Sélection optimale de transformations pour l'augmentation de donnée

par

Antoine HARLÉ

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC PROJET EN GÉNIE, CONCENTRATION GÉNIE DES SYSTÈMES
M. Ing.

MONTRÉAL, LE "18 MARS 2020"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

REMERCIEMENTS

Je tiens tout d'abord à remercier monsieur Marco Pedersoli, mon directeur de projet et enseignant-chercheur à l'ÉTS, pour m'avoir offert l'opportunité de réaliser ce projet de recherche et m'avoir prodiguer de précieux conseils tout au long de mon stage.

Je tiens aussi à remercier monsieur Bruno Ménard, directeur logiciel à Teledyne DALSA, et monsieur Stéphane Dalton, directeur technique à Teledyne DALSA, pour avoir lancé ce projet, mis tous les moyens à ma disposition pour le bon déroulement de mon projet et construire un cadre de travail accueillant. Je tiens aussi à montrer ma reconnaissance au reste de l'équipe Axon de Teledyne DALSA, monsieur Masih Aminbeidokhti et monsieur Guillaume Lagrange, pour leurs conseils et nos discussions enthousiasmantes.

Je veux exprimer ma gratitude envers monsieur Éric Granger, enseignant-chercheur à l'ÉTS, pour avoir accepté d'être mon co-directeur et aider à conclure ce projet.

Je souhaite aussi remercier l'équipe pédagogique de Polytech Sorbonne qui m'a encadrée durant ce projet, monsieur Wael Bachta et monsieur Frédéric Plumet, pour leurs soutiens et disponibilités.

Enfin, je tiens aussi à exprimer ma gratitude à Mitacs accélération qui a financé avec Teledyne DALSA se projet.

Sélection optimale de transformations pour l'augmentation de donnée

Antoine HARLÉ

RÉSUMÉ

L'augmentation de donnée a démontré une grande capacité à améliorer la généralisation des connaissances des modèles en apprentissage profond. Devenue indispensable pour tirer les meilleures performances de ces derniers, elle nécessite en revanche des moyens humains et/ou computationnels non négligeables. Ce projet vise à réduire, voir supprimer, ces contraintes en réalisant l'apprentissage de l'augmentation de donnée optimale conjointement a celle du modèle. Le projet s'est limité à la tache de classification pour la vision par ordinateur.
La solution retenue, après une analyse de la littérature, est inspirée des méthodes constituant l'état de l'art dans les domaines de l'augmentation de donnée (AutoAugment & RandAugment) et du méta-apprentissage (K-RMD). Habituellement séparé en deux taches distinctes, ce projet se base sur l'hypothèse que la *recherche des augmentations optimales et l'apprentissage d'un modèle peuvent être réalisés ensemble*.
L'implémentation de K-RMD et d'une dizaine de transformations différentiables a permis de montrer qu'il était possible, sans expertise particulière ni décuplement de capacités de calcul, d'atteindre ou surpasser l'état de l'art en augmentation de donnée.
Les expériences réalisées ont aussi permis de mettre en évidence la difficulté de la recherche de transformations optimales de manière locale, ainsi que deux nouvelles théories qu'il conviendrait d'explorer dans de futurs travaux. La première est que l'*apprentissage de l'augmentation de donnée tend à réduire sa diversité* et, dans certains cas, limite en conséquence les performances du modèle. La seconde est que l'*augmentation de donnée rehausse la capacité requise par un réseau pour modéliser des distributions de données* ; donnant une nouvelle perspective sur l'augmentation de donnée pour des réseaux à fortes contraintes de capacité.

Mots-clés: Vision par ordinateur, Apprentissage machine, Augmentation de donnée

Optimal Selection of Data Augmentation Transformations

Antoine HARLÉ

ABSTRACT

Data augmentation has demonstrated a great ability to improve the generalization of knowledge of deep learning models. Essential to get the best performance from them, it does, however, require significant human and/or computational resources. This project aims to reduce, or remove, these constraints by performing jointly the learning of optimal data augmentation and model weights.
The project was limited to the classification task for computer vision.
The solution retained, after an analysis of the literature, is inspired by state-of-the-art methods in the fields of data augmentation (AutoAugment & RandAugment) and meta-learning (K-RMD). Usually separated into two distinct tasks, this project is based on the assumption that *looking for optimal augmentations and learning model weights can be done together*.
The implementation of K-RMD and a dozen differentiable transformations allowed to show that it was possible, without particular expertise or a great increase in computing capacity, to reach or surpass state-of-the-art data augmentation methods.
The experiments carried out also highlighted the difficulty of the search for optimal transformations locally, as well as two new theories which should be explored in future work. The first is that *learning data augmentation tends to reduce its diversity* and, in some cases, limits the performance of the model accordingly. The second is that the *data augmentation increases the capacity required by a network to model data distribution*; giving a new perspective on data augmentation applied on networks with high capacity constraints.

Keywords: Computer vision, Machine Learning, Data augmentation

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CNN	Réseau de neurones convolutif (Convolutional Neural Network)
DA	Data Augmentation (Augmentation de donnée)
DrMAD	Distilling Reverse-Mode Automatic Differentiation - Fu, Luo, Feng, Low & Chua
ÉTS	École de Technologie Supérieure
GPU	Graphics Processing Unit (Processeur graphique)
HG	Hyper-gradient (Gradient sur les hyper paramètres)
HP	Hyper-parameter (hyper paramètre)
PBA	Population Based Augmentation (Augmentation basée sur les populations) - Ho, Liang, Stoica, Abbeel & Chen
RL	Reinforcement Learning (Apprentissage par Renforcement)
RMD	Reverse-mode differentiation (différentiation en mode inverse) - Maclaurin, Duvenaud & Adams
TF	Transformations (ou Augmentations)

LISTE DES SYMBOLES ET UNITÉS DE MESURE

α	Facteur de mélange de distributions. Voir section 2.1.1.2.
A	Module d'augmentation de données. Voir section 2.1.2.
$\mathcal{D}$	Distribution des transformations. Voir section 2.1.1.2.
$\mathcal{D}_\alpha$	Distribution des transformations adoucie par un facteur α. Voir section 2.1.1.2.
ϵ	Scalaire proche de 0. Voir section 2.1.1.2.
λ	Facteur de pondération des pertes non supervisées. Voir section 2.1.2.2.
$\mathcal{L}$	Perte ou fonction de coût. Voir section 2.1.2.2.
m	Amplitude de distorsion d'une transformation. Voir section 2.1.1.1.
p	Probabilité d'application d'une transformation. Voir section 2.1.1.1.
θ	Ensemble des paramètres des transformations. Voir section 2.1.2.
$\mathcal{T}$	Ensemble des transformations. Voir section 2.1.1.1.
$\mathcal{T}_{no\_grad}$	Ensemble des transformations non différentiables par rapport à l'amplitude. Voir section 2.1.2.
t	Transformation (Fonction). Voir section 2.1.1.1.
W	Ensemble des paramètres, poids, du modèle. Voir section 2.1.2.
X_{tr}	Ensemble des données d'entraînements. Voir section 2.1.2.
X_{val}	Ensemble des données d'entraînements. Voir section 2.1.2.

INTRODUCTION

Dans le domaine de l'apprentissage machine (machine learning) appliquée à la vision, un problème récurrent est le manque de donnée disponible pour l'entraînement des systèmes. En effet, il est en général nécessaire de fournir, au moins, plusieurs milliers d'images exemples a un système pour obtenir des résultats satisfaisants.

Une solution répandue a ce problème est l'augmentation de donnée (DA). Elle consiste à générer artificiellement de nouveaux exemples à partir des données d'origine, en leur appliquant diverses transformations. Il est alors possible de démultiplier la taille d'une base de données et par conséquent d'entraîner des modèles plus performants.

La DA comporte cependant plusieurs limitations. La principale est que les exemples générés sont de moindres qualités que des exemples réels de la tâche à accomplir. D'autant plus que toutes les transformations ne sont pas appropriées. La sélection de bonnes transformations requiert une connaissance à priori des données, pas toujours disponibles, ainsi que du temps d'ingénieurs.

Ainsi, l'objectif de cette recherche est de fournir un système de DA, capable de fournir des résultats similaires ou supérieurs à des augmentations sélectionnées par l'homme. La solution doit demander des ressources (computationnelles et temporelles) comparables à celles requises par le modèle de base pour son apprentissage. Tout en diminuant ou retirant le besoin de connaissance a priori de la tache.

Dans un premier temps, une revue de la littérature sera réalisée pour concrétiser le problème technique ainsi que les solutions envisageables. Ensuite, la méthodologie employée durant cette recherche sera expliquée. Suivie de la présentation des résultats de ce travail et d'une conclusion.

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Problématique

L'augmentation de donnée (DA) consiste a générer artificiellement des données afin d'agrandir, d'augmenter, une base de données. On cherche à générer des exemples appartenant à la même distribution que la base de données réelle sans connaître l'expression de cette distribution. La difficulté est alors de sélectionner les transformations qui apportent un maximum de variation, d'information, tout en produisant un exemple de la distribution de départ (voir Rotation 30°, figure 1.1).
Cette sélection demande en général des connaissances spécifiques aux données et du temps d'ingénieur, car toute transformation n'est pas appropriée à toutes données. Par exemple, une translation de 30 pixels d'une image de 32x32 pixels fera perdre la majeure partie de l'information, mais serait sans doute bénéfique pour une image de 256x256 pixels (voir Translation 30 pixels, figure 1.1). Tandis qu'une rotation trop importante d'un 6, exemple de la distribution des 6, créerait un exemple de 9 appartenant à une distribution différente et toujours supposé être un exemple de 6 (voir Rotation 180°, figure 1.1). Ce genre de mauvaises transformations peut corrompre la base de données et être néfaste a l'apprentissage d'un modèle performant.

Il n'est pas toujours évident de dire quelles sont les bonnes transformations pour une distribution de données. C'est pourquoi les domaines du méta-apprentissage, et de l'augmentation de donnée automatique sont très actifs. Ainsi, l'objectif de ce projet est de se passer de l'expertise humaine pour la sélection des transformations appropriée tout en conservant ou améliorant les performances des modèles ; précédemment entraîné en profitant de cette expertise.

1.2 Solutions

Il est important de noter que l'augmentation de donnée sert, la plupart du temps, à augmenter la capacité de **généralisation** d'un modèle. En effet, un problème récurrent en apprentissage

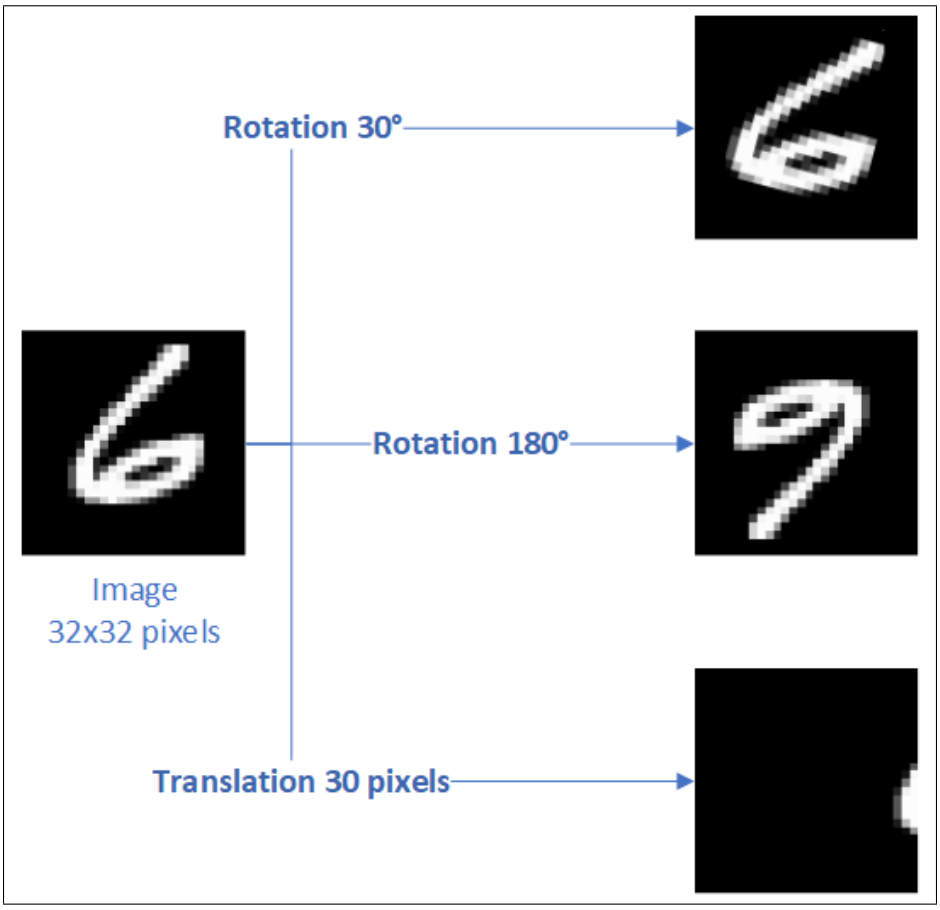

Figure 1.1 Différentes transformations d'un chiffre

machine est qu'un modèle peut-être extrêmement performant sur des données d'*entraînement*, il n'en sera pas pour autant aussi performant en utilisation réelle, bien au contraire ! Ce problème est, en général, synonyme de *surapprentissage*.

Pour limiter ces mauvaises surprises, il est commun de diviser les bases de données en *données d'entraînements* et en *données de tests* (voir figure 1.2). Cette dernière base de données a pour but de simuler une utilisation réelle, ne doit pas être utilisée par l'algorithme d'apprentissage et permet ainsi d'évaluer un modèle avec le minimum de biais.

De plus, pour limiter le surapprentissage, on aura tendance à séparer une partie des données d'entraînement pour produire une base de *données de validations*. Cette dernière sert à avoir une estimation de la capacité de généralisation d'un modèle, ce dernier n'ayant pas directement appris sur ces données. Ainsi, en méta-apprentissage, il est fréquent d'utiliser l'erreur sur la

donnée de validations comme métriques à diminuer.

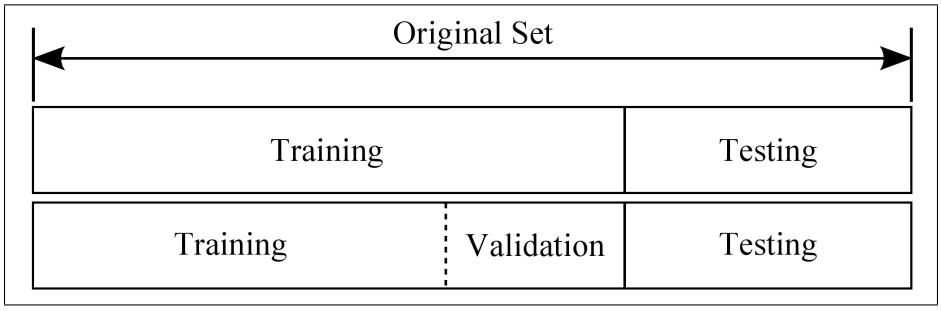

Figure 1.2 Division des bases de données

Les sections suivantes présentent les différentes solutions actuelles pour apprendre réaliser de l'augmentation de données automatique.

1.2.1 Augmentation de donnée automatique

1.2.1.1 AutoAugment

AutoAugment -Cubuk, Zoph, Mane, Vasudevan & Le (a)- est une technique populaire d'augmentation de donnée qui consiste à apprendre les meilleures séquences de transformations à appliquer aux données. Les transformations sont discrètes et paramétrées par leurs amplitudes et probabilités d'application.

L'approche d'AutoAugment consiste à rechercher les meilleurs couples de TF à appliquer aléatoirement aux données pour diminuer l'erreur de validation. Ces ensembles de couples de TF constituent des politiques que AutoAugment recherche avec un algorithme d'apprentissage par renforcement (RL).

Une des limitations majeures de cette approche est son coût computationnel. En effet, l'approche par RL nécessite de réaliser de nombreux entraînements du même modèles afin d'arriver à une politique optimale. Un entraînement complet doit être réalisé pour chaque ensemble de TF testé ! En pratique, pour rendre cette méthode utilisable, AutoAugment travaille sur un problème

simplifié, un modèle moins complexe avec un sous-ensemble des données d'entraînement, plus rapide à converger. Cubuk *et al.* (a) font l'hypothèse que la solution optimale obtenue sur ce problème simplifié se généralise au problème réel. La solution obtenue serait proche de l'optimale pour le modèle souhaité sur l'ensemble des données.

Malgré la simplification du problème, AutoAugment reste extrêmement coûteux, une recherche se mesurant en millier d'heures de GPU (voir tableau 1.1) ! Pour pallier ce problème, plusieurs améliorations d'AutoAugment ont été proposées.

Fast AutoAugment -Lim, Kim, Kim, Kim & Kim- accélère le processus en conditionnant l'espace de recherche de l'algorithme. Lim *et al.* cherche une politique maximisant la précision d'un modèle, mais aussi, cherche à faire correspondre les prédictions du modèle sur les données d'entraînement et les données transformées. Ils s'inspirent aussi d'une optimisation bayésienne, utilisée pour la recherche d'hyper paramètres optimaux, pour réduire le coût de la procédure.

Population Based Augmentation (PBA) -Ho *et al.*-, plutôt que d'apprendre une politique fixe, propose d'apprendre un planning de TF, une politique évoluant en fonction des stades de l'entraînement du modèle. Il est intéressant de noter que *Ho & al.* s'inspirent, quant a eux, d'algorithmes évolutionnaires, aussi utilisés pour la recherche d'hyper paramètres.

Tableau 1.1 Comparaison des coûts estimés, en heures GPU, de recherche des politiques optimales. GPUs - AutoAugment : Tesla P100, Fast AutoAugment : Tesla V100, PBA : Titan XP

Données	**AutoAugment**	**Fast AutoAugment**	**PBA**
CIFAR-10	5000	3.5	5
SVHN	1000	1.5	1
ImageNet	15000	450	-

Malgré les améliorations apportées par Fast AutoAugment et PBA, la recherche des TF/politiques optimales reste relativement coûteuse (voir tableau 1.1) par rapport à l'entraînement classique d'un modèle.

1.2.1.2 RandAugment

Récemment, RandAugment -Cubuk, Zoph, Shlens & Le (b)- a observé que l'hypothèse d'AutoAugment sur le problème simplifié n'était pas valide. En effet, ils ont noté la politique optimale obtenue sur un problème simplifié serait sous-optimale une fois transférée a un problème plus complexe. La solution optimale étant dépendante de la taille du modèle et la quantité de données d'entraînement.
De plus, Cubuk *et al.* (b) ont obtenu des performances similaires à AutoAugment sans recherche de politiques particulières. La sélection des TF est réalisée de manière aléatoire, sélectionnée depuis une distribution uniforme. Les seuls paramètres constituant l'espace de recherche sont le nombre de transformations (N) appliquées à chaque échantillon et une amplitude de distorsion (M) commune a toutes les transformations. Ces paramètres pouvant être adaptés à chaque problème avec une recherche par quadrillage, il ne serait pas utile de réaliser une recherche coûteuse de politiques.

1.2.2 Apprentissage d'hyper paramètres

La DA automatique peut être vue comme une recherche des meilleurs hyper paramètres (HP) définissant les TF appliqués aux données.
Cette section présente les méthodes basés sur pour l'apprentissage des hyper paramètres d'un modèle.

1.2.2.1 Méthodes boîtes noires

Les approches classiques pour la recherche d'HP consistent à considérer les modèles comme des boîtes noires et ne considèrent que les HP et l'erreur de validation sans prendre en considération la structure interne du problème.
Parmi les plus simples, on peut citer les recherches par quadrillage, les recherches par arbres ou les recherches aléatoires -Bergstra, Bardenet, Kégl & Bengio-Feurer & Hutter-. Certaines approches visent, quant à elles, à modéliser la réponse, la performance par rapport à une métrique, en fonction des HP et des données. L'optimisation bayésienne -Snoek, Larochelle & Adams- en est l'exemple le plus connu. On peut aussi citer les méthodes évolutionnaires, basées sur des populations -Jaderberg, Dalibard, Osindero, Czarnecki, Donahue, Razavi, Vinyals, Green, Dunning, Simonyan, Fernando & Kavukcuoglu-, comme PBA -Ho *et al.*- qui sélectionnent les HP durant l'entraînement de multiples modèles.
Dans une autre perspective, Finn, Abbeel & Levine proposent avec *MAML* (Model-Agnostic Meta-Learning) d'apprendre des HP en réalisant de multiples mini-apprentissages (entraînement avec quelques échantillons d'apprentissages). Ils recherchent avec MAML la meilleure initialisation des poids d'un modèle pour minimiser l'erreur de validation.
Ces méthodes sont relativement coûteuses, car elles requièrent de répéter l'entraînement d'un modèle de multiples fois pour converger. Ce coût devenant prohibitif plus le nombre d'HP est grand, l'espace de recherche s'étendant en conséquence.

1.2.2.2 Optimisation à deux niveaux

Une approche différente consiste à considérer l'optimisation des HP d'une manière similaire à l'optimisation des paramètres du modèle (poids du réseau). De même que la recherche des poids optimaux est réalisée par descente de gradient, il est possible de rechercher les HP optimaux par descente de gradient -Feurer & Hutter-. La difficulté principale étant de calculer le gradient sur les HP, ou hyper gradient (HG). Il est important de noter que les HP à optimiser doivent être continus afin d'être différentiable.

Ce problème est communément formulé comme une optimisation à deux niveaux. L'optimisation des poids du modèle w constituant le *problème interne*, ou *objectif bas niveau* g_S (équation 1.2), tandis que l'optimisation des HP λ constitue le *problème externe*, ou *objectif haut niveau* F (attendue) et f_S (échantillonnée) (équation 1.1). Avec S une variable aléatoire constituant le contexte. Voir -Shaban, Cheng, Hatch & Boots- pour plus de détails.

$$\min_{\lambda} F(\lambda) := E_S[f_S(w_S^*(\lambda), \lambda)] \tag{1.1}$$

$$w_S^*(\lambda) \approx_{\lambda} \arg\min_{w} g_S(w, \lambda) \tag{1.2}$$

La résolution de ce problème n'est pas triviale du fait de la dépendance du problème externe aux HP, les poids du modèle étant aussi dépendants des HP. En pratique, cela suppose que pour connaître l'effet réel des HP sur le problème interne doit être résolu, le modèle doit avoir fini de converger. Formellement, la relation entre les poids finaux du modèle et les HP est appelée fonction de meilleure réponse (Best-response function). L'estimation de cette fonction devient plus complexe avec l'augmentation du nombre d'HP et de poids.

Maclaurin *et al.* propose de calculer de manière *exacte* l'HG en propageant le calcul des gradients à travers *toute* la procédure d'entraînement. Ils procèdent à une différentiation en mode inverse (RMD). Connaissant la dynamique de l'algorithme d'entraînement, ils inversent cette dynamique lors du calcul des gradients afin de connaître l'influence exacte des HP à chaque pas d'optimisation.

Une limitation majeure de la méthode de Maclaurin *et al.* est son coût computationnel et en mémoire. Cette approche nécessitant de réaliser l'inverse de la procédure d'entraînements et de garder en mémoire les poids du modèle *autant de fois que le nombre d'itérations réalisé*.
Franceschi, Donini, Frasconi & Pontil propose quant à eux de réaliser le calcul de l'HG en parallèle du processus d'optimisation de l'objectif bas niveau. Cependant, cette procédure de différentiation directe (FMD) reste relativement coûteuse en temps et en mémoire.

Similairement, Baydin, Cornish, Rubio, Schmidt & Wood et Chandra, Meijer, Andow, Arroyo-Fang, Dea, George, Grueter, Hosmer, Stumpos, Tempest & Yang calculent des HG conjointement au processus d'optimisation, mais ces derniers s'appuient, non pas sur les données de validation, mais, sur les données d'entraînements afin de réutiliser les gradients requis lors de l'optimisation de bas niveaux.

Dans le but de réduire la quantité de ressources nécessaires, Luketina, Berglund, Greff & Raiko et Shaban *et al.* montrent qu'il est possible d'obtenir des résultats comparables avec bien moins de ressources. En limitant le nombre d'itérations considérées, il est possible d'obtenir une approximation de l'HG.

Shaban *et al.* propose ainsi une méthode tronquée sur K itération (K-RMD) qui peut être vue comme une approximation temporelle de l'HG. Elle permet ainsi d'optimiser l'objectif haut niveau avec un coût computationnel et en mémoire comparable aux coûts de résolution du problème interne. Cette approche ne nécessitant plus que de réaliser l'inverse de la procédure d'entraînements et de garder en mémoire les poids du modèle *K fois*.

Fu *et al.* proposent, quant à eux, un autre type d'approximation de l'HG. En réalisant une approximation sur la trajectoire suivie pour la résolution du problème interne, DrMAD -Fu *et al.*- calcule une approximation de la direction de l'HG. La RMD étant calculée non pas sur la trajectoire exacte, mais sur une trajectoire raccourcie, définie par l'état de départ et de fin (voir figure 1.3).

En revanche, malgré les promesses de cette méthode, le manque de résultat d'expériences laisse incertain quant à sa fiabilité -Jules-.

D'autres approches telles que Pedregosa ou MacKay, Vicol, Lorraine, Duvenaud & Grosse cherchent aussi à calculer l'HG approximativement.
MacKay *et al.* proposent, par exemple, de modéliser directement la fonction de meilleure réponse

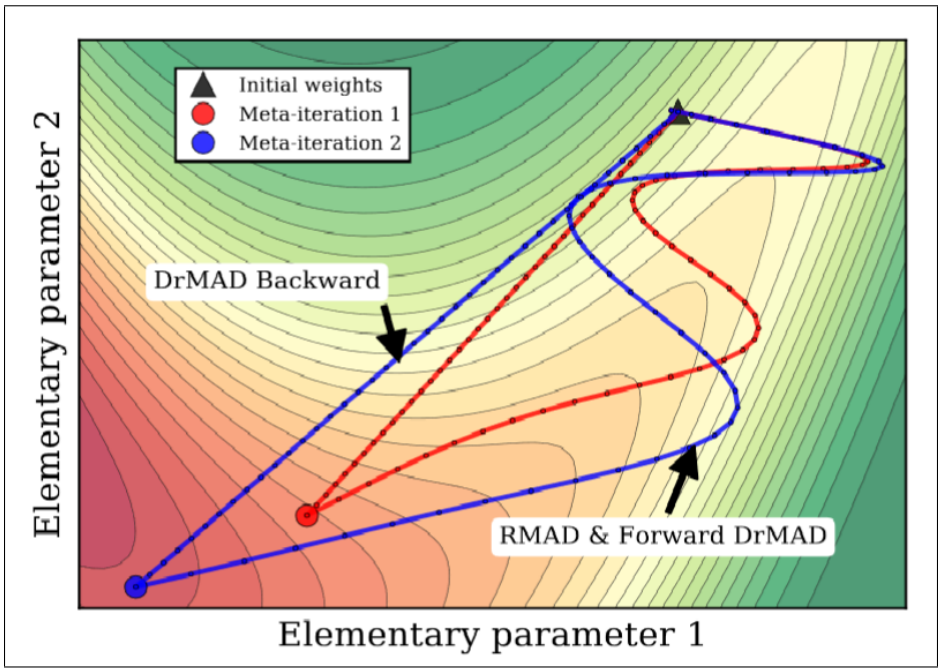

Figure 1.3 Comparaison des 'chemins' utilisée pour l'optimisation des hyper paramètres par DrMAD et RMD (RMAD).

avec un réseau à part. Cela leur permet de ne pas propager le gradient à travers le modèle de base pour obtenir un HG. Le problème externe est alors résolu par descente de gradient sur la fonction de meilleure réponse modélisé. Une des limitations de cette approche est que la modélisation de la fonction de meilleure réponse n'est pas triviale et que l'entraînement du modèle de cette fonction est réalisé au cours de la recherche des poids du modèle de base. Par conséquent, l'entraînement global peut facilement être instable.

1.2.3 Méthodes à base de GAN

Les réseaux génératifs adverses (GAN) sont des modèles capables de générer des échantillons réalistes d'une distribution de données. En conséquence, il est possible de les utiliser pour réaliser de l'augmentation de donnée.

Mirza & Osindero sont parmi les premiers à proposer des GAN conditionnés utilisables pour la DA. En effet, les GAN classiques peuvent composer des images vraisemblables, mais ne donnent aucun contrôle sur les types d'échantillons produit. Sans indications sur leurs classes, il n'est pas possible de les utiliser pour de l'apprentissage supervisé. En conditionnant la génération d'images de manière à produire des échantillons de classes spécifiques, Mirza & Osindero rendent possible d'étendre la base de données d'entraînement.
Directement pour la DA, Antoniou, Storkey & Edwards propose avec DAGAN (Data Augmentation Generative Adversarial Networks) de conditionner le GAN avec des images de la classe à générer. Dans cette configuration, on peut voir l'apprentissage du GAN comme une recherche de TF génériques. Dans une extension de ce principe, Lemley, Bazrafkan & Corcoran et Perez & Wang mettent en avant la possibilité de conditionner la génération avec plusieurs échantillons de la même classe.
Dans une idée similaire, Li, Xu, Zhu & Zhang et Tran, Pham, Carneiro, Palmer & Reid réalisent, quant à eux, l'entraînement d'un modèle (Classificateur) et du GAN réalisant la DA conjointement. Zhang, Wang, Liu & Ling opte pour une approche dans la même direction, avec DADA (Deep Adversarial Data Augmentation), en fusionnant les rôles de discriminateur (nécessaire aux GAN) et de classificateur.
Plutôt que d'apprendre des TF génériques, Ratner, Ehrenberg, Hussain, Dunnmon & Ré propose, avec TANDA (Transformation Adversarial Networks for Data Augmentations), de se baser sur des TF définis par l'utilisateur. L'objectif de TANDA est de produire des TF complexes en générant des séquences de TF (compositions des TF définies par l'utilisateur).

Bien que ces méthodes soient prometteuses, en pratique, elles font face à plusieurs limitations non négligeables.
En effet, contrairement a la majorité des méthodes des sections 1.2.1 et 1.2.2 se basant directement sur les performances du modèle, les approches à base de GAN cherchent à simuler la distribution de donnée d'entraînement puis à générer des images tirées de cette distribution. Ce choix peut finir par être limitant dans le cas où les données d'entraînement ne représentent pas toute la

distribution réelle (ou, en pratique, la distribution de validation/test) et ne pas améliorer la généralisation du modèle.

De plus, les GAN font face à d'importants soucis de stabilité qui les rendent particulièrement complexes à entraîner -Kodali, Abernethy, Hays & Kira-, le cas échéant. Cela induit un manque réel de fiabilité pour une méthode qui veut supporter l'entraînement d'un modèle probablement déjà complexe.

CHAPITRE 2

MÉTHODOLOGIE

2.1 Apprentissage de l'augmentation de donnée

Les prochaines sections présentent le module d'augmentation de donnée ainsi que les détails sur le processus d'optimisation en ligne de ce dernier.

2.1.1 Augmentation de donnée

Dans le but de minimiser les coûts supplémentaires dus à l'apprentissage de la DA, ce dernier est réalisé conjointement à l'apprentissage des poids du modèle. Pour ce faire, un module de DA a été créé avec pour tâche de fournir des versions augmentées des images qui lui sont fournies. Ce module est placé en amont du modèle dans la chaîne de traitement des données afin de fournir constamment de "nouvelles" données et permettre la propagation de l'HG (voir figure 2.1).

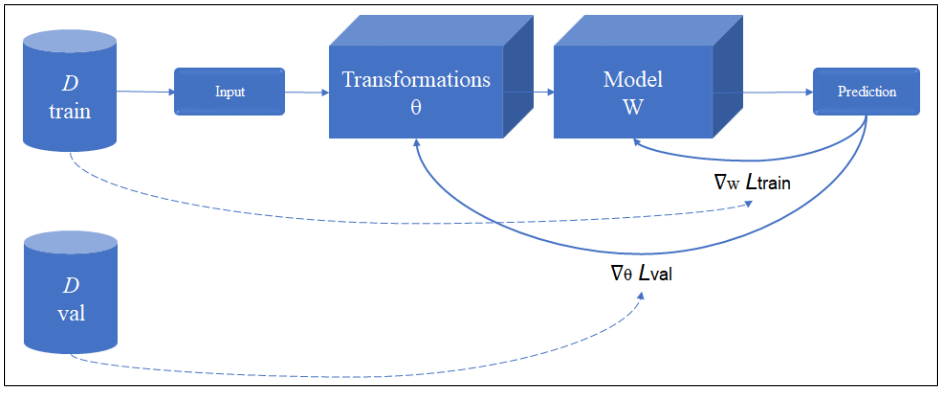

Figure 2.1 Vue d'ensemble du problème de DA joint.

Le module de DA peut être découpé en trois étapes appliquées à chaque image fournie :

1. Sélection des TF à appliquer.
2. Sélection des amplitudes de TF.

3. Application des TF.

Cette chaîne peut-être répétée plusieurs fois pour une même image, dans le cas de transformations séquentielles à une même image. Les TF séquentielles peuvent aussi être vues comme l'a application d'une seule TF complexe résultant de la combinaison de plusieurs TF simples (voir figure 2.2).

Tout le traitement est réalisé par lots de données afin de prendre avantage des capacités de traitement sur GPU.

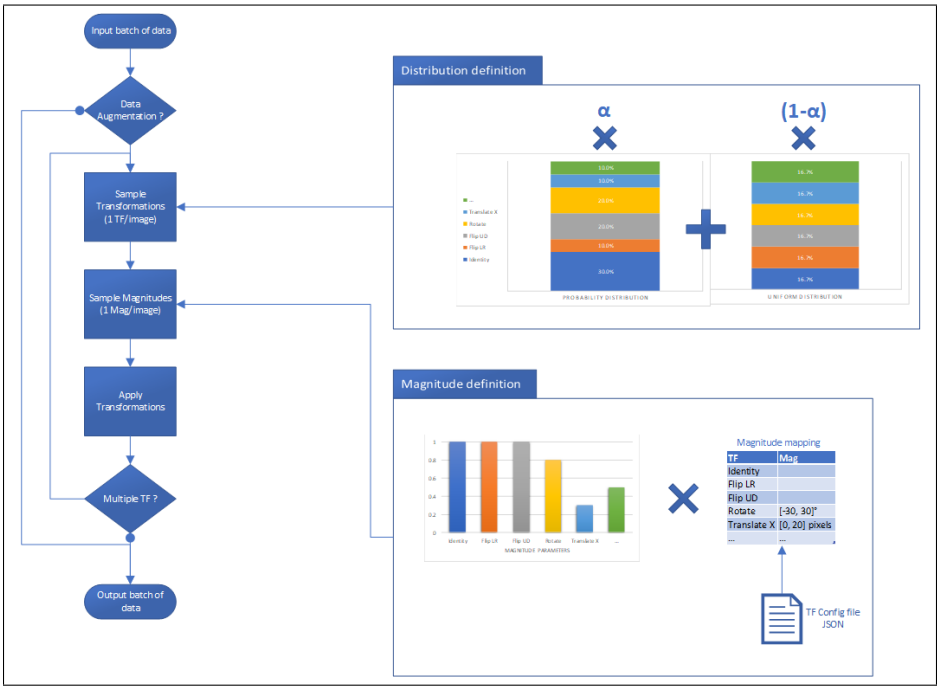

Figure 2.2 Traitement des lots d'images par le module de DA

2.1.1.1 Définition des transformations

D'une manière similaire à AutoAugment -Cubuk *et al.* (a)-, les TF utilisées par le module de DA peuvent être définies par des triplets :

- Transformation (Fonction à appliquer) *t*.
- Probabilité d'application *p*.

- Amplitude de distorsion *m*.

Ce choix de formalisation est relativement générique, permettant une implémentation flexible, et facilite, a posteriori, l'évaluation du système développée (voir Chapitre 3).
Concrètement, un des objectifs principaux de ce projet est d'apprendre les paramètres optimaux de chaque TF, c'est-à-dire les triplets *(t, p, m)*. Pour ce faire, l'algorithme de méta-apprentissage retenu (voir section 2.1.2) nécessite que les TF soient différentiables par rapport à *p* et *m*.
Pour ce faire, les TF utilisées par le module de DA ont été portées en PyTorch afin de mettre à profit la fonctionnalité de différentiation automatique de cette librairie. Principalement basées sur l'utilisation de Kornia et inspirée de Pillow (voir section 2.2.3), *11* transformations ont pu être redéfinies pour constituer l'ensemble $\mathcal{T}$ des TF :

- Transformations géométriques :
  - Identité (Identity).
  - Miroir (Flip).
  - Rotation (Rotate).
  - Translation (Translate).
  - Cisaillement (Shear).
- Transformations de teintes (Ajustement des couleurs) :
  - Contraste (Contrast).
  - Coloration (Color).
  - Luminosité (Brightness).
  - Finesse (Sharpness).
  - Posteriser (Posterize).
  - Solariser (Solarize).

Ce choix de TF a été réalisé pour être le plus proche possible des augmentations, basées sur Pillow, d'AutoAugment -Cubuk *et al.* (a)-.
De plus, il est important de noter que les TF *Identité*, *Miroir*, *Posteriser* et *Solariser* (constituant

$\mathcal{T}_{no\_grad} \subseteq \mathcal{T}$) ne sont ***pas*** *différentiables* par rapport a m ; et ne permettent pas, par conséquent, l'apprentissage de ce paramètre, pour des raisons détaillées dans la section 2.1.2.4.
Il est aussi significatif de mentionner que bien qu'elle soit inspirée de Pillow, les TF, développées ici, diffèrent en deux points de leurs origines : le traitement des images est réalisé *par lot* et les images augmentées peuvent différer légèrement. Plus spécifiquement, ces différences viennent d'une interpolation différente pour les TF *Rotation* et *Cisaillement* ; ainsi que de valeurs de pixels différentes pour les TF *Contraste*, *Coloration*, *Luminosité* et *Finesse* malgré des visuels similaires (voir figure 2.3).

Figure 2.3 Exemple de différences entre les transformations de Pillow et PyTorch.

En pratique, le module de DA utilise des augmentations configurées à partir d'un fichier JSON contenant :

- 'Name' : Nom de l'augmentation.
- 'Function' : Fonction utilisée par l'augmentation parmi la liste de TF portée en PyTorch.
- 'Param' : Détails sur les paramètres de l'augmentation (si nécessaire).
  - 'Min' : Valeur minimale d'amplitude (non normalisée) de l'augmentation.
  - 'Max' : Valeur maximale d'amplitude (non normalisée) de l'augmentation.
  - 'Axis' : Axe d'application de l'augmentation.
  - 'Absolute' : Indique si l'amplitude de l'augmentation doit être absolue ou relative à la dimension des données.
  - 'InvScale' : Indique si l'échelle d'amplitude doit être inversée (voir section 2.1.1.3).

2.1.1.2 Sélection des transformations

L'ensemble des probabilités p_t définissent une distribution $\mathcal{D}$ depuis laquelle les TF à appliquer seront échantillonnés.
Pour des raisons qui seront détaillées dans la section 2.1.2.3, cette distribution est adoucie ($\mathcal{D}_\alpha$), modérée avec un facteur α par une distribution uniforme (équation 2.1). α étant compris dans $[0, 1-\epsilon]$. Sauf indication du contraire, les résultats présentés dans le chapitre 3 utilisent $\epsilon = 1e^{-3}$.

$$D_\alpha = \alpha \times \mathcal{D} + (1-\alpha) \times \mathcal{U}(0, \Omega(\mathcal{D})) \tag{2.1}$$

Dans le cas où plusieurs TF sont appliquées séquentiellement, l'ordre d'application peut être significatif. Par exemple, les combinaison *(Miroir vertical, Rotation, Miroir vertical)* et *(Miroir vertical, Miroir vertical, Rotation)*, équivalent à une simple rotation, sont différentes augmentations. Ainsi, il peut être intéressant d'apprendre cet ordre. Pour ce faire, une variation

du module de DA n'utilisant pas directement chaque TF indépendamment, mais toutes les combinaisons possibles des TF (pour une taille de séquence fixe) ont été testées.

2.1.1.3 Sélection des amplitudes

Pour des raisons de cohérence et de simplicité, toutes les TF utilisent des amplitudes m_t normalisées comprises dans [0, 1].
Pour les transformations géométriques, une amplitude proche de 1 signifiant une distorsion maximale, et vice-versa. En revanche, les transformations de teintes, étant similaires aux augmentations d'AutoAugment-Cubuk *et al.* (a)-, n'utilisent pas la même échelle. En effet, leurs héritages de Pillow impliquent que les TF Posteriser et Solariser utilisent une échelle inverse (distorsion maximale pour m_t=0 et vice-versa). Tandis que les TF Contraste, Coloration, Luminosité et Finesse pressentent une distorsion minimale pour m_t=0.5 et maximale aux bornes (m_t=0 signifiant une diminution maximale et m_t=1 une augmentation maximale).
Pour unifier l'utilisation des transformations, il est possible de définir les augmentations utilisées par le module de manière a ce qu'elles utilisent une échelle commune de magnitude, similaire aux TF géométriques (voir figure 2.4). Pour cela, il suffit d'inverser l'échelle d'amplitude pour les TF Posteriser et Solariser. Tandis que le reste des TF de couleurs sont séparées en deux : les TF ″+″ augmentant la teinte, avec une échelle classique, et les TF ″-″ diminuant la teinte, avec une échelle inversée.

Les amplitudes m_t sont donc converties différemment en fonction de chaque TF, du type, des limites et des échelles de leurs amplitudes. Ces conversions produisent finalement un intervalle duquel est échantillonné aléatoirement *une* amplitude pour une TF appliquée à *une* image.

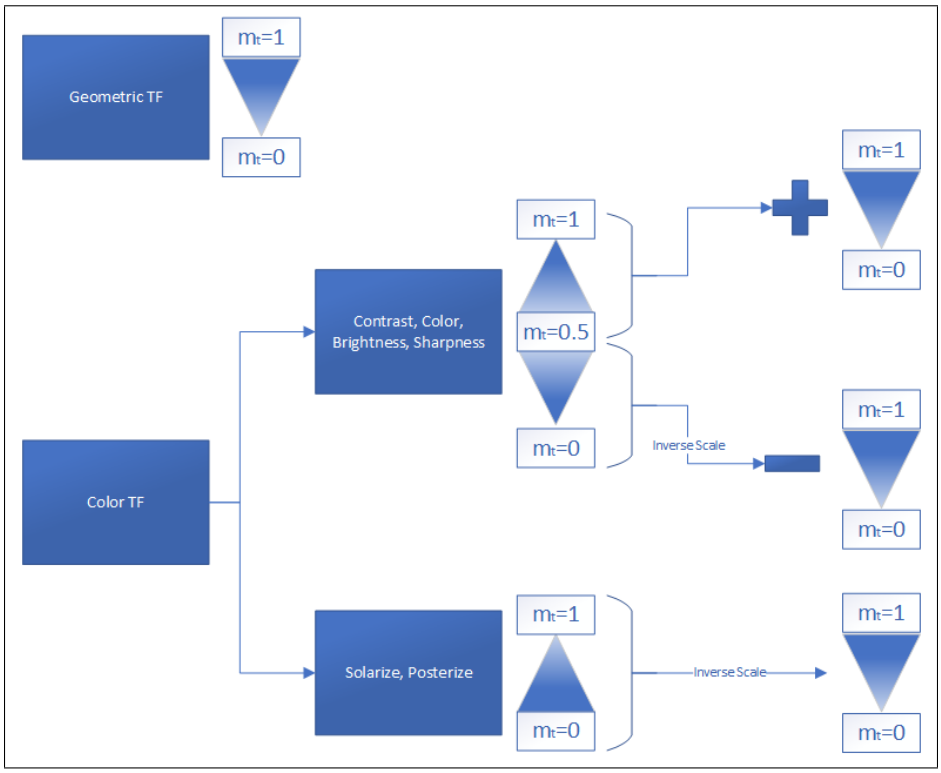

Figure 2.4 Principes de division des transformations de teintes

2.1.2 Optimisation des paramètres

2.1.2.1 Algorithme d'apprentissage

Comme montré dans le chapitre 1.2, il existe de multiples méthodes pour apprendre les TF optimales à appliquer à des données. Cependant, la majorité représente un important coût (calcul et mémoire) supplémentaire ou peu fiable. Parmi les méthodes présentées, les méthodes basées sur l'HG (section 1.2.2) semblent être celles fournissant le meilleur compromis.

On pourra noter que certaines approches visent à rechercher les meilleurs HP en se basant sur l'erreur d'entraînement -Chandra *et al.*-, et non l'erreur de validation. Mais, malgré leurs efficacités, elles ne sont fondamentalement pas appropriées pour l'apprentissage de TF.
En effet, avoir pour objectif la diminution de l'erreur d'entraînements est synonyme de chercher

les meilleures performances. En revanche, elle occulte l'importance de la *généralisation* de ces performances à d'autres données que celles d'entraînements.
Par conséquent, ces méthodes sont très efficaces pour optimiser des HP liés à la dynamique d'entraînement tel que le taux d'apprentissage ou d'autres paramètres des algorithmes d'optimisations, mais sont limitées en dehors de ceux-ci. Dans le cas des HP pour les TF, une optimisation basée seulement sur l'erreur d'entraînement encouragerait les HP produisant une distorsion minimale des données. La transformation des données ayant pour conséquence de rendre la tâche du modèle plus difficile. Ce qui n'est pas une solution optimale, loin de là. L'augmentation de donnée permettant, dans la majorité des cas, une nette amélioration des performances en utilisation réelle.

Ainsi, comme vu dans la section 1.2.2.2, il est possible de définir l'apprentissage optimal des TF comme un problème d'optimisation à deux niveaux (équations 2.2 & 2.3). Dans le cas de l'augmentation de donnée, l'objectif de haut niveau est défini par la minimisation de l'erreur de *validation* par les paramètres θ des TF (équation 2.2). W constituant les poids du modèle, X_{tr} et X_{val} respectivement les données d'entraînements et de validation et A l'augmentation de données.

$$\theta^* = \arg\min_{\theta} \mathcal{L}(X_{val}, W^*) \tag{2.2}$$

$$W^* = \arg\min_{W} \mathcal{L}(A_\theta(X_{tr}), W) \tag{2.3}$$

Inspirée par les travaux de Shaban *et al.*, ce problème est résolu par K-RMD, en propageant le gradient sur l'erreur de validation à travers les K derniers états du modèle jusqu'aux paramètres des TF (voir figure 2.5).

En pratique, l'implémentation de K-RMD est basée sur Higher et réalisée par l'algorithme 2.1.

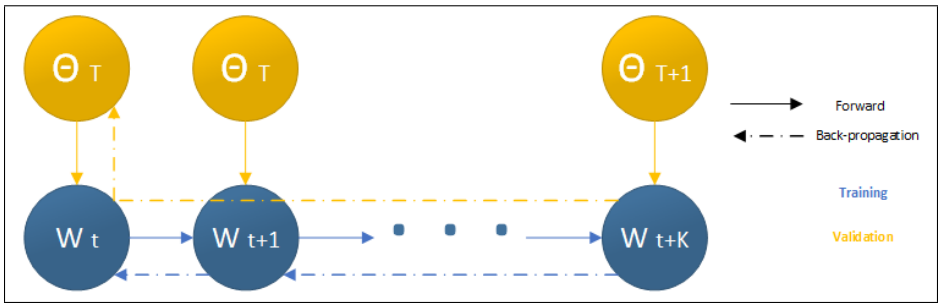

Figure 2.5 Principe de K-RMD

Algorithme 2.1 Optimisation jointe du modèle et du module de DA

Input : *Modèle* $\boldsymbol{Net_W}$*, Module de DA* $\boldsymbol{A_\theta}$*, Données d'entraînements* $\boldsymbol{X_{tr}}$*, Données de validation* $\boldsymbol{X_{val}}$*, Fréquence de mise à jour* $\boldsymbol{K}$
Output : *Modèle* $\boldsymbol{Net_{W'}}$ *et Module de DA* $\boldsymbol{A_{\theta'}}$ *mis à jour*
Initialisation du compteur d'états : $k = 0$
for $imgs_{tr}, labels_{tr} \leftarrow X_{tr}$ **do**
    Estimation de la fonction de coût : $\mathcal{L}_{tr}(Net, A, imgs_{tr}, labels_{tr})$
    Sauvegarde de l'état de Net
    $k+ = 1$
    Mise à jour de Net : $W' \leftarrow inner\_optimizer(\mathcal{L}_{tr})$
    if $k = K$ **then**
        $imgs_{val}, labels_{val} \leftarrow X_{val}$
        $pred_{val} \leftarrow Net(imgs_{val})$
        Estimation de la fonction de coût : $\mathcal{L}_{val}(pred_{val}, labels_{val})$
        Mise à jour de A : $\theta' \leftarrow upper\_optimizer(val_{loss})$
        Suppression des états de Net en mémoire
        $k = 0$
    end
end

2.1.2.2 Fonctions de coûts

En apprentissage machine, la bonne définition des fonctions de coût est primordiale. En effet, cette dernière conditionnera ce que le système apprendra. En apprentissage supervisé, la fonction de coût est la seule information qui guide les poids d'un modèle, à travers un vaste espace de recherche, vers des valeurs optimales. Valeurs minimisant la fonction de coût, du point de vue

du modèle, et permettant d'accomplir la tâche souhaitée, du point de vue du développeur.

Dans notre cas, le problème à deux niveaux sous-entend deux fonctions de coûts avec des significations différentes.
La première fonction de coût est celle basée sur les données d'*entraînements*. Systématiquement utilisée, elle doit représenter l'efficacité dans la tache souhaitée. Pour apprendre la DA optimale conjointement a l'apprentissage du modèle, cette fonction a été construite inspirée par la fonction de coût utilisée par Xie, Dai, Hovy, Luong & Le pour l'apprentissage non supervisé. La fonction de coût utilisée ici se décompose en trois sous fonction :

- **Perte supervisée** $\mathcal{L}_{sup}$: Corresponds à la distance entre le *label prédit* par le modèle à partir d'une image *non augmentée* et le label de cette image. Dans le contexte de classification multi classe, la fonction la plus utilisée est l'*Entropie croisée*, encourageant une bonne classification des échantillons. C'est aussi celle retenue pour ce projet.
- **Perte non supervisée** $\mathcal{L}_{unsup}$: Corresponds à la distance entre le *label prédit* par le modèle à partir d'une image *augmentée* et le label de cette image. Une *Entropie croisée* est aussi utilisée pour cette perte.
- **Divergence** D_{KL} : Corresponds à la divergence entre la *distribution prédite* par le modèle à partir d'une image *non augmentée* et la *distribution prédite* à partir d'une image *augmentée*. En effet, la dernière couche d'un réseau de classification fournit une distribution indiquant la probabilité d'un échantillon d'appartenir à chaque classe. Ainsi, il est possible de mesurer la distance, la divergence, entre les distributions afin d'encourager des prédictions similaires entre images augmentée et non augmentée, indépendamment de la justesse de ces prédictions. Cela permet d'encourager l'invariance des prédictions aux TF, valorisant la capacité de généralisation du modèle. Une mesure de divergence commune est la *Divergence de Kullback–Leibler*, retenue pour ce projet.

La fonction de coût finale $\mathcal{L}_{tr}$ est alors la somme de ses trois fonctions de coûts (voir équation 2.4). De plus, les pertes non supervisées et la divergence sont pondérées par un facteur λ afin

d'équilibrer l'importance des pertes non supervisées par rapport à celles supervisées. Dans le cas particulier où $\lambda = 0$, seule la perte *non supervisée* afin de toujours mettre a contribution la DA. Sauf indication du contraire, les résultats présentés dans le chapitre 3 utilisent un facteur $\lambda = 1$.

$$\begin{aligned} \mathcal{L}_{tr} = W_{loss} \times (\mathcal{L}_{sup} + \lambda \times (\mathcal{L}_{unsup} + D_{KL})), si \lambda > 0 \\ \mathcal{L}_{tr} = W_{loss} \times \mathcal{L}_{unsup}, sinon \end{aligned} \tag{2.4}$$

La seconde fonction de coût est celle basée sur les données de *validations* $\mathcal{L}_{val}$. Celle-ci a, habituellement, pour objectif de représenter la capacité de généralisation des connaissances d'un modèle à de nouvelles données. Plus faible est la perte sur les données de validation, plus grande sont les chances que les performances du modèle soient conservées lors de son exploitation.

$$\mathcal{L}_{val} = cross\_entropy(y_{pred}, y_{true}) + Reg_{mag} \tag{2.5}$$

Les prochaines sections présentes les spécificités relie a l'apprentissage des différents types de paramètres.

2.1.2.3 Optimisation des probabilités

L'apprentissage des probabilités p des TF a en réalité une signification plus large que seulement leurs probabilités d'applications. En effet, en plus de cela, elle représente aussi l'importance qui va être donnée à l'échantillon augmenté dans la fonction de coût.

Habituellement, les pertes sont évaluées par échantillons et, par la suite, moyennées pour obtenir une unique valeur scalaire, la perte, que l'on cherche à minimiser. Ici, chaque échantillon augmenté est pondéré en fonction des TF qui lui ont été appliqués, avant que la moyenne soit calculée. Concrètement, cela signifie qu'une image augmentée par une TF avec une haute

probabilité ($p_t \rightarrow 1$), donc jugée bénéfique pour le modèle, aura plus de poids dans la perte. Tandis qu'une image augmentée par une TF par une TF avec une faible probabilité ($p_t \rightarrow 0$) se verra octroyée moins de poids dans la perte, car possiblement trop altérée pour encore appartenir à la distribution de donnée que l'on cherche à modéliser.

En pratique, les augmentations étant réalisées par lots, la pondération de la perte est réalisée par une matrice normalisée de manière à conserver la valeur moyenne de la perte (voir équation 2.6). Pour des raisons de simplicité et de rapidité, la construction de cette matrice ne prend pas en compte l'ordre d'application de multiples TF.

Pour comprendre l'utilité d'une telle pondération, il est important de comprendre comment sont sélectionnés (voir section 2.1.1.2). En effet, la combinaison de la distribution de TF $\mathcal{D}$ avec une distribution uniforme ($\mathcal{D} \rightarrow D_\alpha$) signifie que même si une TF t est jugée néfaste, elle peut toujours être échantillonnée. En conséquence, l'impact sur la perte de l'image augmentée par t est réduit pour signifier une confiance réduite sur la validité de l'échantillon. De même, si les TF avec une haute probabilité ne sont pas sélectionnées aussi fréquemment qu'elle devrait l'être à cause du mélange de distribution, leurs importances dans la perte sont rehaussées pour compenser.

$$\begin{aligned} W_{loss} &= w_{loss} / Mean(w_{loss}) \\ w_{loss}(i) &= \sum_{t \subseteq \mathcal{T}_{sampled}(i)} p_t, p \subseteq \mathcal{D} \end{aligned} \tag{2.6}$$

Il peut-être légitime de se poser la question de la raison pour laquelle la distribution de TF $\mathcal{D}$ est adoucie ($\mathcal{D}_\alpha$), modérée par une distribution uniforme. Cette pratique a deux origines liées : la propagation du gradient et la stabilité.

En effet, la méthode d'optimisation retenue (K-RMD) nécessite de propager un gradient jusqu'aux hyper paramètres, les probabilités p. Or il n'est pas possible de rétro propager directement un gradient à travers $\mathcal{D}$ duquel nous avons échantillonné. Pour compenser cela, il est nécessaire d'utiliser un substitut permettant la propagation. Ce problème, commun dans le domaine de l'apprentissage par renforcement, peut-être résolu en faisant appel à une *estimation de la fonction de coût* (REINFORCE - Williams) ou à une *estimation des dérivées* (grâce à une reparametrisation). La solution retenue est basée sur REINFORCE. En pondérant la perte par

une fonction de p, on obtient une estimation de la fonction de coût réel.
Un des inconvénients des méthodes en apprentissage par renforcement est qu'elles sont relativement instables, la recherche de paramètre est souvent très bruitée. Aussi il est fréquent que de telles méthodes ne convergent tout simplement pas. C'est pour limiter cela que REINFORCE est adoucie par une distribution uniforme fixe (donnant $\mathcal{D}_\alpha$). Cette approche a le double avantage de faciliter le retour de TF qui aurait été éliminée durant l'entraînement, en plus de stabiliser l'apprentissage. Si les TF sont échantillonnés directement depuis $\mathcal{D}$ et que la probabilité p_t d'une TF t approche de 0, cela signifie qu'elle sera sélectionnée de moins en moins, voir plus du tout ($p_t = 0$). Or pour qu'un gradient soit propagé jusqu'à p_t, lui permettant d'évoluer, il est nécessaire que $t$usois appliquée.

Enfin, il peut être intéressant de noter qu'après chaque mise à jour, les paramètres p sont ajustés de manière à toujours définir une distribution, vérifiant l'équation 2.7.

$$\sum_{t \subseteq \mathcal{T}} p_t = 1 \tag{2.7}$$

2.1.2.4 Optimisation des amplitudes

L'apprentissage des amplitudes m présente un problème différent de l'apprentissage des probabilités, vu dans la section précédente. En effet, d'un côté, la propagation du gradient jusqu'à m_t est dépendant de l'implémentation des TF t, et de l'autre, la dynamique du système à encourager l'utilisation d'une forme de régularisation.
Afin d'autoriser la propagation directe du gradient jusqu'à m_t, il est nécessaire que t soit une fonction faisant appel à un paramètre *réel continu* et qu'elle *conserve* le type de m_t. En conséquence, l'ensemble des $\mathcal{T}_{no\_grad}$ ne permettent pas l'apprentissage de m. Les TF *Identité* et *Miroir* en font parties car elles ne font pas du tout appel à m_t. Tandis que *Solariser* et *Posteriser* ne permettent pas la propagation du gradient, car elles ne conservent pas le type de m_t (conversion en entier et booléen, respectivement).

De plus, dans le cas de l'apprentissage joint aux probabilités, un terme de régularisation fonction de m est ajouté à la perte de validation afin d'encourager des TF de hautes amplitudes (voir équation 2.8). Ce choix vient de l'observation que l'apprentissage joint de p et m offre à la DA deux degrés de libertés par TF. Une conséquence de cela est qu'il est possible pour une TF proche de l'identité, mais une probabilité haute ($m_t \rightarrow 0, p_t \rightarrow 1$). Ce cas de figure n'est pas souhaitable, car il sous-entend de multiplier les TF *Identité*, d'une manière inefficace, qui plus est. Ainsi, il a été décidé d'encourager des amplitudes maximales lors de l'apprentissage joint de p et m en se basant sur deux hypothèses :

- Une TF t produit une plus grande distorsion pour une magnitude m_t plus grande, générant un échantillon augmenté plus éloigné de l'échantillon d'origine, et, par conséquent, encourage une plus grande diversité dans les données d'entraînements.
- Dans le cas où une TF t ne possède aucune valeur de m_t pour laquelle elle est bénéfique à l'entraînement du modèle, l'importance de cette TF dans la perte sera affaiblie : $p_t \rightarrow 0$

$$Reg_{mag} = Mean(\{(m_t - 1)^2, t \subseteq \mathcal{T}\}) \tag{2.8}$$

Enfin, il peut être intéressant de noter qu'après chaque mise à jour, les paramètres m sont ajustés de manière à toujours être compris dans les bornes d'amplitudes normalisées ($m \subseteq [0, 1]$).

2.2 Matériel et contexte de recherche

2.2.1 Matériel

Le matériel utilisé par ce projet a été fourni par Teledyne DALSA.
Le développement de ce projet a été réalisé à l'aide d'un Razer Blade :

- *CPU* : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
- *GPU* : GTX 1060 Max-Q Design

- *GPU externe* : TITAN RTX, GeForce GTX 1080Ti

Les résultats, présentés dans le chapitre 3, ont principalement été obtenus à l'aide du système suivant :

- *CPU* : Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz
- *GPU* : TITAN RTX, TITAN Xp (x2)

On pourra noter que les calculs présentés dans le chapitre 3 n'utilisent qu'un seul GPU à la fois (*TITAN RTX*) afin d'obtenir des résultats comparables.

2.2.2 Environnement

Les développements et tests de ce projet ont été réalisés sous *Ubuntu* dans des conteneurs *Docker*. Cela a permis d'assurer un contrôle plus fin de l'environnement d'exécution et des librairies, présentées dans la section suivante.

2.2.3 Librairies

La totalité du code utilisé durant ce projet a été développée en *Python 3.5*. Ce langage a été choisi, car il permet de développer rapidement, supporte les principales librairies d'apprentissage profond et constitue le langage principal utilisé par l'équipe Axon de Teledyne DALSA.

La libraire d'apprentissage profond retenue pour l'implémentation de l'apprentissage de DA est *PyTorch 1.3*. L'utilisation de telles librairies est particulièrement intéressante pour les capacités d'optimisations et de différentiation automatique (Autograd en PyTorch). L'utilisation de *TensorFlow* a aussi été explorée au début du projet, mais abandonnée, car demandant des temps de développement plus long que PyTorch et disposant de moins d'implémentation des algorithmes étudiés.

La définition des TF pour la DA (voir section 2.1.1.1) a été principalement basée sur *Kornia 0.2.0* -Riba, Mishkin, Ponsa, Rublee & Bradski-, une librairie basée sur PyTorch, permettant le traitement d'images de manière différentiable. De plus, les TF de teintes ont été développées pour être similaires à celles disponibles dans *Pillow* (PIL). Ces choix ont été faits pour être aussi proches que possible des augmentations d'AutoAugment -Cubuk *et al.* (a)-, basées sur Pillow.

L'implémentation de l'algorithme de méta-apprentissage (voir section 2.1.2) est, quant à elle, basée sur *Higher 0.1.5* -Grefenstette, Amos, Yarats, Htut, Molchanov, Meier, Kiela, Cho & Chintala-. Cette librairie facilite le calcul de gradient d'ordre supérieur, des HG, avec PyTorch.

CHAPITRE 3

RÉSULTATS

3.1 Métriques

Les résultats présentés dans ce chapitre utilisent plusieurs types de métriques.
Les performances du modèle sont suivies par :

- **Perte d'entraînement** : fournis par la fonction de coût 2.4 (détaillée dans la section 2.1.2.2). Elle est directement utilisée pour la recherche des poids du modèle, dans le but de la *minimiser*.
- **Perte de validation** : fournis par la fonction de coût 2.5 (détaillée dans la section 2.1.2.2). Elle est utilisée pour la recherche des HP (paramètres de la DA), dans le but de la *minimiser*.
- **Précision** : Évaluée sur les données de tests (indépendantes de l'entraînement du modèle), elle caractérise le pourcentage de bonne classification du modèle. C'est la métrique principale utilisée, dans ce projet, pour comparer les modèles. L'objectif est de *maximiser* la précision du modèle.
- **Score F1** : Comme la précision, le score F1 est évalué sur les données de tests. Cette métrique est plus fine que la précision, car elle prend aussi en compte la mesure du rappel du modèle. Lors de l'utilisation de base de données équilibrée, comme c'est le cas dans les résultats présentés, le Score F1 est très proche de la précision. Ainsi, il est, ici, utilisé comme validation de la mesure de précision. Décomposé en mesure par classe, le score F1 permet de suivre les performances de classification de chaque classe indépendamment. L'objectif est de *maximiser* le score F1 de *chacune* des classes, en s'assurant qu'il n'y a pas de classe laissée à l'abandon.

Durant l'entraînement, les paramètres de la DA (*probabilités* et *amplitudes* des TF) sont suivis pour étudier l'apprentissage de la DA (voir figure 3.1). En plus de l'évolution de ces paramètres avec les époques, on s'intéressera aussi aux *valeurs moyennes des paramètres* de chaque TF

durant l'entraînement.

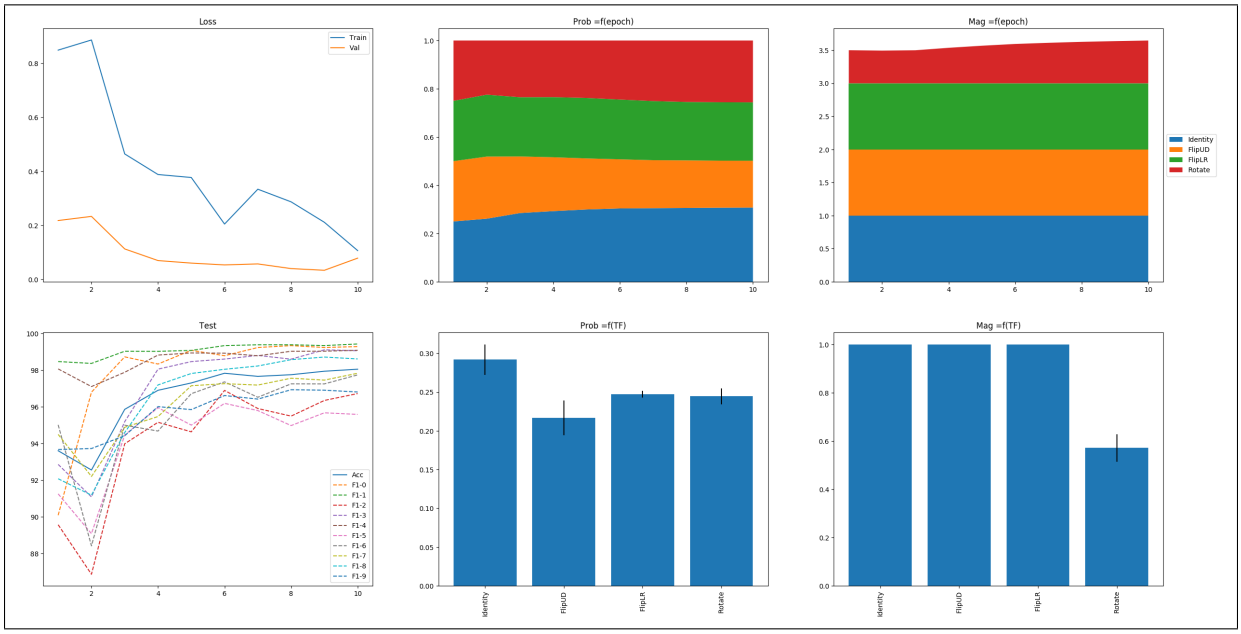

Figure 3.1 Exemple de graphique de résultats d'apprentissage. En haut à gauche, les courbes de pertes. En bas a gauche les courbes de précision et de scores F1. Les graphiques du milieu et de droite présentent respectivement, les probabilités et les amplitudes moyennes des transformations. Ceux du haut contiennent l'évolution avec les époques ; ceux du bas, les valeurs moyennes et leurs écarts-types.

Enfin, les *temps d'apprentissages* ainsi que l'*empreinte mémoire* seront rapportés afin d'avoir une vision globale du rapport coût/performances.

3.2 Présentation des modèles et données

3.2.1 Modèles

Plusieurs réseaux de neurones convolutionnels (CNN), avec des capacités différentes, ont été utilisés au cours de ce projet :

- **LeNet** -Lecun, Bottou, Bengio & Haffner- : Ce réseau fait partit des architectures les plus simples des CNN. Il a une faible capacité par rapport à la moyenne (avec 60.000 paramètres tandis que les CNN, de nos jours, sont fréquemment au-dessus de 100 millions de paramètres). Ce réseau a été principalement utilisé durant les premières étapes du développement de ce projet.
- **ResNet18** -He, Zhang, Ren & Sun- : Les réseaux de neurones résiduels (ResNet), à la différence des réseaux ordinaires, présentent une architecture dans laquelle des 'raccourcis' sont présents. Des couches du réseau peuvent être ignorées ou combinées avec les sorties de précédentes couches (les résidus). Cela permet notamment de donner plus de flexibilités au modèle et de prévenir la disparition du gradient, un problème courant en apprentissage profond. Le ResNet18 est une configuration particulière de cette architecture.
- **WideResNet50** -Zagoruyko & Komodakis- : Les réseaux de neurones résiduels larges (WideResNet ou WRN) utilisent des blocs de ResNet pour produire des réseaux plus larges et moins profonds que d'autres modèles avec des capacités similaires. Ils permettent ainsi d'obtenir des modèles avec une capacité plus grande que les ResNet, donc un potentiel de performance plus important, tandis qu'ils nécessitent moins de temps d'entraînements que des modèles à capacités égales, mais plus profonds. Le WideResNet50 est une configuration particulière de cette architecture.

3.2.2 Données

Différentes bases de données ont été utilisées afin d'évaluer les performances du module de DA :

- MNIST -Lecun *et al.*- : Cette base de données contient 70.000 exemples de chiffres (entre 0 et 9) écrits à la main en niveau de gris. Idéal pour tester des méthodes d'apprentissage machine, elle a été utilisée principalement pour les premières étapes du développement de ce projet. Elle constitue un problème de reconnaissance de formes relativement simple. On pourra noter que pour des raisons de simplicité et de compatibilité les images de cette base de données ont été altérées. N'ayant à l'origine qu'un seul canal de couleur, ce canal a été dupliqué pour obtenir des images à 3 canaux, similaires aux images couleur.

- CIFAR10 -Krizhevsky- : Cette base de données contient 60.000 exemples de petites images de 10 classes mutuellement exclusives (Avion, Voiture, Oiseau, Chat, Cerf, Chien, Crapaud, Cheval, Bateau, Camion). Elle constitue un problème plus complexe que MNIST. Les modèles entraînés sur ces données bénéficient beaucoup de DA, la quantité d'exemples étant relativement faible par rapport a la complexité du problème.

Pour ce projet, chacune de ces bases de données a été séparée en base de données d'entraînement, de validation et de tests.

3.3 Apprentissage de l'augmentation de donnée

3.3.1 Présentation

Afin de vérifier que le système développé accompli bien ça tache principale, l'apprentissage des TF pour la DA, des tests ont été réalises. Concrètement, les valeurs optimales des TF étant inconnues, la capacité évaluée a été celle d'éliminer des TF connues pour être néfastes à l'entraînement. La quantité de ces mauvaises TF connue étant limitée, un certain nombre de TF artificielles néfastes ont été ajoutées (des TF déjà implémentés avec des configurations spécifiquement pensées pour être néfastes, voir tableau I-3).
Une première expérience a été menée sur MNIST sur laquelle il est connu que la TF miroir horizontal est dommageable. Les résultats obtenus sont rapportés dans la figure 3.2 et un échantillon d'entraînement sur la figure 3.3.

Une première expérience a été menée sur CIFAR10 sur laquelle il est connu que la TF miroir horizontal est, aussi, dommageable. Les résultats obtenus sont rapportés dans la figure 3.5 et un échantillon d'entraînement sur la figure 3.6.

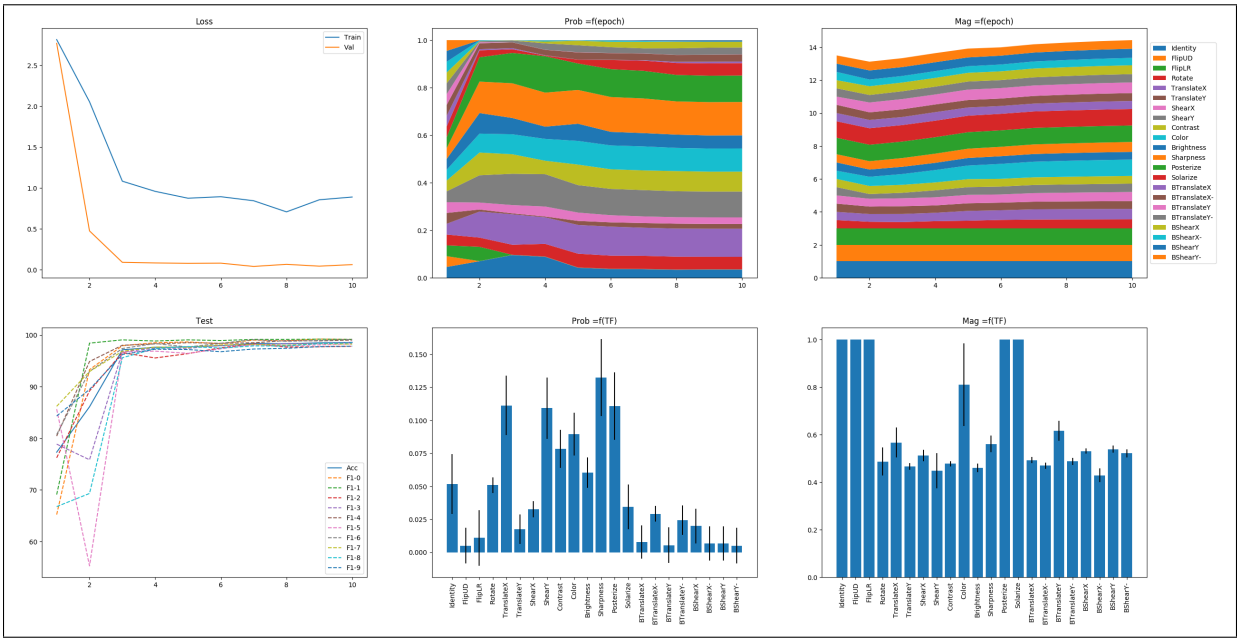

Figure 3.2 Résultat de ResNet18 sur MNIST avec le module de DA
(*N=3, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$*)*

3.3.2 Interprétation

Les résultats obtenus sur MNIST (figure 3.2) montrent que les TF néfastes connues (FlipUD, BTranslateX, BTranslateX-, BTranslateY, BTranslateY-, BShearX, BShearX-, BShearY et BShearY-) sont nettement séparées du reste des TF. De plus, la figure 3.3 montre que la DA tend a mettre des poids plus faibles aux échantillons que l'on ne pourrait plus associes a leurs classes d'origine. Cela laisse penser que le module de DA est bien capable d'apprendre.

On pourra cependant noter que d'autres TF, a priori bénéfiques, telles que FlipLR, TranslateY et ShearX, ne présentent pas des probabilités des sélections aussi hautes qu'attendu. On peut expliquer cette observation par le fait que les TF appliquées sont des combinaisons de TF de base et il est possible que les combinaisons avec ces TF ne donnent pas d'aussi bons résultats que lorsqu'elles sont appliquées seules. Il est aussi tout à fait possible que l'apprentissage soit limite dans le cas de MNIST par la perte de validation. En effet, l'observateur attentif aura remarqué que les performances du ResNet (perte de validation et précision) se stabilisent très rapidement. La perte étant proche de 0 (et la précision de 100%) et n'évoluant quasiment plus,

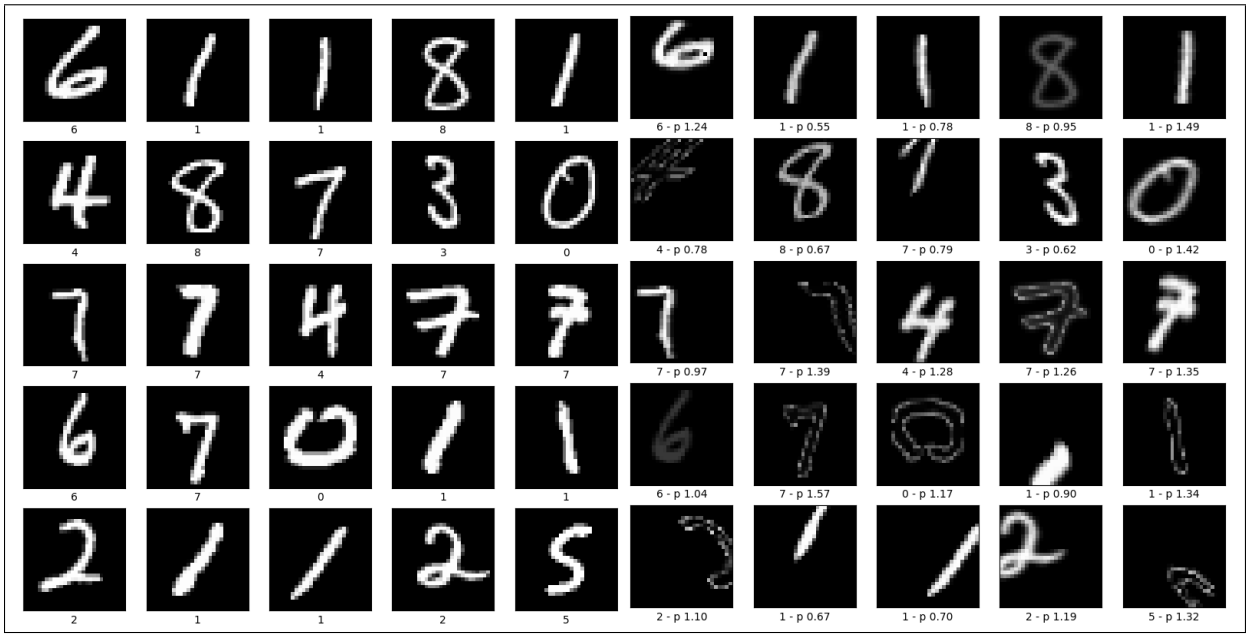

Figure 3.3 Échantillon d'entraînement de ResNet18 sur MNIST (époque 20) avec le module de DA (*N=3, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$). Les échantillons d'origines sont à gauche, tandis que les échantillons augmentés sont à droite. Les classes ainsi que les poids attribués à chaque échantillon augmenté ("- p") sont en dessous de chaque image.

l'algorithme de méta-apprentissage manque d'informations pour le choix des TF.

Les résultats obtenus sur CIFAR10 (figure 3.5) confirment les observations réalisées sur MNIST. Les TF néfastes connues sont dissociées du reste des TF, mais une partie des TF qu'on aurait jugée, a priori, bénéfique (TranslateY, ShearY) semble aussi mise à part. Les poids attribués aux échantillons (figure 3.6 tendent aussi a mettre des poids inférieurs aux images trop distordues. On pourra noter que les probabilités varient beaucoup au cours de l'entraînement, sans se stabiliser réellement autour d'une valeur. Cela laisse penser qu'il n'y a pas une configuration de DA previligie, mais que la configuration optimale est variable (cette hypothèse est de nouveau abordée dans la section 3.4.2.2 avec les valeurs de K).

De plus, il est intéressant d'observer que les amplitudes tendent généralement à être faibles au début de l'entraînement et croissantes par la suite. Cela correspond a l'intuition que l'on pourrait avoir du modèle nécessitant peu de distorsion/augmentation a début de l'entraînement pour

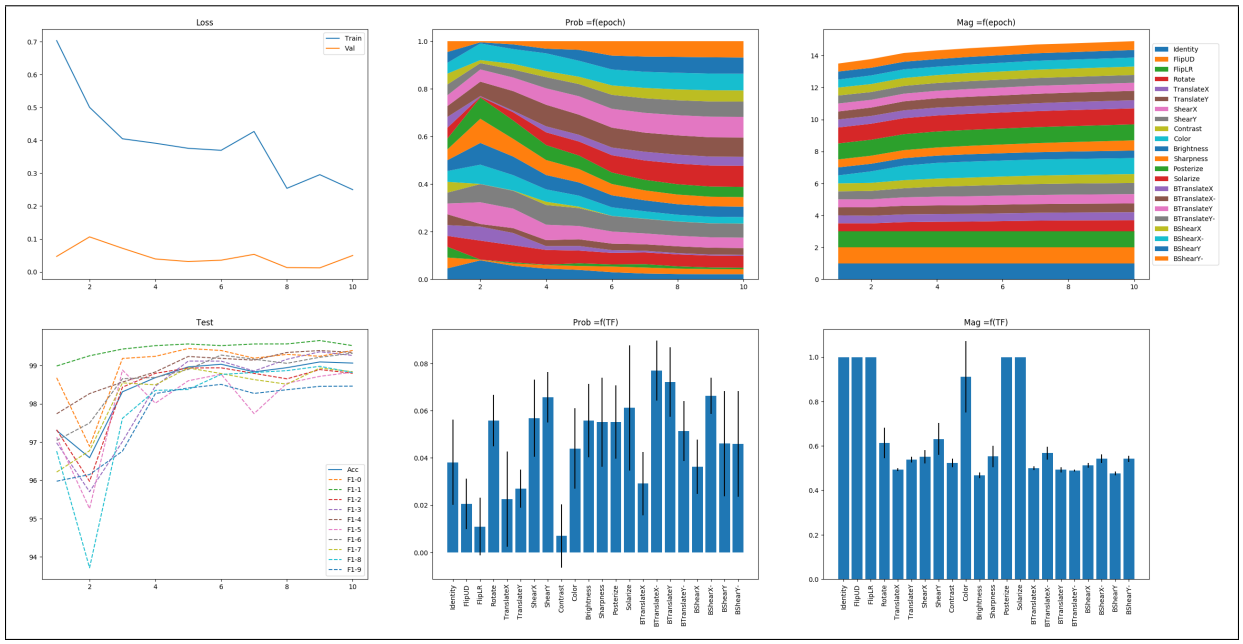

Figure 3.4 Résultat de ResNet18 sur MNIST avec le module de DA
(*N=1, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$*)*

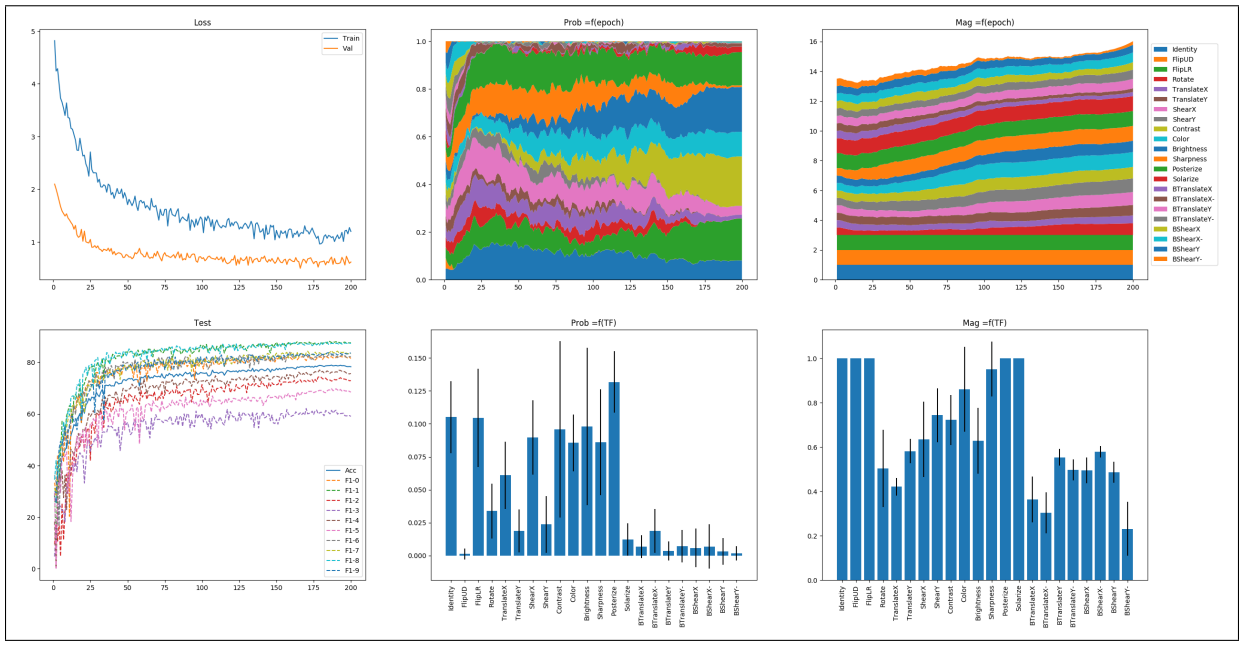

Figure 3.5 Résultat de ResNet18 sur CIFAR10 avec le module de DA
(*N=3, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$*)*

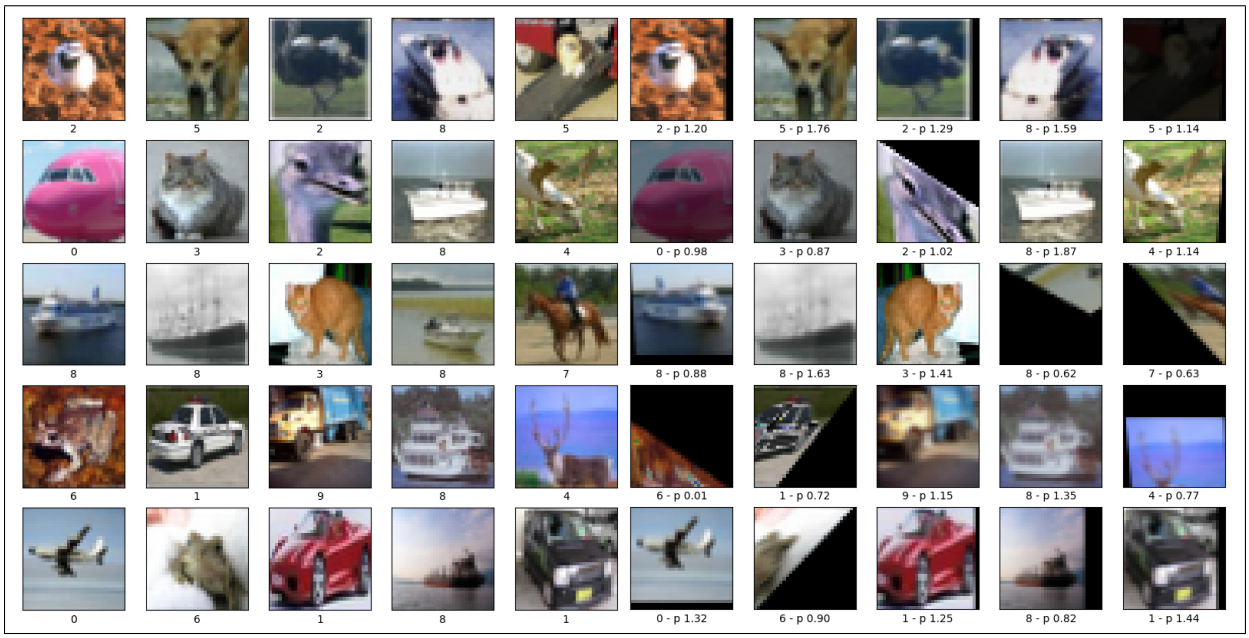

Figure 3.6 Échantillon d'entraînement de ResNet18 sur CIFAR10 (époque 20) avec le module de DA (*N=3, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$). Les échantillons d'origines sont à gauche, tandis que les échantillons augmentés sont à droite. Les classes ainsi que les poids attribués à chaque échantillon augmenté ("- p") sont en dessous de chaque image.

apprendre a partir des données réelles. Pour ensuite, chercher des distorsions de plus en plus importantes, des exemples de plus en plus difficiles pour augmenter sa capacité de généralisation (et réduire la perte de validation). Il est cependant aussi possible que cette augmentation soit seulement le résultat de la diminution du terme de régularisation des amplitudes (voir section 2.1.2.4), ce qui pourrait expliquer l'absence de diminution des amplitudes des TF néfastes connues.

La comparaison des figures 3.2 et 3.4 permets de mettre en évidence une limitation importante de notre approche. En effet, *plus faible est le nombre de TF séquentielles N, moins le méta-apprentissage est efficace*. Comme le montre la figure 3.4, les TF ne pressentent pas les paramètres attendus, avec de hautes probabilités pour les mauvaises TF connues. Un nombre réduit de combinaison N signifie un nombre réduit de TF sélectionnes et appliquées. Induisant invariablement, moins de retours sur l'influence de chaque TF et donc compliquant la tâche

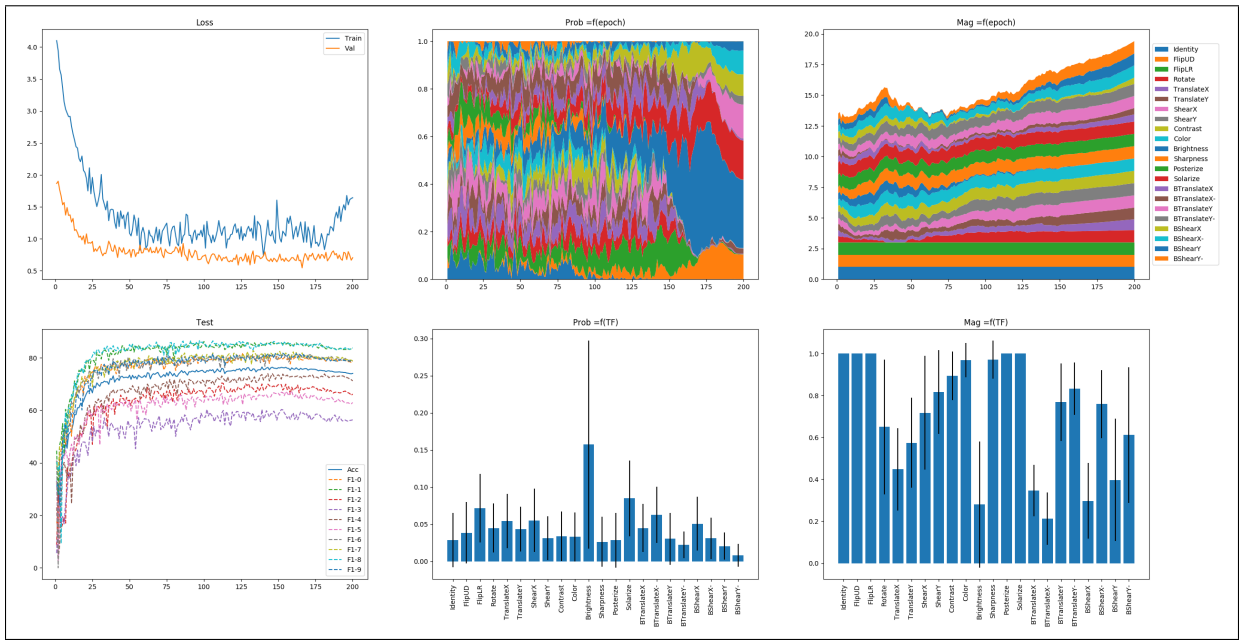

Figure 3.7 Résultat de ResNet18 sur CIFAR10 avec le module de DA (*N=1, TF=Mauvaise - I-3, K=1,* $\alpha = 0.5$*)*

du méta-apprentissage. La sélection stochastique des TF aggrave encore la situation, toutes les TF n'étant pas directement évaluées sur chaque image. Cette hypothèse est encouragée par la comparaison des figures 3.5 et 3.7.

Ceci dit, les capacités d'apprentissage réelles du module de DA seront approfondies, grâce à la comparaison avec RandAugment, dans les sections 3.5 et 3.6.

3.4 Étude des hyper paramètres pour l'augmentation de donnée

3.4.1 Configuration des transformations

3.4.1.1 Présentation

Avant de chercher les meilleurs HP pour la DA, il est important de s'assurer que les TF fournies sont bien configurées. Les configurations des TF comprennent, principalement, les fonctions

utilisées ainsi que les plages d'amplitudes autorisées. Concrètement, deux configurations différentes ont été étudiées :

1. *Configuration de base* (voir tableau I-1) : C'est la configuration apportée par AutoAugment -Cubuk *et al.* (a)- et, plus tard, utilisée par RandAugment -Cubuk *et al.* (b)-.
2. *Configuration large avec échelles inversées* (voir tableau I-2) : Une configuration offrant de, globalement, plus larges plages d'amplitudes que la configuration de base et décomposant les TF de teintes (voir section 2.1.1.3).

Un résumé des résultats obtenus avec les deux configurations sur le module de DA et avec notre implémentation de RandAugment est rapporté sur le tableau 3.1. On pourra noter qu'à défaut d'apprendre le paramètre d'amplitude des TF, l'amplitude des TF de RandAugment est fixée, ici, au maximum (M=1).
Les tests sont réalisés en 3 exemplaires afin d'évaluer la stabilité des résultats.

Tableau 3.1 Résultats de différente configuration de transformations pour un ResNet18 sur CIFAR10

DA	Configuration	Précision	Écart-type
Module DA (*N=3, K=1,* $\alpha = 0.5$)	Base	79.4%	0.7
Module DA (*N=3, K=1,* $\alpha = 0.5$)	Larges & échelles inversées	78.9%	0.2
RandAugment (*N=3, M=1*)	Base	87.4%	0.3
RandAugment (*N=3, M=1*)	Larges & échelles inversées	86.96%	0.1

3.4.1.2 Interprétation

La comparaison des configurations de TF semble montrer que la configuration de base procure les meilleures performances, la différence de performance restant relativement faible. Malgré des performances inférieures, la configuration large avec échelles inversées est celle qui a été retenue pour les expériences suivantes. En effet, après études de la configuration de base (tableau I-1), on peut remarquer que les plages d'amplitudes autorisent pour les TF géométriques sont relativement petites par rapport aux configurations possibles. Un bon exemple étant la Rotation

réduite entre [-30°,30°] tandis qu'on pourrait la définir entre [-180°,180°]. Cela sous-entend que cette configuration a été modifiée pour mieux s'adapter aux données et/ou a l'algorithme d'AutoAugment -Cubuk *et al.* (a)-. Dans les deux cas, une telle modification n'est pas souhaitable lors de l'étude de l'apprentissage des paramètres de DA, car biaise et manquant de généralités. De plus, comme soulevé dans la section 2.1.1.3, la définition des TF de teintes fait que la configuration d'AutoAugment est inappropriée. L'amplitude ayant un sens différent dans le cas de ces TF.

3.4.2 Hyper paramètres du module d'augmentation de donnée

3.4.2.1 Présentation

Afin de pouvoir évaluer efficacement le module de DA, il était nécessaire de connaître les effets des HP sur l'apprentissage. Les HP étudiés sont :

- *La fréquence K de mise à jour des paramètres de DA* : Ici, elle correspond aussi au nombre K d'états utilisés par l'algorithme de méta-apprentissage (voir section 2.1.2). Ainsi, les paramètres de DA sont modifiés toutes les K itérations du modèle.
- *Le nombre de TF séquentielles N* : Il peut aussi être vu comme la taille N des combinaisons de TF de base appliquée a chaque image.
- *Le facteur de mélange* α : Définis l'adoucissement de la distribution de laquelle sont échantillonnés les TF (voir section 2.1.1.2).
- *Le gel des amplitudes* : Il est possible de fixer les amplitudes des TF ou de les apprendre conjointement aux probabilités. Par défaut, les amplitudes sont aussi apprises.

Une approche de recherche par quadrillage a été utilisée avec un ResNet18 sur CIFAR10 (avec une répartition 50%/50% des données en apprentissage/validation). La configuration de base des TF (voir section 3.4.1.1) a été utilisée.
Une première évaluation a aussi été réalisée au cours du développement avec LeNet. Elle a notamment permis d'informer la seconde recherche avec le ResNet18. Mais ces résultats ayant

été obtenus sur de précédentes versions du module de DA, ils ne seront pas présentés ici.
Les tests sont réalisés en 3 exemplaires afin d'évaluer la stabilité des résultats.

La fréquence K a des conséquences directes sur les performances ainsi que les ressources nécessaires pour réaliser l'apprentissage. Ainsi l'évolution avec ce paramètre de la précision (figure 3.8), du temps d'apprentissage (figure 3.9) et de l'empreinte mémoire (figure 3.10) ont été spécifiquement étudiées.

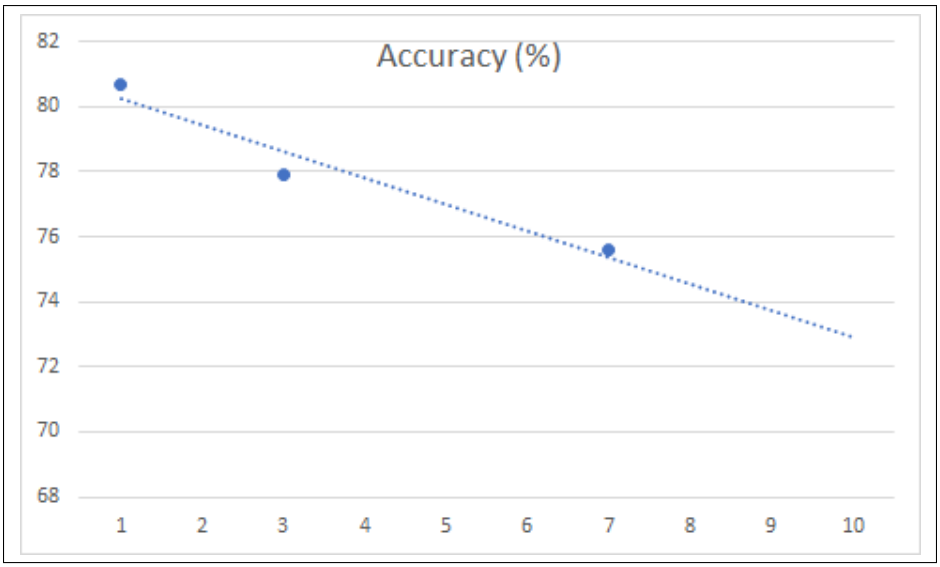

Figure 3.8 Évolution de la précision en fonction de K

Pour ce qui est des autres HP, les tendances étant nettement moins marques et les résultats moins visuels, seuls les graphiques jugés pertinents, figure 3.11 et 3.12, sont présentées. Par souci de transparence, les résultats de la recherche d'HP sont disponibles dans l'annexe II.

3.4.2.2 Interprétation

L'évolution des performances avec le nombre d'états utilisés K pointe dans la direction de faibles valeurs de K pour les meilleurs résultats. La meilleure précision (figure 3.8) et empreinte mémoires (figure 3.10) étant atteinte pour K=1. Seul le temps d'entraînements bénéficie de plus

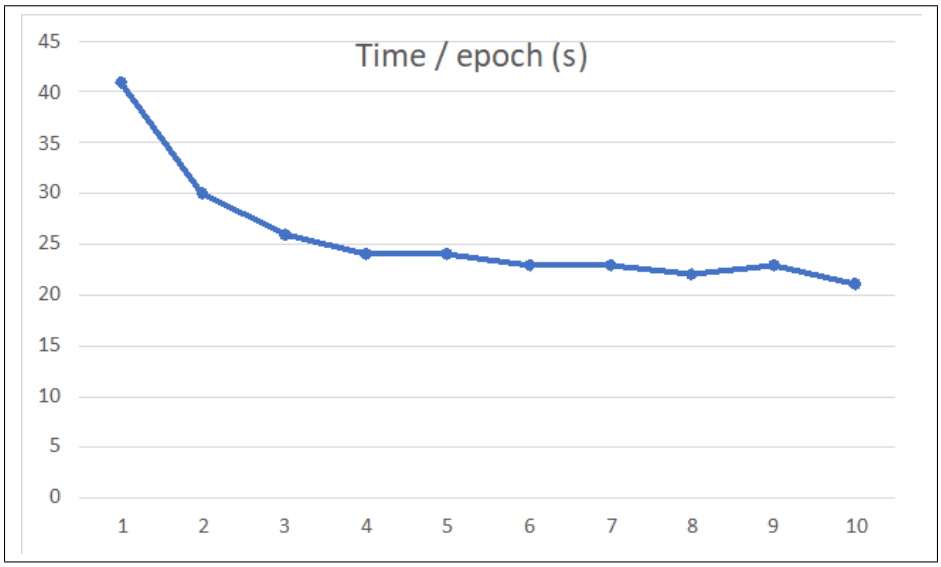

Figure 3.9 Évolution du temps d'apprentissage (par époques) en fonction de K

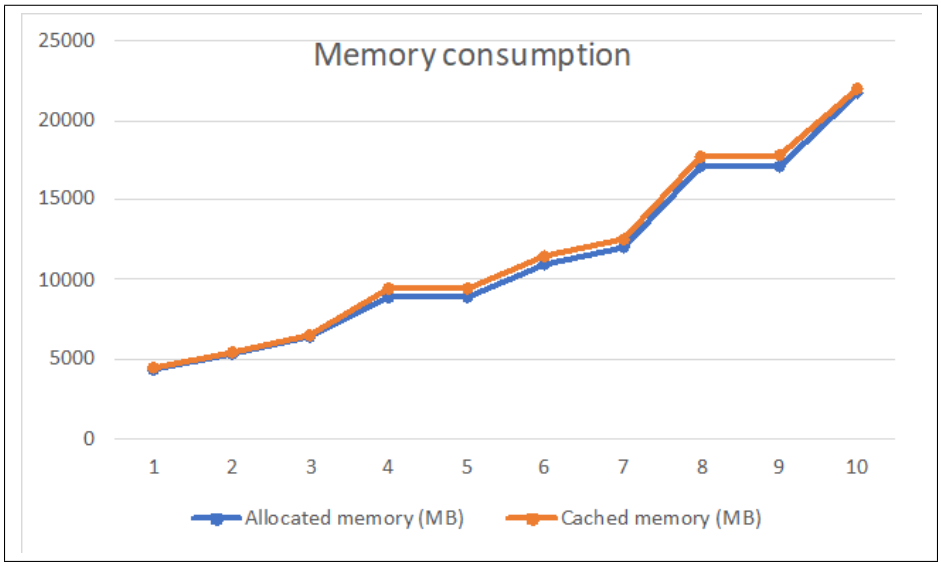

Figure 3.10 Évolution de l'empreinte mémoire avec K

haute valeur de K (figure 3.9).

Les résultats pour l'empreinte mémoire et le temps d'entraînement ne sont pas très surprenants. La méthode de méta-apprentissage (voir 2.1.2) nécessitant de stocker en mémoire K fois les états du modèle, son évolution est relativement linéaire par rapport a K. On remarquera cependant que,

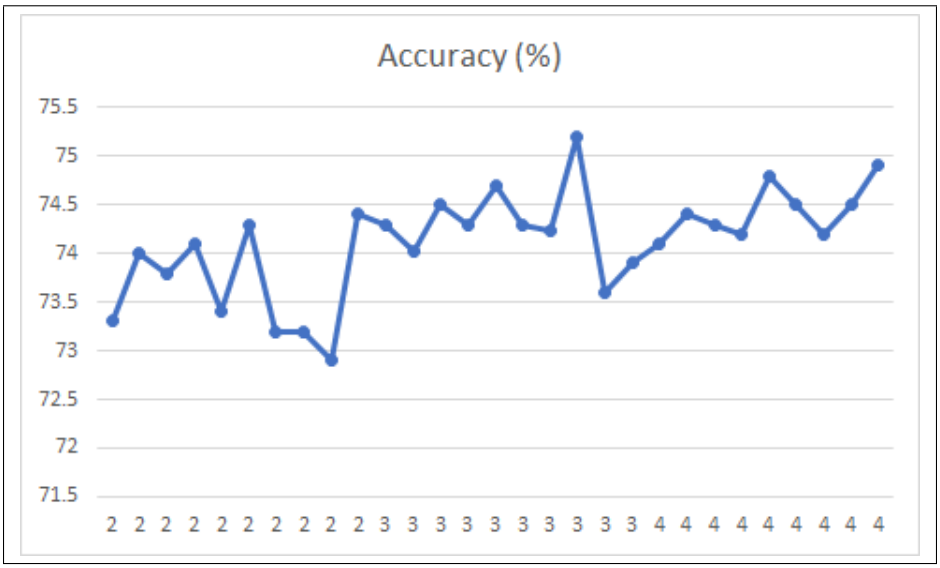

Figure 3.11 Évolution de la précision avec N

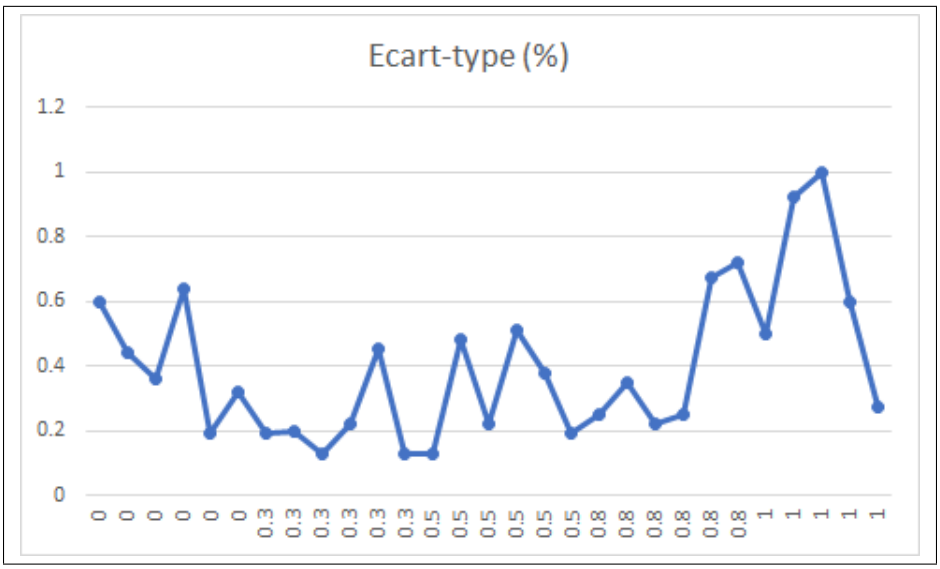

Figure 3.12 Évolution de l'écart-type avec α

dans notre cas, cette courbe est inférieure à la linéarité. Cela vient probablement de la gestion sous-jacente de la mémoire par Higher (voir section 2.2.3). Ensuite, pour ce qui est du temps d'entraînement, les mises à jour des paramètres de DA étant réalisées plus fréquemment pour de faibles valeurs de K, il n'est pas étonnant d'observer des temps d'entraînements décroissants

avec K.

En revanche, l'évolution de la précision avec K (figure 3.8) est, quant à elle, moins attendue. En effet, en théorie, plus le nombre d'états K utilisé est grand plus la convergence vers des paramètres de DA optimaux est suppose être rapide et, par voie de conséquence, devrait amener a de meilleures précisions. En revanche, il est important de noter qu'ici K définit aussi la *fréquence de mise à jour* des paramètres de DA.
De plus, tous les résultats obtenus semblent montrer qu'il n'y a pas *une* configuration optimale de DA, les paramètres de DA se stabilisant rarement autour d'une valeur. Cette observation laisse penser que la configuration optimale de DA n'est pas une configuration, mais un *planning* de configuration, que la configuration optimale change avec les époques. Cette hypothèse n'est pas nouvelle et est fondatrice dans l'approche de PBA -Ho *et al.*- (voir section 1.2.1).
Avec cette optique, il est plus cohérent d'obtenir de meilleure précision avec une valeur de K faible, car la mise à jour des paramètres de DA est plus fréquente. Cela implique que la DA est capable de s'adapter plus rapidement aux 'besoins' changeant du modèle au cours de l'entraînement. Le planning de configuration final finissant donc par être plus proche de l'optimalité.

Concernant le nombre de TF séquentielles N, les résultats semblent montrer que les meilleures performances sur CIFAR10 sont obtenues pour des combinaisons de 3 ou 4 TF (N=3, N=4). Il est intéressant de remarquer que dans le cas de RandAugment, la valeur de N optimale est plutôt autour de 2 ou 3. Ainsi, il semble que l'apprentissage des paramètres des TF de base permette de supporter des combinaisons plus complexes. Les combinaisons complexes ont plus de chance d'apporter une distorsion trop importante aux données, les excluant de la distribution que l'ont cherche à modéliser. Dans cette perspective, l'apprentissage permet de réduire les chances d'obtenir de telle distorsion. Autorisant, par conséquent, la découverte de TF plus complexes qui ont le potentiel d'accroître la capacité de généralisation du modèle.

Une telle préférence pour de hautes valeurs de N semble aussi aller dans le sens de l'hypothèse que *plus faible est le nombre de TF séquentielles N, moins le méta-apprentissage est efficace* (voir section 3.3.2).
Cependant, on pourra noter que, dans le cas de N=5, les données sont tellement distordues que le modèle n'arrive pas du tout à converger.

Pour ce qui du facteur de mélange α, l'influence de ce dernier sur les résultats est moins nette. Néanmoins, on s'aperçoit qu'un $\alpha = 0.5$ semble fournir un bon compromis entre précision et stabilité en moyenne.
Il est aussi pertinent de constater que l'écart-type augmente nettement quand α se rapproche de 1. Comme abordée dans la section 2.1.2.3, plus la distribution adoucie $\mathcal{D}_\alpha$ tend vers la distribution réelle $\mathcal{D}$, plus le problème se rapproche d'un apprentissage par renforcement et, donc, a tendance a être instable.

Finalement, le gel des amplitudes ne présente pas non plus un effet net sur la précision en général. On peut distinguer une légère tendance reliant le gel des amplitudes à une diminution de l'écart-type et de la précision. Cette tendance va dans le sens de ce qu'on pourrait attendre, car un gel des amplitudes signifie moins de variables et donc plus de stabilité ; mais aussi une capacité moindre de la DA à s'adapter, ce qui pourrait limiter les précisions atteignables.

3.5 Étude comparative à l'état de l'art

3.5.1 Présentation

Cette section présente une évaluation du module de DA avec une comparaison des modèles sans DA, avec RandAugment et avec apprentissage de la DA. Les ResNet18 et WideResNet50 sont testées sur CIFAR10 (figures 3.13, 3.14, 3.15 et 3.16), dans un premier temps, puis sur SVHN. Un résumé des résultats est disponible dans les tableaux 3.2, 3.3, .

Enfin, une comparaison des performances de RandAugment et le module de DA avec les TF néfastes sont rapportés dans le tableau 3.4.
Les tests sont réalisés en 3 exemplaires afin d'évaluer la stabilité des résultats.

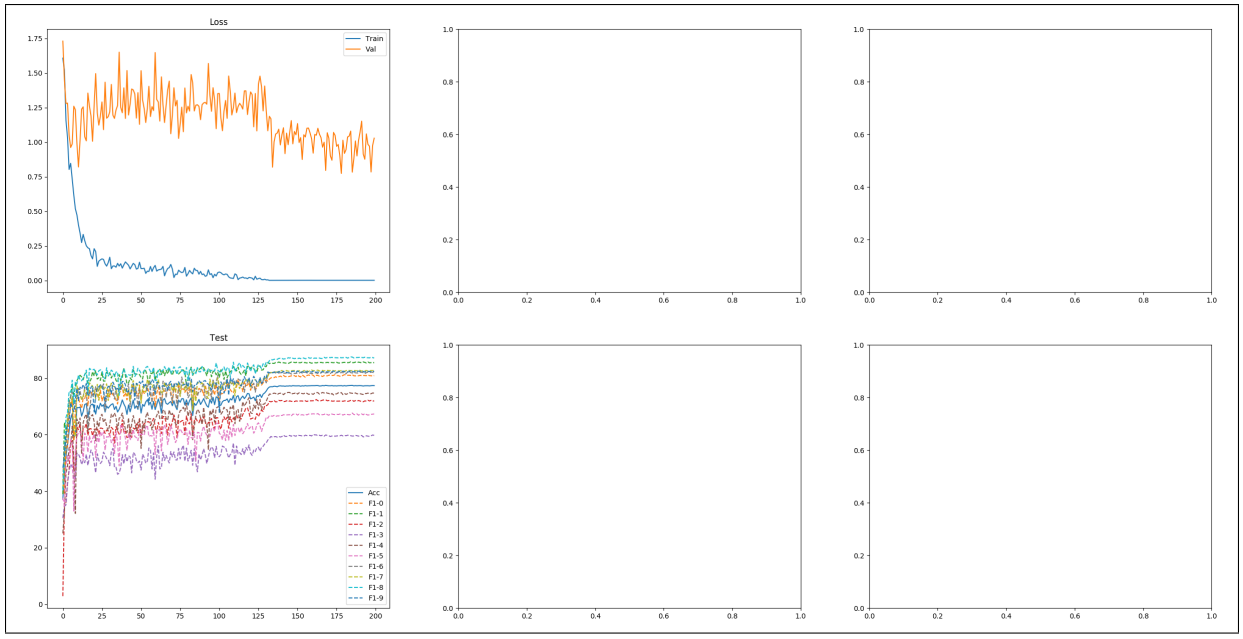

Figure 3.13 Résultat de ResNet18 sur CIFAR10 sans DA

Tableau 3.2 Comparaison résultats des systèmes de DA pour un ResNet18 sur CIFAR10 (Configuration des transformations large avec échelles inverses)

DA	**Précision**	**Écart-type**	**Époque (s)**	**Mémoire (MB)**
Aucun	77.3%	0.1%	7	3.5k
Module DA (*N=3, K=1,* $\alpha = 0.5$)	79.4%	0.7%	43	4.5k
RandAugment (*N=3, M=1*)	86.96%	0.1%	11	3.8k
RandAugment (*N=2, M=1*)	86.8%	0.5%	10	3.8k
RandAugment (*N=3, M=0.17*)	84.0%	0.1%	11	3.8k

3.5.2 Interprétation

Avant toute autre interprétation, il est important de remarquer l'effet radical de la DA sur les performances et les capacités de généralisations d'un modèle. Maintes fois expérimentés par

Figure 3.14 Résultat de ResNet18 sur CIFAR10 avec RandAugment
(*N=2, TF=Larges & échelles inversées - I-2)*

Figure 3.15 Résultat de ResNet18 sur CIFAR10 avec RandAugment
(*N=3, TF=Larges & échelles inversées - I-2)*

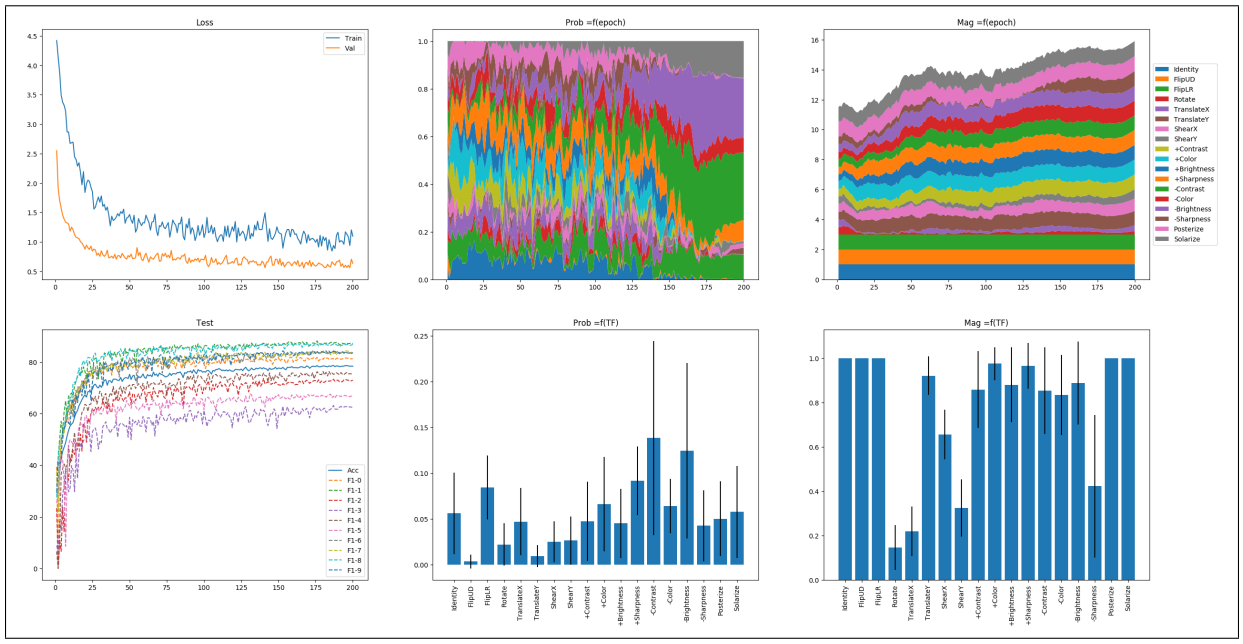

Figure 3.16 Résultat de ResNet18 sur CIFAR10 avec le module de DA (*N=3, TF=Larges & échelles inversées - I-2, K=1,* $\alpha = 0.5$*)*

Tableau 3.3 Comparaison résultats des systèmes de DA pour un WideResNet50 sur CIFAR10 (Configuration des transformations large avec échelles inverses)

DA	**Précision**	**Écart-type**	**Époque (s)**	**Mémoire (MB)**
Aucun	72.9%	0.65%	17	10.2k
Module DA (*N=3, K=1,* $\alpha = 0.5$)	76.8%	1.0%	124	20.8k
RandAugment (*N=3, M=1*)	89.1%	0.2%	36	11.8k

d'autres, les bénéfices de la DA sont encore une fois visibles en comparant les entraînements des modèles sans DA aux entraînements avec DA (figure 3.13 et figures 3.14, 3.15, 3.16). Tandis que sans DA, les pertes de validation diminuent initialement puis augmentent rapidement, révélant un problème connu de sur-apprentissage ; avec DA, les pertes de validations suivent une décroissance plus longue et vers des valeurs bien inférieures. Cela pointe dans la direction de capacités de généralisation bien supérieures qui se reflètent dans les précisions des modèles (tableaux 3.2 et 3.3). De plus, là où la DA classique (fixe et réduites à quelques TF de base) tend seulement à rallonger le temps de décroissance de la perte de validation (et améliorer la

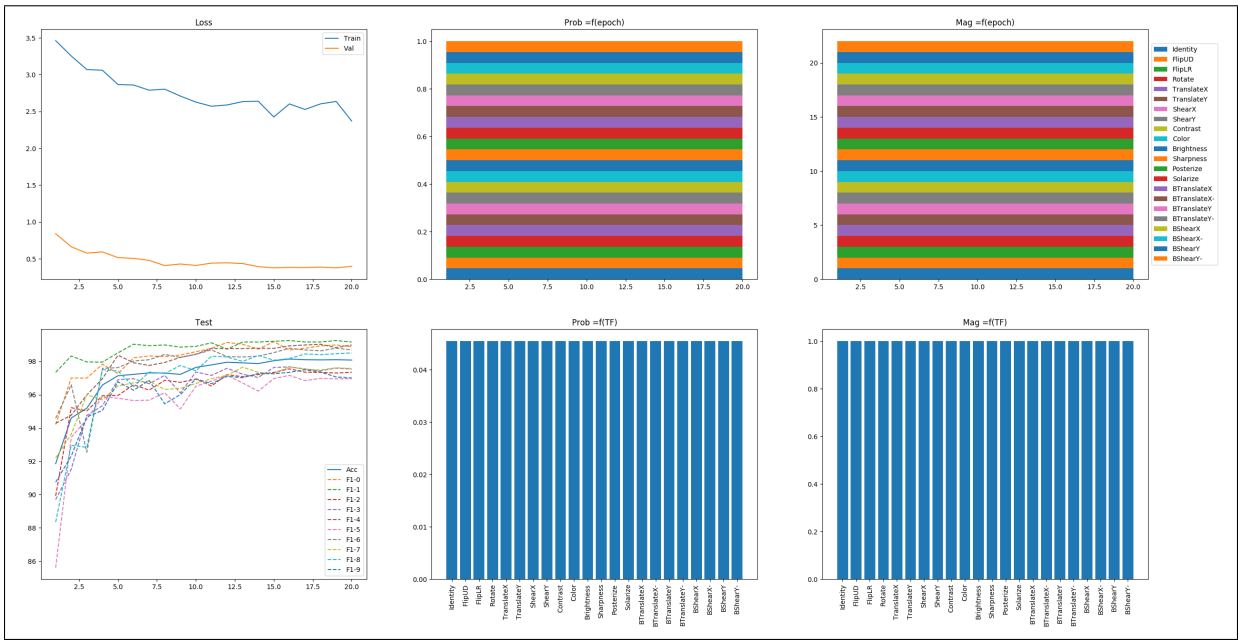

Figure 3.17 Résultat de LeNet sur MNIST avec RandAugment
(*N=3, TF=Mauvaise - I-3)*

Figure 3.18 Résultat de LeNet sur CIFAR10 avec RandAugment
(*N=3, TF=Mauvaise - I-3)*

Tableau 3.4 Comparaison résultats des systèmes de DA
(avec transformations néfastes)

Données	Modele	DA	Précision	Écart-type
MNIST	LeNet	Module DA (*N=3, K=1,* α = 0.5)	98.3%	0.1%
MNIST	LeNet	RandAugment (*N=3, M=1*)	97.7%	0.1%
MNIST	ResNet18	Module DA (*N=3, K=1,* α = 0.5)	98.7%	0.1%
MNIST	ResNet18	RandAugment (*N=3, M=1*)	98.7%	0.2%
CIFAR10	LeNet	Module DA (*N=3, K=1,* α = 0.5)	76.5%	0.3%
CIFAR10	LeNet	RandAugment (*N=3, M=1*)	72.9%	0.1%
CIFAR10	ResNet18	Module DA (*N=3, K=1,* α = 0.5)	78.1%	1.3%
CIFAR10	ResNet18	RandAugment (*N=3, M=1*)	85.7%	0.2%

généralisation) ; ici, RandAugment et le module de DA permettent une décroissance très longue de la perte de validation et semblent *prévenir le sur-apprentissage*, tout simplement. Cette observation est gage de la très grande diversité apportée aux TF par ces méthodes de DA par rapport aux méthodes classiques.

L'étude des résultats de cette étude (tableaux 3.2, 3.3) met en évidence que l'hypothèse selon laquelle *l'apprentissage de la DA est bénéfique semble erronée*. En effet, les performances rapportent par RandAugment sont bien supérieures a celles du module de DA, recherchant les paramètres optimaux.

L'apprentissage étant visiblement préjudiciable, malgré étant a priori capable d'éliminer les TF néfastes (section 3.3, il semblerait qu'il ne soit pas capable d'apprendre les TF menant aux meilleures performances. Plus précisément, la recherche des paramètres optimaux étant réalisée de manière locale (K=1), seuls les effets immédiatement visibles sur la perte de validation sont pris en compte. Les TF néfastes engendrant une corruption des données, elles ont un effet relativement immédiat sur les performances du modèle en les diminuant ou prévenant leurs augmentations. En revanche, une TF bénéfique aura sans doute un effet bien plus indirect et différé sur les performances. Rendant de prime abord l'entraînement plus difficile (diminution de la perte d'entraînement plus lente), mais permettant une meilleure généralisation grâce à la *diversité* qu'elle a apportée aux échantillons, à la longue. Cette corrélation entre diversité et

meilleure généralisation est aussi mise en avant par Cubuk *et al.* (b).
Avec cette perspective, l'apprentissage de la DA (particulièrement localement) tend à réduire la diversité apportée par les TF, en préférant certaines TF à d'autres et en limitant potentiellement les amplitudes de ces dernières. Cette hypothèse est encouragée quand on compare les résultats obtenus avec RandAugment (spécifiquement sur le tableau 3.2).

On pourra noter à partir des tableaux 3.2 et 3.3 que les ressources computationnelles (empreinte mémoire et temps d'apprentissage) requises par RandAugment et le module de DA respectent l'objectif fixé dans la section 1.1, en conservant dans l'ensemble le même ordre de grandeur.
Cependant, là où le module de DA prend maximum *7* fois plus de temps, RandAugment n'en prend que *2* fois plus. Cette différence s'explique aisément par les calculs supplémentaires requis pour le méta-apprentissage. Cela dit, la librairie utilise (*Higher* -Grefenstette *et al.*-) étant relativement récente, il n'est pas a exclure que le code nécessite optimisation. La même remarque peut être faite du code développé pour ce projet.
Concernant l'empreinte mémoire, il n'est pas surprenant de voir que le module de DA nécessite jusqu'à *2* fois plus de mémoire, car l'algorithme de méta-apprentissage (K-RMD) utilise un K de 1.

Pour terminer, la comparaison des performances de RandAugment et du module de DA avec les TF néfastes (tableau 3.4) permet de mène a une nouvelle hypothèse sur les propriétés de l'apprentissage. En effet, il semble que dans la majorité des cas étudiés l'apprentissage de la DA ne soit pas bénéfique pour le modèle. En revanche, cela semble différent, dans les cas de MNIST et avec l'utilisation de modèle de faible capacité tel que LeNet.
L'apprentissage de la DA avec MNIST semble mener, dans le pire cas, a des performances similaires à RandAugment. Une explication de cette particularité peut venir du fait que MNIST, à la différence de CIFAR10, propose des classes qui *ne sont pas toutes mutuellement exclusives*. La distribution de toutes les instances de '6' se superposant a la distribution des '9', par exemple. Ainsi, si l'on peut voir la DA comme une manière d'étendre les distributions de données dont ondispose ; l'apprentissage de la DA permet de l'étendre en essayant d'éviter les superpositions

de distribution qui aurait pour conséquence la *corruption des labels*. Tandis que RandAugment ne prend pas en compte ces superpositions, et risque de transformer un échantillon d'une classe vers une classe différente, sans changer de label. Cela dit, cette hypothèse reste encore à vérifier, car, si l'on pourrait penser que les classes '6' et '9' de MNIST serait les plus atteintes, l'étude des scores F1 des figures 3.17 et 3.18 ne pointe pas spécifiquement ces classes, mais plutôt les classes '2', '3' et '5'.
Une seconde hypothèse semble ressortir de ses résultats, *l'augmentation de données peut avoir des effets plus néfastes sur les modèles à faibles capacités*. On peut remarquer que les résultats sur RandAugment engendrent une plus grande baisse de performance sur LeNet (faible capacité) que sur ResNet18 (plus grande capacité). Dans la perspective où la capacité d'un modèle définit la complexité des distributions qu'il est capable de modéliser, on peut comprendre que l'apprentissage de la DA, plus sélective que RandAugment, soit plus performant sur un modèle tel que LeNet.

3.6 Discussion

Les sections précédentes ont permis de montrer que l'apprentissage de la DA n'est pas trivial et mérite de rester un domaine de recherche actif.

La section 3.3 a pu mettre en évidence que le manque de retours sur l'effet des paramètres rendait le processus de méta-apprentissage particulièrement difficile.
La sélection stochastique des TF et de leurs amplitudes pour chaque échantillon de données amène plus de diversités dans la DA, mais aussi du bruit, une incertitude sur le choix des paramètres. Une TF avec une certaine amplitude peut être néfaste sur une image, mais bénéfique sur une autre ; une amplitude différente, quant à elle, peut amener un résultat tout autre.
L'évolution constante des performances du modèle ne facilite pas non plus les choses. L'effet des variations des HP optimise par K-RMD (sections 1.2.2.2 et 2.1.2) n'étant souvent pas direct. Les TF corrompant les données que ce soit au niveau des images (section 3.3.2) ou des labels (section 3.5.2) ont un effet relativement direct sur les performances. En revanche, les

TF bénéfiques, et les HP de manière générale, ont des effets beaucoup plus long terme. C'est pourquoi les premières méthodes de méta-apprentissage (section 1.2.2) tendent à réaliser des mises à jour de ces paramètres seulement après des entraînements complets. Tandis que K-RMD n'est qu'une *approximation* de ces méthodes exactes, avec un champ de vision bien plus réduit sur l'entraînement. Réduite encore par la faible valeur de K sélectionné (voir section 3.4).

Ensuite, tandis que la section 3.5 a mis en avant l'hypothèse que l'*apprentissage de la DA réduisait la diversité* et par conséquent les performances des modèles, on ne peut s'empêcher de penser au revirement des auteurs d' AutoAugment -Cubuk *et al.* (a)- et RandAugment -Cubuk *et al.* (b)-. Avec la publication d' AutoAugment proposant une méthode d'apprentissage de la DA, une série de publication, elles aussi proposant d'*apprendre* la DA, ont suivies (voir section 1.2.1). Pourtant, RandAugment égale ou dépasse toutes les méthodes précédentes et ceux avec un apprentissage dérisoire de la DA, seulement basée sur une recherche par quadrillage de N et M (voir section 3.4). Cependant, les auteurs de RandAugment mettent l'emphase sur le fait que c'est l'apprentissage disjoint de la DA et du modèle qui est à l'origine de ces différences de performance ; et pas spécifiquement, l'apprentissage de la DA tout court.

Enfin, la section 3.5 met aussi en exergue une théorie qui n'a pas été réellement étudiée, a notre connaissance. Théorie selon laquelle *l'augmentation de donnée rehausse la capacité requise par un réseau pour modéliser des données*. Comme souligne par Frankle & Carbin, les modèles utilisent en apprentissage profond tendent a être très largement en surcapacité par rapport aux complexités des problèmes. Cette surcapacité serait la raison pour laquelle, dans la majorité des cas, plus la *diversité* apportée par la DA est grande, meilleures sont les performances. En revanche, dans des cas extrêmes, où les modèles ont de fortes contraintes de capacités ou des données très complexes a modélisé, il est probable que l'apprentissage de la DA soit plus avisée qu'une méthode telle que RandAugment.

Toutes ces observations nous amènent à penser que le domaine de la DA est encore en pleine transformation et que la recherche dans ce domaine a encore beaucoup de potentiel !

CONCLUSION ET RECOMMANDATIONS

4.1 Conclusion

L'augmentation de donnée est capitale pour tirer les meilleures performances des modèles utilisés en apprentissage profond, particulièrement dans le domaine de la vision par ordinateur. Elle requiert cependant des ressources non négligeables en termes humains (expertise sur la tache) et/ou computationnels (apprentissage automatique de l'augmentation de donnée). L'objectif de ce projet était de réduire, voir supprimer, ces limitations.

La solution développée pour ce projet est le résultat d'une étude approfondie de la littérature combinant les domaines de l'augmentation de donnée et du méta-apprentissage. Un module d'augmentation de donnée capable d'apprendre conjointement avec le modèle constitue l'aboutissement du développement de cette solution.
Une dizaine de transformations d'images ont été redéfinies afin de pouvoir apprendre indépendamment leurs différents paramètres ; et un algorithme de méta-apprentissage (*K-RMD*), minimisant les ressources computationnelles requises, intégrées.
De plus, ce module est capable, amputé de sa composante de méta-apprentissage, de reproduire l'approche de RandAugment ; l'état de l'art en augmentation de donnée paru au cours du projet.

Les expériences réalisées ont permis de montrer qu'avec les mêmes ressources computationnelles et sans expertise particulière, il est possible d'obtenir des performances équivalentes ou supérieures à l'état de l'art avec notre module d'augmentation de donnée. Les résultats obtenus laissent envisager que le module développé serait particulièrement efficace pour des *applications embarquées* ou des *données avec une grande complexité*.
La recherche effectuée a notamment permis de mettre en avant des difficultés liées à l'appren-

tissage de l'augmentation de donnée, ainsi que de nouvelles théories sur le fonctionnement de l'augmentation de donnée qu'il conviendrait d'approfondir dans de prochaine recherche.

4.2 Recommandations

Les expériences réalisées ainsi que leurs analyses, dans la section 3.6, ont révélé de nouvelles pistes de recherche qui mériteraient d'être explorées :

- Approfondir l'hypothèse selon laquelle *l'augmentation de donnée rehausse la capacité requise par un réseau pour modéliser des distributions de données* en utilisant des données plus complexes que celles de ce projet (ImageNet -Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg & Fei-Fei-, par exemple) et d'autres modèles a faibles capacités.
- Confirmer l'hypothèse selon laquelle l'*apprentissage de l'augmentation de donnée tend à réduire sa diversité* avec des algorithmes de méta-apprentissage différents (voir chapitre 1).
- Expérimenter le découplage du nombre d'états utilisés par l'algorithme de méta-apprentissage, K-RMD, et la fréquence de mise à jour des hyper paramètres. Il serait alors possible de conserver une haute fréquence de mise à jour tout en ayant une vision plus large sur l'entraînement et l'effet des hyper paramètres sur ce dernier (voir section 3.4.2.2 pour plus de détails).
- Explorer l'efficacité de la présélection des transformations par l'apprentissage en amont d'un système basé sur RandAugment. L'apprentissage de l'augmentation s'étant révélé particulièrement précieux pour éliminer les transformations néfastes (voir section 3.3).
- Étudier l'apprentissage joint d'hyper paramètres supplémentaires. L'algorithme de méta-apprentissage implémenté (K-RMD) réalise déjà l'apprentissage de paramètres basé sur les données de validations. Il est donc possible d'apprendre d'autres hyper paramètres que ceux lies a l'augmentation de données pour un coût computationnel supplémentaire négligeable.

- Tester la combinaison des algorithmes K-RMD et DrMAD (voir section 1.2.2.2) afin de réduire encore l'empreinte mémoire du méta-apprentissage. Cette combinaison reviendrait à des approximations supplémentaires et compliquerait alors davantage l'apprentissage, déjà complexe (voir section 3.3.2).

ANNEXE I

CONFIGURATIONS DES TRANSFORMATIONS

Tableau-A I-1 Configuration de transformations de base

Nom	Fonction	Axe	Min	Max	Échelle
Identity	identity				
FlipUD	flip	Y			
FlipLR	flip	X			
Rotate	rotate		-30°	30°	Linéaire
TranslateX	translate	X	-33%	33%	Linéaire
TranslateY	translate	Y	-33%	33%	Linéaire
ShearX	shear	X	-0.3	0.3	Linéaire
ShearY	shear	Y	-0.3	0.3	Linéaire
Contrast	contrast		0.1	1.9	Linéaire
Color	color		0.1	1.9	Linéaire
Brightness	brightness		0.1	1.9	Linéaire
Sharpness	sharpness		0.1	1.9	Linéaire
Posterize	posterize		4 bits	8 bits	Linéaire
Solarize	solarize		1	256	Linéaire

Tableau-A I-2 Configuration de transformations large avec échelles inversées

Nom	Fonction	Axe	Min	Max	Échelle
Identity	identity				
FlipUD	flip	Y			
FlipLR	flip	X			
Rotate	rotate		-180°	180°	Linéaire
TranslateX	translate	X	-50%	50%	Linéaire
TranslateY	translate	Y	-50%	50%	Linéaire
ShearX	shear	X	-1.0	1.0	Linéaire
ShearY	shear	Y	-1.0	1.0	Linéaire
+Contrast	contrast		1.0	1.9	Linéaire
+Color	color		1.0	1.9	Linéaire
+Brightness	brightness		1.0	1.9	Linéaire
+Sharpness	sharpness		1.0	1.9	Linéaire
-Contrast	contrast		0.1	1.0	Inverse
-Color	color		0.1	1.0	Inverse
-Brightness	brightness		0.1	1.0	Inverse
-Sharpness	sharpness		0.1	1.0	Inverse
Posterize	posterize		4 bits	8 bits	Inverse
Solarize	solarize		1	256	Inverse

Tableau-A I-3 Configuration de transformations de base avec mauvaises transformations

Nom	Fonction	Axe	Min	Max	Échelle
Identity	identity				
FlipUD	flip	Y			
FlipLR	flip	X			
Rotate	rotate		-30°	30°	Linéaire
TranslateX	translate	X	-33%	33%	Linéaire
TranslateY	translate	Y	-33%	33%	Linéaire
ShearX	shear	X	-0.3	0.3	Linéaire
ShearY	shear	Y	-0.3	0.3	Linéaire
Contrast	contrast		0.1	1.9	Linéaire
Color	color		0.1	1.9	Linéaire
Brightness	brightness		0.1	1.9	Linéaire
Sharpness	sharpness		0.1	1.9	Linéaire
Posterize	posterize		4 bits	8 bits	Linéaire
Solarize	solarize		1	256	Linéaire
BTranslateX	translate	X	75%	95%	Linéaire
BTranslateX-	translate	X	-95%	-75%	Linéaire
BTranslateY	translate	Y	75%	95%	Linéaire
BTranslateY-	translate	Y	-95%	-75%	Linéaire
BShearX	shear	X	0.9	1.2	Linéaire
BShearX-	shear	X	-1.2	-0.9	Linéaire
BShearY	shear	Y	0.9	1.2	Linéaire
BShearY-	shear	Y	-1.2	-0.9	Linéaire

ANNEXE II

RÉSULTAT DE LA RECHERCHE D'HYPER PARAMÈTRES DU MODULE D'AUGMENTATION DE DONNÉE

Les résultats du tableau II-1 sont obtenus sur 3 entraînements par configuration avec un module de DA (*TF=Base - I-1, K=1*) et un ResNet18 sur CIFAR10 (50%/50% données entraînements/validations) pour 200 époques.

L'interprétation de ces résultats et des détails sur les configurations présentées sont disponibles dans la section 3.4.

Tableau-A II-1 Résultat de la recherche d'hyper paramètres du module d'augmentation de donnée

N	α	**Gel des amplitudes**	**Précision**	**Écart-type**
2	0	Oui	74.3%	0.2%
2	0	Non	73.3%	0.6%
2	0.3	Oui	74.0%	0.3%
2	0.3	Non	73.8%	0.2%
2	0.5	Oui	74.1%	0.1%
2	0.5	Non	73.4%	0.1%
2	0.8	Oui	74.3%	0.2%
2	0.8	Non	73.2%	0.3%
2	1	Oui	73.2%	0.7%
2	1	Non	72.9%	0.5%
3	0	Oui	74.4%	0.4%
3	0	Non	74.3%	0.4%
3	0.3	Oui	74.0%	0.2%
3	0.3	Non	74.5%	0.1%
3	0.5	Oui	74.3%	0.5%
3	0.5	Non	74.7%	0.2%
3	0.8	Oui	74.3%	0.4%
3	0.8	Non	74.2%	0.2%
3	1	Oui	75.2%	0.9%
3	1	Non	73.6%	1.0%

Tableau-A II-2 Résultat de la recherche d'hyper paramètres du module d'augmentation de donnée

N	α	**Gel des amplitudes**	**Précision**	**Écart-type**
4	0	Oui	73.9%	0.6%
4	0	Non	74.1%	0.2%
4	0.3	Oui	74.4%	0.2%
4	0.3	Non	74.3%	0.5%
4	0.5	Oui	74.2%	0.5%
4	0.5	Non	74.8%	0.4%
4	0.8	Oui	74.5%	0.25%
4	0.8	Non	74.2%	0.7%
4	1	Oui	74.5%	0.3%
4	1	Non	74.9%	0.3%
5	0	Oui	37.5%	0.0%
5	0	Non	36.7%	
5	0.3	Oui	37.2%	0.1%
5	0.3	Non	36.2%	
5	0.5	Oui	37.1%	0.3%
5	0.5	Non	36.8%	
5	0.8	Oui	36.7%	0.2%
5	0.8	Non	36.5%	
5	1	Oui	37.2%	0.95%
5	1	Non	34.1%	

LISTE DE RÉFÉRENCES

Antoniou, A., Storkey, A. & Edwards, H. Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks. *Artificial Neural Networks and Machine Learning – ICANN 2018*, (Lecture Notes in Computer Science), 594–603.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M. & Wood, F. Online Learning Rate Adaptation with Hypergradient Descent. Repéré à http://arxiv.org/abs/1703.04782.

Bergstra, J., Bardenet, R., Kégl, B. & Bengio, Y. Algorithms for Hyper-Parameter Optimization.

Chandra, K., Meijer, E., Andow, S., Arroyo-Fang, E., Dea, I., George, J., Grueter, M., Hosmer, B., Stumpos, S., Tempest, A. & Yang, S. Gradient Descent : The Ultimate Optimizer. Repéré à http://arxiv.org/abs/1909.13371.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. & Le, Q. V. AutoAugment : Learning Augmentation Policies from Data. Repéré à http://arxiv.org/abs/1805.09501.

Cubuk, E. D., Zoph, B., Shlens, J. & Le, Q. V. RandAugment : Practical data augmentation with no separate search. Repéré à http://arxiv.org/abs/1909.13719.

Feurer, M. & Hutter, F. Hyperparameter Optimization. Dans Hutter, F., Kotthoff, L. & Vanschoren, J. (Éds.), *Automated Machine Learning : Methods, Systems, Challenges* (pp. 3–33). Springer International Publishing. doi : 10.1007/978-3-030-05318-5_1.

Finn, C., Abbeel, P. & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. Repéré à http://arxiv.org/abs/1703.03400.

Franceschi, L., Donini, M., Frasconi, P. & Pontil, M. Forward and Reverse Gradient-Based Hyperparameter Optimization. *International Conference on Machine Learning*, pp. 1165–1173. Repéré à http://proceedings.mlr.press/v70/franceschi17a.html.

Frankle, J. & Carbin, M. The Lottery Ticket Hypothesis : Finding Sparse, Trainable Neural Networks. Repéré à http://arxiv.org/abs/1803.03635.

Fu, J., Luo, H., Feng, J., Low, K. H. & Chua, T.-S. DrMAD : Distilling Reverse-Mode Automatic Differentiation for Optimizing Hyperparameters of Deep Neural Networks. Repéré à http://arxiv.org/abs/1601.00917.

Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K. & Chintala, S. Generalized Inner Loop Meta-Learning. Repéré à http://arxiv.org/abs/1910.01727.

He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. Repéré à http://arxiv.org/abs/1512.03385.

Ho, D., Liang, E., Stoica, I., Abbeel, P. & Chen, X. Population Based Augmentation : Efficient Learning of Augmentation Policy Schedules. Repéré à http://arxiv.org/abs/1905.05393.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C. & Kavukcuoglu, K. Population Based Training of Neural Networks. Repéré à http://arxiv.org/abs/1711.09846.

Jules, M. S. Experiments With Scalable Gradient-based Hyperparameter Optimization for Deep Neural Networks. 56.

Kodali, N., Abernethy, J., Hays, J. & Kira, Z. On Convergence and Stability of GANs. Repéré à http://arxiv.org/abs/1705.07215.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. 86(11), 2278–2324. doi : 10.1109/5.726791.

Lemley, J., Bazrafkan, S. & Corcoran, P. Smart Augmentation - Learning an Optimal Data Augmentation Strategy. 5, 5858–5869. doi : 10.1109/ACCESS.2017.2696121.

Li, C., Xu, K., Zhu, J. & Zhang, B. Triple Generative Adversarial Nets. Repéré à http://arxiv.org/abs/1703.02291.

Lim, S., Kim, I., Kim, T., Kim, C. & Kim, S. Fast AutoAugment. Repéré à http://arxiv.org/abs/1905.00397.

Luketina, J., Berglund, M., Greff, K. & Raiko, T. Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters. Repéré à http://arxiv.org/abs/1511.06727.

MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D. & Grosse, R. Self-Tuning Networks : Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. Repéré à http://arxiv.org/abs/1903.03088.

Maclaurin, D., Duvenaud, D. & Adams, R. P. Gradient-based Hyperparameter Optimization through Reversible Learning. Repéré à http://arxiv.org/abs/1502.03492.

Mirza, M. & Osindero, S. Conditional Generative Adversarial Nets. Repéré à http://arxiv.org/abs/1411.1784.

Pedregosa, F. Hyperparameter optimization with approximate gradient. Repéré à http://arxiv.org/abs/1602.02355.

Perez, L. & Wang, J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Repéré à http://arxiv.org/abs/1712.04621.

Ratner, A. J., Ehrenberg, H. R., Hussain, Z., Dunnmon, J. & Ré, C. Learning to Compose Domain-Specific Transformations for Data Augmentation. Repéré à http://arxiv.org/abs/1709.01643.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E. & Bradski, G. Kornia : an Open Source Differentiable Computer Vision Library for PyTorch. Repéré à http://arxiv.org/abs/1910.02190.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. Repéré à http://arxiv.org/abs/1409.0575.

Shaban, A., Cheng, C.-A., Hatch, N. & Boots, B. Truncated Back-propagation for Bilevel Optimization. Repéré à http://arxiv.org/abs/1810.10667.

Snoek, J., Larochelle, H. & Adams, R. P. Practical Bayesian Optimization of Machine Learning Algorithms. Repéré à http://arxiv.org/abs/1206.2944.

Tran, T., Pham, T., Carneiro, G., Palmer, L. & Reid, I. A Bayesian Data Augmentation Approach for Learning Deep Models. Repéré à http://arxiv.org/abs/1710.10564.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. 8(3), 229–256. doi : 10.1007/BF00992696.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T. & Le, Q. V. Unsupervised Data Augmentation for Consistency Training. Repéré à http://arxiv.org/abs/1904.12848.

Zagoruyko, S. & Komodakis, N. Wide Residual Networks. Repéré à http://arxiv.org/abs/1605.07146.

Zhang, X., Wang, Z., Liu, D. & Ling, Q. DADA : Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification. Repéré à http://arxiv.org/abs/1809.00981.