but also presents a fresh perspective that underscores its particular efficacy in the field of image processing. We release our code at https://github.com/b-faye/Adaptative-Context-Normalization.

1. ITRODUCTION

Normalization, a common data processing operation [1], equalizes variable amplitudes, aiding single-layer network convergence [2]. In multilayer networks, data distribution changes necessitate various normalization techniques, such as activation, weight, and gradient normalization. Batch Normalization (BN) [3] is a common method for stabilizing multilayer neural network training by standardizing layer activations using batch statistics. While it allows for 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]. Another approach, Mixture Normalization (MN) [5], was introduced to handle data distribution assumptions. MN differs from BN by accommodating data samples from various distributions. It employs the Expectation-Maximization (EM) algorithm [6] to cluster data based on components and normalizes each sample using component-specific statistics. MN enhances convergence, particularly in convolutional neural networks, compared to BN. However, the use of the Expectation-Maximization (EM) algorithm in mixture normalization can lead to a significant increase in computation time, limiting its efficiency. EM estimates the parameters over the entire dataset before the training of the neural network, which enables the provision of normalization parameters.

To tackle this issue, we introduce a novel normalization method called Adaptative Context Normalization (ACN). ACN incorporates a prior knowledge structure called "context", which can be defined as a cluster or class of samples sharing similar characteristics. Building contexts relies on experts' knowledge to group data effectively, but it may result from a partitioning done with another clustering algorithm. For instance, in a dataset with classes and superclasses (groupings of classes), each superclass can be treated as a distinct context. In domain adaptation, each specific domain within the dataset can be considered as independent context. Operating on the hypothesis that activations can be represented as a Gaussian mixture model, ACN normalizes these activations during deep neural network training to estimate parameters for each mixture component. It's worth noting that the choice of a Gaussian distribution assumption is made to ensure a coherent basis for comparison. However, it is important to emphasize that alternative hypotheses for data distribution could also be considered in different scenarios and experiments. As an integral layer within the deep neural network, ACN plays a pivotal role in standardizing activations originating from the same context using the parameters acquired through backpropagation. This process facilitates the estimation of parameters for each mixture component, ultimately enhancing the data representation's discriminative

capacity with respect to the target task.
This study introduces three significant contributions:

- We introduce the concept of "context", formed by expert knowledge similar to superclasses or specific scenarios in image classification. Our innovative method, **ACN**, acts as a layer in the neural network. ACN supervises and estimates normalization parameters specific to Gaussian mixture components according predefined contexts.
- We apply ACN to various deep neural network architectures for image processing using Vision Transformer and Convolutional Neural Networks, serving as a layer at different levels. Extensive experiments consistently show that ACN accelerates training processes, improving generalization performance.
- We build upon ACN's strong image classification performance, extending its application to image domain adaptation. This aims to enhance model performance across diverse data distributions, effectively addressing the associated challenges.

2. RELATED WORK

2.1. Batch Normalization

To ensure clear understanding and consistency in our model, and to facilitate comparison with existing work, we adopt the same notations as those used in [5]. Let's consider $x \in \mathbb{R}^{N\times C\times H\times W}$, a 4-D activation tensor in a convolutional neural network (CNN), where the axes N, C, H, and W represent the batch size, number of channels, height, and width respectively. Batch normalization (BN) is performed on the mini-batch $B = \{x_{1:m} : m \in [1, N] \times [1, H] \times [1, W]\}$, where x is flattened across all but channel:

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

where $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ and $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ represent, respectively, the mean and variance of B, while $\epsilon > 0$ is a small value that handles numerical instabilities. If samples within the mini-batch are from the same distribution, the transformation shown in Equation 1 generates a zero mean and unit variance distribution. This constraint allows stabilizing the distribution of the activations and thus benefits training.

2.2. Variants of Batch Normalization

Several extensions of BN [4] have been proposed, including Layer Normalization (LN). LN is tailored for recurrent neural networks and aims to eliminate dependencies among activations by normalizing each neuron activation with respect to its respective layer. Another technique is Instance Normalization (IN), which normalizes each sample individually, focusing on removing style information, especially in images. IN improves the performance of specific deep neural networks and finds widespread application in tasks like image style transfer. Similarly, Group Normalization (GN), divides neuron activations into groups and independently normalizes activations in these groups. Like IN, GN is effective for visual tasks with small batch sizes, such as object detection and segmentation. Lastly, Mixture Normalization (MN) [5], which uses a Gaussian Mixture Model (GMM) to normalize activations based on multiple modes of variation in their underlying distribution, suitable when the distribution of activations exhibits multiple modes. The general transformation $x \to \hat{x}$ is shared across all these variants and is applied on the flattened x along the spatial dimension $L = H \times W$, as follows:

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

where $B_i = \{j : j_N \in [1, N], j_C \in [i_C], j_L \in [1, L]\}$, and $i = (i_N, i_C, i_L)$ a vector indexing the activations $x \in \mathbb{R}^{N\times C\times L}$.
In MN algorithm, each x_i in B_i is normalized with the mean and standard deviation of the mixture component it belongs to. The probability density function p_θ can be represented as a GMM. Let $x \in \mathbb{R}^D$, and $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, ..., K\}$, then we have:

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

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 k-th component, μ_k is the mean vector, and Σ_k is the covariance matrix. The probability that x was generated by the k-th 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 Equation (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 k-th 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 (Equations(4),(3))

3. PROPOSED METHOD: ADAPTATIVE CONTEXT NORMALIZATION

In this framework, clusters of samples sharing common characteristics and behaviors are defined by the "context". Each context is characterized by a unique input scalar (r), and activations originating from samples within a specific context undergo normalization. This normalization process is facilitated through parameters $\theta_r = \{\mu_r, \sigma_r\}$, which are acquired through neural network backpropagation. To ensure coherence, we adhere to the assumption of Gaussian mixture distributions in activations, aligning with the predefined understanding of contexts. It is crucial to note that various experiments may explore alternative hypotheses regarding data distribution or normalization methods.
Let x be an activation tensor in the $\mathbb{R}^{N\times C\times H\times W}$ space, where B_i represents a group of activations resulting from flattening x along the $L = H \times W$ axis. Each x_i within B_i is normalized along the C axis, using the parameters associated with its context, $\theta_{r_i} = \{\mu_{r_i}, \sigma_{r_i}\}$:

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

The parameters $\theta = \{\mu_r, \sigma_r\}_{r=1}^T$, learned for each of the T mixture components, undergo updates during the normalization process applied to the activations from the corresponding context.Each normalized activation $\hat{x}_i$ can be viewed as an input to a sub-network composed of the linear transform (Equation 5), followed by the other processing done by the original network. During the training process, as outlined in Algorithm 1, it is necessary to propagate the gradient of the loss function ℓ, through the transformation. Furthermore, it is essential to calculate the gradients with respect to the parameters of the ACN transform. This computation is accomplished using the chain rule, as illustrated in the following expression (before simplification):

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

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

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

During inference, we can either normalize activations based on their corresponding context as shown in Algorithm 1

Algorithm 1: Training an Adaptative Context-Normalized Network

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

Output: Adaptative Context-Normalized deep neural network for inference, Net^{inf}_{ACN}

$Net^{tr}_{ACN} = Net$ // *Training ACN deep neural network*

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

- Add transformation $\hat{x}_i = ACN_{\theta_{r_i}}(x_i)$ to Net^{tr}_{ACN} (Equation 5)
- Modify each layer in Net^{tr}_{ACN} with input x_i to take $\hat{x}_i$ instead

Train Net^{tr}_{ACN} to optimize the parameters: $\Theta = \Theta \cup \{\mu_r, \sigma_r\}_{r=1}^T$

$Net^{inf}_{ACN} = Net^{tr}_{ACN}$ // *Inference ACN network with frozen parameters*

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

- Retrieve the parameters associated with the context of x_i: $\theta_{r_i} = \{\mu_{r_i}, \sigma_{r_i}\}$
- transform $\hat{x}_i = ACN_{\theta_{r_i}}(x_i)$ using Equation 5

or treat all contexts collectively (as mixture normalization), by transforming Equation 3, as follows:

$$\hat{x}_i = \sqrt{T}\sum_{r=1}^{T} \tau_r(x_i)\hat{x}^r_i, \tag{6}$$

with $v^r_i = x_i - \mathbb{E}_{B_i}[\hat{\tau}_r(x).x]$, $\hat{x}^r_i = \frac{v^r_i}{\sqrt{\mathbb{E}_{B_i}[\hat{\tau}_r(x).(v^r)^2]+\epsilon}}$, where $\hat{\tau}_r(x_i) = \frac{\tau_r(x_i)}{\sum_{j\in B_i}\tau_r(x_j)}$, In Equation 6, T represents the number of contexts, and we assume constant prior probabilities ($\lambda_r = \frac{1}{T}, r = 1, ..., T$).

4. EXPERIMENTS

In the proposed experiments, we will assess the performance of the adaptative context normalization method ACN and compare it to BN and MN. We also evaluate our results against ACN-base, a simplified variant of ACN. In ACN-base, two separate neural networks estimate the T context parameters $\theta = \{\mu_r, \sigma_r\}_{r=1}^T$. For a given input x_i in the ACN-base layer, the context identifier r_i of x_i is initially encoded using one-hot encoding and serves as input for both networks. The first neural network outputs μ_{r_i}, while the second provides $\sigma^2_{r_i}$, which are then used to normalize x_i. The experiments will cover different architectural setups, including Convolutional Neural Networks (CNNs) (as described in

Sections 4.2, and 4.4) and Vision Transformers (ViT) [7] (as outlined in Section 4.3). These approaches will be evaluated across a range of tasks, including classification and domain adaptation.

4.1. Datasets

The experiments conducted in this study utilize several commonly used benchmark datasets in the classification community, including:

- **CIFAR-10 and CIFAR-100:** These datasets include 50,000 training images and 10,000 test images, each sized 32×32 pixels. CIFAR-10 features 10 distinct classes, while CIFAR-100 has 100 classes organized into 20 superclasses [8].
- **Tiny ImageNet:** A dataset that is a reduced version of the ImageNet dataset, containing 200 classes with 500 training images and 50 test images per class [9].
- **MNIST digits:** Contains 70,000 grayscale images of size 28×28 pixels representing the 10 digits, with approximately 6,000 training images and 1,000 test images per class [10].
- **SVHN:** A dataset with over 600,000 digit images, focused on digit and number recognition in natural scene images [11].

These datasets provide a diverse set of challenges for evaluating the proposed approaches in various tasks.

4.2. A Comparative Study: Adaptative Context Normalization vs. Mixture Normalization

In this experiment, we employ a shallow deep Convolutional Neural Network (ConvNet) architecture as described in the MN paper. This network consists of four convolutional layers with ReLU activation, each followed by a batch normalization layer. One challenge associated with batch normalization (BN) is the utilization of non-linear functions (e.g., ReLU) subsequent to activation normalization. Through the application of ConvNet, we showcase how Adaptative Context Normalization (ACN) addresses this issue and improves convergence and overall performance by substituting a BN layer with ACN within the ConvNet.

During training on CIFAR-10, CIFAR-100, and Tiny ImageNet datasets, we employ the MN method, which involves estimating a Gaussian mixture model through Maximum Likelihood Estimation (MLE). For comparison purposes, the three components discovered by the Expectation-Maximization (EM) algorithm on MN are utilized as distinct contexts ($T = 3$) for ACN and ACN-base, enabling the normalization of samples 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 [12, 13]. To compare normalization methods, we replace the third BN layer in ConvNet with an MN layer and repeat this process for ACN and ACN-base layers.

Figure 1 demonstrates that ACN and its smaller variant, ACN-base, exhibit faster convergence compared to BN and MN. This accelerated convergence results in 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. This positive trend persists across different numbers of classes (10, 100, 200), even with an increase in the learning rate from 0.001 to 0.005. Increasing the learning rate exacerbates the gap in convergence, demonstrating the capacity of our normalization technique to effectively leverage higher learning rates during training.

In our upcoming experiments, we aim to demonstrate that adaptative context normalization can be implemented in a single step for specific scenarios, offering a potential reduction in time complexity compared to the more intricate mixture normalization. However, it's worth noting that in the experiments described in the following sections, we do not utilize Gaussian components as context, rendering MN inapplicable as a baseline for comparison.

4.3. Using Superclasses as Contexts in Adaptative Normalization

In our adaptative context normalization approach, superclasses serve as prior knowledge. Each superclass represents a distinct context. ACN is integrated into the Vision Transformer (ViT) for CIFAR-100 classification using Keras baseline [14]. The core innovation in the experiment lies in the use of CIFAR-100 superclasses as contexts for predicting the dataset's 100 classes, specifically in the case of ACN method.

We employed three distinct models: the base ViT model sourced from Keras [14], a modified version incorporating a Batch Normalization (BN) layer as its initial component, and two alternative models that replaced the BN layer with ACN-base and ACN layers. Training employed early stopping based on validation performance, and images were preprocessed 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. The AdamW optimizer, with a learning rate of 10^{-3} and a weight decay of 10^{-4}, was selected to prevent overfitting and optimize model parameters [12, 13]. Table 1 showcases the substantial performance improvements achieved by our novel Adaptative Context Normalization (ACN), when compared to Batch Normalization (BN) training the ViT architecture from scratch on CIFAR-100. ACN approach exhibit an accuracy improvement of approximately **12%** over BN. Additionally, it's worth noting that ACN-base and ACN demonstrate faster convergence compared to BN, requiring less training time to achieve superior performance.

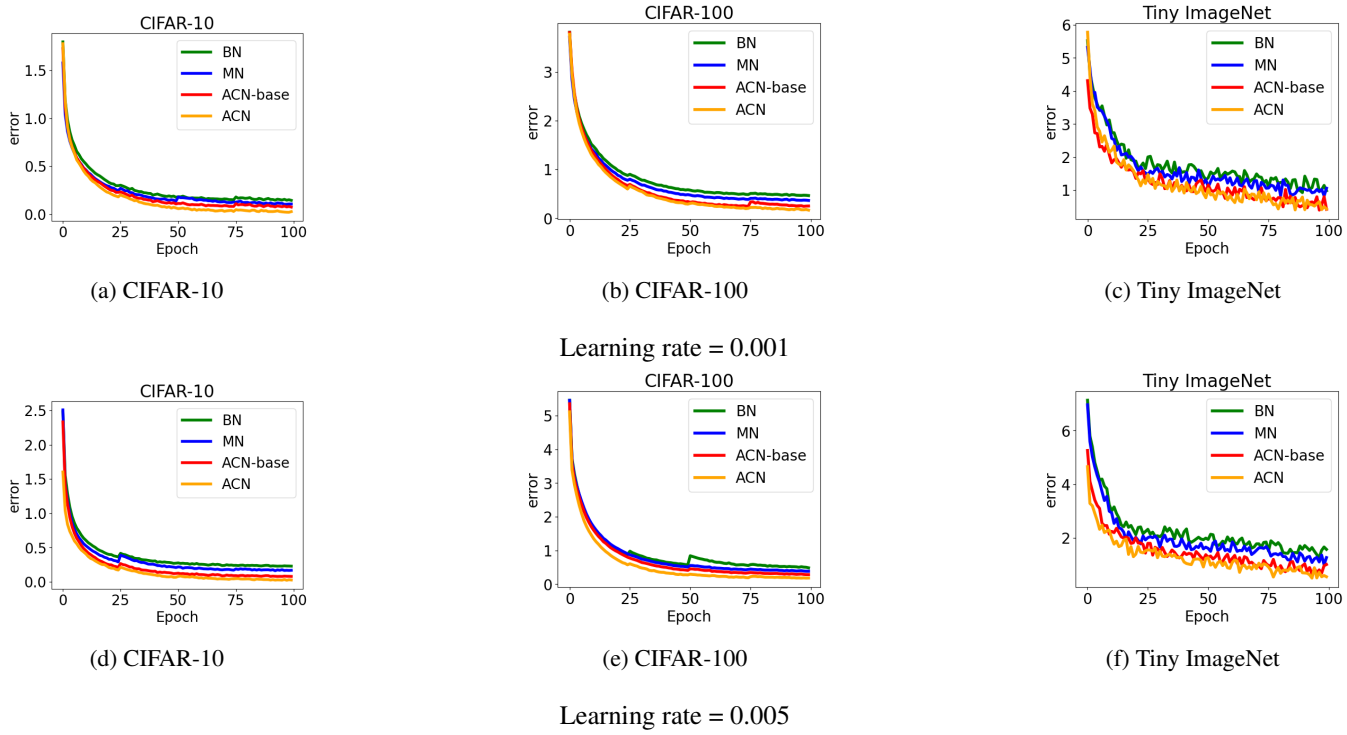


Fig. 1: Test error curves when ConvNet architecture is trained under different learning rate.

model	accuracy	precision	recall	f1-score
ViT+BN	55.63	8.96	90.09	54.24
ViT+ACN-base	65.87	**23.36**	98.53	65.69
ViT+ACN	**67.38**	**22.93**	**99.00**	**67.13**

Table 1: Evaluating CIFAR-100 Performance with ViT Architecture [14] integrating BN, ACN-base, and ACN with superclasses as contexts.

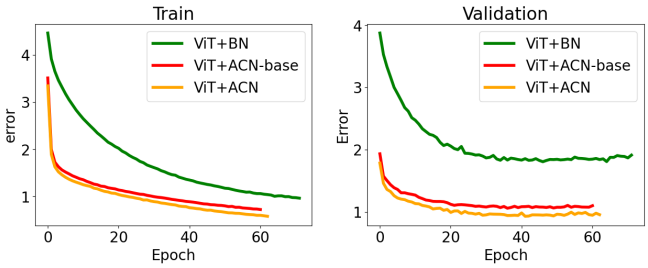


Fig. 2: Comparing Training and Validation Error Curves: ACN-base and ACN in ViT Architecture on CIFAR-100 show faster convergence and lower validation loss, enhancing learning efficiency and classification compared to BN.

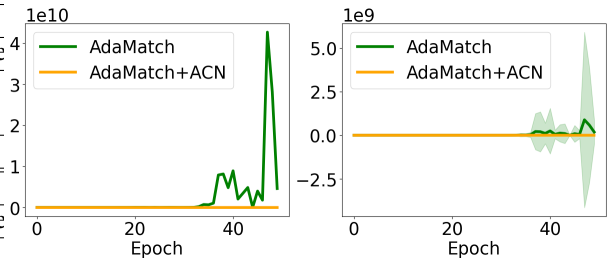
This observation is further supported by the train and validation loss comparison depicted in Figure 2, highlighting that ACN facilitates accelerated learning while enhancing classification performance. These findings suggest that the proposed method ACN not only stabilizes data distributions and mitigates internal covariate shift but also significantly reduces training time for improved results. ViT+ACN-base and ViT+ACN achieve outstanding performance, surpassing all known ViT models when trained from scratch on the CIFAR-100 dataset.

4.4. Domain Adaptation: Using Source and Target Domains as Contexts

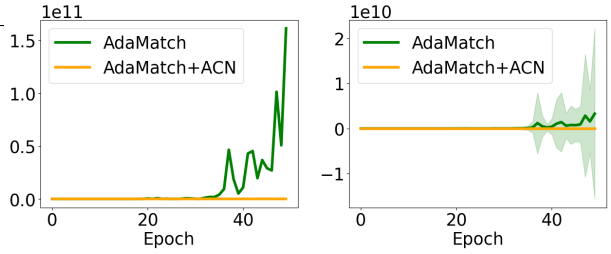
In this experiment, we demonstrate that adaptative context normalization's (ACN) proficiency in enhancing local representations can lead to significant improvements in domain adaptation. Domain adaptation, as explained in [15], involves leveraging knowledge acquired by a model from a related domain, where there is sufficient labeled data, to enhance the model's performance in a target domain with limited labeled data. In this scenario, we consider two contexts: the "source domain" and the "target domain". To illustrate this, we apply ACN in conjunction with AdaMatch [16], a method that combines the tasks of unsupervised domain adaptation (UDA), semi-supervised learning (SSL), and semi-supervised domain adaptation (SSDA). In UDA, we have access to a labeled dataset from the source domain and an unlabeled dataset from the target domain. The goal is to train a model that can effectively generalize to the target dataset. It's worth noting that the source and target datasets exhibit variations in distribution. Specifically, we utilize the MNIST dataset as

MNIST (source domain)				
model	accuracy	precision	recall	f1-score
AdaMatch	97.36	87.33	79.39	78.09
AdaMatch+ACN-base	**99.26**	**99.20**	**99.32**	**99.26**
AdaMatch+ACN	98.92	98.93	98.92	98.92
SVHN (target domain)				
model	accuracy	precision	recall	f1-score
AdaMatch	25.08	31.64	20.46	24.73
AdaMatch+ACN-base	43.10	53.83	43.10	47.46
AdaMatch+ACN	**54.70**	**59.74**	**54.70**	**54.55**

Table 2: Comparing model performance: AdaMatch vs. AdaMatch+ACN-base and AdaMatch+ACN on MNIST (source) and SVHN (target) datasets.



(a) Source domain (MNIST)



(b) Target domain (SVHN)

Fig. 3: Gradient Variance Evolution: AdaMatch and AdaMatch+ACN models during training on source (MNIST) and target (SVHN) domains. Left: Max gradient variance per epoch. Right: Average gradient variance per epoch.

the source dataset, while the target dataset 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 model (AdaMatch) [17] trained from scratch with wide residual networks [18] on dataset pairs, serves as the reference model. The model is trained using the Adam [13] optimizer and a cosine decay schedule to decrease the initial learning rate, which is initialized at 0.03. ACN is used as initial layer in AdaMatch to incorporate the context identifier (source domain and target domain) into the image normalization process. As the labels in the source domain are known, the model provides a better representation of this domain compared to the target domain, where the labels are unknown. This advantage is leveraged by considering the context of the source domain during inference to enhance model's performance on the target domain.

It is clear that in a broader context, Table 2 demonstrates a significant improvement in validation metrics with the use of adaptative context normalization. This improvement is evident through an increase in accuracy of **18.02%** with ACN-base and **29.62%** with ACN. This enhancement notably bolsters the performance of the AdaMatch model, resulting in significantly accelerated convergence during training. Figure 3 shows the valuable role that adaptative context normalization plays in stabilizing the gradient throughout the training process, benefiting both the source and target domains. As a result, this contributes to faster convergence and an overall improvement in model performance.

5. CONCLUSION

We present Adaptive Context Normalization (ACN) as a method to improve deep neural network training, with a particular focus on applications in image processing. ACN boosts stability, speeds up convergence, supports higher learning rates, and accommodates various activation functions. Unlike Mixture Normalization, ACN uses a context-based approach, enabling supervised learning of Gaussian component parameters for faster estimation. ACN offers non-linear decision boundaries. Through experiments, it has demonstrated superior performance when compared to established methods like batch normalization and mixture normalization, proving its value in domains like domain adaptation and image classification.

Our aim is to create an unsupervised variant of ACN that dynamically acquires context during training, bypassing the need for it as an input. This involves cluster discovery through normalization in training, potentially enhancing deep neural network convergence and performance by customizing data representation to the target task.

6. ACKNOWLEDGEMENTS

We thank Grid5000 for providing the computational resources that enabled us to conduct experiments within the framework of the LabCom partnership.

7. REFERENCES

[1] Agnan Kessy, Alex Lewin, and Korbinian Strimmer, "Optimal whitening and decorrelation," *The American Statistician*, vol. 72, no. 4, pp. 309–314, 2018.

[2] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-

Robert Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

[3] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[4] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao, "Normalization techniques in training dnns: Methodology, analysis and application," *arXiv preprint arXiv:2009.12836*, 2020.

[5] Mahdi M Kalayeh and Mubarak Shah, "Training faster by separating modes of variation in batch-normalized models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 6, pp. 1483–1500, 2019.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, "CIFAR (canadian institute for advanced research)," 2009.

[9] Ya Le and Xuan Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, pp. 3, 2015.

[10] Yann LeCun and Corinna Cortes, "MNIST handwritten digit database," 2010.

[11] Pierre Sermanet, Soumith Chintala, and Yann 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.

[12] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

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

[14] Khalid Salama, "Implementing the vision transformer (vit) model for image classification," 2021, https://github.com/keras-team/keras-io/tree/master.

[15] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia, "A brief review of domain adaptation," *Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020*, pp. 877–894, 2021.

[16] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin, "Adamatch: A unified approach to semi-supervised learning and domain adaptation," *arXiv preprint arXiv:2106.04732*, 2021.

[17] Sayak Paul, "Unifying semi-supervised learning and unsupervised domain adaptation with adamatch," 2019, https://github.com/keras-team/keras-io/tree/master.

[18] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.