

# Context Normalization Layer with Applications

1<sup>st</sup> Bilal FAYE

LIPN, UMR CNRS 7030  
Sorbonne Paris Nord University  
Villetaneuse, France  
faye@lipn.univ-paris13.fr

2<sup>nd</sup> Hanane AZZAG

LIPN, UMR CNRS 7030  
Sorbonne Paris Nord University  
Villetaneuse, France  
azzag@univ-paris13.fr

3<sup>th</sup> Mustapha Lebbah

DAVID Lab, University of Versailles,  
Université Paris-Saclay,  
Versailles, France  
mustapha.lebbah@uvsq.fr

4<sup>th</sup> Mohamed-Djallel DILMI

LIPN, UMR CNRS 7030  
Sorbonne Paris Nord University  
Villetaneuse, France  
dilmi@lipn.univ-paris13.fr

5<sup>th</sup> Djamel BOUCHAFFRA

Center for Development of Advanced Technologies  
Algiers, Algeria  
djamel.bouchaffra@gmail.com

**Abstract**—Deep neural networks (DNNs) have gained prominence in many areas such as computer vision (CV), natural language processing (NLP), robotics, and bioinformatics. While their deep and complex structure enables powerful representation and hierarchical learning, it poses serious challenges (e.g., internal covariate shift, vanishing/exploding gradients, overfitting, and computational complexity), during their training phase. Neuron activity normalization is an effective strategy that lives up to these challenges. This procedure consists in promoting stability, creating a balanced learning, improving performance generalization and gradient flow efficiency. Traditional normalization methods often overlook inherent dataset relationships. For example, batch normalization (BN) estimates mean and standard deviation from randomly constructed mini-batches (composed of unrelated samples), leading to performance dependence solely on the size of mini-batches, without accounting for data correlation within these batches. Conventional techniques such as Layer Normalization, Instance Normalization, and Group Normalization estimate normalization parameters per instance, addressing mini-batch size issues. Mixture Normalization (MN) utilizes a two-step process: (i) training a Gaussian mixture model (GMM) to determine components parameters, and (ii) normalizing activations accordingly. MN outperforms BN but incurs computational overhead due to GMM usage. To overcome these limitations, we propose a novel methodology that we named "Context Normalization" (CN). Our approach assumes that the data distribution can be represented as a mixture of Gaussian components. However, unlike MN that assumes a-priori that data are partitioned with respect to a set of Gaussian distributions, CN introduces the notion of concept that accounts for data relationship via a neural network classification scheme. Samples that are gathered within a cluster define a context. The estimation of the Gaussian components parameters is conducted through a supervised neural network-based concept classification. CN is more precise when clusters are thick and not sparse. Extensive comparative experiments conducted on various datasets demonstrates the superiority of CN over BN and MN in terms of convergence speed and performance generalization. In fact, CN outperforms BN and MN with a convergence speed margin of 5% and a performance margin of 10%. These results reveal the importance and the need of capturing inherent data context to learn the Gaussian component parameters. Our proposed approach harnesses data relationships, and therefore enhances deep learning models in various applications.

## I. INTRODUCTION

Normalization is a general process that transforms data to possess specific statistical properties. Various methods exist for data normalization, with some of the most common techniques including centering, scaling, standardizing, decorrelating, and whitening [1]. Input normalization can be used in deep neural networks (DNNs) training to remove the magnitude difference between features, speeding convergence during linear model optimization [2]. In layered neural networks, as the input is only directly connected to the first weight matrix, it is not clear how the input impacts the optimization landscape with respect to other weight matrices. The initial weights are typically not normalized, and this can indeed have a significant impact on the gradient optimization procedure. To overcome this, some methods employ weight-initializing techniques to obtain equal variances for layer input/output gradients across different layers [3]. However, due to the updating of the weight matrices during training, the equal variance property across layers may be broken. From this perspective, it is important to normalize activations during training, to obtain similar benefits of normalizing inputs.

Different normalization techniques, including activation normalization, weight normalization, and gradient normalization, are employed to enhance the training performance of DNNs. To normalize activations, the most common technique is Batch Normalization (BN) [4]. BN has been proposed to solve the problem caused by the changing distribution of the inputs of each layer during training, called internal covariate shift. In fact, the distribution change problem often requires lower learning rates and careful parameter initialization, which slows down the training process and makes it difficult to train models with saturating nonlinearity. As a layer of the neural network, BN methods perform the normalization on each training mini-batch with its respective mean and variance. This mini-batch-wise approach allows for a more appropriate representation of the data and, therefore, faster processing, thus increasing the performance of the neural network in terms of convergence.

Despite the good performance, the effect of BN depends on the mini-batch size, and it is not obvious how to apply it to some DNNs architectures. To address this, several variants and alternative methods have been proposed [5].

BN and its few extensions can be studied from the viewpoint of Fisher kernels that arise from generative probability models. Kalayeh and al. [6] show that assuming samples within a mini-batch are from the same probability density function, then BN is identical to the Fisher vector of a Gaussian distribution. More specifically, each instance in the mini-batch is assigned to a component of the Gaussian Mixture Model (GMM), where the GMM approximates the probability density function of the input activations. Instead of computing one set of statistical measures from the entire population (of instances in the mini-batch) as BN does, Mixture Normalization (MN) [6] proposes a normalization on sub-populations which can be identified by disentangling modes of the distribution, estimated via GMM. In addition to speeding up training, MN allows better accuracy results to be achieved.

In this perspective, we propose a novel normalization technique called context normalization (CN). In fact, assuming that the data are well modeled by a mixture of several components, each sample in the mini-batch is normalized using the mean and variance of the associated component. Indeed, the capability of GMM to approximate any continuous distribution with arbitrary precision has been demonstrated by [7]. Building upon this foundation, our paper follows a similar track but introduces a novel method. In particular, we define a context that can come from various sources that describe the structure of the dataset. A context can be conceptualized as a coherent cluster of samples that share common characteristics and can be effectively grouped together. Each context can be viewed as a component of the Gaussian mixture with its own probability density function. By normalizing samples from the same context with the parameters learned during backpropagation, CN allows an estimation of the mean and variance of each mixture component thus improving the discrimination power of the data representation according to the target task.

In summary, the main contributions of this work are as follows:

- We propose Context Normalization (CN), a novel approach that utilizes defined contexts to capture underlying distribution variations. In CN, each sample in a mini-batch is normalized using the mean and standard deviation specific to its context. By treating contexts as components of a Gaussian mixture, we learn their parameters during model training, eliminating the need for the EM algorithm. This leads to improved efficiency and simplified implementation of CN.
- Through a comprehensive set of experiments, we demonstrate that CN not only accelerates model convergence, but also achieves superior final test accuracy. These results highlight the effectiveness of our proposed method in improving the overall performance of models.

For consistency, we use the variable notation proposed in

Table I for all sections.

Variable	Definition
$T$	number of context
$K$	number of components
$N$	mini-batch size
$C$	channels
$H$	height
$W$	width
$Net$	neural network
$\Theta$	trainable parameters of neural network
$x \in \mathbb{R}^{N \times C \times H \times W}$	4-D activation tensor
$B = \{x_{1:m}\}$	flattened $x$ across axis $N$ , $H$ and $W$
$B_i$	flattened $x$ across axis $H$ and $W$
$\mu_r$	the mean on the context $r$
$\sigma_r$	standard deviation on the context $r$
$onehot(.)$	One-Hot Encoding function

TABLE I: Table of notations

## II. RELATED WORK

### A. Batch Normalization

Let  $x \in \mathbb{R}^{N \times C \times H \times W}$  denote the activation for a given neuron in a layer of a convolutional neural network where  $N$ ,  $C$ ,  $H$  and  $W$  are respectively the batch, channel, height, and width axes. Batch normalization (BN) [4] standardizes with  $m$  samples mini-batch  $B = \{x_{1:m} : m \in [1, N] \times [1, H] \times [1, W]\}$  with flattened  $x$  across all but the channel axis by:

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

where  $\mu_B = \frac{1}{m} \sum_i x_i$ ,  $\sigma_B^2 = \frac{1}{m} \sum_i (x_i - \mu_B)^2$  are the mean and variance respectively, and  $\epsilon > 0$  a small number to prevent numerical instability.

Due to the constraints introduced by standardization, additional learnable parameters  $\gamma$  and  $\beta$  are introduced to eliminate the linear regime of nonlinearity of some activations:

$$\tilde{x}_i = \gamma \hat{x}_i + \beta \quad (2)$$

During inference, population statistics are needed for deterministic inference. They are usually computed by running the average over the training iterations, as follows:

$$\begin{cases} \hat{\mu} = (1 - \lambda)\hat{\mu} + \lambda\mu_B \\ \hat{\sigma}^2 = (1 - \lambda)\hat{\sigma}^2 + \lambda\sigma_B^2 \end{cases} \quad (3)$$

If the samples within the mini-batch are from the same distribution, the transformation in Equation (1) generates a zero mean and unit variance distribution. This zero-mean and unit-variance constraint allows stabilizing the distribution of the activations and thus benefits training.

This mini-batch-wise approach makes it possible to have a more suitable representation of the data and, therefore, faster processing, thus increasing the performance of the neural network in terms of convergence. Despite the good performance, the effect of BN is dependent on the mini-batch size, and the discrepancy between training and inference limits its usage in complex networks (e.g. recurrent neural networks). To tackle these challenges and address parameter estimation on unrelated samples, numerous variants, and alternative methods have been proposed.

## B. Variants of Batch Normalization

Some extensions to batch normalization (BN) have been proposed, in particular, Layer Normalization (LN), Instance Normalization (IN), Group Normalization (GN) and Mixture Normalization (MN) [5], [6].

The general transformation  $x \rightarrow \hat{x}$  according to a mini-batch, on the flattened spatial domain ( $L = H \times W$ ), can be written as follows

$$v_i = x_i - \mathbb{E}_{B_i}(x), \quad \hat{x}_i = \frac{v_i}{\sqrt{\mathbb{E}_{B_i}(v^2) + \epsilon}}, \quad (4)$$

given  $B_i = \{j : j_N \in [1, N], j_C \in [i_C], j_L \in [1, L]\}$ , where  $i = (i_N, i_C, i_L)$  a vector indexing the activations  $x \in \mathbb{R}^{N \times C \times L}$ .

**Layer Normalization (LN)** eliminates inter-dependency of batch-normalized activations by calculating mean and variance based on specific layer neuron inputs. LN is effective for recurrent networks but may face challenges with convolutional layers due to variations in visual information across the spatial domain.

**Instance Normalization (IN)** is a normalization technique that normalizes each sample individually, focusing on removing style information, especially in images. By computing mean values and standard deviations in the spatial domain, IN improves the performance of specific deep neural networks (DNNs) and finds widespread application in tasks like image style transfer [8] [9].

**Group Normalization (GN)** is a normalization technique that divides neurons into groups and independently standardizes layer inputs for each sample within the groups. It excels in visual tasks with limited batch sizes, like object detection and segmentation.

**Mixture Normalization** In the context of deep neural networks (DNNs), the distribution of activations is almost certain to have multiple modes of variation due to the non-linearities. The batch normalization (BN) [4] hypothesis that a Gaussian distribution can model the generative process of mini-batch samples is less valid. To address this, Mixture Normalization (MN) [6] investigates BN from the viewpoint of Fisher kernels, which arise from generative probability models. Rather than using a mean and standard deviation calculated over entire instances within a mini-batch, MN uses a Gaussian Mixture Model (GMM) to affect each instance in the mini-batch to a component and then normalizes with respect to multiple means and standard deviations associated with different modes of variation in the underlying data distribution.

MN normalizes each sample in the mini-batch using the mean and standard deviation of the mixture component to which the sample belongs to. The probability density function  $p_\theta$  that describes the data can be parameterized as a Gaussian Mixture Model (GMM). Let  $x$  in  $\mathbb{R}^D$ , if  $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \dots, K\}$ ,

$$p(x) = \sum_{k=1}^K \lambda_k p(x|k), \quad s.t. \quad \forall_k : \lambda_k \geq 0, \quad \sum_{k=1}^K \lambda_k = 1, \quad (5)$$

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  $k^{th}$  Gaussian in the mixture model  $p(x)$ ,  $\mu_k$  the mean vector and  $\Sigma_k$  is the covariance matrix.

The probability that  $x$  has been generated by the  $k^{th}$  Gaussian component in the mixture model can be defined as:

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

Based on these assumptions and the general transform in Equation (4), the Mixture Normalizing Transform for a given  $x_i$  is defined as

$$\hat{x}_i = \sum_{k=1}^K \frac{\tau_k(x_i)}{\sqrt{\lambda_k}} \hat{x}_i^k, \quad (6)$$

given

$$v_i^k = x_i - \mathbb{E}_{B_i}[\hat{\tau}_k(x).x], \quad \hat{x}_i^k = \frac{v_i^k}{\sqrt{\mathbb{E}_{B_i}[\hat{\tau}_k(x).(v^k)^2] + \epsilon}}, \quad (7)$$

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  $x_i$  with respect to the mini-batch  $B_i$  in the estimation of the statistical measures of the  $k^{th}$  Gaussian component.

With this approach, Mixture Normalization can be done in two stages:

- estimation of mixture model's parameters  $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \dots, K\}$  by Expectation-Maximization (EM) [10] algorithm.
- normalization of each sample with respect to the estimated parameters (Equation (7)) and aggregation using posterior probabilities (Equation (6)).

On convolutional neural networks, this method allows Mixture Normalization to achieve better results than batch normalization in terms of convergence and accuracy in supervised learning tasks.

## III. PROPOSED METHOD: CONTEXT NORMALIZATION

### A. Method description

Based on the Mixture Normalization (MN) hypothesis proposed by [6] (ref. to Figure 1), our Context Normalization (CN) approach operates under a similar assumption that data can be effectively represented by a mixture of multiple components, as opposed to batch normalization (BN) [4]). In the Context Normalization (CN) approach, a fundamental concept is introduced, namely, the notion of context, which represents a cluster of samples sharing common characteristics that can be efficiently grouped together. Unlike the Expectation-Maximization (EM) algorithm [10] typically employed for parameters estimation in each component, CN utilizes a deep neural network to learn these parameters through context-based normalization. In our approach, we assign a unique

identifier to each context and utilize it for normalization during training. Samples within the same context share the same identifier, allowing for alignment in a shared space that aligns with the target task. This approach not only facilitates the normalization of samples within the same context but also enables the estimation of optimal parameters for all contexts, promoting the convergence of the model. By leveraging these context identifiers, our approach enhances the alignment and adaptability of the model to different contexts, leading to improved performance.

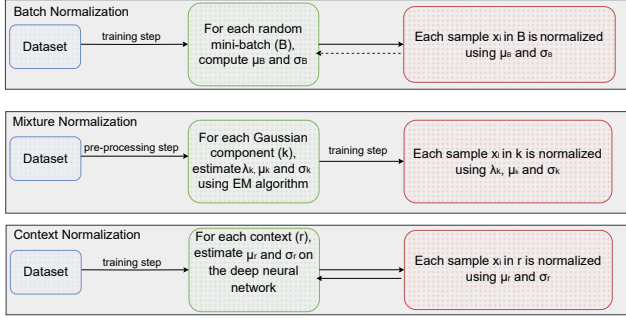


Fig. 1: A concise overview of the processing steps involved in Batch Normalization (BN), Mixture Normalization (MN), and Context Normalization (CN). The dashed line in the Batch Normalization diagram indicates a mini-batch parameter update, highlighting a key step in the process.

### B. Parameters learning

For a given  $x_i$  in  $B_i$ , the mean ( $\mu_r$ ) and standard deviation ( $\sigma_r$ ) are estimated based on the context  $r$  associated to  $x_i$ . This estimation process is outlined in Algorithm 1 and visually depicted in Figure 2. Subsequently, these estimated parameters are used to normalize  $x_i$  according to Equation (8). The CN transform can be added to a deep neural network to provide a better representation of the input data. In Algorithm 1,  $\hat{x}_i = \text{CN}(x_i, r)$  indicates that the parameters  $\mu_r$  and  $\sigma_r$  are to be learned depending on each training example according to the context  $r$ . Algorithm 2 summarizes the procedure for training context-normalized networks.

$$\hat{x}_i \leftarrow \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} \quad (8)$$

In algorithm 1, we use an affine transformation followed by element-wise linearity:

$$\begin{aligned} \alpha_r &\leftarrow \text{Embedder}_r(\text{onehot}(r)) : W_r \text{ onehot}(r) + b_r, \\ \mu_r &\leftarrow \text{Embedder}_\mu(\alpha_r) : W_\mu \alpha_r + b_\mu, \\ \sigma_r^2 &\leftarrow \text{Embedder}_\sigma(\alpha_r) : W_\sigma \alpha_r + b_\sigma \end{aligned}$$

where  $W$  and  $b$  are learned parameters of the Embedder model and  $r$  is the context associated to  $x_i$ . The  $\text{onehot}(\cdot)$  is the function that performs one-hot encoding on the categorical

---

#### Algorithm 1: CN- Context Normalizing Transform

---

**Input :** Values of  $x$  over the mini-batch  
 $B_i = \{x_i\}_{i=1}^m$ ; context  $r$  associated to  $x_i$   
**Output:**  $\{\hat{x}_i = \text{CN}(x_i, r)\}$   
 $\alpha_r \leftarrow \text{Embedder}_r(\text{onehot}(r))$  // identifier embedding  
 $\mu_r \leftarrow \text{Embedder}_\mu(\alpha_r)$  // mean estimation  
 $\sigma_r^2 \leftarrow \text{Embedder}_\sigma(\alpha_r)$  // variance estimation  
 $\hat{x}_i = \text{CN}(x_i, r)$ : Normalize  $x_i$  using Equation 8

---



---

#### Algorithm 2: CN-Training (Context-Normalized Deep Neural Network)

---

**Input :** Deep neural network  $Net$  with trainable parameters  $\Theta$ ; subset of activations  
 $B_i = \{x_i\}_{i=1}^m$   
**Output:** Context-Normalized deep neural network for training,  $Net_{CN}^{tr}$   
 $Net_{CN}^{tr} = Net$  // Training CN deep neural network  
**for**  $i \leftarrow 1$  to  $m$  **do**  
    • Add Algorithm 1 transformation:  $\hat{x}_i = \text{CN}(x_i, r)$   
    // normalize  $x_i$  using associated context,  $r$   
    • Modify each layer in  $Net_{CN}^{tr}$  with  $x_i$  to take  $\hat{x}_i$  instead  
**end**  
Train  $Net_{CN}^{tr}$  to optimize the parameters:  
 $\Theta = \Theta \cup \{\mu_r, \sigma_r\}_{r=1}^T$

---

variable representing the context  $r$ , which is embedded in the continuous space  $\alpha_r$ .

During the training process, it is necessary to backpropagate the gradient of the loss function, denoted as  $\ell$ , through the transformation. Additionally, the gradients with respect to the parameters of the CN transform need to be computed. This is achieved using the chain rule, as shown in the following expression (before simplification):

$$\begin{aligned} \frac{\partial \ell}{\partial \mu_r} &= \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial \mu_r} = -\frac{\partial \ell}{\partial \hat{x}_i} \cdot (\sigma_r^2 + \epsilon)^{-1/2} \\ \frac{\partial \ell}{\partial \sigma_r^2} &= \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial \sigma_r^2} = \frac{\mu_r + x_i}{2(\sigma_r^2 + \epsilon)^{3/2}} \end{aligned}$$

CN transform is a differentiable operation in deep neural networks that normalizes input data. By applying CN, the model can continuously learn from input distributions and adapt its representations to the target task, leading to improved performance. This normalization helps mitigate the influence of variations in input distributions, allowing the model to focus on relevant patterns and features. The differentiability of CN enables efficient gradient flow during training, facilitating parameter updates and learning from the normalized data while preserving differentiation through the normalization process. Overall, CN plays a vital role in enhancing model



performance by promoting effective learning and adaptability through data normalization. It demonstrates higher flexibility compared to MN due to its ability to establish consistent data grouping based on provided contexts, without the need for additional algorithms. This is advantageous over MN since the Expectation-Maximization (EM) algorithm employed in MN can exhibit slower convergence. In the specific case of classifying dog images, where data scarcity is a challenge, the method addresses this issue by partitioning the dog class into subclasses. This approach enables the acquisition of specific features applicable to all dogs, facilitating the normalization of images within the dog superclass and creating a more coherent and easily learnable feature space. Importantly, the context identifier used for learning the normalization parameters is unrelated to the images themselves. Instead, it can be viewed as noise, contributing to the regularization of the deep neural network during training, similar to techniques like dropout, thereby enhancing the generalization performance of the model [11].

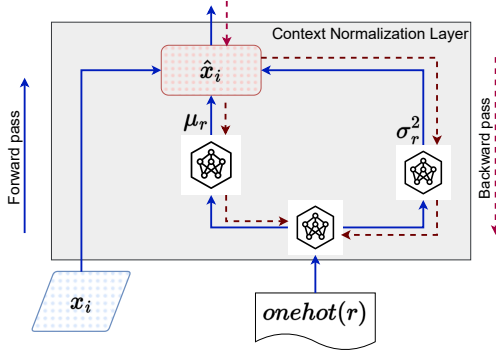


Fig. 2: Context Normalization Layer applied to a given activation  $x_i$ . The context identifier ( $r$ ) is encoded by a neural network, the output of which is then used as input to two different neural networks to generate a mean ( $\mu_r$ ) and a standard deviation ( $\sigma_r$ ), respectively, for normalizing  $x_i$ .

After the model has converged, we enter a scenario similar to mixture normalization, where the parameters of each context are known and dependent on the target task. During the inference step, we have the option to normalize the data directly using CN or consider all the contexts collectively. This enhanced approach is referred to as CN+ (ref. Algorithm 3). For a given  $x_i$  in  $B_i$  and using the Equation 6, CN+ can be formulated as

$$\hat{x}_i = \sqrt{T} \sum_{r=1}^T \tau_r(x_i) \hat{x}_i^r, \quad (9)$$

given

$$v_i^r = x_i - \mathbb{E}_{B_i}[\hat{\tau}_r(x) \cdot x], \quad \hat{x}_i^r = \frac{v_i^r}{\sqrt{\mathbb{E}_{B_i}[\hat{\tau}_r(x) \cdot (v^r)^2] + \epsilon}},$$

where

$$\hat{\tau}_r(x_i) = \frac{\tau_r(x_i)}{\sum_{j \in B_i} \tau_r(x_j)},$$

under the assumption that prior probabilities ( $\lambda_r = \frac{1}{T}, r = 1, \dots, T$ ) are constant;

---

**Algorithm 3:** CN-Inference (Context-Normalized Deep Neural Network)

---

**Input :** Deep Neural Network  $Net_{CN}^{tr}$  with trainable parameters  $\Theta \cup \{\mu_r, \sigma_r\}_{r=1}^T$  (ref. Algorithm 2); mini-batch of activation  $\{x_i\}_{i=1}^m$ ;  $choice \in \{CN, CN+\}$

**Output:** Context-Normalized deep neural network for inference,  $Net_{CN}^{inf}$

$Net_{CN}^{inf} \leftarrow Net_{CN}^{tr}$  // Inference CN deep neural network // with frozen parameters

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

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

Normalize using context( $r$ ) parameters ( $\mu_r$  and  $\sigma_r$  learned during training) associated to  $x_i$ .

$\hat{x}_i \leftarrow CN(x_i, r)$

**end**

**end**

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

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

• Normalize  $x_i$  using Equation 9 // Normalization + aggregation

**end**

**end**

---

#### IV. EXPERIMENTS

In the proposed experiments, we will assess the performance of the context normalization method and compare it to BN and MN. Detailed descriptions of the methods under consideration are provided in Table II.

CN-Patches can be used as layer on Transformer-based architecture [12] like Vision Transformer (ViT) [13]. ViT model takes as input a grid of non-overlapping contiguous image patches. CN-Patches consist of normalizing image patches with a vector  $\mu$  and  $\sigma$  that are learned from the context identifier embedding (ref. Figure 2 and Algorithm 1). Images are grouped by context (each context is a component of the mixture of Gaussian distributions), and for each context, a unique vector (context identifier) is associated. The context

Model	Description
BN	model with batch normalizing transform (ref. Section II-A)
MN	model with mixture normalizing transform (ref. Section II-B)
CN	Context Normalization model which uses CN-Training (ref. Algorithm 2) in the training step and CN method (ref. Algorithm 3) in the inference step. Two versions of CN are used in the experiments: CN on Patches (CN-Patches) and CN on Channels (CN-Channels).
CN+	Context Normalization model which uses CN-Training (ref. Algorithm 2) in training step and CN+ method (ref. Algorithm 3) in inference step

TABLE II: Normalization method used in the experiments

identifier is first embedded for a better representation, and the resulting vector is used to learn the corresponding parameters  $\mu$  and  $\sigma$  independently. This process is repeated for each input (image) to the neural network during training. CN-Patches is a way of integrating the context information into the image patch normalization process. It allows the image patch representation to be adapted to the context of the image, which improves the overall performance of the model.

CN-Channels is designed to be applied directly to images. The parameters  $\mu$  and  $\sigma$  are vectors of size the number of channels. They are learned independently according to the context by using context identifiers. CN-Channels incorporates the context identifier into the image normalization process. In this case, the context identifier embedding is used to directly adjust the image representation per channel. This allows the model to adapt the image representation to the context, which improves the overall performance of the model.

#### A. Datasets

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

- CIFAR-10: A dataset with 50,000 training images and 10,000 test images, each of size  $32 \times 32$  pixels, distributed across 10 classes [14].
- CIFAR-100: Derived from the Tiny Images dataset, it consists of 50,000 training images and 10,000 test images of size  $32 \times 32$ , divided into 100 classes grouped into 20 superclasses [15].
- MNIST digits: Contains 70,000 grayscale images of size  $28 \times 28$  representing the 10 digits, with around 6,000 training images and 1,000 testing images per class [16].
- VERI-Wild: Comprises 416,314 vehicle images with 40,671 identities, captured from different angles and at different times [17].
- SVHN: A challenging dataset with over 600,000 digit images, focusing on recognizing digits and numbers in natural scene images [18].

Table III illustrates the mapping of each dataset to the corresponding experiments in which it is employed.

#### B. Context Normalization vs Mixture Normalization on CIFAR

In this experiment, we use a shallow convolutional neural network architecture as described in Table IV.

The mixture normalization approach replaces the BN layer in conv3 with mixture normalization layer, and the context normalization experiment uses CN-Channels as the first CIFAR ConvNet layer, incorporating context information directly into the base image. According to [6], we experiment with two different learning rates, one 5 times larger than the other. The same is done with weight decay. The mini-batch is set to 256, and all models are trained for 100 epochs using RMSprop [19] with 0.9 momentum. During training on CIFAR-10 and CIFAR-100, we fit a Gaussian mixture model by Maximum Likelihood Estimation (MLE). We use K-menas++ [20] to initialize the centers of the mixture component and Expectation-Maximization (EM) [10] to estimate

Dataset	Task
CIFAR-10	This dataset is used for training CIFAR ConvNet (ref. Table IV), ensuring consistent conditions with the study on mixture normalization (ref. Section IV-B).
CIFAR-100	Three experiments (ref. Sections IV-B, IV-C, and IV-D) use this dataset: comparing with mixture normalization methods, utilizing superclass structure as context, and blending with MNIST digits for domain adaptation.
MNIST digits	MNIST digits is employed in two domain adaptation experiments, serving as the blended dataset with CIFAR-100 in the first (ref. Section IV-D) and as the source domain in the second (ref. Section IV-F).
VERI-Wild	This dataset facilitates the analysis of context parameters obtained through model training using an image similarity measurement strategy (ref. Section IV-E), thereby constructing the American night.
SVHN	SVHN is used in conjunction with MNIST digits as the target dataset to train the AdaMatch model in unsupervised domain adaptation (ref. Section IV-F).

TABLE III: Summary of datasets and associated training experiments

layer	type	size	kernel	(stride, pad)
input	input	$3 \times 32 \times 32$	—	—
conv1	conv+bn+relu	$64 \times 32 \times 32$	$5 \times 5$	(1, 2)
pool1	max pool	$64 \times 16 \times 16$	$3 \times 3$	(2, 0)
conv2	conv+bn+relu	$128 \times 16 \times 16$	$5 \times 5$	(1, 2)
pool2	max pool	$128 \times 8 \times 8$	$3 \times 3$	(2, 0)
conv3	conv+bn+relu	$128 \times 8 \times 8$	$5 \times 5$	(1, 2)
pool3	max pool	$128 \times 4 \times 4$	$3 \times 3$	(2, 0)
conv4	conv+bn+relu	$256 \times 4 \times 4$	$3 \times 3$	(1, 1)
pool4	avg pool	$256 \times 1 \times 1$	$4 \times 4$	(1, 0)
linear	linear	10 or 100	—	—

TABLE IV: CIFAR ConvNet architecture on mixture normalization paper for a comparison of MN and BN for small and large learning rate regimes

the parameters of the mixture model  $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \dots, K = 3\}$ . In mixture normalization, each sample is normalized using the mean and standard deviation of the mixture component to which it belongs. Context normalization treats each Gaussian component as an individual context, enabling sample normalization within each component. It employs separate multi-layer perceptrons (MLPs) to estimate the mean and standard deviation for each context using the one-hot encoded context identifier (*onehot(.)*).

Our primary objective is not to achieve state-of-the-art results, which require computationally expensive architectures and careful parameter tuning. Instead, we aim to demonstrate that by replacing or incorporating our context normalization technique, the convergence rate can be improved, leading to superior test accuracy. This showcases the significant impact of our approach in enhancing model performance. Increasing the learning rate from 0.001 to 0.005 increases the convergence gap, showing the ability of context normalization as mixture normalization to take advantage of higher learning rates for training. From Table V and Figure 3 we can see that regardless of the weight decay and learning rates, CN models not only

CIFAR-10		
model	(lr, weight decay)	test accuracy (%)
BN-1	(0.001, 1e-4)	86.9
BN-2	(0.001, 2e-5)	85.9
BN-3	(0.005, 1e-4)	82.56
BN-4	(0.005, 2e-5)	83.71
MN-1	(0.001, 1e-4)	87.08
MN-2	(0.001, 2e-5)	87.9
MN-3	(0.005, 1e-4)	82.07
MN-4	(0.005, 2e-5)	84.71
CN-1	<b>(0.001, 1e-4)</b>	<b>87.32</b>
CN-2	<b>(0.001, 2e-5)</b>	<b>88.15</b>
CN-3	<b>(0.005, 1e-4)</b>	<b>83.85</b>
CN-4	<b>(0.005, 2e-5)</b>	<b>87.09</b>
CIFAR-100		
model	(lr, weight decay)	test accuracy (%)
BN-1	(0.001, 1e-4)	57.48
BN-2	(0.001, 2e-5)	57.69
BN-3	(0.005, 1e-4)	50.82
BN-4	(0.005, 2e-5)	52.25
MN-1	(0.001, 1e-4)	61.6
MN-2	<b>(0.001, 2e-5)</b>	<b>61.9</b>
MN-3	(0.005, 1e-4)	53.08
MN-4	(0.005, 2e-5)	52.7
CN-1	<b>(0.001, 1e-4)</b>	<b>61.79</b>
CN-2	(0.001, 2e-5)	60.53
CN-3	<b>(0.005, 1e-4)</b>	<b>56.08</b>
CN-4	<b>(0.005, 2e-5)</b>	<b>53.54</b>

TABLE V: Performance Evaluation on CIFAR-10 and CIFAR-100 using the CIFAR ConvNet architecture (ref. Table IV) with the incorporation of Batch Normalization (BN), Mixture Normalization (MN), and Context Normalization (CN). Each algorithm is followed by a corresponding number, representing the learning rate and weight decay.

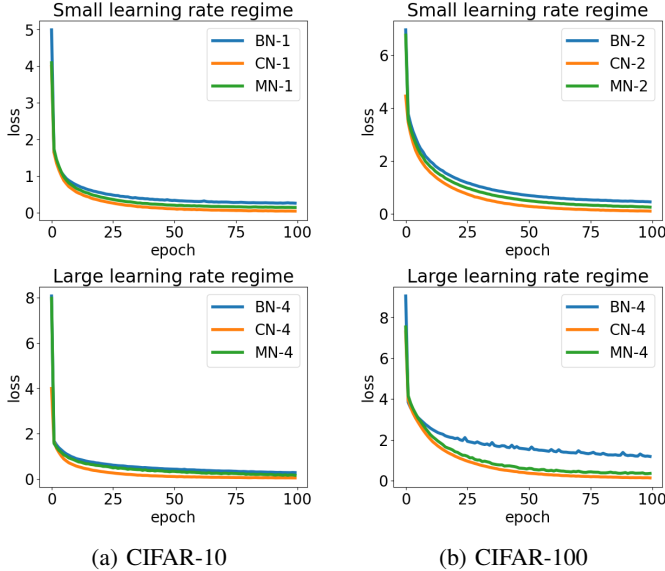


Fig. 3: Comparative analysis of validation error curves for the CIFAR ConvNet architecture (ref. Table IV) trained under different learning rate and weight decay configurations.

converge much faster than their corresponding BN and MN counterparts but also achieve better accuracy.

After model training, each context’s parameters are known. The use of CN+ in the inference step leads to the results shown in Table VI. The CN method (ref. Table V) is faster than the

CIFAR-10		
model	(lr, weight decay)	test accuracy (%)
CN+1	<b>(0.001, 1e-4)</b>	<b>87.90</b>
CN+2	<b>(0.001, 2e-5)</b>	<b>88.28</b>
CN+3	(0.005, 1e-4)	82.85
CN+4	<b>(0.005, 2e-5)</b>	<b>87.50</b>
CIFAR-100		
model	(lr, weight decay)	test accuracy (%)
CN+1	(0.01, 1e-4)	62.01
CN+2	(0.001, 2e-5)	60.53
CN+3	(0.005, 1e-4)	55.80
CN+4	(0.005, 2e-5)	53.15

TABLE VI: Performance Evaluation on CIFAR-10 and CIFAR-100 using CIFAR ConvNet architecture (ref. Table IV). Introducing CN+ which is an extended version of CN for Inference, encompassing all contexts as Mixture Normalization. Each CN+ variant is denoted by a number, representing the iterative exploration of learning rate and weight decay adjustments.

CN+ method (ref. Table VI) and gives approximately the same results. In the following experiments, we use the CN method in the inference step (ref. Algorithm 3).

In the following experiments, we aim to demonstrate that context normalization can be implemented in a single step in specific scenarios, unlike mixture normalization. This approach would lead to a decrease in time complexity.

### C. Context Normalization using superclass as context

As described in Section IV-A, the CIFAR-100 dataset incorporates a superclass structure in addition to the class partition. Our proposed context normalization leverages this superclass information as “prior knowledge” for classification. Each superclass corresponds to a context, identified by a one-hot encoded vector of size 20 (the number of superclasses). We integrated the context normalization technique into the Vision Transformer (ViT) architecture, using Context Normalization on Patches (CN-Patches) and Context Normalization on Channels (CN-Channels). 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. The AdamW optimizer with a learning rate of  $10^{-3}$  and weight decay of  $10^{-4}$  was chosen to prevent overfitting and optimize model parameters [21], [22].

Table VII demonstrates the significant performance improvement of context normalization over batch normalization (BN) when using the ViT architecture trained from scratch on CIFAR-100. Both CN-Patches and CN-Channels approaches outperform BN by approximately 10% and 18% in terms of accuracy and top-5 accuracy. The train and test loss comparison in Figure 4 further supports this observation, showing that

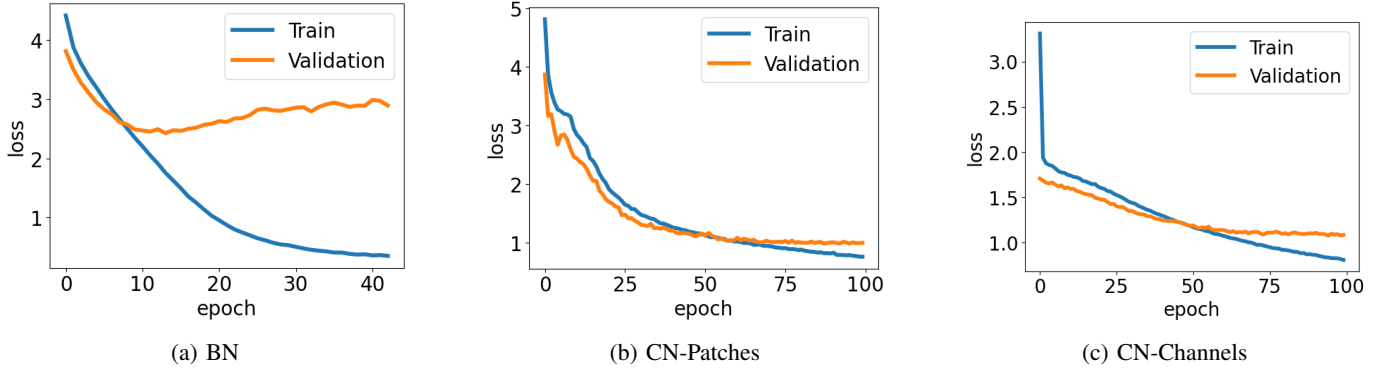


Fig. 4: Validation loss curves on CIFAR-100 when the ViT architecture is trained with different normalization methods.

model	test accuracy	test top-5-accuracy
ViT	52.37%	80.98%
ViT+BN	53.35%	79.68%
<b>ViT+CN-Patches</b>	<b>63.80%</b>	<b>99.74%</b>
<b>ViT+CN-Channels</b>	<b>62.48%</b>	<b>99.83%</b>

TABLE VII: Comparison of the two Context Normalization methods on CIFAR-100: Context Normalization on Patches (CN-Patches) and Context Normalization on Channels (CN-Channels), with normalization to the mean and standard deviation of the dataset (ViT) and input normalization using batch normalization (BN).

CN-Patches and CN-Channels accelerate the learning process and enhance classification performance. These results indicate that the proposed approaches stabilize the distribution and mitigate internal covariate shift, resulting in a superior data representation aligned with the target task.

#### D. Image classification on blended dataset

As deep neural networks require a certain amount of labeled data for effective training, it is well known that the lack of a large enough corpus of accurately labeled high-quality data can produce disappointing results. Data augmentation [23] is one way to overcome this problem. However, current approaches generate the data with a distribution that may be quite different from the original one, and this data bias will often lead to suboptimal performance [24]. Inspired by the transfer learning [25], we study in this subsection a new approach using the proposed context normalization. The goal of this approach is to improve the performance of the model on a target dataset by training the model on the combined dataset and then applying the trained model to the target dataset. In this framework, the proposed normalization technique allows to obtain a domain adaptation.

Specifically, we train ViT models with the same settings as in Section IV-C with context normalization approaches (CN-Channels and CN-Patches) on the combined dataset CIFAR-100 and MNIST digits. We target two contexts  $r \in \{1, 2\}$ , corresponding to the datasets and the context identifier is encoded by  $\text{onehot}(r = 1) = (1, 0)$  for images in CIFAR-100 and  $\text{onehot}(r = 2) = (0, 1)$  for images in MNIST digits.

The parameters  $\mu$  and  $\sigma$  of each Gaussian distribution are then learned using these vectors after embedding. The trained models (on the blended dataset) are then applied to the CIFAR-100 test dataset.

It is important to mention that the baseline models (ViT with standard preprocessing and ViT with batch normalization) collapsed in this blended dataset as the two datasets have different structures, and simple normalization does not allow a suitable representation of the data. Context normalization, on the other hand, gives an adaptive representation per dataset (according to the contexts), which makes training possible. As shown in Table VIII, models with context normalization technique achieve good results on the blended dataset. It is also interesting to notice that this performance in terms of accuracy is biased by the MNIST digits dataset, which is less difficult to learn than CIFAR-100. More precisely, the model with CN-Patches achieves 55.04% accuracy and 81.83% top-5 accuracy, which exceeds the results of all baseline models (ref. Figure VII) trained on CIFAR-100. The model with CN-Channels gives 50.99% accuracy and 78.55% top-5 accuracy.

model	accuracy	top-5-accuracy
ViT	—	—
ViT+BN	—	—
<b>ViT+CN-Patches</b>	<b>77.09%</b>	<b>90.92%</b>
<b>ViT+CN-Channels</b>	<b>74.92%</b>	<b>89.28%</b>

TABLE VIII: Blended dataset CIFAR-100 and MNIST digits: results of models based on the two normalization methods (CN-Patches and CN-Channels). Baseline models collapsed in this combined dataset.

The Latent space normalization (CN-Patches) has a better performance than the CN-Channels in this experiment. The drop in performance for the CN-Channels method can be explained by the fact that the normalization is applied directly to images located in a space with several different data characteristics.

#### E. Self-supervised learning for image similarity estimation

This experiment is motivated by the interpretability of context normalization parameters of each context obtained



after training. To illustrate this, we construct the American night with context normalization on the VERI-Wild dataset. American night [26] is a set of cinematic techniques used to simulate a night scene while filming in daylight. To accomplish this, we make a partition of two contexts on the dataset:  $r \in \{Day, Night\}$ . A context vector  $onehot(r = Day) = (1, 0)$  is assigned to day images and  $onehot(r = Night) = (0, 1)$  to night images. We label pairs of images containing the same object with 1, and otherwise with 0.

We use context normalization on the backbone of a siamese network (ViT architecture) with contrastive loss [27] to estimate the similarity between images. The aim of this experiment is to reveal the behavior of the parameters learned by the model and to understand how context information has an influence on the normalization process. The use of a Siamese network makes it possible to measure the similarity between the images and to evaluate the efficiency of the normalization in preserving the relevant information.

After training, we normalize day images with the appropriate parameters ( $\mu_{Day}$  and  $\sigma_{Day}$ ), then denormalize (scale and shift) with the night parameters ( $\mu_{Night}$  and  $\sigma_{Night}$ ), to construct night images, as shown in Figure 5. The images become darker, giving the night effect known as the American night (day for night) used for cinematic production in Hollywood, where a filter is applied to emphasize the light in the blue channel. A reverse process can be applied to night images to obtain day images, as shown in Figure 6, with brighter images reflecting the daylight effect.



Fig. 5: Simulation of a night image on a day image: night image is obtained by normalizing day image with Day parameters, then scale and shift with Night parameters.

#### F. Context normalization in domain adaptation

In this experiment, we show that due to its strength in local representation, context normalization can yield substantial gains in domain adaptation. Domain adaptation [28] is a technique for using knowledge learned by a model from a related domain with sufficient labeled data to improve the performance of the model in a target domain with insufficient labeled data. In this case, two contexts can be considered:  $r \in \{\text{source domain, target domain}\}$ . As an illustration, we use context normalization with AdaMatch [29], a method that

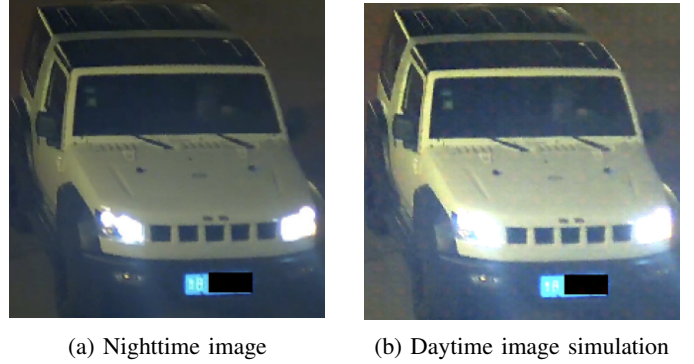


Fig. 6: Simulation of day image on night image: day image is obtained by normalizing night image with Night parameters, then scale and shift with Day parameters

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 source labeled dataset and a target unlabeled dataset. Then the task is to learn a model that can generalize well to the target dataset. The source and the target datasets vary in terms of distribution. We use the MNIST dataset (ref. Section IV-A) as the source dataset, while the target dataset is SVHN (ref. Section IV-A). Both datasets have various varying factors in terms of texture, viewpoint, appearance, etc. Their domains, or distributions, are different from one another. As in [29], we use wide residual networks [30] for the dataset pairs. The model is trained using the Adam [22] optimizer and a cosine decay schedule to reduce the initial learning rate, which is initialised at 0.03. Context normalization is used as the first layer of AdaMatch to incorporate the context identifier (source domain and target domain) into the image normalization process.

model	source data (MNIST)	target data (SVHN)
AdaMatch	79.39%	20.46%
<b>AdaMatch+CN-Channels</b>	<b>99.21%</b>	<b>43.80%</b>

TABLE IX: Test accuracy of AdaMatch and AdaMatch with context normalization (AdaMatch+CN-Channels) using source domain (MNIST) as context identifier.

model	source data (MNIST)	target data (SVHN)
AdaMatch	79.39%	20.46%
<b>AdaMatch+CN-Channels</b>	<b>94.45%</b>	<b>23.22%</b>

TABLE X: Test accuracy of AdaMatch and AdaMatch with context normalization (AdaMatch+CN-Channels) using target domain (SVHN) as context identifier.

In general, the clear improvement that context normalization brings in terms of validation can be seen in Tables IX and X. Normalizing with CN-Channels takes the target task into account. The source domain (MNIST) is labeled as opposed to the target domain (SVHN), which could explain why using MNIST as the context identifier in AdaMatch+CN-Channels (ref. Tables IX and X) gives better results.

## V. CONCLUSION

We have proposed a novel approach called "context normalization" (CN) that enhances deep neural network training in terms of training stability, fast convergence, higher learning rate, and viable activation functions. Similar to the conventional mixture normalization (MN) method, our approach is driven by the hypothesis that any continuous function can be approximated in some sense by a weighted sum of Gaussian distributions with finite mean vectors and covariance matrices. In other words, our methodology assumes that the data distribution is a mixture of Gaussian models. However, unlike the mixture normalization technique that invokes the expectation maximization (EM) algorithms to estimate the Gaussian components parameters, our proposed methodology relies on the notion of concept that represents a cluster of related data. In fact, a supervised deep neural network is built and trained in order to learn the Gaussian components parameters. Once these optimal values are determined after convergence, they are utilized during the CN procedure performed on a deep neural network activation layer. CN alleviates the slow estimation of Gaussian component parameters inherent to EM in the scenario of large datasets. Furthermore, unlike MN, CN provides non linear decision boundaries between context which reflects more reality. Our experimental results demonstrate the superiority of context normalization over batch normalization and mixture normalization, showcasing enhanced convergence and generalization performance. The proposed method, when applied specifically to images, introduces CN-Channels and CN-Patches for training, and CN and CN+ for inference. With its flexibility to adapt various representations and tasks, context normalization proves to be a valuable tool in some application such as image classification.

Our short-term perspective consists in merging seamlessly a gradient free optimization algorithm with a gradient-based error optimizer in order to reach global convergence. We believe that this objective will boost training of deep neural networks further. This precision margin gained allows gaining insight into the neuron activation level sensitivity.

## REFERENCES

- [1] A. Kessy, A. Lewin, and K. Strimmer, "Optimal whitening and decorrelation," *The American Statistician*, vol. 72, no. 4, pp. 309–314, 2018.
- [2] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [5] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training dnns: Methodology, analysis and application," *arXiv preprint arXiv:2009.12836*, 2020.
- [6] M. M. Kalayeh and M. 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.
- [7] D. M. Titterton and A. R. M., "A. f. smith, and ue makov, 1985 statistical analysis of infinite mixture distributions."
- [8] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *arXiv preprint arXiv:1610.07629*, 2016.
- [9] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189.
- [10] 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.
- [11] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [13] 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," *arXiv preprint arXiv:2010.11929*, 2020.
- [14] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (canadian institute for advanced research)," 2009. [Online]. Available: <http://www.cs.toronto.edu/~kriz/cifar.html>
- [15] —, "CIFAR-100 (canadian institute for advanced research)," 2009. [Online]. Available: <http://www.cs.toronto.edu/~kriz/cifar.html>
- [16] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [17] Y. Lou, Y. Bai, J. Liu, S. Wang, and L.-Y. Duan, "VERI-wild: A large dataset and a new method for vehicle re-identification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3235–3243.
- [18] P. Sermanet, S. Chintala, and 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.
- [19] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning," *Technical report*, 2017.
- [20] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *proceedings of the eighteenth annual acm-siam symposium on discrete algorithms*, ser. *soda'07*. philadelphia, pa, usa: Society for industrial and applied mathematics., 2007.
- [21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
- [22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [23] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.
- [24] Y. Xu, A. Noy, M. Lin, Q. Qian, H. Li, and R. Jin, "Wemix: How to better utilize data augmentation," *arXiv preprint arXiv:2010.01267*, 2020.
- [25] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.
- [26] G. Haro, M. Bertalmío, and V. Caselles, "Visual acuity in day for night," *International Journal of Computer Vision*, vol. 69, pp. 109–117, 2006.
- [27] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.
- [28] A. Farahani, S. Voghoci, K. Rasheed, and H. 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.
- [29] D. Berthelot, R. Roelofs, K. Sohn, N. Carlini, and A. Kurakin, "Adamatch: A unified approach to semi-supervised learning and domain adaptation," *arXiv preprint arXiv:2106.04732*, 2021.
- [30] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.