Normalisation Contextuelle : Une Nouvelle Approche pour la Stabilité et l'Amélioration des Performances des Réseaux de Neurones

Bilal FAYE*, Hanane AZZAG**, Mustapha Lebbah***, Fangchen FANG****

*LIPN, UMR CNRS 7030, Université Sorbonne Paris Nord, 93430 Villetaneuse, France
faye@lipn.univ-paris13.fr
**LIPN, UMR CNRS 7030, Université Sorbonne Paris Nord, 93430 Villetaneuse, France
azzag@univ-paris13.fr
***Laboratoire DAVID, Université Paris Saclay, 78035 Versailles, France
mustapha.lebbah@uvsq.fr
****Laboratoire L2TI, Université Sorbonne Paris Nord, 93430 Villetaneuse, France
fangchen.feng@univ-paris13.fr

Résumé. L'apprentissage des réseaux de neurones est confronté à des défis majeurs liés au changement de distribution en couches, perturbant ainsi la convergence et les performances des modèles. La Normalisation par lot (BN) a révolutionné ce domaine, mais repose sur l'hypothèse simplifiée d'une seule composante gaussienne par lot. Pour remédier à cela, la Normalisation par Mélange (MN) a adopté une approche basée sur le modèle de mélange gaussien (GMM), mais avec des coûts computationnels importants liés à l'algorithme Espérance-Maximisation (EM) pour déterminer des composantes. Notre solution, la Normalisation Contextuelle (CN), regroupe des observations similaires en "contextes" pour une représentation locale, sans nécessiter d'algorithme de construction de ces contextes. Les paramètres de normalisation sont appris de manière similaire aux poids du modèle, assurant rapidité, convergence et performances supérieures par rapport à BN et MN.

1 Introduction

La normalisation est une opération courante en traitement des données, impliquant des transformations pour obtenir une représentation avec des propriétés statistiques spécifiques Kessy et al. (2018). Elle est utilisée pour égaliser les amplitudes des variables, accélérant ainsi la convergence dans les réseaux de neurones monocouches LeCun et al. (2002). Cependant, dans les réseaux multicouches, la normalisation des entrées peut ne pas avoir d'impact significatif sur les autres couches, car les distributions de données évoluent avec les mises à jour des poids du réseau. Pour garantir la stabilité de l'apprentissage à travers toutes les couches, diverses techniques de normalisation sont nécessaires, notamment la normalisation des activations, des poids et des gradients. La normalisation par lot (batch) (BN) Ioffe et Szegedy (2015) est une méthode courante pour stabiliser l'apprentissage dans les réseaux de neurones multicouches.

Elle standardise les activations de chaque couche en utilisant la moyenne et l'écart-type du batch auquel elles appartiennent. Cette méthode permet l'utilisation de taux d'apprentissage plus élevés, accélérant ainsi l'apprentissage. Cependant, BN présente deux inconvénients majeurs : elle dépend de la taille du batch d'entraînement, ce qui peut nuire aux performances avec de petits batchs, et elle suppose que les observations d'un même batch proviennent d'une même distribution, ce qui peut être trop restrictif pour des données complexes et variées. Des variantes spécialisées de la BN ont été proposées pour remédier aux limitations liées à la taille du batch Huang et al. (2020). Une autre approche, la normalisation par mélange (MN) Kalayeh et Shah (2019), a été introduite pour contourner les limitations découlant de l'hypothèse d'une composante unique. Dans le contexte de MN, il est considéré que les observations d'un même lot ne sont pas nécessairement issues de la même distribution. Elle utilise l'algorithme Espérance-Maximisation (EM) Dempster et al. (1977) pour regrouper les observations en fonction de leurs composantes et normalise ensuite chaque observation en utilisant les statistiques de sa composante respective. MN améliore la performance et la convergence par rapport à BN, notamment pour les réseaux de neurones convolutifs. Cependant, l'utilisation d'EM dans MN entraîne une augmentation significative du temps de calcul, limitant ainsi son efficacité. Pour résoudre ce problème, nous proposons une nouvelle approche de normalisation nommée Normalisation Contextuelle (CN). CN partage l'hypothèse de MN concernant la distribution des observations au sein d'un batch, mais elle introduit une structure appelée "contexte" pour regrouper les observations en ensembles cohérents sans recourir à l'algorithme EM. Chaque contexte suit une distribution gaussienne dans un espace latent de représentation, et les observations d'un même contexte sont normalisées en utilisant des paramètres appris lors de la rétropropagation du gradient. Cette approche facilite la tâche du réseau de neurones, accélère l'apprentissage et surmonte les limitations de MN liées au temps de calcul élevé.

Cette étude présente trois contributions significatives :

— CN : Méthode novatrice pour une normalisation adaptative grâce à des contextes prédéfinis, simulant des métaclasses et des clusters dans des espaces latents gaussiens. Les paramètres, acquis pendant l'entraînement, supplantent l'algorithme EM.
— Une couche adaptable à diverses architectures de réseaux de neurones, y compris les Transformeurs et les réseaux de neurones convolutifs.
— Inspirés par des résultats prometteurs en classification, CN est déployée en adaptation de domaine, boostant performance et convergence de l'algorithme AdaMatch. Berthelot et al. (2021).

2 État de l'art

2.1 Normalisation par lot (BN)

Soit $x \in \mathbb{R}^{N \times C \times H \times W}$, un tenseur d'activation dans un réseau de neurones convolutif, où les axes N, C, H, et W représentent respectivement le lot (batch), le nombre de canaux, la hauteur et la largeur. On pose B, le sous-ensemble résultant de l'aplatissement de x sur tous les axes sauf celui des canaux. Pour toute activation x_i dans $B = \{x_{1:m} : m \in [1, N] \times$

$[1, H] \times [1, W]\}$, la standardisation avec BN s'écrit sous la forme :

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

où $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ et $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ représentent respectivement la moyenne et la variance de B, tandis que $\epsilon > 0$ est une petite valeur qui gère les instabilités numériques. La transformation décrite dans l'Équation (1) génère une distribution ayant une moyenne nulle et une variance unitaire si les observations à l'intérieur du batch proviennent de la même distribution (hypothèse utilisée par BN). Cette contrainte de moyenne nulle et de variance unitaire permet de stabiliser la distribution des activations et, par conséquent, d'accélérer l'apprentissage du réseau de neurones. Bien que la BN présente de bonnes performances, elle est sensible à la taille du batch et repose sur l'hypothèse limitée d'une composante unique pour l'ensemble des observations d'un même batch. Afin de relever ces défis, de nombreuses variantes et méthodes alternatives ont été développées Kalayeh et Shah (2019); Huang et al. (2020), et nous en présenterons quelques-unes dans la section suivante.

2.2 Variantes de la BN

Quelques extensions de BN ont été proposées, notamment la normalisation par couche (LN) la normalisation par instance (IN), la normalisation par groupe (GN), et la normalisation par mélange (MN) Kalayeh et Shah (2019); Huang et al. (2020).
La transformation générale $x \rightarrow \hat{x}$, commune à toutes ces variantes, est appliquée sur x avec un aplatissement sur la dimension spatiale $L = H \times W$, et peut être formulée de la manière suivante :

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

où $B_i = \{j : j_N \in [1, N], j_C \in [i_C], j_L \in [1, L]\}$, et $i = (i_N, i_C, i_L)$ est un vecteur d'indexation pour les activations $x \in \mathbb{R}^{N \times C \times L}$.
LN : Élimine les interdépendances entre activations en normalisant chaque neurone de manière spécifique à sa couche. Convient aux réseaux de neurones récurrents.
IN : Normalise chaque activation individuellement, utile pour supprimer les informations de style, notamment en traitement d'images.
GN : Divise les neurones en groupes et normalise indépendamment les activations au sein de ces groupes. Elle est efficace pour les tâches visuelles avec de petites tailles de batchs, telles que la détection d'objets et la segmentation.
MN : Utilise un modèle de mélange gaussien (GMM) pour normaliser les activations en fonction de plusieurs modes de variation dans leur distribution sous-jacente, adapté lorsque la distribution des activations présente plusieurs modes.

MN normalise chaque activation x_i dans B_i en utilisant la moyenne et l'écart-type de la composante du mélange à laquelle x_i appartient. La fonction de densité de probabilité p_θ peut être représentée sous la forme d'un GMM.
Soit $x \in \mathbb{R}^D$, et $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, ..., K\}$, alors on a : $p(x) = \sum_{k=1}^{K} \lambda_k p(x|k)$, s.t. $\forall_k$: $\lambda_k \geq 0$, $\sum_{k=1}^{K} \lambda_k = 1$, où $p(x|k) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}{2}\right)$, est la k-ème composante, μ_k la moyenne et Σ_k la matrice de covariance.
La probabilité que x ait été généré par la k-ème composante gaussienne peut être définie

comme suit : $\tau_k(x) = p(k|x) = \frac{\lambda_k p(x|k)}{\sum_{j=1}^{K} \lambda_j p(x|j)}$. En se basant sur ces hypothèses et la transformation générale dans l'Équation (2), la normalisation de x_i, est définie comme suit :

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

avec

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

où $\hat{\tau}_k(x_i) = \frac{\tau_k(x_i)}{\sum_{j \in B_i} \tau_k(x_j)}$, est la contribution normalisée de x_i sur B_i dans l'estimation des statistiques de la k-ème composante gaussienne.
Avec cette approche, MN peut être alors appliquée en deux étapes :

1. Estimation des paramètres de chaque composante Gaussienne $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, ..., K\}$ à l'aide de l'algorithme Espérance-Maximisation (EM) Dempster et al. (1977).
2. Normalisation de chaque x_i avec les paramètres de la composante à laquelle elle appartient (Équation (3)).

En comparaison avec la BN, la MN favorise une meilleure convergence et des performances accrues dans les réseaux de neurones convolutifs sur des tâches d'apprentissage supervisé.

3 Méthode Proposée : Normalisation Contextuelle (CN)

3.1 Description de la Méthode

CN, à l'instar de la normalisation par mélange, modélise efficacement les données sous l'hypothèse d'un mélange de composantes gaussiennes dans un espace latent. Dans cette approche, un "contexte" est défini comme un groupe d'observations partageant des caractéristiques similaires. Les observations au sein d'un même contexte sont regroupées et normalisées avec des paramètres appris par le réseau, le tout sans nécessiter le recours à l'algorithme EM coûteux. Chaque contexte est identifié par un identifiant unique r, simplifiant ainsi le processus de normalisation. CN peut être formulée comme l'équation 2 lorsque

$$B_i = \{j : j_N \in [i_N], j_C \in [i_C], j_L \in [1, L]\}$$

3.2 Apprentissage des paramètres

Considérons un tenseur d'activation x dans l'espace $\mathbb{R}^{N \times C \times H \times W}$, où B_i, un groupe d'activations résultant de l'aplatissement de x le long de l'axe $L = H \times W$. Chaque x_i dans B_i est normalisée avec les paramètres (μ_r, σ_r) du contexte r auquel il est associé :

$$\hat{x}_i \leftarrow \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} \tag{5}$$

La moyenne (μ_r) et l'écart-type (σ_r) sont apprises comme des poids du réseau de neurones par normalisation des activations résultantes de la transformation des observations du contexte r,

Algorithme 1 : CN- Transformation : Normalisation d'une activation

Entrée : $x_i \in B_i$; contexte r associé à x_i
Sortie : $\{\hat{x_i}$ = CN$(x_i, r)\}$

$\alpha_r \leftarrow onehot(r)$ // *encodage de la variable catégorielle* r

$\mu_r \leftarrow Encodeur_\mu(\alpha_r)$ // *estimation de la moyenne*

$\sigma_r^2 \leftarrow Encodeur_\sigma(\alpha_r)$ // *estimation de la variance*

$\hat{x}_i$ = CN(x_i, r) : Normaliser x_i avec l'Équation 5

Algorithme 2 : CN- Entraînement : Normalisation d'une activation

Entrée : Réseau de neurones Net avec des paramètres Θ ; $B_i = \{x_i\}_{i=1}^m$;
Sortie : Réseau de neurones Net_{CN}^{tr} normalisé

$Net_{CN}^{tr} = Net$ // *Initialisation des paramètres du réseau de neurones normalisé avec CN*
pour $i \leftarrow 1$ *à* m **faire**
- — Identifier le contexte r associé à x_i
- — Appliquer l'Algorithme 1 : $\hat{x}_i =$ CN(x_i, r)
- — Modifier chaque couche de Net_{CN}^{tr} en remplaçant x_i par $\hat{x}_i$

Propager le gradient dans Net_{CN}^{tr} pour optimiser : $\Theta = \Theta \cup \{\mu_r, \sigma_r\}_{r=1}^T$

comme indiqué dans l'Algorithme 1. Dans l'algorithme 1, nous utilisons une transformation affine de la forme :

$$\begin{cases} \alpha_r \leftarrow onehot(r) \\ \mu_r \leftarrow Encodeur_\mu(\alpha_r) : W_\mu\ \alpha_r + b_\mu \\ \sigma_r^2 \leftarrow Encodeur_\sigma(\alpha_r) : W_\sigma\ \alpha_r + b_\sigma \end{cases} \tag{6}$$

Dans l'Équation 6, W et b sont les paramètres de $Encodeur$, et r est le contexte associé à x_i. La fonction onehot$(.)$ effectue l'encodage one-hot de la variable catégorielle r, pour donner α_r.

Durant la phase d'apprentissage du réseau de neurones avec une normalisation contextuelle, conformément à l'Algorithme 2, il est nécessaire de rétropropager le gradient de la fonction de perte, ℓ, à travers la transformation. De plus, les gradients par rapport aux paramètres de la transformation CN doivent être calculés. Cela est réalisé en utilisant la règle de la chaîne, comme le montre l'expression suivante (avant simplification) :

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

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

CN est une opération différentiable qui normalise les activations dans les réseaux de neurones, favorisant l'adaptation aux distributions d'entrée pour de meilleures performances après

convergence. Après cette convergence, CN crée des composantes gaussiennes dans un espace latent, fournissant une représentation adaptée à la tâche.

Lors de l'inférence, nous avons le choix entre normaliser les activations avec CN ou considérer tous les contextes collectivement. Cette amélioration est appelée CN+ et est décrite dans l'Algorithme 3. Soit T, le nombre de contextes, pour un x_i quelconque dans B_i, en reformulant l'Équation 7, CN+ s'écrit comme suit :

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

avec

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

où

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

en supposant que les probabilités a priori ($\lambda_r = \frac{1}{T}, r = 1, ..., T$) sont constantes.

Algorithme 3 : CN- Inférence

Entrée : Réseau de neurones Net_{CN}^{tr} avec les paramètres $\Theta \cup \{\mu_r, \sigma_r\}_{r=1}^{T}$ (réf. Algorithme 2) ; $B_i = \{x_i\}_{i=1}^{m}$; $choix \in$ {CN, CN+} ;

Sortie : Réseau de neurones avec des paramètres gelés Net_{CN}^{inf} ; B_i normalisé

$Net_{CN}^{inf} \leftarrow Net_{CN}^{tr}$ // *Initialisation du réseau de neurones pour l'inférence avec des paramètres gelés*

si $choix =$ *CN* **alors**

  pour $i \leftarrow 1$ *à* m **faire**

    — Pour un contexte r donné

    — Prendre les paramètres μ_r et σ_r correspondants

    — Normaliser x_i : $\hat{x}_i \leftarrow \text{CN}(x_i, r)$

si $choix =$ *CN+* **alors**

  pour $i \leftarrow 1$ *à* m **faire**

    — Calculer $\hat{x}_i$ avec l'Équation 7 // *Normalisation + agrégation*

4 Expériences

Dans cette section, nous comparons la normalisation contextuelle (CN) à d'autres méthodes, notamment la normalisation par lot (BN) et la normalisation par mélange (MN), sur différentes architectures, dont les Réseaux de Neurones Convolutifs (CNN) (voir Section 4.2 et Section 4.4) et les Transformeurs de Vision (ViT) Dosovitskiy et al. (2020) (voir Section 4.3). Nous évaluons ces approches sur diverses tâches, notamment la classification et l'adaptation de domaine.

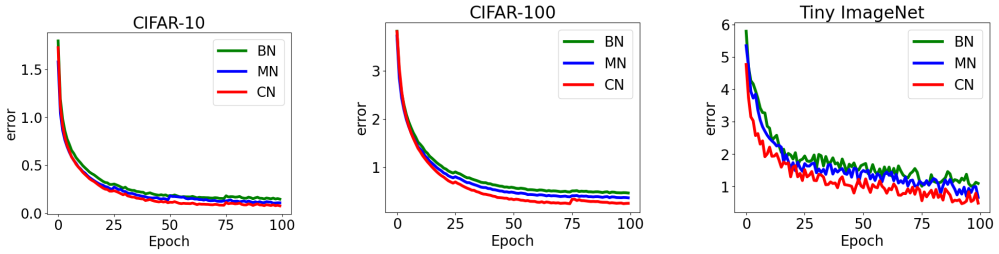
4.1 Ensembles de données

Les expériences de cette étude s'appuient sur plusieurs jeux de données de référence couramment utilisés en classification. **CIFAR-10** et **CIFAR-100** comprennent respectivement 50,000 images d'entraînement et 10,000 images de test, avec une taille de 32×32 pixels Krizhevsky et al. (2009a,b). CIFAR-100, une extension de CIFAR-10, se divise en 100 classes regroupées en 20 superclasses. **Tiny ImageNet** est une version réduite d'ImageNet avec 200 classes Le et Yang (2015). **MNIST digits** contient 70,000 images représentant les 10 chiffres LeCun et Cortes (2010). **SVHN** focalise sur la reconnaissance des chiffres dans des scènes naturelles, totalisant plus de 600,000 images Sermanet et al. (2012). Cette consolidation permet d'optimiser l'espace tout en préservant les informations essentielles.

4.2 Etude Comparative : Normalisation Contextuelle (CN) vs. Normalisation par Mélange (MN) et Normalisation par Lot (BN) sur CIFAR et Tiny ImageNet

Dans cette expérience, nous utilisons un réseau de neurones convolutifs (ConvNet+BN) peu profonde. Ce réseau est composé de quatre couches de convolution avec activation ReLU, chacune suivie par une couche de BN. Une des limites de BN est la baisse de performance lors de l'application de fonctions non linéaires (par exemple, ReLU) après la normalisation des activations. À travers ConvNet+BN, nous illustrons comment la CN résout ce problème et améliore la convergence et les performances globales en remplaçant une couche de BN par une couche de CN sur le réseau. Pour ce faire, nous commençons par appliquer l'algorithme d'Estimation du Maximum de Vraisemblance (MLE) Dempster et al. (1977) sur chaque ensemble de données (CIFAR-10, CIFAR-100 et Tiny ImageNet) afin de déterminer les paramètres de chaque composante gaussienne, en utilisant $k = 3$. Ensuite, nous créons trois modèles distincts sur chaque ensemble de données : le premier modèle, ConvNet+BN, consiste en des couches de normalisation BN uniquement. Le deuxième modèle, nommé ConvNet+MN, est obtenu en remplaçant la troisième couche BN de ConvNet+BN par une couche de MN. Enfin, le troisième modèle, appelé ConvNet+CN, est obtenu en remplaçant la troisième couche BN de ConvNet+BN par une couche de CN. Les paramètres des composantes gaussiennes ainsi obtenus sont utilisés par la couche MN dans ConvNet+MN pour normaliser les activations, tandis que la couche CN dans ConvNet+CN utilise ces composantes gaussiennes comme contextes, pour normaliser les activations du même contexte, avec des paramètres (μ_r et σ_r^2 avec $r = 1, ..., 3$) qui sont appris pendant l'entraînement du réseau ConvNet+CN.

Les modèles sont entraînés dans des conditions uniformes sur 100 époques. Pendant la phase d'apprentissage, l'accuracy de chaque modèle est enregistré tous les 25 époques sur les données de validation, ce qui permet de suivre de près l'évolution des performances, comme le montre la Table 1. Le modèle ConvNet+CN, qui intègre la normalisation contextuelle, présente des performances supérieures sur l'ensemble des époques enregistrés, et cet avantage s'intensifie quel que soit le nombre de classes dans l'ensemble de données à prédire (10, 100 et 200). Ce même effet est également observé lors de l'utilisation de l'approche CN+ en inférence sur le modèle ConvNet+CN, comme décrit dans la Table 2. En plus d'améliorer les performances, la normalisation contextuelle CN dans ConvNet+CN accélère la convergence du réseau de neurones, comme illustré sur la Figure 1. Ce phénomène est cohérent avec les améliorations de performances observées.



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny ImageNet

FIG. 1 – Evolution de l'erreur durant l'entraînement des modèles ConvNet+BN, ConvNet+MN et ConvNet+CN, sur les ensembles de données CIFAR-10, CIFAR-100 et Tiny ImageNet.

CIFAR-10				
modèle	25 époques	50 époques	75 époques	100 époques
ConvNet+BN	84,34	86,49	86,41	86,90
ConvNet+MN	84,45	86,60	86,6	87,07
ConvNet+CN	**85,99**	**86,93**	**87,09**	**87,11**

CIFAR-100				
modèle	25 époques	50 époques	75 époques	100 époques
ConvNet+BN	57,41	59,74	59,82	59,82
ConvNet+MN	56,90	59,80	59,80	60,10
ConvNet+CN	**57,74**	**61,01**	**61,01**	**61,01**

Tiny ImageNet				
modèle	25 époques	50 époques	75 époques	100 époques
ConvNet+BN	37,17	37,17	37,17	37,17
ConvNet+MN	38,18	38,17	38,5	38,5
ConvNet+CN	**38,53**	**40,08**	**40,08**	**40,08**

TAB. 1 – Évolution de l'accuracy (%) sur les ensemble de données de validation de CIFAR-10, CIFAR-100 et Tiny ImageNet pour chaque modèle : ConvNet+BN, ConvNet+MN et ConvNet+CN. La validation se fait après chaque 25 époques d'apprentissage.

En résumé, à travers les résultats présentés dans la Table 1 et la Figure 1, notre méthode de normalisation contextuelle démontre sa capacité à améliorer les performances et à accélérer la convergence des réseaux de neurones convolutifs complexes. Elle surmonte les limitations de la BN liées à l'utilisation de fonctions non linéaires (comme ReLu dans cet exemple) et dépasse même l'approche utilisant MN.

L'approche présentée dans cette expérimentation utilise des composantes gaussiennes trouvées via MLE comme contextes, mais c'est fastidieux. Dans les sections suivantes, nous simplifierons la définition des contextes sans recourir à des algorithmes complexes. La MN ne sera plus utilisée, car elle exige la préconstruction de composantes gaussiennes (une limite). Étant donné que CN se comporte de manière similaire à CN+, elle sera privilégiée pour l'inférence, étant plus économique en ressources.

CN+			
Ensemble de données	CIFAR-10	CIFAR-100	Tiny ImageNet
accuracy (%)	**88,09**	60,28	**40,10**

TAB. 2 – Évaluation de l'accuracy (%) de ConvNet+CN (CN+, décrite dans l'Algorithme 3 est utilisée comme méthode d'inférence) sur les données de test de CIFAR-10, CIFAR-100 et Tiny ImageNet.

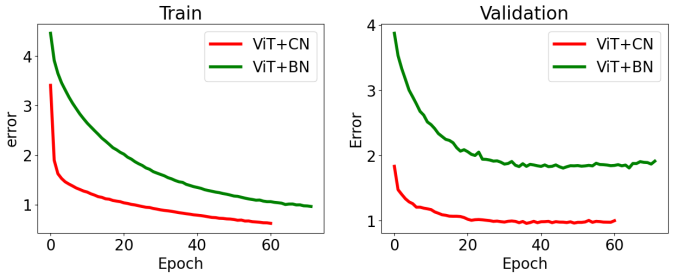


FIG. 2 – Évolution de l'erreur sur les ensembles d'entraînement et de validation de CIFAR-100 pendant l'entraînement des modèles ViT+BN et ViT+CN.

modèle	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**

TAB. 3 – Évaluation des performances de deux modèles ViT sur le jeu de données de test de CIFAR-100, l'un utilisant la normalisation par lots (ViT+BN) et l'autre la normalisation contextuelle (ViT+CN).

4.3 Normalisation Contextuelle avec les Superclasses de CIFAR-100 comme Contextes

Cette expérience vise à créer des contextes dans un ensemble de données, sans nécessiter d'algorithme, contrairement à celle décrite dans la Section 4.2. Nous utilisons l'exemple de CIFAR-100, qui comprend 100 classes à prédire et 20 superclasses regroupant ces classes. Les superclasses sont utilisées comme contextes pour notre approche de normalisation CN. Dans cette approche, les observations appartenant au même contexte (superclasse) r sont normalisées avec les mêmes paramètres (μ_r et σ_r) appris pendant l'entraînement du réseau de neurones. Le réseau de base utilisé est un modèle ViT issu de Keras Salama (2021). Deux variantes sont construites à partir de ce modèle. La première incorpore une couche de normalisation par lots (BN) en tant que première couche pour normaliser les images (ViT+BN), tandis que la seconde intègre directement notre couche de normalisation contextuelle (CN) en tant que première couche du modèle ViT pour normaliser les entrées (ViT+CN). Les deux modèles sont entraînés avec des paramètres identiques sur 200 époques.

La Figure 2 illustre une convergence plus rapide du modèle ViT+CN par rapport au modèle ViT+BN. Cette observation est cohérente sur les ensembles d'entraînement et de validation, renforçant l'idée que la normalisation contextuelle CN accélère l'apprentissage des réseaux de neurones. En plus de cette accélération, CN améliore la performance, comme démontré dans la Table 3, où des différences significatives sont notées sur l'ensemble des métriques. L'incorporation de CN, comme illustré par ViT+CN, permet d'obtenir une augmentation de l'accuracy de **10%**, un niveau d'accuracy jamais atteint par un modèle ViT entraîné à partir de zéro sur l'ensemble de données CIFAR-100.

Cette expérience démontre une méthode simple pour créer des contextes sans algorithme, accélérant la convergence et améliorant la performance des réseaux de neurones. Cependant,

la structure de superclasse n'est pas toujours facile à établir. La prochaine section présente une application typique de CN, où les contextes se révèlent clairement.

4.4 La Normalisation Contextuelle en Adaptation de Domaine

L'adaptation de domaine Farahani et al. (2021) transfère des connaissances du domaine source au domaine cible malgré les différences de distribution. Les modèles d'apprentissage automatique rencontrent des défis en raison de ces variations de distribution. En utilisant la normalisation contextuelle, on crée des représentations spécifiques à chaque domaine en fonction de la tâche cible, résolvant les problèmes liés aux différences de distribution, améliorant les performances et accélérant la convergence. Afin d'illustrer cette approche, nous prenons l'exemple de l'algorithme AdaMatch Berthelot et al. (2021), en considérant les domaines source et cible comme des contextes distincts ($r \in \{\text{domaine source, domaine cible}\}$). Nous utilisons un modèle de base, AdaMatch Paul (2019), créé à partir de réseaux résiduels de grande taille Zagoruyko et Komodakis (2016), pour servir de modèle de référence. À partir de ce modèle de référence, nous en construisons un autre (AdaMatch+CN) en intégrant une couche CN en tant que première couche du réseau. Cette couche permet de normaliser les données directement en fonction de leur contexte ou de leur domaine d'origine. Dans cette expérience, l'ensemble de données MNIST est utilisé en tant que domaine source, avec des étiquettes connues, tandis que l'ensemble de données SVHN sert de domaine cible, avec des étiquettes inconnues.

Afin de surveiller la stabilité des deux modèles, AdaMatch et AdaMatch+CN, lors de

MNIST (domaine cible)					**SVHN (domaine cible)**				
modèle	accuracy	precision	recall	f1-score	modèle	accuracy	precision	recall	f1-score
AdaMatch	97,36	87,33	79,39	78,09	AdaMatch	25,08	31,64	20,46	24,73
AdaMatch+CN	**99,26**	**99,20**	**99,32**	**99,26**	**AdaMatch+CN**	**43,10**	**53,83**	**43,10**	**47,46**

TAB. 4 – Évaluation de la performance des modèles AdaMatch et AdaMatch+CN sur les données de validation des ensembles de données source (MNIST) et cible (SVHN).

l'entraînement sur le domaine cible, nous enregistrons la variabilité du gradient. Une faible variance du gradient est un indicateur de la stabilité du réseau, ce qui peut accélérer la convergence et améliorer les performances globales. Sur la Figure 3, on peut clairement observer que le modèle AdaMatch+CN affiche une variance du gradient considérablement réduite par rapport au modèle AdaMatch. Cela met en évidence le rôle bénéfique de la normalisation contextuelle (CN) dans la stabilisation du réseau en influençant le gradient, ce qui a un impact positif sur l'accélération de la convergence du modèle. Cet effet se traduit par une amélioration significative des performances, tant sur le domaine source (MNIST) que sur le domaine cible (SVHN), comme en témoigne la Table 4, où l'on observe une augmentation de l'accuracy de **18%** sur le domaine cible.

En somme, l'adaptation de domaine est un exemple concret de l'efficacité de la normalisation contextuelle. En associant chaque contexte à un domaine, cette approche réduit la variabilité du gradient, favorisant la stabilité de l'apprentissage. Cela se traduit par une convergence plus rapide, comme le montre l'exemple d'AdaMatch, et des performances de généralisation améliorées sur l'ensemble des domaines (source et cible).

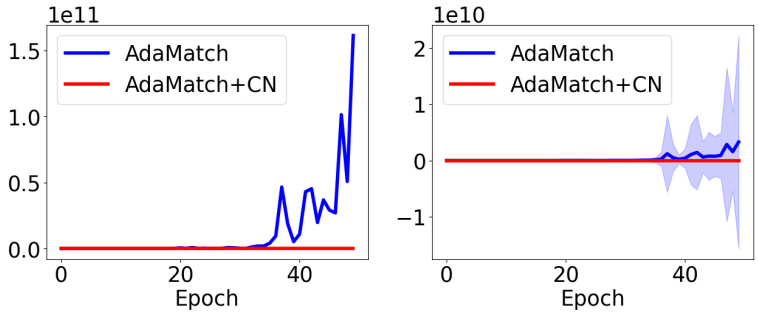


FIG. 3 – Variation de la variance du gradient sur le domaine cible (SVHN) pendant l'entraînement des deux modèles : AdaMatch et AdaMatch+CN. Graphique de gauche : Variation du gradient maximal. Graphique de droite : Variation du gradient moyen.

5 Conclusion

La normalisation contextuelle (CN) représente une avancée significative dans l'entraînement des réseaux de neurones. En partageant une philosophie similaire avec la normalisation par mélange (MN), CN élimine le besoin de l'estimation coûteuse de l'algorithme Espérance-Maximisation (EM) utilisée par MN, en apprenant directement les paramètres des distributions gaussiennes à partir de groupes de données nommés "contextes". Nos expériences ont démontré que CN surpassait à la fois la normalisation par lots (BN) et la normalisation par mélange (MN), offrant une meilleure convergence et des performances de généralisation améliorées. Trois méthodes ont été explorées pour définir les contextes : l'utilisation des composantes d'un modèle de mélange gaussien (GMM), l'identification des superclasses dans un ensemble de données, et l'application en adaptation de domaine en considérant chaque domaine comme un contexte distinct.

À court terme, nous visons à créer une variante non supervisée de CN en éliminant le besoin d'entrée du contexte, optant plutôt pour son estimation pendant l'apprentissage du réseau de neurones.

Références

Berthelot, D., R. Roelofs, K. Sohn, N. Carlini, et A. Kurakin (2021). Adamatch : A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv :2106.04732*.

Dempster, A. P., N. M. Laird, et D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B 39*(1), 1–38.

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. (2020). An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*.

Farahani, A., S. Voghoei, K. Rasheed, et H. R. Arabnia (2021). A brief review of domain adaptation. *Advances in Data Science and Information Engineering : Proceedings from ICDATA 2020 and IKE 2020*, 877–894.

Huang, L., J. Qin, Y. Zhou, F. Zhu, L. Liu, et L. Shao (2020). Normalization techniques in training dnns : Methodology, analysis and application. *arXiv preprint arXiv :2009.12836*.

Ioffe, S. et C. Szegedy (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR.

Kalayeh, M. M. et M. Shah (2019). Training faster by separating modes of variation in batch-normalized models. *IEEE transactions on pattern analysis and machine intelligence 42*(6), 1483–1500.

Kessy, A., A. Lewin, et K. Strimmer (2018). Optimal whitening and decorrelation. *The American Statistician 72*(4), 309–314.

Krizhevsky, A., V. Nair, et G. Hinton (2009a). CIFAR-10 (canadian institute for advanced research).

Krizhevsky, A., V. Nair, et G. Hinton (2009b). CIFAR-100 (canadian institute for advanced research).

Le, Y. et X. Yang (2015). Tiny imagenet visual recognition challenge. *CS 231N 7*(7), 3.

LeCun, Y., L. Bottou, G. B. Orr, et K.-R. Müller (2002). Efficient backprop. In *Neural networks : Tricks of the trade*, pp. 9–50. Springer.

LeCun, Y. et C. Cortes (2010). MNIST handwritten digit database.

Paul, S. (2019). Unifying semi-supervised learning and unsupervised domain adaptation with adamatch. https ://github.com/keras-team/keras-io/tree/master.

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

Sermanet, P., S. Chintala, et Y. LeCun (2012). Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pp. 3288–3291. IEEE.

Zagoruyko, S. et N. Komodakis (2016). Wide residual networks. *arXiv preprint arXiv :1605.07146*.

Summary

Neural network training encounters challenges from distribution shifts between layers, impacting model convergence and performance. While Batch Normalization (BN) transformed the field, it relies on a simplified assumption of a single Gaussian component per batch. Mixture Normalization (MN) addresses this with a Gaussian Mixture Model (GMM) approach, yet incurs significant computational costs with the Expectation-Maximization (EM) algorithm. Our solution, Contextual Normalization (CN), introduces "contexts" by grouping similar observations for local representation, eliminating the need for context construction algorithms. Learning normalization parameters akin to model weights ensures speed, convergence, and outperforms BN and MN.