Un Mixup Local pour empêcher les intrusions de variétés

Raphaël BAENA, Lucas DRUMETZ, Vincent GRIPON

IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France
prenom.nom@imt-atlantique.fr

Résumé – Utilisé dans le domaine de l'apprentissage supervisé, Mixup est une méthode de régularisation dépendante des données qui consiste à générer par interpolation linéaire de nouveaux exemples. Cette méthode améliore la généralisation sur des jeux de données standard d'apprentissage. Cependant, plusieurs auteurs ont souligné que Mixup peut générer des exemples en dehors de la distribution d'entrée ou parfois même contradictoires, pouvant être ainsi néfastes pour l'apprentissage. Dans cet article, nous introduisons *Local Mixup*, une méthode où les interpolations entre points distants sont pondérées plus faiblement dans la fonction de coût. Sous certaines hypothèses, nous prouvons que *Local Mixup* permet de contrôler un compromis bias/variance et de retrouver dans les cas extrêmes Mixup et un modèle sans mixage. Sur des jeux de données standard de vision par ordinateur, nous montrons que *Local Mixup* améliore la généralisation.

Abstract – Deployed in the context of supervised learning, Mixup is a data-dependent regularization technique that consists in linearly interpolating input samples and associated outputs. It has been shown to improve accuracy when used to train on standard machine learning datasets. However, authors have pointed out that Mixup can produce out-of-distribution virtual samples and even contradictions in the augmented training set, potentially resulting in adversarial effects. In this paper, we introduce *Local Mixup* in which distant input samples are weighted down when computing the loss. In constrained settings we demonstrate that *Local Mixup* can create a trade-off between bias and variance, with the extreme cases reducing to vanilla training and classical Mixup. Using standardized computer vision benchmarks, we also show that *Local Mixup* can improve accuracy.

1 Introduction

En apprentissage automatique, de nombreux modèles peuvent apprendre sans erreur les données d'entraînement sans pour autant généraliser correctement à de nouvelles données. C'est pourquoi il est souvent plus utile d'apprendre un modèle remettant partiellement en question les données d'entraînement (biais élevé) mais exhibant des régularités aidant à une bonne généralisation (variance faible).

Pour aider à mieux généraliser, des méthodes de régularisation ont été introduites [6] ; parmi elles, les techniques dites d'*augmentation de données* se sont révélées particulièrement efficaces. En particulier, *Mixup* est une technique introduite dans [16] présentées comme une méthode d'augmentation de données. L'idée est d'augmenter le jeu d'entraînement par des interpolations linéaires entre paires : cette interpolation est valable tant pour les entrées (signaux) que pour les sorties (labels) correspondantes. Ils montrent que *Mixup* diminue l'erreur de généralisation sur plusieurs jeux de données standard et également en apprentissage avec peu de données [11, 5].

Cependant, les exemples *virtuels* générés par interpolations linéaires peuvent dans certains cas être sources de contradictions ou se trouver en dehors de la distribution d'entrée. Ce phénomène est décrit et nommé *intrusion de la variété* dans [7]. Ainsi, il n'est pas évident que *Mixup* soit toujours bénéfique. Plus généralement, la question se pose de savoir si *Mixup* pourrait être amélioré afin d'éviter la génération de tels exemples contradictoires.

Dans cet article, nous présentons *Local Mixup*, une méthode où les exemples *virtuels* sont pondérés dans la fonction de coût. Le poids de chaque exemple dépend de la distance entre les deux points utilisés pour effectuer l'interpolation $(\mathbf{x}_i, \mathbf{x}_j)$. En particulier, cette méthode peut interdire complètement les interpolations entre des points trop distants afin de réduire le risque d'intrusion de variétés comme nous en discuterons plus tard.

Nos contributions principales sont les suivantes :

— Nous introduisons *Local Mixup*, une méthode qui dépend d'un seul paramètre et qui généralise *Mixup* et un réseau sans mixage.

— En dimension 1, nous prouvons que *Local Mixup* permet de sélectionner un compromis biais/variance.

— En plus grande dimension, nous montrons que *Local Mixup* peut atteindre une précision plus importante que *Mixup* sur des jeux de données de vision par ordinateur.

— De manière générale, nos travaux contribuent à apporter une meilleure compréhension de l'impact de *Mixup* pendant la phase d'entraînement.

2 Etat de l'art

Notations On note f la fonction paramétrique du modèle à entraîner et $\mathcal{F}$ l'espace d'hypothèses, i.e, l'ensemble de toutes les paramétrisations de f. Pour entraîner le modèle, on utilise une fonction de coût $\mathcal{L}$ (ex. la cross entropie) qui mesure l'écart entre la sortie du modèle et la sortie attendue.

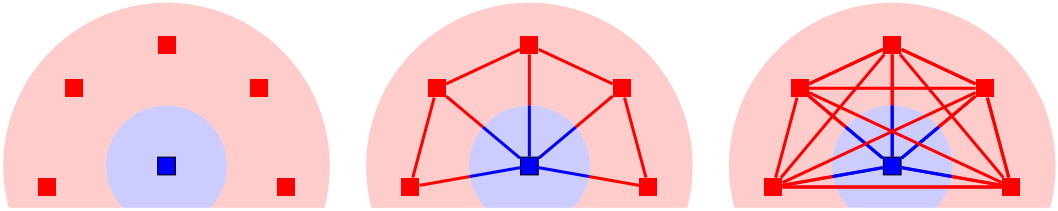

FIGURE 1 – Illustration de *Local Mixup*. Sur la gauche, pas de mixage. Au milieu *Local Mixup* où seuls les exemples proches sont interpolés (pas de contradiction). A droite *Mixup* où toutes les interpolations sont possibles (contradictions possibles).

Augmentation de données et mixup Pour améliorer la généralisation, on peut utiliser des méthodes de régularisation [6]. Parmi ces méthodes, l'augmentation de données est une forme de régularisation qui dépend des données [7]. Récemment, de telles méthodes reposant sur du mixage ont été présentées [16, 13, 15, 4, 8, 9, 3, 10, 2, 14, 12]. La méthode pionnière de mixage est *Mixup* [16], où les exemples virtuels $(\tilde{x}, \tilde{y})$ sont générés par interpolation linéaire entre paires d'exemples. La fonction f^* qui minimise le critère de *Mixup* est définie comme :

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n^2} \mathbb{E}_\lambda \left[\sum_{\mathcal{D}^2_{train}} \mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right) \right],$$

$$\text{où } [\tilde{\mathbf{x}}_{i,j,\lambda}, \tilde{\mathbf{y}}_{i,j,\lambda}]^T = \lambda[\mathbf{x}_i, \mathbf{y}_i]^T + (1-\lambda)[\mathbf{x}_j, \mathbf{y}_j]^T.$$

Mixup encourage une linéarité entre les exemples [16] ; cette propriété est questionnée par plusieurs auteurs qui essayent d'expliquer théoriquement et expérimentalement *Mixup*. Dans [7], les auteurs décrivent et présentent l'intrusion de variétés. Ce phénomène est illustré dans la Figure 1, schéma de droite, où les exemples virtuels générés par Mixup entre les points rouges se retrouvent en dehors de la variété pour la classe rouge. La méthode [7], dénommée *Adamixup*, utilise un réseau additionnel pour supprimer de telles interpolations.

3 Mixup en dimension 1

Les preuves de nos résultats sont disponibles sur une version plus longue de notre article [1]. On considère le cas simple où notre modèle f est défini sur $\mathbb{R}$. On souhaite montre que Mixup a pour effet de moyenner les interpolations et que la solution qui minimise le coût de Mixup est linéaire par morceau. Sans perte de généralité, on considère que le jeu d'entraînement est ordonné par ordre croissant selon les entrées, i.e, $\mathcal{D}_{train} = \{x_i, y_i\}$ et $x_i \leq x_{i+1}$.

Pour un exemple virtuel donné $\tilde{x}$, la fonction de coût de *Mixup* implique que la sortie du réseau $f^*(\tilde{x})$ est déterminée par l'ensemble $\mathcal{E}(\tilde{x})$ de toutes les combinaisons convexes qui peuvent générer $\tilde{x}$ à partir de deux entrées x_i and x_j : $\mathcal{E}(\tilde{x}) = \{i, j, \lambda_{i,j} | \tilde{x} = \lambda_{ij} x_i + (1 - \lambda_{ij})\, x_j\}$. On prouve que pour $x \in [x_0, x_n]$, $f^*(\tilde{x})$ est le barycentre des sorties données par $\mathcal{E}(\tilde{x})$.

Lemme 3.1. $\forall \tilde{x} \in [x_0, x_{n-1}]$,

$$f^*(\tilde{x}) = \frac{1}{card(\mathcal{E}(\tilde{x}))} \sum_{(i,j,\lambda_{i,j}) \in \mathcal{E}(\tilde{x})} \lambda_{i,j} y_i + (1 - \lambda_{i,j}) y_j. \tag{1}$$

Une conséquence de ce lemme est le théorème suivant :

Théorème 3.2. *La fonction f^* qui minimise la fonction de coût sur le jeu d'entraînement est une fonction continue par morceaux sur $[x_0, x_{n-1}]$, linéaire sur chaque segment $[x_i, x_{i+1}]$ et définie par l'équation* (1)*.*

En pratique, on ne cherche par à inférer une fonction f^* qui minimise le coût car elle risque ne pas généraliser correctement. On cherche plutôt une fonction f qui minimise suffisamment le coût pour garder l'effet régularisant. Toutefois on remarquera que f tend vers une fonction qui moyenne l'ensemble des combinaisons convexes et qui conduit donc à un modèle à faible variance.

4 Local Mixup

4.1 Graphe de localité

On propose de construire un graphe sur un jeu d'entraînement où les arrêtes sont pondérées par $\mathbf{W}$; une matrice déterminée par une fonction ϕ des distances entre les exemples $(D[i, j])$. Les poids forts correspondent aux paires d'exemples les plus proches. On considère plusieurs fonctions de pondération : le graphe des K plus proches voisins, où les pondérations sont 1 ou 0 ; Un graphe seuillé où $\mathbf{W}[i, j] = \phi(D[i, j])$ et $\phi(d) = \mathbf{1}_{d \leq \varepsilon}$; ou un graphe en exponentielle décroissante où $\mathbf{W}[i, j] = \exp(-\alpha \mathbf{D}[i, j])$. La fonction de coût est pondérée à partir de la matrice des poids des arêtes $\mathbf{W}$:

$$L_{\text{local mixup}} = \sum_{\mathcal{D}^2_{train}} \mathbf{W}[i, j] \mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right). \tag{2}$$

Les cas extrêmes où les poids sont à zéro correspondent à des exemples virtuels qui sont éliminés lors de la descente de gradient, ce qui fait que seulement les interpolations locales sont considérées, d'où le nom *Local Mixup*.

4.2 Faible dimension

Dans cette partie, on montre que *Local Mixup* permet de contrôler le compromis bias/variance d'un modèle entraîné. On considère une version simplifiée du problème en dimension une et avec un graphe des K plus proches voisins.

4.2.1 Local Mixup et le compromis biais/variance

Sur des cas simples, on montre que Mixup a pour effet de réduire la variance du model.

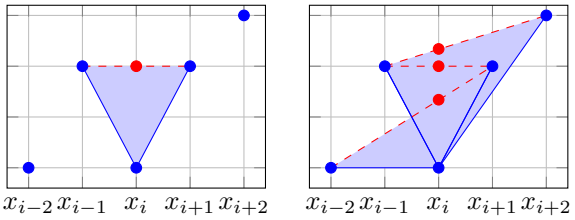

FIGURE 2 – On représente ici les interpolations déterminants $f_K^*(x_i)$ pour différentes valeurs de K. En bleu les $2K$ interpolations directes, en rouge interpolations indirectes. A droite $K = 2$ et à gauche $K = 1$. Les points sur l'abscisse x_i déterminent $f_K^*(x_i)$.

Définition 4.1 (Biais et Variance)**.** *Soit un jeu d'entraînement $\mathcal{D}_{train}$ et une fonction f de $\mathcal{X}$ dans $\mathcal{Y}$. On définit biais et variance comme :*

— Biais *: $Bias(f)^2 = \mathbb{E}_{train}[(f(x) - y)^2]$.*

— Variance *: $Var(f) = \mathbb{E}_{train}[(f - \mathbb{E}_{train}[f])^2]$.*

On considère deux cas d'étude. Premier cas, le domaine d'entrée $\mathbb{Z}/n\mathbb{Z}$ est périodique et le nombre d'exemples est fini. Dans le second cas, le domaine d'entrée est $\mathbb{Z}$ infini et les sorties sont indépendantes et identiquement distribuées (i.i.d).

Cas périodique

On considère un jeu d'entraînement $\mathcal{D}_{train}$ périodique. Dans ce cas, on peut écrire explicitement la fonction f_K^* qui minimise le coût de *Local Mixup* que l'on illustre sur la Figure 2. On représente les segments des interpolations qui interviennent dans f_K^* le barycentre des interpolation : directes entre x_i et ses voisins directs (en bleu), et indirectes (en rouge) entre les points autres que x_i qui intersectent x_i. Lorsque K augmente, l'influence des interpolations indirectes prédomine.

On a le théorème suivant :

Théorème 4.1 (Convergence de f_K^* dans le cas périodique)**.** *Lorsque K tend vers l'infini on a :*

$$\forall x_i \in \mathbb{Z}/n\mathbb{Z},$$
$$f_K^*(x_i) \to \mathbb{E}_{D_{train}}[y], \tag{3}$$
$$Bias^2(f_K^*) \to \mathbb{E}_{train}[(y_i - \mathbb{E}_{train}[y])^2], \tag{4}$$
$$Var(f_K^*) = \mathbb{E}_{train}[(f_K^*(x_i) - \mathbb{E}_{train}[f_K^*(x_i)])^2] \to 0, \tag{5}$$
$Var(f_K^*)$ *est décroissante pour K assez grand.*

Ce théorème signifie deux choses : 1) dans le cas de *Mixup* la fonction f^* qui minimise le coût a une variance nulle et converge vers $\mathbb{E}_{train}[y]$. 2). Pour K assez grand la variance de la fonction qui minimise le coût *Local Mixup* est décroissante, ce qui prouve que *Local Mixup* peut bien contrôler le compromis biais/variance.

Cas avec sorties i.i.d

On considère que le jeu d'entraînement est composé de l'ensemble $\{x \mid \exists y, (x, y) \in \mathcal{D}_{train}\} = \mathbb{Z}$ et y_i i.i.d. selon une loi de probabilité R et de variance σ^2.

Théorème 4.2. *Pour un signal avec des sorties i.i.d, la variance est bornée par :*

$$\frac{4^2\sigma^2}{K^2} \leq Var(f_K(x_i)) \leq \frac{8\sigma^2}{K}. \tag{6}$$

4.2.2 Invariance des modèles linéaires

En considérant un modèle linaire, on peut montrer que *Mixup* et *Local Mixup* conduisent à la même solution.

Théorème 4.3. *Pour un modèle linéaire : $f(x) = ax+b$, $a, b \in \mathbb{R}$, la fonction f^* qui minimise la coût de* Mixup *ou* Local Mixup *est la même.*

4.3 Grande dimension et constante de Lipschitz

Les preuves en faible dimension sont limitées : l'effet de moyenne apparaît car tous les points x dans l'intervalle $[x_1, x_n]$ peuvent s'écrire comme la combinaison convexe d'au moins une paire. Les contradictions illustrées auparavant apparaissent lorsque plusieurs combinaisons peuvent correspondre au même x. En grande dimension, la probabilité de cet évènement est faible. Toutefois, plus on autorise des interpolations plus le jeu d'entraînement est enrichi d'exemples virtuels, ce qui peut impacter la marge entre les classes et impacter le compromis biais/variance comme nous le démontrons dans [1].

5 Expériences

5.1 Faible dimension

Comme énoncé dans l'introduction et [7], *Mixup* conduit à des interpolations pouvant être contradictoires pour le réseau. Pour illustrer ce phénomène, on considère un jeu d'entraînement 2d composé de deux spirales enroulées où de telles interpolations sont courantes. Pour cette expérience on utilise un graphe seuillé avec pour seuil ε. Dans [1] on justifie l'utilisation de ce graphe en particulier. Pour de faibles valeurs de ε beaucoup de poids sont mis à zéro et correspondent à des interpolations qui ne sont pas prises en compte pour le calcul du coût.

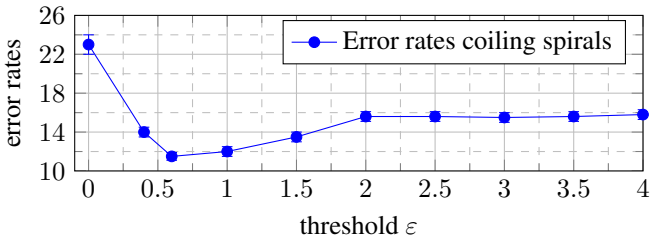

FIGURE 3 – Taux d'erreur (moyenné sur 1000 runs) en fonction de ε pour les deux spirales enroulées. Les extrêmes correspondent à un réseau sans *Mixup* ($\varepsilon = 0$) et à *Mixup* ($\varepsilon > 4$)

Sur la figure 3, on remarque une réduction importante de l'erreur pour *Mixup* et *Local Mixup* par rapport à un modèle

TABLE 1 – Taux d'erreur(%) sur CIFAR-10, Cifar-100, Fashion-MNIST et SVHN. Les valeurs sont moyennées sur 100 runs pour CIFAR-10, 10 runs pour fashion-MNIST et SVHN. L'erreur moyenne et les intervalles de confiance sont donnés.

METHOD	CIFAR-10 / Resnet18	CIFAR-100 / Resnet18	Fashion-MNIST / Densenet	SVHN / LeNet
Baseline	4.98 ± 0.03	30.6 ± 0.27	6.20 ± 0.2	10.01 ± 0.15
Mixup	4.13 ± 0.03	29.23 ± 0.4	6.36 ± 0.16	8.31 ± 0.14
Local Mixup	4.03 ± 0.03	29.08 ± 0.34	5.97 ± 0.2	8.20 ± 0.13

sans mixage. Comme attendu, *Local Mixup* obtient une erreur bien plus faible que *Mixup*.

5.2 Grande dimension et classification

Pour montrer l'avantage de notre méthode sur sur des données réelles plus complexes nous avons réalisé des expériences sur plusieurs problèmes de classification standard (voir table 1). Pour ces expériences on utilise un graphe avec une exponentielle décroissante paramétrée par α. On note que *Local Mixup* présente toujours l'erreur la plus faible.

5.3 Comparaison avec *Adamixup*

On compare également notre méthode avec *Adamixup*, une autre méthode capable d'éviter l'intrusion de variétés. Contrairement à *Adamixup*, notre méthode *Local Mixup* n'élimine pas forcément les interpolations causant des intrusions de variétés mais les pondère faiblement. Sur CIFAR-10, notre méthode dépasse *Adamixup* : les erreurs sont de $4.11\% \pm 0.12$ pour *Adamixup* et $3.89\% \pm 0.12$ pour *Local Mixup* (moyenne sur 5 tests). On note aussi que notre méthode converge plus rapidement sur MNIST [1]. Un autre avantage et que notre méthode est simple et comporte un nombre de paramètres bien moins important. Sur (CIFAR-10) 836 milliers de paramètres pour *Local Mixup* contre 1117 pour *Adamixup*.

6 Conclusion

Dans cet article, nous avons introduit *Local Mixup*, une méthode simple pour inclure la notion de localité dans Mixup : chaque exemple interpolé est pondéré par la distance du segment d'interpolation. Cette méthode comporte un hyperparamètre qui permet d'avoir un ensemble continu de solutions entre *Mixup* et un modèle sans *Mixup*. Avec des hypothèses simplificatrices on montre que *Local Mixup* peut contrôler le compromis biais/variance d'un modèle entraîné. De manière plus générale, on montre que *Local Mixup* peut influer sur la constante de Lipschitz d'un modèle. Sur des données réelles on montre que notre méthode *Local Mixup* conduit à une meilleure généralisation que celle apportée par *Mixup* ou par une architecture sans mixage.

Dans de futurs travaux, on pourrait chercher de nouvelles métriques et une construction de graphe plus pertinente, ainsi qu'une méthode déterministe pour sélectionner notre hyperparamètre, par exemple par l'utilisation de mesures topologiques (histogramme, diagramme de persistance).

Références

[1] Raphael BAENA, Lucas DRUMETZ et Vincent GRIPON. *Preventing Manifold Intrusion with Locality : Local Mixup*. `https://arxiv.org/abs/2201.04368`. 2022. arXiv : `2201.04368 [cs.LG]`.

[2] John CHEN, Samarth SINHA et Anastasios KYRILLIDIS. "StackMix : A complementary Mix algorithm". In : *arXiv preprint arXiv :2011.12618* (2020).

[3] Hsin-Ping CHOU et al. "Remix : Rebalanced mixup". In : *European Conference on Computer Vision*. Springer.

[4] Terrance DEVRIES et Graham W TAYLOR. "Improved regularization of convolutional neural networks with cutout". In : *arXiv preprint arXiv :1708.04552* (2017).

[5] Guneet S DHILLON et al. "A baseline for few-shot image classification". In : *arXiv preprint arXiv :1909.02729* (2019).

[6] Ian GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE. *Deep learning*. MIT press, 2016.

[7] Hongyu GUO, Yongyi MAO et Richong ZHANG. "Mixup as locally linear out-of-manifold regularization". In : *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[8] Dan HENDRYCKS et al. "Augmix : A simple data processing method to improve robustness and uncertainty". In : *arXiv preprint arXiv :1912.02781* (2019).

[9] Jang-Hyun KIM, Wonho CHOO et Hyun Oh SONG. "Puzzle mix : Exploiting saliency and local statistics for optimal mixup". In : *International Conference on Machine Learning*. PMLR. 2020.

[10] Zicheng LIU et al. "AutoMix : Unveiling the Power of Mixup". In : *arXiv preprint arXiv :2103.13027* (2021).

[11] Puneet MANGLA et al. "Charting the right manifold : Manifold mixup for few-shot learning". In : *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020.

[12] Alexandre RAME, Remy SUN et Matthieu CORD. "MixMo : Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks". In : *arXiv preprint arXiv :2103.06132* (2021).

[13] Vikas VERMA et al. "Manifold mixup : Better representations by interpolating hidden states". In : *International Conference on Machine Learning*. PMLR. 2019.

[14] Wenpeng YIN et al. "BATCHMIXUP : Improving Training by Interpolating Hidden States of the Entire Mini-batch". In : ().

[15] Sangdoo YUN et al. "Cutmix : Regularization strategy to train strong classifiers with localizable features". In : *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

[16] Hongyi ZHANG et al. "mixup : Beyond empirical risk minimization". In : *arXiv preprint arXiv :1710.09412* (2017).