# Évaluation réaliste de l'apprentissage *Few-Shot* transductif

par

Olivier VEILLEUX

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE
M. Sc. A.

MONTRÉAL, LE 10 MAI, 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

**PRÉSENTATION DU JURY**

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Ismail Ben Ayed, directeur de mémoire
Département de génie des systèmes, École de technologie supérieure

M. Marco Pedersoli, président du jury
Département de génie des systèmes, École de technologie supérieure

M. José Dolz, membre du jury
Département de génie logiciel et des TI, École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 20 AVRIL, 2022

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

**REMERCIEMENTS**

Je tiens d'abord à remercier mon directeur de recherche, Professeur Ismail Ben Ayed, pour son soutien, sa confiance et la motivation qu'il m'a amenés tout au long de la réalisation de ma maîtrise. Grâce à son aide, son expertise et ses conseils, j'ai su me développer professionnellement et acquérir des connaissances et des outils qui me seront utiles tout au long de ma carrière d'ingénieur.

Merci à tous les professeurs et collègues que j'ai côtoyés et qui ont contribué à mon développement au cours de mon cheminement universitaire à l'École de technologie supérieure et au LIVIA.

Merci à mes amis, notamment Guy, Jonathan, Mark et Simon qui me sont très chers et occupent une place très spéciale dans ma vie. Merci à vous pour l'inspiration, les conseils, les encouragements et le support que vous m'amenez. Merci pour tous les fous rires, souvenirs et beaux moments que nous avons partagés ensemble et pour ceux à venir.

Enfin, un gros merci à famille et spécialement à mes parents, Nathalie et Simon pour leur amour inconditionnel. Ils m'ont toujours soutenu, inspiré, et encouragé dans mon cheminement scolaire, mes activités et projets. Merci de m'avoir appris à être persévérant et résilient, ce malgré tous les obstacles que la vie peut nous présenter. Merci de m'avoir transmis toutes les bonnes valeurs qui font de moi le jeune homme que je suis aujourd'hui.

**Évaluation réaliste de l'apprentissage *Few-Shot* transductif**

Olivier VEILLEUX

## RÉSUMÉ

La communauté de l'apprentissage *Few-Shot* porte un grand intérêt de recherche pour les méthodes transductives. Ces méthodes posent leur inférence en exploitant à la fois les données annotées de la base de support et celles non-annotées de la base de *query*. Elles ont su surpassé de manière significative les performances des méthodes inductives. Nous remettons en question les performances de ces méthodes par le fait que le processus d'évaluation actuel n'est pas adéquat. Les méthodes sont évaluées en fonction de leur capacité de généralisation afin de classifier correctement les données de la base de *query*. Dans la littérature actuelle, la distribution des classes des données *query* est uniforme. Le fait de connaître au préalable la distribution des classes des données à classifier ne reflète pas le défi d'un problème de classification réaliste. De plus, l'inférence de certaines méthodes transductives s'effectuent en exploitant cette fausse information a priori. Afin de générer des tâches de classification aléatoires, nous proposons d'échantillonner les probabilités marginales des classes de la base de *query* en suivant la distribution de Dirichlet. Nous avons évaluées les méthodes de l'état de l'art sur trois bases de données de référence : *mini*-Imagenet, *tiered*-Imagenet et *Caltech-UCSD Birds 200* (CUB) et en utilisant deux architectures de réseau de neurones : ResNet-18 et WRN28-10. Toutes les méthodes transductives souffrent de perte de performances lorsqu'évaluées à l'aide de tâches aléatoirement distribuées. Il est également intéressant d'observer que les méthodes les plus performantes dans la littérature de l'apprentissage *Few-Shot* actuelle sont celles qui souffrent le plus de notre scénario de test, soit en subissant des pertes de performances allant jusqu'à près de 20%. Nous proposons une méthode transductive ($\alpha$-TIM) pour s'attaquer à ce nouveau problème de classification réaliste. Nous utilisons la divergence-$\alpha$ pour généraliser la fonction de coût exploitant l'information mutuelle. Cette forme de divergence est robuste en faisant face à différentes sévérités de déséquilibre de classes. De plus, nos résultats expérimentaux montrent que notre méthode surpassent les performances de toutes les méthodes de l'état de l'art évaluées. Le code de ce projet se trouve dans le répertoire GitHub suivant : https://github.com/oveilleux/Realistic_Transductive_Few_Shot.

**Mots-clés:** apprentissage *Few-Shot*, déséquilibre des classes, distribution de Dirichlet, divergence-$\alpha$, inférence transductive

**Realistic Evaluation of Transductive Few-Shot Learning**

Olivier VEILLEUX

## ABSTRACT

The Few-Shot learning community has a strong research interest in transductive methods. Transductive inferance exploits both labeled data from the support set and unlabeled data from the query set. These methods were able to significantly surpass the performance of inductive methods. We question the performances of these methods based on the fact that the current evaluation process might not be adequate. The methods are evaluated based on their ability to generalize in order to correctly classify the query set's data. In the current literature, the query set's class distribution is uniform. Knowing beforehand the class distribution of the data to be classified does not reflect the challenge of a realistic classification problem. Moreover, the inference of certain transductive methods is carried out by exploiting this false a priori information. In order to produce random classification tasks, we propose to sample the marginal probabilities of the query set's classes by following the Dirichlet's distribution. We evaluated state-of-the-art methods on three benchmarks : *mini*-Imagenet, *tiered*-Imagenet, and *Caltech-UCSD Birds 200* (CUB) and using two neural network architectures : ResNet-18 and WRN28-10. All transductive methods suffer from performance loss when evaluated on randomly distributed tasks. It's also interesting to observe that the best performing methods in the current FSL literature are the ones that suffer the most from this test scenario, undergoing performance losses of up to almost 20%. We propose a transductive method ($\alpha$-TIM) to tackle this new realistic classification problem. We use the $\alpha$-divergence to generalize the loss function exploiting mutual information. This divergence's shape is appropriate to deal with different severities of class imbalance. Moreover, our experimental results demonstrate that our method outperforms all corresponding state-of-the-art methods. The code of this project can be found in the following GitHub repository : https://github.com/oveilleux/Realistic_Transductive_Few_Shot.

**Keywords:** Few-Shot Learning, Class Imbalance, Dirichlet's Distribution, $\alpha$-Divergence, Transductive Inference

## TABLE DES MATIÈRES

Page

## LISTE DES TABLEAUX

Page

## LISTE DES FIGURES

Page

## LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

| | |
|---|---|
| BN | Normalisation de batch |
| CUB | Caltech-UCSD Birds 200 |
| CE | Entropie croisée |
| CNN | Réseau de neurones à convolutions |
| ETS | École de Technologie Supérieure |
| FSL | Apprentissage *Few-Shot* (*Few-Shot Learning*) |
| KL | Kullback-Leibler |
| RN | Residual Network |
| TBN | Normalisation de batch transductive |
| WRN | Wide Residual Network |

## LISTE DES SYMBOLES ET UNITÉS DE MESURE

| | |
|---|---|
| $\mathcal{A}$ | Apprentissage |
| $\mathcal{B}$ | Batch de données |
| $\mathcal{B}_{\mathcal{S}}$ | Batch de données de support |
| $\mathcal{B}_{\mathcal{Q}}$ | Batch de données *query* |
| $\mathcal{D}_{base}$ | Base de données d'entraînement |
| $\mathcal{D}_{test}$ | Base de données de test |
| $\mathcal{D}_{val}$ | Base de données de validation |
| $f_{\theta}$ | Modèle pré-entraîné |
| $f_{\boldsymbol{W}}$ | Classificateur |
| $\mathcal{H}(Y)$ | Entropie marginale de $Y$ |
| $\mathcal{H}(Y\|X)$ | Entropie conditionnelle de $Y$ en sachant $X$ |
| $\mathcal{I}(X;Y)$ | Information mutuelle entre $X$ et $Y$ |
| $\mathcal{L}$ | Fonction de coût |
| $n_k^{\mathcal{Q}}$ | Nombre de données de données de *query* par classe |
| $\mathbf{p}_i$ | Vecteur de probabilité d'assignation de classe de la donnée $i$ |
| $\mathcal{Q}$ | Base de *query* |
| $\mathcal{S}$ | Base de support |
| $\mathcal{T}$ | Tâche de classification *Few-Shot* |
| $\boldsymbol{W}$ | Poids du classificateur |
| $\mathcal{X}$ | Donnée |
| $\mathcal{X}_{\mathcal{Q}}$ | Donnée de la base de *query* |
| $\mathcal{X}_{\mathcal{S}}$ | Donnée de la base de support |
| $\mathcal{Y}_{base}$ | Classes des données de la base d'entraînement |

| | |
|---|---|
| $\mathcal{Y}_{test}$ | Classes des données de la base de test |
| $\mathcal{Y}_{val}$ | Classes des données de la base de validation |
| $\mathcal{Y}_{\mathcal{Q}}$ | Classes des données de la base de *query* |
| $\mathcal{Y}_{\mathcal{S}}$ | Classes des données de la base de support |
| $\boldsymbol{y}_i$ | Vecteur encodé sous forme *one-hot* contenant la classe véritable de la donnée $i$ |
| $\mathcal{Z}$ | Donnée représentée sous forme de vecteur de caractéristiques |
| $\mathbb{1}_K$ | Vecteur unitaire à $K$ dimensions |

# INTRODUCTION

Le développement de l'apprentissage machine a amené à l'utilisation d'algorithmes de classification d'images dans plusieurs contextes. Par exemple, la classification d'images par ordinateur peut être utilisée dans le domaine médical afin d'émettre des diagnostiques préliminaires, dans le domaine de la sécurité et surveillance pour l'identification d'individus, dans le domaine industriel afin d'assurer la qualité des produits, etc. Récemment, les modèles de classifications d'images offrent des performances très précises. La majorité de ces modèles sont de réseaux de neurones à convolutions (CNN) qui se basent sur le principe de l'apprentissage profond. Un entraînement extensif est requis afin que ces modèles puissent offrir des performances de classification satisfaisantes. Pour se faire, ces derniers sont généralement entraînés de manière supervisée à l'aide d'une quantité énorme de données annotées. Les modèles sont très efficaces pour effectuer la classification d'images appartenant aux classes rencontrées lors de l'entraînement. Par contre, ces derniers éprouvent de la difficulté à classifier des données appartenant à de nouvelles classes (jamais vues lors de l'entraînement). Il serait contre-productif de considérer le réentraînement complet des modèles afin que ces derniers soient en mesure de reconnaître de nouvelles classes de données. Cela engendrerait non seulement des coûts énormes pour l'acquisition et l'annotation de nouvelles données, en temps d'entraînement et ressources informatiques de calculs (GPU), mais aussi en efforts d'annotations fournis par des humains.

L'apprentissage *Few-Shot* (FSL) (Fei-Fei, Fergus & Perona (2006); Miller, Matsakis & Viola (2000); Vinyals, Blundell, Lillicrap, Kavukcuoglu & Wierstra (2016)) propose une solution à ce problème. Les concepts du FSL sont inspirés de la manière dont les humains apprennent. L'apprentissage de nouvelles notions par les humains s'effectue en exploitant les connaissances déjà acquises. En seulement quelques exemples et en tissant des liens avec notre bagage de connaissances, il nous est possible d'assimiler et de comprendre de nouveaux concepts. C'est ce que le FSL tente d'appliquer pour l'apprentissage effectué par ordinateur. La littérature propose l'adaptation des modèles préentraînés en utilisant seulement quelques exemples de données

annotées appartenant à de nouvelles classes. En d'autres termes, le FSL tente de généraliser les modèles à l'aide d'un niveau de supervision très faible.

La capacité d'adaptation des méthodes est évaluée à l'aide de tâches $\mathcal{T}$ composées de la base de support $\mathcal{S}$ contenant les exemples annotés et de la base de *query* $\mathcal{Q}$ contenant les exemples non annotés à classifier. La métrique permettant de mesurer leur performance est le taux de précision de classification (le nombre d'exemples bien classifié sur le nombre d'exemples total). Dans la méthode d'évaluation actuelle, les données à classifier sont représentées de manière uniforme parmi les différentes classes possibles. Prenons comme exemple un contexte où les classes sont : chien, chat et cheval. La distribution uniforme des données à classifier signifie qu'il y a le même nombre d'exemples parmi chaque classe, par exemple 3 images de chiens, 3 de chats et 3 de chevaux. Cette distribution uniforme des données à classifier ne représente pas un problème de classification réel. Il est clairement non réaliste de prétendre que nous allons rencontrer à occurrence égale les différentes classes de données.

Dans les dernières années, les efforts de recherches ont mené à des méthodes FSL de plus en plus performantes. Certaines recherches se sont tournées vers les méthodes transductives (Finn, Abbeel & Levine (2017); Yanbin *et al.* (2019); Liu, Song & Qin (2020a); Qiao *et al.* (2019); Boudiaf *et al.* (2020); Hou, Chang, Bingpeng, Shan & Chen (2019); Dhillon, Chaudhari, Ravichandran & Soatto (2020); Guo & Cheung (2020); Hu *et al.* (2020a); Hu, Gripon & Pateux (2020b); Liu, Schiele & Sun (2020b); Wang, Xu, Liu, Zhang & Fu (2020); Yang *et al.* (2020); Ziko, Dolz, Granger & Ben Ayed (2020)) qui proposent d'exploiter à la fois les données annotées et les données non annotées lors de leur adaptation. Comme ces méthodes utilisent les données $\mathcal{Q}$, certaines d'entre elles exploitent la fausse information a priori à propos de l'uniformité des classes des données à classifier. De plus, certaines méthodes devenues immensément populaires et ayant inspiré de nombreuses autres solutions (Gordon, Bronskill, Bauer, Nowozin & Turner

(2019); Zhen *et al.* (2020)), par exemple MAML (Finn *et al.* (2017)) exploitent cette fausse assomption par l'utilisation de la normalisation de batch transductive (TBN).

Bien que les méthodes atteignent des performances de classification satisfaisantes, il est important de se questionner si leurs performances sont légitimes et représentatives d'un problème de classification réaliste. L'hypothèse que les données à classifier sont distribuées uniformément parmi les classes ne correspond pas à un problème de classification réel ou la répartition des classes est inconnue. De plus, il est à se demander si les bonnes performances des méthodes transductives sont en partie dues à l'environnement de test actuel. Avec l'engouement alloué aux méthodes transductives, il est à se demander si la procédure de test pour la courante littérature FSL est adéquate. Ce problème d'hypothèse irréaliste dans la procédure de test peut sembler bénin, mais ce dernier peut amener les recherches vers de fausses conclusions.

Ce projet de recherche vise à remettre en question les procédures de tests de la littérature FSL actuelle. Pour ce faire, nous proposons une méthode d'échantillonnage aléatoire qui permet de générer des tâches de classification dont la répartition des données parmi les classes est inconnue au préalable. Afin de valider notre hypothèse, nous effectuons une évaluation des méthodes de l'art en suivant notre procédure de test réaliste. De plus, nous proposons une méthode qui exploite les divergences-$\alpha$ afin d'attaquer ce nouveau problème de classification.

Le premier chapitre introduit et explique plus en détail le problème d'apprentissage *Few-Shot* pour la classification d'images. Ce chapitre comporte également une revue de la littérature des méthodes s'attaquant au problème de classification FSL. Nous y présentons aussi les différents types d'entraînement, soit le méta-apprentissage et l'apprentissage par transfert, utilisés dans la littérature FSL. De plus, nous expliquons les distinctions entre les méthodes inductives et transductives.

Le deuxième chapitre introduit la problématique de manière plus formelle et motive ce courant projet de recherche. Nous y présentons une étude sur deux méthodes des plus compétitives dans la littérature, et dont l'implémentation exploite le biais de la procédure de test utilisée couramment.

Le troisième chapitre présente notre méthode d'échantillonnage de tâches qui exploitent les concepts des distributions de Dirichlet. Notre méthode est également comparée à d'autres formulations d'échantillonnage utilisées dans la littérature actuelle.

Nous introduisons dans le quatrième chapitre notre méthode $\alpha$-TIM qui est une généralisation de la méthode TIM (Boudiaf *et al.* (2020)). Nous proposons de modifier la formulation du terme de l'information mutuelle par l'utilisation de la divergence-$\alpha$ de Tsallis (Tsallis (1988)).

Nous présentons au cinquième chapitre nos protocoles et résultats expérimentaux. Nous y affichons les performances de classification des méthodes de l'état de l'art et de notre méthode $\alpha$-TIM lorsqu'évaluées dans l'environnement de test réaliste.

La dernière section de ce mémoire est la conclusion qui résume ce travail de recherche et où nous suggérons de nouvelles pistes de recherche.

Ce travail de maîtrise a mené à une publication dans la conférence *Neural Information Processing Systems* (NeurIPS) (Veilleux *et al.* (2021))).

# CHAPITRE 1

# REVUE DE LITTÉRATURE

Ce chapitre présente une introduction au problème d'apprentissage *Few-Shot* (FSL), ainsi qu'une revue et analyse de méthodes de l'état de l'art pour ce type de problème de classification.

## 1.1 Formulation du problème d'apprentissage *Few-Shot*

Tout comme la majorité des problèmes de classification en apprentissage machine, l'apprentissage *Few-Shot* (FSL) est constitué de deux phases, soient l'entraînement, suivi par la phase de test. Une base de données $\mathcal{D}_{base}$ est utilisée pour effectuer l'entraînement du modèle. Le modèle en question est généralement un réseau de neurones à convolutions profond (CNN). Le but de l'entraînement est d'apprendre les paramètres $\theta$ du modèle à partir des données annotées de la base $\mathcal{D}_{base}$. Les paramètres optimisés du modèle lui permettent de représenter les données $\mathcal{X}$ dans un espace à dimensions réduites. À la sortie du CNN les données $\mathcal{X}$ sont représentés sous forme de vecteurs de caractéristiques $\mathcal{Z}$. Une fois l'entraînement terminé, le modèle peut donc être vu comme étant un extracteur de caractéristiques : $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$.

Lors de la phase de test, le modèle est évalué par rapport à ses performances de classification sur de multiples tâches $\mathcal{T}$. Les tâches $\mathcal{T}$ de classification sont composées de deux sous-ensembles, soient de la base de support $\mathcal{S}$ et de la base de *query* $\mathcal{Q}$. La base de support $\mathcal{S}$ contient des données $\mathcal{X}_\mathcal{S}$ appartenant à un nombre $K$ de classes différentes (*K-way*). Pour chacune des classes ($K$), $N$ échantillons de données étiquetées sont présents (*N-shot*). La base de *query* quant à elle contient des données non étiquetées $\mathcal{X}_\mathcal{Q}$ partageant les mêmes classes que celle des données de la base de support $\mathcal{S}$. La figure 1.1 illustre une tâche *Few-Shot*. Pour chaque tâche $\mathcal{T}$, le but est que le modèle préentraîné $f_\theta$ sur $\mathcal{D}_{base}$ classifie correctement les données *query* $\mathcal{Q}$ en ayant accès seulement qu'à quelques données annotées par classe (généralement $N = 1 \parallel 5$) , celles de la base de support $\mathcal{S}$. La difficulté du problème provient du fait que les classes des données de la base d'entraînement $\mathcal{D}_{base}$ ($\mathcal{Y}_{base}$) sont différentes de celles des tâches de classification $\mathcal{T}\{\mathcal{S}, \mathcal{Q}\}$ ($\mathcal{Y}_{test}$), mathématiquement dénoté : $\mathcal{Y}_{base} \cap \mathcal{Y}_{test} = \emptyset$. Le modèle est donc évalué en fonction de

sa capacité de généralisation pour la classification de données appartenant à des classes n'ayant pas été vue lors de l'entraînement de base, et ce dans un contexte très peu supervisé.


FIGURE 1.1 Exemple d'une tâche 1-*shot* 5-*way* de problème *Few-Shot*. Les données annotées de la base de support $\mathcal{X}_S$ sont représentées en couleur et les données non-annotées à classifier de la base de *query* sont représentées en tons de gris

## 1.2 Méthodes d'entraînement (méta-apprentissage vs. transfert d'apprentissage)

La littérature de l'apprentissage *Few-Shot* présente deux types de méthodes d'entraînement.

Beaucoup de méthodes FSL se basant sur le principe du méta-apprentissage (Vinyals *et al.* (2016) ; Finn *et al.* (2017) ; Snell, Swersky & Zemel (2017) ; Sung *et al.* (2018)). Le méta-apprentissage également connu sous le nom d'entraînement épisodique, il peut être défini par le concept d'apprendre à apprendre. Cela veut dire que l'entraînement du modèle se fait au travers d'épisodes qui simulent l'environnement de test auquel le modèle aura à faire face.

Un épisode est composé d'une batch de support $\mathcal{B}_{\mathcal{S}}$ et d'une batch de *query* $\mathcal{B}_{\mathcal{Q}}$. Les batch sont échantillonnés parmi les données de la base d'entraînement $\mathcal{D}_{base}$. D'abord, $K$ classes sont aléatoirement sélectionnées parmi $\mathcal{Y}_{base}$. $\mathcal{B}_{\mathcal{S}}$ qui représente des bases de support $\mathcal{S}$ est composé de $N$ exemples de données appartenant au $K$ classes sélectionnées parmi la base d'entraînement $\mathcal{D}_{base}$. $\mathcal{B}_{\mathcal{Q}}$ qui représente des bases de *query* $\mathcal{Q}$ est composé d'exemples de données appartenant aux mêmes $K$ classes sélectionnées.

L'entraînement épisodique vise à apprendre des paramètres $\theta$ conduisant à une bonne classification des données des batch de *query* $\mathcal{B}_{\mathcal{Q}}$ et ayant uniquement accès qu'à quelques exemples de données annotées, ceux des batch de support $\mathcal{B}_{\mathcal{S}}$. Les paramètres $\theta$ sont appris par un processus d'optimisation qui pénalise le modèle lors de mauvaises classifications. La logique d'optimisation qui pénalise le modèle est propre à chaque méthode qui s'appuie sur les principes du méta-apprentissage.

Plusieurs méthodes de l'état de l'art (Wang, Chao, Weinberger & van der Maaten (2019) ; Boudiaf *et al.* (2020) ; Chen *et al.* (2020) ; Dhillon *et al.* (2020) ; Liu *et al.* (2020a) ; Ziko *et al.* (2020)) adoptent les principes du transfert d'apprentissage. Les méthodes s'appuyant sur le transfert d'apprentissage cherchent à obtenir une fois l'entraînement terminé, un modèle $f_\theta$ permettant d'extraire les caractéristiques discriminantes des données et ainsi les représentés sous dimensions réduites à la sortie du réseau $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$.

L'apprentissage des paramètres $\theta$ est effectué par un entraînement supervisé sur les données de $\mathcal{D}_{base}$. Les données de $\mathcal{D}_{base}$ sont aléatoirement divisées en batch $\mathcal{B}$ de même taille. Les batch $\mathcal{B}$ sont présentées au CNN. À la sortie du réseau, une fonction de coût est calculée par rapport aux prédictions faites pour chaque batch $\mathcal{B}$. Les paramètres $\theta$ du réseau sont optimisés pour

assurer la minimisation de la fonction de coût. La mise à jour des paramètres $\theta$ est effectué lors de la rétropropagation du gradient de la fonction de coût, pour chaque batch. La fonction de coût généralement utilisée est l'entropie croisée (CE) entre la prédiction de classe $p_{ik}$ et la classes réelles $y_{ik}$. Cette dernière est définie de la façon suivante :

$$\min_{\theta} \mathrm{CE} \tag{1.1}$$

$$\mathrm{CE} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.2}$$

La prédiction de classe, $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \theta, \boldsymbol{W}\right)$ est en fait la probabilité d'appartenance de la donnée $x_i$ à la classe $k$, calculé par la fonction softmax appliquée aux données à la sortie du réseau :

$$p_{ik} \propto \exp\left(W_k^T f_\theta(x_i)\right) \tag{1.3}$$

La classe réelle de chaque donnée $x_i$ est dénoté par le vecteur $\boldsymbol{y}_i = (y_1, \ldots, y_K)$ encodé sous forme *one-hot*.

Lors de la phase de test, le modèle préentraîné $f_\theta$ est ajusté (*fine-tuning*) en fonction de chaque tâche *Few-Shot* qui lui sont présentées. Les ajustements apportés à $f_\theta$ sont en fait l'apprentissage des paramètres $\boldsymbol{W}$ d'un classificateur $f_{\boldsymbol{W}} : \mathcal{Z} \to \Delta_K$ propre à chaque tâche. Les paramètres $\boldsymbol{W}$ font partie de l'ensemble de paramètres $\theta$, ils correspondent à ceux de la dernière couche du réseau qui est un classificateur linéaire. Les méthodes reposant sur le principe du transfert d'apprentissage se distinguent l'une d'entre elles par l'algorithme utilisé pour l'apprentissage du classificateur $f_{\boldsymbol{W}}$.

## 1.3 Inférence inductive vs. transductive

Les méthodes de la littérature *Few-Shot* se distinguent selon leur manière de construire le classificateur $f_W$. Ce dernier permet d'émettre les prédictions de classes lors de l'inférence. Il existe des méthodes inductives et transductives.

### 1.3.1 Inductive

Lors de l'inférence, les méthodes inductives construisent leur classificateur $f_W$ en se basant sur les statistiques des données étiquetées. Ce processus est comparable à l'apprentissage supervisé qui utilise également un ensemble de données annotées. Dans le cas du problème *Few-Shot*, les données annotées sont celles de la base de support $\mathcal{S}$. Pour chaque tâche $\mathcal{T}$, un nouveau classificateur $f_W$ est appris en se basant sur $\mathcal{S}$. L'apprentissage du classificateur peut être mathématiquement noté : $f_W = \mathcal{A}\left(f_\theta, \mathcal{S}\right)$. La prédiction des classes des données de la base de *query* $\mathcal{Q}$ s'effectue un exemple à la fois. Les prédictions émises sont donc indépendantes les unes des autres.

### 1.3.2 Transductive

Le classificateur $f_W$ des méthodes transductives est construit en exploitant les statistiques des données étiquetées ainsi que celles des données non-étiquetées. Le processus est comparable à l'apprentissage semi-supervisé. Encore une fois, pour chaque tâche $\mathcal{T}$, un nouveau classificateur $f_W$ est appris, mais en se basant sur $\mathcal{S}$ et $\mathcal{Q}$. L'apprentissage du classificateur transductif est mathématiquement dénoté : $f_W = \mathcal{A}\left(f_\theta, \mathcal{S}, \mathcal{Q}\right)$. La classification de $\mathcal{Q}$ est donc effectué en un seul temps. Les prédictions émises sont dépendantes de la distribution des données $\mathcal{Q}$ à classifier.

### 1.3.3 Résumé méthode d'inférence

En résumé, deux familles de méthodes sont présentes dans la littérature *Few-Shot*, le méthodes inductives et transductives.

À l'inférence, les méthodes inductives construisent leur classificateur $f_W$ en exploitant uniquement les statistiques des données annotées $\mathcal{S}$. La distribution des données non-étiquetées $Q$ n'a aucune influence sur les poids $W$ de $f_W$. Cela fait en sorte que $f_W$ peut émettre des prédictions pour n'importe quel nouvel exemple non-étiqueté, car leur prédiction se fait indépendemment les unes des autres.

Les méthodes transductives quant à elles construisent leur classificateur $f_W$ en se basant sur les statistiques des données annotées $\mathcal{S}$ et des données non-annotées $Q$. Le fait d'introduire de l'information par rapport aux données à classifier lors de la construction de $f_W$ limite sa capacité de généralisation. En effet, les prédictions de nouvelle données non-annotées n'ayant pas été vu lors de la construction de $f_W$, risquent d'être imprécises. Comme les prédictions émises par $f_W$ sont dépendantes de la distribution des données non-étiquetées, le classificateur devrait donc être reconstruit lors de l'inclusion de nouvelles données à classifier afin d'exploiter leur statistiques. Ce cas de figure n'impacte peu dans le contexte de la classification *Few-Shot*, car le classificateur est reconstruit à chaque nouvelle tâche $\mathcal{T}$ et les bases de *query* sont préalablement échantillonnées.

Les méthodes transductives ont gagné en popularité dans la littérature FSL récente. Ces méthodes permettent d'obtenir des performances de classification supérieure. En effet, l'exploitation de l'information sur la distribution des données à classifier lors de la construction du classificateur permet d'obtenir de meilleurs résultats de classification. Par contre, il faut faire attention à leur méthode de construction du classificateur, comme certaines d'entre elles peuvent introduire de fausses connaissances a priori sur la distribution des données à classifier.

### 1.3.4 Normalisation de batch

La normalisation de batch a été introduite dans la littérature de l'apprentissage machine par (Ioffe & Szegedy (2015)). Cette procédure permet de réduire le changement de la forme de la distribution des données entre chacune des couches du réseau. La normalisation de batch assure que chaque dimension des activations aux couches du réseau suivent une distribution normale.

Cette imposition de distribution permet aux réseaux de neurones profonds d'avoir un meilleur flux du gradient, réduit la dépendance à l'initialisation des paramètres et permet l'utilisation de taux d'apprentissage plus grands (Ben Ayed (2019)). La normalisation de batch est maintenant indispensable lors de l'entraînement de réseaux de neurones profonds.

Tel que son nom l'indique, la normalisation de batch est appliqué sur chaque mini-batch $\mathcal{B}$ présentées au réseau. Les mini-batch $\mathcal{B}$ sont constitué d'activations $x$ ayant $d$-dimensions $x = (x^{(1)}, ..., x^{(d)})$. Lors de la phase d'entraînement, les activation $x$ de la mini-batch $\mathcal{B} = \{x_1^{(d)}, ..., x_m^{(d)}\}$ contenant $m$ échantillons, sont normalisées sur chacune de leur dimension $d$ en utilisant leurs propres statistiques, soit la moyenne et la variance de la batch. La moyenne $\mu_{\mathcal{B}}$ et la variance $\sigma_{\mathcal{B}}^2$ de chaque dimension $k$ sont calculées de la façon suivante :

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{1.4a}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \tag{1.4b}$$

Les statistiques de l'équation 1.4 sont utilisées pour normaliser chaque dimensions $k$ des activations de la mini-batch $\mathcal{B} = \{x_1^{(d)}, ..., x_m^{(d)}\}$ de la manière suivante :

$$\widehat{x_i} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{1.5}$$

Les activations normalisées $\widehat{x}$ sont ensuite mises à l'échelle et décalées dans chacune de leur dimension $k$ en utilisant les paramètres appris $\gamma^k$ et $\beta^k$. Le paramètre $\gamma^k$ est en fait l'écart-type et $\beta^k$ est la moyenne des activations.

$$y_i \leftarrow \gamma \widehat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{1.6}$$

Lors de la phase d'entraînement, la normalisation de batch s'effectue en utilisant les statistiques $\mu_{\mathcal{B}}$, $\sigma^2_{\mathcal{B}}$ de la batch courante. Ces statistiques sont donc recalculées tel qu'aux équations 1.4a et 1.4b pour la normalisation de chaque batch. Lors de la phase de test, il est pris pour acquis que le modèle n'a pas accès préalablement aux statistiques des données à classifier. La normalisation de batch lors de l'inférence est donc effectuée en se basant sur les statistiques de la population d'entraînement. Durant l'entraînement, une moyenne variable (*running mean*) $E[x]$ et une variance variable (*running variance*) $Var[x]$ ont été calculées et mises à jour en exploitant les données contenues dans les batch d'entraînement. À l'inférence, les valeurs de moyennes et de variances utilisées sont maintenues fixes. La normalisation des données de test est effectuée en utilisant la moyenne variable (*running mean*) $E[x]$ et la variance variable (*running variance*) $Var[x]$.

$$\widehat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \tag{1.7}$$

De cette manière, aucune information a priori sur la distribution des données à classifier n'est introduite lors de l'inférence.

#### 1.3.4.1 Normalisation de batch transductive

La majorité des méthodes de la littérature FSL utilisent et dépendent de la normalisation de batch lors du préentraînement de leur modèle. Les méthodes d'apprentissage par transfert utilisent la normalisation de batch tel qu'expliqué à la sous-section 1.3.4.

Les méthodes exploitant le méta-apprentissage effectuent leur entraînement à l'aide d'épisodes (tel qu'expliqué à la section 1.2). Lors du méta-apprentissage, la normalisation des épisodes s'effectue grâce aux statistiques d'épisodes (moyenne et variance) calculées à partir des données des bases $\mathcal{S}$ et $\mathcal{Q}$. Afin de respecter les principes de la normalisation de batch et ainsi ne pas introduire d'information par rapport aux données de test, lors de la phase de méta-test, les méthodes devraient utilisées la moyenne variable (*running mean*) $E[x]$ et la variance

variable (*running variance*) $Var[x]$ calculées lors de la phase de méta-apprentissage. Ce n'est pas toutes les méthodes par méta-apprentissage qui respectent ce principe. En effet, tel que pointé par (Nichol, Achiam & Schulman (2018) ; Du, Zhen, Shao & Snoek (2021)) certaines méthodes très populaires telles que MAML (Finn *et al.* (2017)) et Versa (Gordon *et al.* (2019)) utilisent le principe de normalisation de batch transductive (TBN) (Bronskill, Gordon, Requeima, Nowozin & Turner (2020)). Ces méthodes ont également inspiré de nombreuses autres méthodes (Yanbin *et al.* (2019); Mishra, Rohaninejad, Chen & Abbeel (2018); Oreshkin, López & Lacoste (2018); Rusu *et al.* (2019); Sung *et al.* (2018); Hou *et al.* (2019); Ye, Hu, Zhan & Sha (2020)). La normalisation de batch transductive se veut que lors de la phase de méta-test, pour chaque épisode les statistiques de normalisation (moyenne et variance) sont calculées à l'aide des données des bases $\mathcal{S}$ et $\mathcal{Q}$, à la place d'utiliser les statistiques calculées lors de la phase de méta-entraînement. La TBN lors du méta-test introduit donc de l'information par rapport aux données de test $\mathcal{Q}$. L'utilisation de TBN par les méthodes fait en sorte que ces dernières sont considérées comme étant transductive, dû au fait que l'adaptation des paramètres du modèle est effectuée en exploitant $\mathcal{S}$ et $\mathcal{Q}$ (section 1.3). Les méthodes exploitant la TBN obtiennent des augmentations significatives en performances, par rapport aux autres méthodes de méta-apprentissage.

## 1.4 Revue des méthodes par méta-apprentissage

### 1.4.1 Matching Networks for One Shot Learning

Matching Network (Vinyals *et al.* (2016)) est une méthode se basant sur l'apprentissage épisodique. À chaque épisode, un noyau d'attention (*attention kernel*) est défini afin de mesurer la similarité entre les données de support : $x_s \in \mathcal{B}_\mathcal{S}$ et les données *query* : $x_q \in \mathcal{B}_\mathcal{Q}$. Bien qu'il n'y ait pas d'avantage clair, les auteurs proposent d'utiliser deux extracteurs de caractéristiques différents, soit un pour les données de support : $f_\theta : \mathcal{X}_\mathcal{S} \rightarrow \mathcal{Z}_\mathcal{S}$ et un deuxième pour les données *query* : $g_\theta : \mathcal{X}_\mathcal{Q} \rightarrow \mathcal{Z}_\mathcal{Q}$. Le noyau d'attention est calculé par la fonction softmax de la similarité cosinus $\cos_{s,q}$ entre les caractéristiques de données de support : $z_s = f_\theta(x_s)$ et *query* : $z_q = g_\theta(x_q)$ :

$$\alpha(x_q, x_s) = \frac{\exp\left(\cos_{s,q}\right)}{\sum_{k=1}^{K} \exp\left(\cos_{s_k,q}\right)} \tag{1.8}$$

Lors du méta-apprentissage, les paramètres $\theta$ des extracteurs de caractéristiques sont appris en minimisant l'entropie croisée entre les prédictions des données *query* ($x_q \in \mathcal{B}_Q$) et leurs vraies annotations :

$$\mathcal{L}_{\text{mathcing}} = -\sum_{i \in \mathcal{B}_Q} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.9}$$

Les prédictions de classe $p_{ik} := \mathbb{P}\left(Y = k | x_q, \mathcal{B}_\mathcal{S}; \theta\right)$ sont émises par le noyau d'attention, soit par la plus grande mesure de similarité :

$$p_{ik} \propto \sum_{i \in \mathcal{B}_Q} \sum_{k=1}^{K} \alpha(x_{qi}, x_{sk}) y_{sk} \tag{1.10}$$

### 1.4.2 Prototypal Networks for Few-shot Learning

Prototypal Network (Snell *et al.* (2017) est une méthode qui exploite les principes du méta-apprentissage. Lors de chaque épisode, cette méthode utilise les données annotées de $\mathcal{B}_\mathcal{S}$ afin de produire des prototypes de classe $c_k$. Le prototype de chaque classe est en fait une représentation moyenne des données annotées dans l'espace des caractéristiques. Les caractéristiques des données de support $\mathcal{B}_\mathcal{S}$ sont extraites à l'aide du modèle $f_\theta : \mathcal{X} \to \mathcal{Z}$. Les prototypes de classes sont obtenus en calculant la moyenne de $\mathcal{B}_\mathcal{S}$ dans l'espace des caractéristiques, et ce pour chaque classe $k$ :

$$\mu_k = \frac{1}{N} \sum_{x_s \in B_S : y_i = k} f_\theta(x_s) \tag{1.11}$$

Pour chaque donnée de la batch de *query* $\mathcal{B}_Q$, une distribution de probabilité d'appartenance à chaque classe est produite $p_{ik} := \mathbb{P}\left(Y = k | x_q \in \mathcal{B}_Q, \mu_k; \theta\right)$. Cette dernière est obtenue en calculant la fonction softmax de la distance entre chaque exemple $x_q$ et chaque prototype de classe $\mu_k$ :

$$p_{ik} \propto \frac{e^{-d(f_\theta(x_{qi}), \mu_k)}}{\sum_{k=1}^{K} e^{-d(f_\theta(x_{qi}), \mu_k)}} \tag{1.12}$$

L'apprentissage des paramètres $\theta$ du modèle est effectué par la minimisation de la fonction de coût $\mathcal{L}_{proto}$ à l'aide de la descente stochastique du gradient. La fonction de coût est l'entropie croisée entre les prédictions du modèle et la distribution réelle des étiquettes :

$$\mathcal{L}_{\text{proto}} = -\sum_{i \in \mathcal{B}_Q} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.13}$$

Cette méthode cherche donc à optimiser les paramètres $\theta$ du modèle afin de produire une bonne représentation des données annotées dans l'espace des caractéristiques. Si l'extraction des caractéristiques est efficace, alors seulement quelques exemples annotés sont nécessaires pour produire des prototypes de classes fiables. De cette manière, l'assignation de classe aux exemples non annotés est effectuée en fonction de la distance minimale avec un certain prototype. *Prototypal Network* est une méthode inductive, car son classificateur est construit en exploitant uniquement les statistiques de $\mathcal{B}_S$.

### 1.4.3 Model Agnostic Meta-Learning for Fast Adaptation of Deep Networks (MAML)

*Model Agnostic Meta-Learning* (MAML) (Finn *et al.* (2017)) est une méthode de méta-apprentissage qui vise à optimiser les paramètres $\theta$ du modèle, de manière à ce que ces derniers soient bien initialisés à la fin de la phase du méta-apprentissage. Les auteurs affirment qu'une bonne initialisation des paramètres $\theta$ sur les tâches d'entraînement permet de facilement les adapter lorsque le modèle rencontre de nouvelles tâches. Si les paramètres du modèle sont optimaux à l'itération $t$, alors seulement une ou quelques itérations de minimisation du gradient seraient requises afin que ces paramètres soient adaptés $\hat{\theta}$ pour la classification de nouvelles tâches. Par exemple, l'adaptation des paramètres $\theta$ à l'itération $t$, s'effectue en minimisant l'entropie croisée sur les données de support :

$$\hat{\theta} = \theta(t) - \alpha \nabla_{\theta} \mathcal{L}\left(\mathcal{B}_{\mathcal{S}}, \theta(t)\right) \tag{1.14}$$

Où $\alpha$ est la valeur du taux d'apprentissage. Une fois que les paramètres sont soi-disant adaptés à la tâche, le modèle est pénalisé s'il est émet des prédictions de classe incorrectes pour les données *query*. La fonction de coût est l'entropie croisée entre les prédictions des données *query* $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \hat{\theta}, \boldsymbol{W}\right)$ et leur classe véridique.

$$\mathcal{L}_{\text{MAML}} = -\sum_{i \in \mathcal{B}_Q} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.15}$$

Les prédictions sont émises par la fonction softmax appliquée aux *logits* à la sortie de la couche de classification :

$$p_{ik} \propto \exp\left(\boldsymbol{W}_k^T f_{\hat{\theta}}(x_i)\right) \tag{1.16}$$

Lors de la manipulation de nouvelle tâches durant la phase de méta-test, les paramètres du modèle sont adaptés en exploitant les statistiques de $\mathcal{B}_{\mathcal{S}}$ tel qu'à l'équation 1.14. Une fois l'adaptation des paramètres effectuée à partir des données de support $\mathcal{B}_{\mathcal{S}}$, les prédictions sont obtenues par le calcul des probabilités d'appartenance softmax à chaque classe, tel qu'à l'équation 1.16.

Malgré que lors du méta-test, l'adaptation des paramètres $\theta$ s'effectue à l'aide des données annotées de la base de support $\mathcal{B}_{\mathcal{S}}$, cette méthode ne peut être considérée comme étant inductive. Les auteurs ont omis de mentionner que lors du méta-test de l'information par rapport aux données de test $\mathcal{Q}$ est introduite par rapport à ceux-ci via la normalisation de batch transductive (TBN). Lors du méta-test, les statistiques de l'épisode ($\mu$ et $\sigma^2$) sont calculées à partir des bases $\mathcal{S}$ et $\mathcal{Q}$ courantes. Tel qu'expliqué à la sous-section 1.3.4, la normalisation de batch devrait s'effectuer à partir de la moyenne variable (*running mean*) $E[x]$ et de la variance variable (*running variance*) $Var[x]$ calculées lors de la phase d'entraînement. MAML est donc une méthode transductive.

### 1.4.4 Learning to Propagate Labels : Transductive Propagation Network for Few-Shot Learning (TPN)

(Yanbin *et al.* (2019)) introduisent la méthode TPN qui exploite les données *query* en plus des données de support afin d'avoir accès à un maximum d'information pour la construction d'un classificateur fiable. Pour se faire, ils proposent de propager les étiquettes des données annotées $\mathcal{S}$ sur les données $\mathcal{Q}$. La méthode comprend un premier réseau de neurones qui est utilisé pour l'extraction des caractéristiques des données : $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$. Une fois tous les exemples de la tâche représentés sous forme de caractéristiques, ces derniers sont considérés comme faisant part d'un et unique ensemble : $z_i \in \mathcal{S} \cup \mathcal{Q}$. Un graphe (matrice : $W \in \mathbb{R}^{(N\times K+|\mathcal{Q}|)\times(N\times K+|\mathcal{Q}|)}$) interconnectant toutes les données de l'ensemble $\mathcal{S} \cup \mathcal{Q}$ est construit. Chaque paire de données représentées dans l'espace des caractéristiques est connectée par un poids $W_{ij}$ qui représente leur similarité gaussienne :

$$W_{ij} = \exp\left(-\frac{1}{2}d\left(\frac{z_i}{\sigma_i}, \frac{z_j}{\sigma_j}\right)\right) \tag{1.17}$$

où $d(\cdot,\cdot)$ est la distance euclidienne entre la paire d'exemples $i$ et $j$. Le paramètre d'échelle de longueur $\sigma$ de chaque exemple $z_i$ est produit par un deuxième réseau de neurones $g_\phi : \mathcal{Z} \to \sigma$.

La propagation des étiquettes aux données *query* est effectuée par l'expression suivante :

$$F^* = (I - \alpha S)^{-1} Y \tag{1.18}$$

où $F^*$ est la matrice de score d'annotation, $I$ est la matrice identité, $S$ est le graphe de poids $W$ normalisé et $\alpha \in (0, 1)$ est un paramètre contrôlant la quantité d'information partagée. La matrice $Y$ dresse la carte d'assignation d'étiquette. Cette dernière est construite de manière à ce que $Y_{ij} = 1$ si la donnée $i$ provient de la base de support et que son étiquette $y_i$ correspond à celle de son binôme $j : y_i = y_j$, sinon $Y_{ij} = 0$.

Les prédictions de classe $p_{ik} := \mathbb{P}\left(Y = k | X \in \mathcal{B}_\mathcal{S} \cup \mathcal{B}_\mathcal{Q}; \theta, \phi, \boldsymbol{W}\right)$ sont obtenues en appliquant la fonction softmax sur la matrice de score d'annotation $F^*$ :

$$p_{ik} \propto \frac{\exp F^*}{\sum \exp F^*} \tag{1.19}$$

Lors du méta-apprentissage, les paramètres $\theta$ et $\phi$ des deux réseaux de neurones $f_\theta$ et $g_\phi$ sont appris en minimisant l'entropie croisée entre les étiquettes prédites de toutes les données de la tâche $(\mathcal{S} \cup \mathcal{Q})$ et leur vraie annotation. La fonction de coût est dénotée :

$$\mathcal{L}_{\text{TPN}} = -\sum_{i \in \mathcal{B}_S \cup \mathcal{B}_Q} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.20}$$

L'inférence de cette méthode lors de la phase de méta-test est transductive, car le classificateur a été construit en exploitant les données $\mathcal{S}$ et $\mathcal{Q}$ de la tâche. Les prédictions de classe sont émises en suivant l'équation 1.19.

## 1.5 Revue des méthodes par entraînement régulier

### 1.5.1 A Closer Look at Few-Shot Classification

Les auteurs du papier *A Closer Look at Few-Shot Classification* (Chen, Liu, Kira, Wang & Huang (2019)) proposent une méthode qui exploitent le transfert d'apprentissage. Lors de la phase de test, ces derniers proposent de garder les paramètres $\theta$ de l'encodeur ($f_\theta$) fixes et de réentraîner un nouveau classificateur $f_{\boldsymbol{W}}$ en utilisant les données annotées $\mathcal{S}$. L'adaptation des poids $\boldsymbol{W}$ du classificateur est effectué par la minimisation de l'entropie croisée :

$$\mathcal{L} = \text{CE} = -\sum_{i \in \mathcal{S}} \sum_{k=1}^{K} y_{ik} \log(p_{ik}) \tag{1.21}$$

Deux méthodes pour émettre les prédictions de classe $p_{ik} := \mathbb{P}(Y = k | X = x_i; \theta, \boldsymbol{W})$ sont proposées :

- *Baseline :* Les prédictions sont produites en applicant la fonction softmax sur les sorties de la couche de classification.

$$p_{ik} \propto \exp\left(W_k^T f_\theta(x_i)\right) \tag{1.22}$$

- *Baseline++ :* Cette version cherche à réduire la variation intra-classe des caractéristiques. Pour se faire, les caractéristiques sont normalisées. Le classificateur pose donc ses prédictions en fonction de la distance cosinus entre les caractéristiques des données et les poids du classificateur.

$$p_{ik} \propto \exp\left(\frac{W_k^T f_\theta(x_i)}{\|W_k\| \, \|f_\theta(x_i)\|}\right) \tag{1.23}$$

### 1.5.2 SimpleShot : Revisiting Nearest-Neighbor Classification for Few-Shot Learning

La méthode *SimpleShot* (Wang *et al.* (2019)) se base également sur le principe du transfert d'apprentissage. Les auteurs proposent une approche basée sur le principe du plus proche voisin pour effectuer la classification des données. Aucune modification à l'extracteur de caractéristiques $f_\theta$ n'est apportée suite au préentraînement. L'adaptation du classificateur est effectuée en utilisant les données de support $\mathcal{S}$ pour calculer le centroïde $\mu_k$ de chaque classe $k$.

$$\boldsymbol{W}_k \leftarrow \mu_k \tag{1.24}$$

La prédiction de classes des données $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \theta, \boldsymbol{W}\right)$ est ensuite effectuée en fonction de la plus petite distance avec un certain centroïde de classe. Trois approches se distinguant par le type de normalisation appliquée aux caractéristiques ont été proposées :

- *Aucune normalisation :* Aucune transformation n'est appliquée aux caractéristiques obtenues à la sortie de l'extracteur de caractéristiques.

$$p_{ik} \propto \exp(\boldsymbol{W}_k^T f_\theta(x_i)) \tag{1.25}$$

- *Normalisation L2 :* Une normalisation L2 est appliquée aux caractéristiques des données.

$$p_{ik} \propto \exp\left(\frac{\boldsymbol{W}_k^T f_\theta(x_i)}{\|\boldsymbol{W}_k\| \, \|f_\theta(x_i)\|}\right) \tag{1.26}$$

- *Centrage et normalisation L2 (CL2) :* D'abord, un centrage des données est effectué en utilisant un prototype moyen des caractéristiques des données de la base d'entraînement $\mathcal{D}_{base}$ :

$$z \leftarrow f_\theta(x) - \frac{1}{|\mathcal{D}_{base}|} \sum_{x \in \mathcal{D}_{base}} f_\theta(x) \tag{1.27}$$

  La normalisation L2 est ensuite appliquée aux caractéristiques centrées. Le classificateur émet donc ses prédictions de la manière suivante :

$$p_{ik} \propto \exp\left(\frac{\boldsymbol{W}_k^T z_i}{\|\boldsymbol{W}_k\| \, \|z_i\|}\right) \tag{1.28}$$

### 1.5.3 A baseline for Few-Shot Image Classification

L'approche proposée dans le papier de (Dhillon *et al.* (2020)) se base également sur le principe du transfert d'apprentissage. Le préentraînement du modèle est effectué de manière standard, soit par la minimisation l'entropie croisée sur les données annotées de $\mathcal{D}_{base}$. L'adaptation du modèle lors de la phase de test diffère par contre des autres méthodes. La couche de classification linéaire entraînée sur $\mathcal{D}_{base}$ est conservée et utilisée pour produire les *logits* $z = \boldsymbol{W} f_\theta + b, z \in \mathbb{R}^{|\mathcal{Y}_{base}|}$ qui sont utilisés comme caractéristiques d'entrées dans la nouvelle couche de classification. Cette nouvelle couche de classification est définie comme étant : $\{\boldsymbol{W'}, b'\}, \boldsymbol{W'} \in \mathbb{R}^{|\mathcal{Y}_{base}| \times |\mathcal{Y}_{task}|}, b' \in \mathbb{R}^{|\mathcal{Y}_{task}|}$. Les auteurs justifient l'utilisation des *logits* du modèle préentraîné comme caractéristiques par le fait que ces derniers sont déjà bien groupés selon les classes. Deux méthodes sont proposées :

- L'initialisation des poids du classificateur en utilisant les données de support. Un prototype correspondant au centroïde $c_k$ des données de support de chaque classe $k$ est défini.

$$\boldsymbol{W}_k \leftarrow c_k \tag{1.29}$$

Les prédictions du classificateur $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \theta, \boldsymbol{W}\right)$ sont émises en fonction de la distance cosinus entre les données à classifier et les prototypes de classes.

$$p_{ik} \propto \exp\left(\frac{\boldsymbol{W}_k^T f_\theta(x_i)}{\|\boldsymbol{W}_k\| \, \|f_\theta(x_i)\|}\right) \tag{1.30}$$

- L'autre méthode proposée consiste à effectuer une adaptation semi-supervisée de tous les paramètres du modèle, ceci inclut également les paramètres de l'extracteur de caractéristiques $f_\theta$. Pour une question de clarté les paramètres du modèle sont dénotés : $\phi = \{\theta', \boldsymbol{W}', b'\}$. La fonction de coût pour l'adaptation des paramètres $\phi$ est composé de l'entropie croisée sur $\mathcal{S}$ et un terme de régularisation. Le terme de régularisation est l'entropie conditionnelle $\mathcal{H}(Y_Q|X_Q)$ des prédictions de classe sachant $\mathcal{Q}$. Ce terme assure de favoriser des prédictions confiantes. Cette méthode d'adaptation des paramètres est transductive, car elle exploite à la fois $\mathcal{S}$ et $\mathcal{Q}$.

$$\mathcal{L} = -\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K} y_{ik}\log(p_{ik}) + \underbrace{\frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K} y_{ik}\log(p_{ik})}_{\text{Terme de régularisation transductif}} \tag{1.31}$$

Les prédictions $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \phi\right)$ sont émises en appliquant la fonction softmax sur les sorties de la couche de classification :

$$p_{ik} \propto \exp\left(W_k'^T f_{\theta'}(x_i)\right) \tag{1.32}$$

### 1.5.4 Prototype Rectification for Few-Shot Learning

(Liu *et al.* (2020a)) introduisent une méthode par transfert d'apprentissage qui vise à produire un classificateur basé sur la distance entre les prototypes de classes et les données de test. Leur méthode inclue également une procédure afin de réduire les biais d'écart intra-classe et inter-classe.

En calculant les prototypes de classes à partir de seulement quelques exemples annotés, un biais par rapport à la distribution intra-classe est introduit. Le but lors de la définition d'un prototype de classe est que ce dernier moyenne les caractéristiques de toutes les données faisant partie de cette classe. Dans le contexte de l'apprentissage *Few-Shot*, l'accès à seulement quelques exemples annotés dans $\mathcal{S}$ limite la capacité de définition d'un prototype pouvant adéquatement représenter toutes les données d'une classe. Afin de réduire ce biais, les auteurs suggèrent l'utilisation de plus de données afin de calculer des prototypes de classes plus représentatifs. L'idée proposée est d'utiliser les données $\mathcal{Q}$ ayant été pseudo-étiquetées en plus des données $\mathcal{S}$ pour calculer les prototypes de classe.

Tel qu'expliqué par les auteurs, le biais inter-classe est abordé dans les problèmes d'adaptation de domaine, où les données d'entraînement et de tests font partie de deux domaines différents. Le but est de minimiser la distance entre les vecteurs représentant les données du domaine source et du domaine cible. Dans le contexte FSL, les données $\mathcal{S}$ et $\mathcal{Q}$ font parti du même domaine. Le biais inter-classe correspond à la différence entre le vecteur de caractéristiques moyen de $\mathcal{S}$ et de $\mathcal{Q}$. Pour réduire ce biais, il est proposé d'appliquer une translation des caractéristiques des données $\mathcal{Q}$ vers celles des données $\mathcal{S}$.

Comme dans la majorité des méthodes par transfert d'apprentissage, les paramètres de l'extracteur de caractéristiques $f_\theta$ entraîné sur $\mathcal{D}_{base}$ sont maintenus fixes. Lors de l'inférence, les caractéristiques extraites sont d'abord transformées à l'aide de l'une des normalisations suivantes :

- *Aucune normalisation :* Aucune transformation n'est appliquée aux caractéristiques obtenues à la sortie de l'extracteur de caractéristiques.
- *Normalisation L2 :* Une normalisation L2 est appliquée aux caractéristiques des données.

$$z \leftarrow \frac{f_\theta(x)}{\|f_\theta(x)\|} \tag{1.33}$$

- *Centrage et normalisation L2 (CL2) :* D'abord, un centrage des données est effectué en utilisant un prototype moyen des caractéristiques des données de la base d'entraînement $\mathcal{D}_{base}$. La normalisation L2 est ensuite appliquée aux caractéristiques centrées.

$$\textbf{Centrage :}\ z \leftarrow f_\theta(x) - \frac{1}{|\mathcal{D}_{base}|} \sum_{x \in \mathcal{D}_{base}} f_\theta(x) \tag{1.34}$$

$$\textbf{Normalisation L2 :}\ z \leftarrow \frac{z}{\|z\|} \tag{1.35}$$

Les prototypes de classes initiaux sont calculés en moyennant les caractéristiques normalisées des données $\mathcal{S}$ :

$$\mu_k = \frac{1}{N} \sum_{z_s \in \mathcal{S} : y_i = k} z_s \tag{1.36}$$

Afin de réduire la variance entre les caractéristiques normalisées de $\mathcal{S}$ et $\mathcal{Q}$, le terme de décalage $\xi$ est calculé et appliqué aux caractéristiques de $\mathcal{Q}$ :

$$\textbf{Terme de décalage :}\ \xi = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} z_i - \frac{1}{|\mathcal{Q}|} \sum_{i \in \mathcal{Q}} z_i \tag{1.37}$$

$$\textbf{Translation des données } \mathcal{Q} \textbf{ :} \; z'_q \leftarrow z_q + \xi \tag{1.38}$$

L'assignation de pseudo-étiquettes aux données $\mathcal{Q}$ se fait par les prédictions confiantes $p_{ik} := \mathbb{P}_{\text{pseudo}}\left(Y = k | Z = z'_i; \boldsymbol{W}\right)$ émises par un classificateur à distance cosinus :

$$\boldsymbol{W}_k \leftarrow \mu_k \tag{1.39}$$

$$p_{ik} \propto \exp\left(\frac{\boldsymbol{W}_k^T z'_i}{\|\boldsymbol{W}_k\| \left\|z'_i\right\|}\right) \tag{1.40}$$

La base de support annotée est ensuite augmentée par l'addition des données *query* pseudo-étiquetées, $\mathcal{S}' = \mathcal{S} \cup \mathcal{Q}_{pseudo}$. Afin d'assurer l'exactitude des prototypes de classe rectifiés, ces derniers sont calculés en moyennant la somme pondérée des caractéristiques de chaque exemple avec leur poids de confiance de prédiction :

$$\mu'_k = \frac{1}{|\mathcal{S}'|} \sum_{i \in \mathcal{S}'} p_{ik} \times z'_i \tag{1.41}$$

Les prototypes rectifiés sont ensuite utilisés pour initialiser le classificateur linéaire :

$$\boldsymbol{W}'_k \leftarrow \mu'_k \tag{1.42}$$

et émettre les prédictions de classe $p_{ik} := \mathbb{P}\left(Y = k | Z = z'_i; \boldsymbol{W'}\right)$ :

$$p_{ik} \propto \exp\left(\frac{\boldsymbol{W}_k'^T z'_i}{\|\boldsymbol{W}'_k\| \, \|z'_i\|}\right) \tag{1.43}$$

### 1.5.5 Laplacian Regularized Few-Shot Learning (LaplacianShot)

LaplacianShot (Ziko *et al.* (2020)) est une méthode transductive régularisée par un terme Laplacien. Elle se base sur le principe du transfert d'apprentissage. Le modèle préentraîné est maintenu fixe et est utilisé pour extraire les caractéristiques des données : $f_\theta : \mathcal{X} \to \mathcal{Z}$. Deux termes sont considérés pour la classification des données. Le premier terme est basé sur l'assignation d'étiquette en fonction de la distance minimale à un prototype de classe $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$. Ce terme est dénoté :

$$\mathcal{N}(Y) = \sum_{i \in Q} \sum_{k=1}^{K} p_{ik} d(z_i - \mu_k) \tag{1.44}$$

où $\mathbf{p}_i = (p_{i1}, \ldots, p_{iK})$ est le vecteur de probabilité d'assignation de la donnée $x_i$. Le vecteur $\mathbf{p}_i$ représente une ligne dans la matrice d'assignation $\mathbf{P}$.

Les prototypes de classes peuvent être définis par la moyenne ou par le centroïde $c_k$ des caractéristiques des données de support :

$$\mu_k = \frac{1}{|\mathcal{S}|} \sum_{z_s \in \mathcal{S} : y_i = k} z_s \qquad \text{ou} \qquad \mu_k \leftarrow c_k \tag{1.45}$$

Afin d'avoir une meilleure représentation des classes des données par chaque prototype, les auteurs proposent également d'utiliser la méthode de rectification de prototypes (Liu *et al.* (2020a)) tel qu'expliqué à la sous-section 1.5.4.

Le deuxième terme est un régularisateur Laplacien. Les données $\boldsymbol{Q}$ sont évalué par paire $(x_p, x_q)$ en fonction de la similarité de leurs caractéristiques. Les mesures de similarité de chaque paire de données sont compilées dans une matrice d'affinité : $\boldsymbol{W} = [w(z_q, z_p)]$. Le terme Laplacien pénalise le classificateur lors de l'attribution d'étiquettes différentes pour une paire de données partageant des caractéristiques similaires. Ce terme est dénoté :

$$\mathcal{L}(Y) = \frac{1}{2} \sum_{q,p \in \boldsymbol{Q}} w(z_q, z_p) \left\| \boldsymbol{p}_q - \boldsymbol{p}_p \right\|^2 \tag{1.46}$$

La méthode cherche à optimiser la précision des prédictions de classe à l'aide d'un processus itératif. Ce processus itératif a comme objectif de minimiser l'énergie $\mathcal{E}(Y)$ des prédictions de classe obtenues par le terme $\mathcal{N}(Y)$ et le terme Laplacien $\mathcal{L}(Y)$ :

$$\mathcal{E}(Y) = \mathcal{N}(Y) + \frac{\lambda}{2} \mathcal{L}(Y) \tag{1.47}$$

où $\lambda$ est un hyperparamètre contrôlant l'importance accordée au terme Laplacien.

### 1.5.6 Leveraging the Feature Distribution in Transfer-based Few-Shot Learning

PT-MAP, proposée dans le papier de (Hu *et al.* (2020b)) est une méthode transductive par transfert d'apprentissage. Cette méthode se distingue par son processus de normalisation appliqué aux caractéristiques des données et par la construction de son classificateur basée sur le principe du transport optimal.

Comme toute méthode par transfert d'apprentissage, le modèle préentraîné sur $\mathcal{D}_{base}$ est utilisé pour extraire les caractéristiques des données, tout en gardant ses paramètres fixes. Les auteurs expliquent qu'il est souvent supposé dans la littérature que les caractéristiques représentant les données d'une même classe suivent une distribution normale (Gaussienne). Cette supposition est par contre très rarement vérifiée. (Hu *et al.* (2020b)) proposent de transformer les caractéristiques des données en les mettant à la puissance et les mettant à l'échelle en fonction de leur norme :

$$z \leftarrow \frac{\log\left(f_\theta(x) + \epsilon\right)^\beta}{\left\|\log\left(f_\theta(x) + \epsilon\right)^\beta\right\|_2} \tag{1.48}$$

Le facteur de mise à la puissance $\beta$ est un hyperparamètre de valeur positive et $\epsilon = 1\text{e}{-6}$ est utilisé pour assurer que toutes les valeurs des caractéristiques soient plus grandes que 0. La mise à la puissance est motivée par le fait qu'elle permet de réduire les biais présents dans la distribution des caractéristiques. La mise à l'échelle des caractéristiques par leur norme assure que leur nouvelle distribution ait une variance unitaire. Cette manipulation permet de donner autant d'importance à toutes les caractéristiques. Les caractéristiques qui avaient préalablement de grandes variances ne prédomineront plus sur celles qui présentaient des variances plus faibles.

Le classificateur de cette méthode est construit de manière itérative par l'estimation des centres de classe $c_k$ et d'une matrice d'assignation de classe : $\mathbf{M} \in \mathbb{R}^{|Q| \times K}$. Les centres de classe $c_k$ sont en fait l'estimation du centre des caractéristiques pour une certaine classe $k$. La matrice $\mathbf{M}$ est calculé en se basant sur le principe du transport optimal. Le transport est effectué entre les distributions $r$ et $c$. La distribution $r$ représente le coût requis pour l'attribution d'étiquette à chaque exemple de $Q$. La distribution $c$ représente la quantité de données non annotées attribuées à chaque classe $k$. Le but est de construire la matrice $\mathbf{M} \in \{\mathbf{M} : \mathbf{M}\mathbb{1}_k = r, \mathbb{1}_{|Q|}\mathbf{M} = c\}$ qui minimise le coût du transport optimal entre $r$ et $c$. Cette minimisation est assurée par l'algorithme de Sinkhorn-Knopp :

$$\mathbf{M} = Sinkhorn(\mathbf{D}, r, c, \lambda) \tag{1.49}$$

où $\mathbf{D}$ est une matrice contenant la distance euclidienne entre chaque exemple de la tâche et chaque centre de classe $c_k$. Lors de l'itération initiale, $c_k^0$ sont estimés à l'aide des données de support. Par la suite, lors de chaque itération $t$ les centres de classe $c_k^t$ sont réestimées en se basant sur les statistiques des données $\mathcal{S}$ ainsi que celle de $\mathcal{Q}$ temporairement annotés par $\mathbf{M}$ :

$$\mu_k = \frac{\sum\limits_{i \in \mathcal{Q}} \mathbf{M}_{ik} \cdot z_i + \sum z_s}{\sum\limits_{i \in \mathcal{Q}} \mathbf{M}_{ik}} \tag{1.50}$$

$$c_k^{t+1} \leftarrow c_k^t + \alpha(\mu_k - c_k^t) \tag{1.51}$$

où $\mu_k$ le centre de classe $k$ temporaire et $\alpha \in [0, 1]$ est l'hyperparamètre du taux d'apprentissage. Cette méthode de mise à jour des centres de classe permet de ne pas exécuter des changements trop rapides. La matrice d'assignation est recalculée à l'aide de l'équation 1.49 en tenant compte des nouveaux $c_k$.

À la fin du processus itératif, l'assignation des étiquettes est émise en fonction de la probabilité d'appartenance la plus confiante :

$$y_i = \underset{k}{\operatorname{argmax}} \, (\mathbf{M}_{ik}) \tag{1.52}$$

### 1.5.7 Transductive Information Maximization For Few-Shot Learning (TIM)

(Boudiaf *et al.* (2020)) proposent une méthode transductive visant la maximisation de l'information mutuelle entre les exemples à classifier et leur prédiction de classe. Les paramètres de l'extracteur de caractéristiques $f_\theta$ appris lors du préentraînement sur $\mathcal{D}_{base}$ sont maintenus fixes. Les caractéristiques des données sont transformées en suivant une normalisation L2 :

$$z \leftarrow \frac{f_\theta(x)}{\|f_\theta(x)\|} \tag{1.53}$$

Un nouveau classificateur est appris pour chaque tâche. Les prédictions $p_{ik} := \mathbb{P}\left(Y = k | X = x_i; \boldsymbol{W}, \theta\right)$ sont émises en fonctions de la matrice de poids apprise $\boldsymbol{W} \in \mathbb{R}^{K \times d}$ où $K$ est le nombre de classes et $d$ est la dimension des caractéristiques :

$$p_{ik} \propto \exp\left(-\frac{\tau}{2}\|w_k - z_i\|^2\right) \qquad \text{et} \qquad \widehat{p}_k = \frac{1}{|Q|}\sum_{i \in Q} p_{ik} \tag{1.54}$$

où $\widehat{p}_k := \mathbb{P}\left(Y_Q = k; \boldsymbol{W}, \theta\right)$ correspond aux prédictions marginales de classe.

Pour chaque tâche, les poids du classificateur sont entraînés en minimisant une fonction de coût semi-supervisée. Cette dernière exploite l'information de l'entièreté des données de la tâche. En effet, les données de $\mathcal{S}$ et $Q$ sont utilisés afin d'optimiser les poids du classificateur. L'exploitation de l'information des données à classifier en fait donc une méthode transductive. La supervision est amenée par l'entropie croisée CE appliquée sur les données de support $\mathcal{S}$ :

$$\text{CE} = -\frac{1}{|\mathcal{S}|}\sum_{i \in S}\sum_{k=1}^{K} y_{ik}\log(p_{ik}) \tag{1.55}$$

Les statistiques de $Q$ sont exploitées par l'information mutuelle entre les exemples de cette base non annotée et les étiquettes leur étant été prédites. L'information mutuelle est composée de deux termes : l'entropie marginale des étiquettes prédites : $\mathcal{H}(Y_Q)$ et l'entropie conditionnelle des étiquettes prédites connaissant les caractéristiques des donnée $Q$ : $\mathcal{H}(Y_Q|X_Q)$. Ces termes sont pondérés par l'hyperparamètre $\lambda$.

$$\mathcal{I}(X_Q;Y_Q) = \underbrace{\frac{1}{|Q|}\sum_{i\in Q}\sum_{k=1}^{K} p_{ik}\log(p_{ik})}_{\mathcal{H}(Y_Q|X_Q)\text{ : entropie conditionnelle}} - \lambda \underbrace{\sum_{k=1}^{K}\widehat{p}_k \log \widehat{p}_k}_{-\mathcal{H}(Y_Q)\text{ : entropie marginale}} \tag{1.56}$$

L'entropie conditionnelle $\mathcal{H}(Y_Q|X_Q)$ cherche à minimiser les prédictions de classe peu confiantes. Cela revient à encourager le classificateur à produire des prédictions confiantes. Tel qu'expliquent (Boudiaf *et al.* (2020)), l'optimisation de ce terme doit être effectuée avec soin, car ce dernier peut trouver son optimum près des sommets (*vertices*) du simplexe. Cette solution triviale mènerait à ce que les prédictions de tous les exemples de $Q$ soient émises dans une seule et même classe.

Le terme d'entropie marginale $\mathcal{H}(Y_Q)$ agit comme régularisateur. En étant maximisée, l'entropie marginale permet d'assurer une distribution uniforme des prédictions marginales de classe $\widehat{p}_k$. Une distribution uniforme de $\widehat{p}_k$ permet par conséquent d'éviter les solutions triviales à classe unique que peut produire l'entropie conditionnelle lorsqu'elle est optimisée seule.

La formulation complète de la fonction de coût est la suivante :

$$\mathcal{L}_{\text{TIM}} = \beta \cdot \text{CE} - \underbrace{\mathcal{I}(X_Q;Y_Q)}_{\text{Terme transductif}} \tag{1.57}$$

où $\beta$ est un hyperparamètre permettant de gérer l'importance accordée au terme de l'entropie croisée supervisée dans la fonction de coût. Les poids du classificateur sont optimisés de manière à minimiser l'entropie croisée et à maximiser l'information mutuelle entre les caractéristiques des données $Q$ et leur prédiction de classe.

## 1.6 Comparaison des méthodes étudiées

Il est intéressant de se pencher sur les performances de classification qu'offrent les différentes méthodes présentées. Afin que la comparaison des méthodes soit juste, il est important que ces dernières soient évaluées en utilisant les mêmes architectures de CNN et que la classification s'effectue sur des bases de données communes. Une comparaison juste des méthodes par méta-apprentissage a été effectuée par (Chen *et al.* (2020)). Le tableau 1.1 complète cette étude comparative avec les méthodes par transfert d'apprentissage présentées à la section 1.5.

Cette étude comparative permet d'observer que les méthodes employant des procédures de méta-entraînement compliquées se font presque toutes surpassées par les performances des méthodes suivant une procédure d'entraînement standard, soit la minimisation de l'entropie croisée sur la base d'entraînement $\mathcal{D}_{base}$. Suite à ces observations, il est pertinent de se questionner sur les efforts de recherches à mettre sur des méthodes employant des procédures d'entraînement compliquées. En effet, pour le problème d'apprentissage *Few-Shot*, il est peut-être plus intéressant de se concentrer sur la recherche des méthodes par transfert d'apprentissage qui ne nécessitent qu'une simple adaptation des paramètres lors de la phase de test. Par exemple, la méthode SimpleShot (Wang *et al.* (2019)) effectue une simple normalisation des caractéristiques extraites par $f_\theta$ et émet ses prédictions de classe à l'aide d'un classificateur à plus proche voisin. Cette approche simple, mais efficace permet d'obtenir des performances de classification significativement meilleures que les méthodes employant le méta-apprentissage.

Les méthodes présentées dans ce chapitre se distinguent par leur méthode d'inférence, soit inductive ou transductive. Le fait que les méthodes transductives aient accès et exploitent l'information des données *query* pour l'adaptation du classificateur leur permettent d'obtenir

des taux de précision de classificateur supérieurs à celles des méthodes inductives. Les gains en performances des méthodes transductives suscitent l'intérêt dans la littérature FSL.

Par contre, il est important de noter que le temps d'inférence et les coûts computationnels de certaines méthodes transductives peuvent être imposants. Prenons comme exemple la méthode de (Dhillon *et al.* (2020)) qui minimise l'entropie de manière semi-supervisée afin de mettre à jour tous les paramètres du réseau. Le temps de calcul du gradient pour effectuer la mise à jour des paramètres fait en sorte que le temps d'inférence est nettement plus grand.

Un autre point important à considérer lors de l'implémentation de méthodes transductives est d'éviter d'introduire de fausses informations a priori sur la distribution des données à classifier. Si le classificateur est construit de manière à respecter des assomptions sur la distribution des données *query*, mais que ces dernières ne les respectent pas, alors les prédictions qu'émettra ce classificateur risquent d'être erronées.

Tableau 1.1 Comparaison des performances de classification des méthodes de l'état de l'art. Les résultats de classification pour les méthodes par méta-apprentissage ont été tirés de l'étude comparative effectuée par (Chen *et al.* (2020)). Les résultats des méthodes par transfert d'apprentissage sont tirés de leur papier respectif

| | | | | ***mini*-ImageNet** | | ***tiered*-ImageNet** | | **CUB** | |
|---|---|---|---|---|---|---|---|---|---|
| | Méthodes | Type apprentissage | Arch. | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot |
| Inductive | MatchingNet (Vinyals *et al.* (2016)) | Méta | Resnet-18 | 52.9 | 68.9 | - | - | 73.5 | 84.5 |
| | ProtoNet (Snell *et al.* (2017) | Méta | | 54.2 | 73.7 | - | - | 73.0 | 86.6 |
| | Baseline (Chen *et al.* (2020)) | Transfert | | 51.8 | 74.27 | - | - | 65.5 | 82.9 |
| | Baseline++ (Chen *et al.* (2020)) | Transfert | | 51.9 | 75.7 | - | - | 67.0 | 83.6 |
| | SimpleShot (Wang *et al.* (2019)) | Transfert | | 63.1 | 79.9 | 69.7 | 84.6 | - | - |
| | SimpleShot (Wang *et al.* (2019)) | Transfert | WRN | 63.5 | 80.3 | 69.8 | 85.3 | - | - |
| Transductive | TPN (Yanbin *et al.* (2019)) | Méta | Resnet-12 | 59.5 | 75.7 | - | - | - | - |
| | Entropy-min (Dhillon *et al.* (2020)) | Transfert | | 62.4 | 74.5 | 68.4 | 83.4 | - | - |
| | MAML (Finn *et al.* (2017)) | Méta | Resnet-18 | 49.6 | 65.7 | - | - | 68.4 | 83.5 |
| | LaplacianShot (Ziko *et al.* (2020)) | Transfert | | 72.1 | 82.3 | 79.0 | 86.4 | - | - |
| | TIM (Boudiaf *et al.* (2020)) | Transfert | | 73.9 | 85.0 | 79.9 | 88.5 | 82.2 | 90.8 |
| | Entropy-min (Dhillon *et al.* (2020)) | Transfert | WRN | 65.7 | 78.4 | 73.3 | 85.5 | - | - |
| | BDCSPN (Liu *et al.* (2020a)) | Transfert | | 70.3 | 81.9 | 78.7 | 86.9 | - | - |
| | LaplacianShot (Ziko *et al.* (2020)) | Transfert | | 74.9 | 84.1 | 80.2 | 87.6 | - | - |
| | TIM (Boudiaf *et al.* (2020)) | Transfert | | 77.8 | 87.4 | 82.1 | 89.8 | - | - |
| | PT-MAP (Hu *et al.* (2020b)) | Transfert | | 82.9 | 88.8 | - | - | 91.6 | 94.0 |

# CHAPITRE 2

# DÉFINITION DU PROBLÈME

## 2.1 Motivation

Dans les dernières années, le problème d'apprentissage *Few-Shot* a suscité beaucoup d'intérêt de la part de la communauté d'apprentissage machine. Les méthodes FSL permettent aux modèles de facilement s'adapter pour la classification de nouvelles données appartenant à des classes n'ayant jamais été rencontrées lors de l'entraînement initial. Le problème FSL tente de représenter la réalité de l'apprentissage des humains. C'est-à-dire, qu'en présentant uniquement quelques exemples annotés aux modèles ces derniers devraient être en mesure de s'adapter et de correctement classifier les nouvelles données rencontrées. Le peu de données annotées requis pour l'adaptation des modèles les rendent intéressantes et applicables pour une multitude de problèmes de classification.

Afin d'évaluer la capacité de généralisation des méthodes, ces dernières sont testées sur des tâches (épisodes) tel que présenté à la section 1.1. Chaque tâche est constituée d'une base de support $\mathcal{S}$ contenant $n$ données annotées appartenant à $K$ différentes classes. La base de *query* $Q$ contient les données à classifier. Dans la littérature FSL actuelle, les données non annotées sont distribuées à occurrence égale parmi les $k$ classes. Cet environnement de test peut être remis en question par le fait qu'il ne représente pas un problème de classification réaliste. En effet, il est peu probable qu'en réalité les classes des données à classifier soient également réparties. Un problème de classification réaliste ne devrait contenir aucune présomption a priori sur la distribution des classes des exemples non annotées.

La littérature FSL s'est d'abord développée autour des méthodes inductives (Vinyals *et al.* (2016); Snell *et al.* (2017); Sung *et al.* (2018); Wang *et al.* (2019); Chen *et al.* (2020)). Tel qu'expliqué à la sous-section 1.3.1, l'inférence de ces méthodes s'effectue indépendamment de la distribution des données *query* à classifier. La distribution uniforme des données *query* n'a donc aucun impact puisque ces méthodes classifient chaque donnée de manière indépendante.

Plus récemment, les efforts de recherche ont été concentrés sur les méthodes transductives (Finn *et al.* (2017); Yanbin *et al.* (2019); Liu *et al.* (2020a); Qiao *et al.* (2019); Boudiaf *et al.* (2020); Hou *et al.* (2019); Dhillon *et al.* (2020); Guo & Cheung (2020); Hu *et al.* (2020a,b); Liu *et al.* (2020b); Wang *et al.* (2020); Yang *et al.* (2020); Ziko *et al.* (2020)). L'adaptation des modèles en exploitant les données $\mathcal{S}$ et $\mathcal{Q}$ a permis à ces derniers d'obtenir des gains en performance substantiels comparativement à celles des méthodes inductives. Par contre, les performances de certaines méthodes transductives peuvent être remises en doute dû au fait que leur implémentation repose sur le biais de distribution uniforme de classe introduit par $\mathcal{Q}$. Par exemple, la méthode TIM (Boudiaf *et al.* (2020)) exploite les probabilités marginales d'appartenance à une classe dans sa fonction de coût. La méthode PT-MAP (Hu *et al.* (2020b)) impose des contraintes basées sur l'uniformité de la distribution des données $\mathcal{Q}$. Ces méthodes performent exceptionnellement bien dans l'environnement de test actuel, mais risquent d'être fortement affectées dans un contexte où la distribution des classes est aléatoire.

Quelques travaux (Triantafillou *et al.* (2019); Lee *et al.* (2019); Chen *et al.* (2020); Ochal, Patacchiola, Storkey, Vazquez & Wang (2021)) ont étudié des méthodes FSL dans un contexte où les classes des données de la base de support $\mathcal{S}$ sont déséquilibrées. Cet axe de recherche est intéressant, car il est possible que lors d'une application réelle certaines classes comportent plus d'exemples annotés que d'autres. Par contre, ces travaux contiennent toujours le biais du balancement des classes dans $\mathcal{Q}$. Dans un problème d'application concret, il est beaucoup plus réaliste de pouvoir contrôler la distribution des classes des données annotées $\mathcal{S}$. La distribution des données à classifier devrait être inconnue, c'est pourquoi la distribution des classes de $\mathcal{Q}$ devrait être aléatoire. Au meilleur de mes connaissances, aucun travail de recherche n'a été effectué afin d'évaluer l'effet d'une distribution aléatoire de classe pour les données $\mathcal{Q}$.

Ce projet de recherche vise à reconsidérer et proposer une méthode plus réaliste pour l'évaluation des méthodes FSL. Au chapitre 3 est proposé une méthode d'échantillonnage de tâches où la distribution des classes dans $\mathcal{Q}$ est aléatoire. Cette méthode d'échantillonnage se base sur les principes de la distribution de Dirichlet. La méthode, $\alpha$-TIM est proposée au chapitre 4 afin de résoudre ce nouveau problème de classification plus réaliste. De plus, inspiré par le travail de

(Chen *et al.* (2020)), une évaluation des méthodes de l'état de l'art dans l'environnement de test proposé est présentée au chapitre 5.

## 2.2 Études de cas

Dans cette section est présentée une étude de cas sur l'introduction du biais de l'uniformité de la distribution des données $\mathcal{Q}$ par les méthodes TIM (Boudiaf *et al.* (2020)) et PT-MAP (Hu *et al.* (2020b)).

### 2.2.1 TIM

Comme il a été expliqué à la sous-section 1.5.7, (Boudiaf *et al.* (2020)) ont formulé une fonction de coût (équation 1.57) qui minimise l'entropie croisée sur les données annotées $\mathcal{S}$ et maximise l'information mutuelle (équation 1.56) entre les données $\mathcal{Q}$ et leur prédiction de classe. Le terme de l'entropie marginale $\mathcal{H}(Y_{\mathcal{Q}})$ agit comme un régularisateur afin d'éviter que la minimisation seule de l'entropie conditionnelle $\mathcal{H}(Y_{\mathcal{Q}}|X_{\mathcal{Q}})$ mène à des solutions triviales à classe unique. Ce terme pénalise le modèle lorsque les probabilités marginales de $\mathcal{Q}$ s'éloignent d'une solution balancée pour les classes. Il peut également être vu comme étant la divergence de Kullback-Leibler (KL) entre la distribution des prédictions marginales $\widehat{\boldsymbol{p}} = (\widehat{p}_1, \ldots, \widehat{p}_k)$ et la distribution de classe uniforme $\boldsymbol{u}_K = \frac{1}{K}\mathbb{1}_K$. Tel que démontré par (Veilleux *et al.* (2021)), le terme de l'entropie marginale est déviré vers la forme de la divergence KL de cette manière :

$$\mathcal{H}(Y_{\mathcal{Q}}) = -\sum_{k=1}^{K} \widehat{p}_k \log\left(\widehat{p}_k\right) = \log(K) - \mathcal{D}_{\mathrm{KL}}(\widehat{\boldsymbol{p}}\|\boldsymbol{u}_K) = \log(K) - \sum_{k=1}^{K} \widehat{p}_k \left(\frac{1}{K}\right) \tag{2.1}$$

Ce terme, bien qu'important et utile dans l'environnement de test FSL actuel est remis en question pour son application à des problèmes de classification réalistes. Dans un contexte où les données $\mathcal{Q}$ ne sont pas uniformément distribuées, ce terme risque d'impacter les performances de la méthode.

Lorsque cette méthode est évaluée sur des tâches ayant des distributions $Q$ aléatoires (procédure d'échantillonnage décrite au chapitre 5) et en utilisant les hyperparamètres recommandés par (Boudiaf *et al.* (2020)), les performances de cette dernière souffrent de pertes allant jusqu'à 20%.

### 2.2.2 PT-MAP

À la sous-section 1.5.6 est expliqué que la méthode de (Hu *et al.* (2020b)) utilise les principes du transport optimal afin de définir une matrice d'assignation de classe $\mathbf{M} \in \mathbb{R}^{|Q| \times K}$ pour chaque tâche rencontrée. Cette matrice cherche à optimiser le coût de transport entre la distribution $r$ qui est le coût d'attribution d'étiquette aux données $Q$ et la distribution $c$ qui est la quantité de données $Q$ à assigner à chaque classe $k$. L'optimisation de la matrice $\mathbf{M} \in \{\mathbf{M} : \mathbf{M}\mathbb{1}_k = r, \mathbb{1}_{|Q|}\mathbf{M} = c\}$ est effectué par l'algorithme de Sinkhorn-Knopp qui contraint les colonnes et les lignes à sommer aux vecteurs pré-définis : $r \in \mathbb{R}^{|Q|}$ et $c \in \mathbb{R}^K$. La prédéfinition de $c = \frac{1}{K}\mathbb{1}_K$ introduit la fausse assomption que les données $Q$ sont également distribuée parmi les $K$ classes. Cette méthode performe exceptionnellement bien dans l'environnement de test FSL actuel, mais risque de subir de fortes pertes de performances dans un contexte où la distribution des classes de $Q$ est aléatoire.

Lorsque cette méthode est évaluée sur des tâches ayant des distributions $Q$ aléatoires (procédure d'échantillonnage décrite au chapitre 5), les performances de cette dernière souffrent de pertes allant jusqu'à 20%.

# CHAPITRE 3

# MÉTHODE D'ÉCHANTILLONNAGE - DISTRIBUTION DE DIRICHLET

Ce chapitre présente la méthode d'échantillonnage proposée pour la génération de tâches plus réalistes et ayant une distribution de classe aléatoire. La méthode proposée se base sur les principes des distributions de Dirichlet. De plus, nous comparons notre méthode proposée avec celles présentes dans la courante littérature FSL.

## 3.1 Distribution de Dirichlet

Nous avons souligné au chapitre 2, le biais présent dans le problème FSL, soit que la distribution des classes des données $Q$ pour chaque tâche est prédéterminée : $p_k = n_k^Q/|Q| = 1/K$. La présente méthode d'échantillonnage introduit la connaissance a priori suivante : les données de chaque classe $k$ sont également distribuée dans la base de *query*. Afin de produire des tâches FSL plus réalistes nous devons proposer une méthode d'échantillonnage aléatoire et qui ne permet d'introduire aucune information a priori quant aux données $Q$ à classifier. Nous avons jugé que la distribution de Dirichlet est idéale pour nos besoins.

La distribution de Dirichlet dénoté : $\texttt{Dir}(\boldsymbol{a})$ est une loi de probabilité de Bayes qui permet de générer des distributions continues catégoriques et multinomiales ayant $K$ classes possibles ($K$-*way*). Nous considérons donc la distribution de Dirichlet afin d'obtenir les probabilités marginales (proportions) de données attribuées à chaque classe de la base de *query*. La distribution de Dirichlet se modélise sur un simplexe à $(K-1)$ dimensions : $\Delta_K = \{\boldsymbol{p} \in [0,1]^K \mid \sum_k p_k = 1\}$, où $p_k$ est le vecteur de probabilité de la classe $k$. Une variable aléatoire $P_k$ est associée à chaque vecteur de $p_k$, cette dernière représente la proportion associée à la classe $k$ sur le simplexe. La distribution de Dirichlet sur le simplexe est dénotée par le vecteur $P = (P_1, \ldots, P_K)$. La distribution de proportion $P \sim \texttt{Dir}(\boldsymbol{a})$ suit une distribution qui est paramétrée par le vecteur $\boldsymbol{a} = (a_1, \ldots, a_K) \in \mathbb{R}^K$.

La distribution de Dirichlet est une généralisation à plusieurs variables de la distribution Beta. La fonction de densité est la suivante :

$$f_{\texttt{Dir}}(\boldsymbol{p};\boldsymbol{a}) = \frac{1}{B(\boldsymbol{a})}\prod_{k=1}^{K} p_k^{a_k-1} \qquad \text{où} \qquad \boldsymbol{p} = (p_1,\ldots,p_K) \in \Delta_K \tag{3.1}$$

La fonction Beta à variables multiples est définie par la fonction Gamma :

$$B(\boldsymbol{a}) = \frac{\prod_{k=1}^{K}\Gamma(a_k)}{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)} \tag{3.2}$$

Lorsque la valeur du paramètre $a$ est plus grand que 0, la fonction Gamma est définie par :

$$\Gamma(a) = \int_0^\infty t^{a-1}\exp(-t)dt \tag{3.3}$$

lorsque la valeur du paramètre $a$ est un nombre entier positif, la fonction est définie par :

$$\Gamma(a) = (a-1)! \tag{3.4}$$

La génération de pseudo-nombres aléatoires $n_k^Q$ est possible à l'aide d'une procédure dérivant du théorème 3.1 de (Devroye, 1986, p. 594). Cette procédure simple permet d'obtenir le vecteur de probabilités aléatoire $(p_1,\ldots,p_k)$ suivant une distribution de Dirichlet paramétrée par : $\boldsymbol{a} = (a_1,\ldots,a_K)$. D'abord, il faut échantillonner de manière aléatoire $K$ exemples : $(n_1,\ldots,n_k)$. Les échantillons sont tirés d'une distribution Gamma à variable simple, cette dernière est définie par la fonction de densité suivante :

$$f_{\texttt{Gamma}}(n; a_k) = \frac{n^{a_k - 1} \exp(-n)}{\Gamma(a_k)} \qquad \text{où} \qquad n \in \mathbb{R} \tag{3.5}$$

Ensuite les proportions de chaque classe sont définis par :

$$p_k = \frac{n_k}{\sum_{k=1}^{K} n_k} \tag{3.6}$$

Le nombre d'exemples appartenant à la classe $k$ est calculé en arrondissant le résultat de la multiplication entre la proportion de classe $p_k$ au nombre total d'exemples dans la base de *query* : $n_k^Q = p_k \cdot |Q|$, où $\sum_k n_k^Q = |Q|$.

Dans le cadre de ce projet de recherche, nous considérons le cas spécial de la distribution symétrique de Dirichlet. Ce phénomène se produit lorsque la distribution est paramétrée par un vecteur composé d'éléments partageant la même valeur : $\boldsymbol{a} = a\mathbb{1}_K$, où $\mathbb{1}_K$ est un vecteur unitaire à $K$ dimensions et $a$ est le paramètre de convergence. L'avantage de la symétrie de la distribution est que chaque classe est considérée de manière égale en évitant d'en favoriser une parmi les autres. La distribution symétrique assure qu'aucune information a priori ne soit introduite, ce qui fait en sorte que la distribution produite est réellement aléatoire. La figure 3.1 illustre la densité de distributions à trois classes ($K = 3$) sur un simplexe à deux dimensions. Le simplexe est représenté par un triangle équilatéral où chaque sommet (*vertice*) est défini par le vecteur de probabilité d'une classe, soit classe 1 : $(1, 0, 0)$, classe 2 : $(0, 1, 0)$ et classe 3 : $(0, 0, 1)$. Les distributions présentées ont respectivement des paramètres de convergence $a$ de : 0.5, 2, 5 et 50. Lorsque $a \to 0$, la distribution converge vers les sommets du simplexe. De telles distributions mènent à des solutions extrêmement déséquilibrées où les données échantillonnées risquent d'appartenir à une seule et même classe. Lorsque $a \to +\infty$, la distribution converge vers le centre du simplexe. Le centre du simplexe représente une distribution uniforme parmi toutes les

classes possibles. Ce cas de figure revient à l'échantillonnage de tâches tel qu'effectué dans la courante littérature FSL, où toutes les classes sont présentes avec des proportions égales.

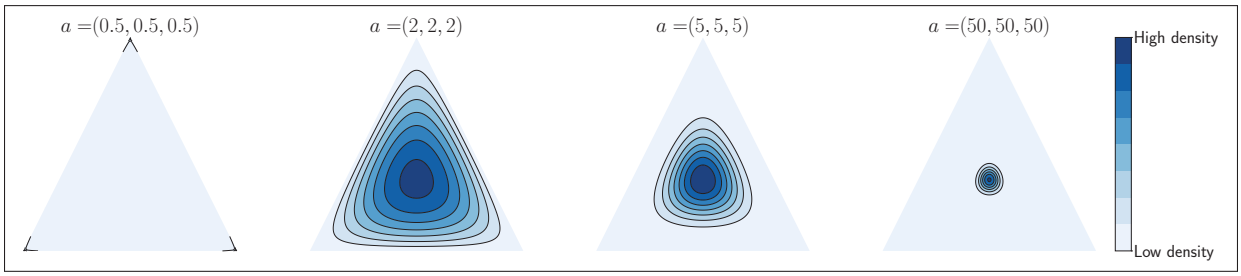


FIGURE 3.1 Fonctions de densité de Dirichlet sur un simplexe à deux dimensions ($K = 3$) pour différents vecteurs de convergence $\boldsymbol{a}$
Tirée de (Veilleux *et al.* (2021))

## 3.2 Comparaison des méthodes d'échantillonnage

Dans la littérature FSL, plusieurs méthodes d'échantillonnage ont été proposées pour étudier l'effet du déséquilibre de classe dans les bases de support $\mathcal{S}$. L'une de ces méthodes est l'échantillonnage suivant un déséquilibre *linéaire* proposée par (Ochal *et al.* (2021)) :

$$n_k^Q = \texttt{round}\left(n_{min}^Q - c + (k-1) \times \left(n_{max}^Q + 2 \times c - n_{min}^Q\right) / (K-1)\right) \tag{3.7}$$

où $k \in (1, \ldots K)$, $n_{min}^Q$ et $n_{max}^Q$ sont les nombres minimal et maximal d'exemples et $c$ est une constante pour l'arrondissement. Dans un scénario de test 5-*way*, tout en limitant le nombre $|Q|$ à 25 (tel que dans le scénario 5-*way*, 5-*shot*), l'échantillonnage de la base de support suivant un déséquilibre *linéaire* pourrait ressembler à : $\{1, 3, 5, 7, 9\}$.

Une autre méthode proposée par (Ochal *et al.* (2021)) est l'échantillonnage suivant un déséquilibre par *étapes*. Ce type de déséquilibre génère une distribution de données étant en minorité pour certaines classes et en majorité pour d'autres. L'échantillonnage est défini par :

$$n_k^Q = \begin{cases} n_{min}^Q, & \text{si } k \leq M \\ n_{max}^Q, & \text{sinon} \end{cases} \tag{3.8}$$

où $M$ est une variable qui paramètre le nombre de classes en minorité dans la distribution. Dans un scénario de test 5-*way*, tout en limitant le nombre $|Q|$ à 25 (tel que dans le scénario 5-*way*, 5-*shot*), l'échantillonnage de la base de support suivant un déséquilibre par *étapes* pourraient ressembler à : $\{1, 6, 6, 6, 6\}$.

Ces méthodes d'échantillonnage permettent bel et bien d'introduire un déséquilibre parmi les classes de $\mathcal{S}$ et auraient également pu être utilisées pour l'échantillonnage de $Q$. Par contre, en connaissant la forme du déséquilibre de la distribution, il reste possible de l'exploiter comme étant une information a priori. Pour illustrer ces propos, la figure 3.2 présente un simplexe à trois classes ($K = 3$) où sont superposés la densité de distribution de Dirichlet et de possibles distributions suivant un déséquilibre *linéaire* et par *étapes*. Malgré que les distributions déséquilibrées *linéairement* et par *étapes* couvrent une certaine région du simplexe, il est toujours possible de déterminer approximativement la tendance de déséquilibre de leur échantillonnage. La distribution de Dirichlet permet d'échantillonner les proportions de classe de manière plus aléatoire sur le simplexe entier, tout en limitant l'accès à la connaissance de la forme du déséquilibre. C'est pour ces raisons que nous jugeons que l'échantillonnage de $Q$ à l'aide de la distribution de Dirichlet reflète mieux les problèmes de classification réels.

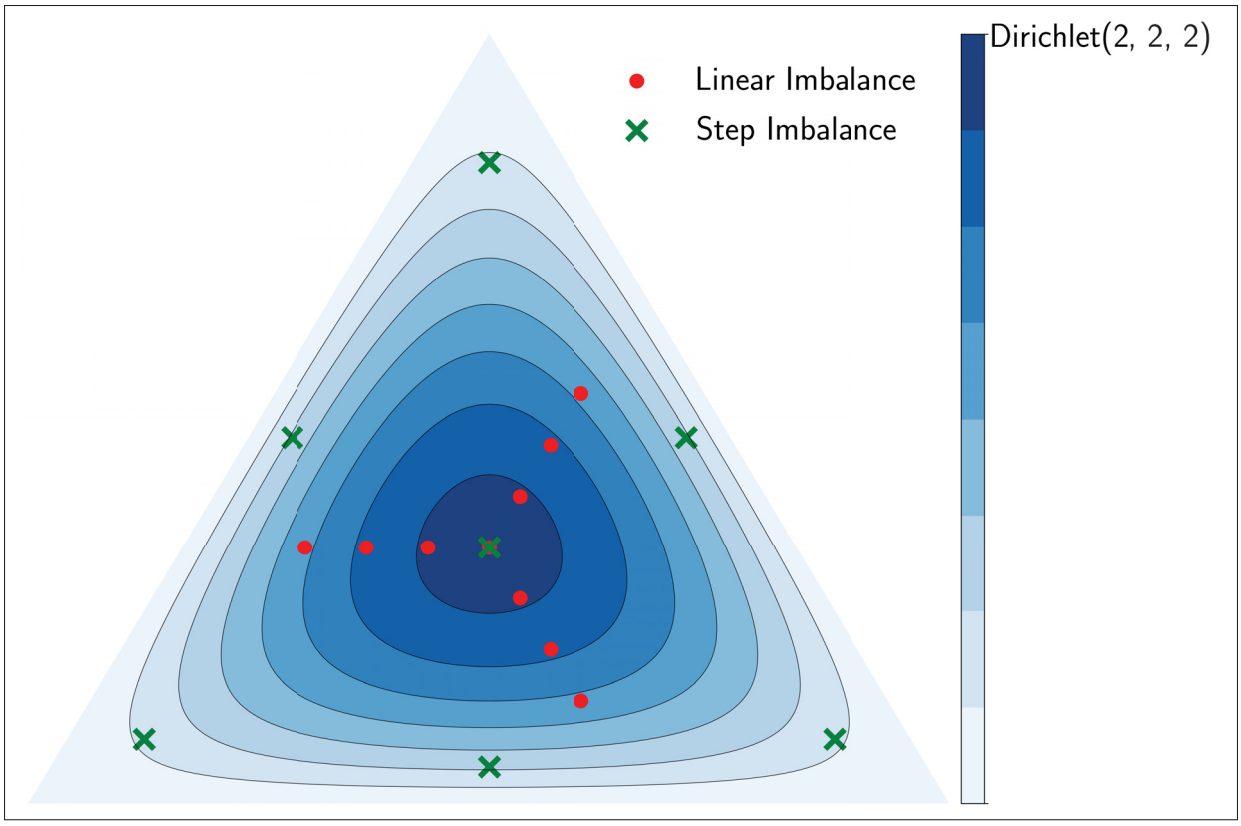


FIGURE 3.2 Comparaison de la distribution de Dirichlet avec des distributions suivant un déséquilibre *linéaire* et par *étapes*
Tirée de (Veilleux *et al.* (2021))

# CHAPITRE 4

# MÉTHODE PROPOSÉE $\alpha$-TIM

Dans ce chapitre est présentée notre méthode transductive : $\alpha$-TIM, afin de résoudre le problème FSL où les tâches de classification sont aléatoirement échantillonnées en suivant une distribution de Dirichlet (décrite au chapitre 3).

Notre méthode $\alpha$-TIM est fortement reliée à la méthode TIM (Boudiaf *et al.* (2020)) qui exploite l'information mutuelle dans sa fonction de coût (équation 1.57). Nous avons expliqué la faiblesse de cette méthode à la sous-section 2.2.1, le biais provenant du terme de l'entropie marginale $\mathcal{H}(Y_Q)$ qui encourage des solutions strictement uniformes. Ce terme affecte négativement les performances de la méthode lorsque cette dernière est évaluée sur des tâches où les proportions de représentation de classe sont aléatoires. La diminution du poids $\lambda$ associé à ce terme, tel qu'effectué lors de la validation des paramètres (section 5.4), atténue les pertes de performances, mais n'est pas suffisant pour l'obtention de résultats très compétitifs. Nous proposons donc une généralisation de la fonction de coût de TIM par l'utilisation d'une forme d'entropie différente.

Le terme d'entropie marginale $\mathcal{H}(Y_Q)$ dans la méthode TIM est défini par l'entropie de Shannon, dont la forme favorise fortement les solutions balancées. Nous proposons de conserver ce terme de régularisation afin d'éviter les solutions triviales à classe unique, mais de le définir en utilisant une forme d'entropie différente. La forme de l'entropie considérée peut permettre une plus grande flexibilité de déviation de la solution uniforme.

Tel qu'expliqué à la sous-section 2.2.1, l'entropie marginale suivant la forme de Shannon peut être interprétée comme étant la divergence KL entre les prédictions marginales $\widehat{\boldsymbol{p}}$ et la distribution uniforme $\boldsymbol{u}_K = \frac{1}{K}\mathbb{1}_K$. Nous proposons d'utiliser la divergence-$\alpha$ de (Tsallis (1988)), premièrement introduite par (Havrda & Charvát (1967)). Cette dernière est une généralisation de la divergence KL. La divergence-$\alpha$ entre la distribution des prédictions : $\boldsymbol{p} = (p_1, \ldots, p_K)$ et des données : $\boldsymbol{q} = (q_1, \ldots, q_K)$ est :

$$\mathcal{D}_\alpha(\boldsymbol{p}\|\boldsymbol{q}) = -\sum_{k=1}^{K} p_k \log_\alpha\left(\frac{q_k}{p_k}\right) = \frac{1}{1-\alpha}\left(1 - \sum_{k=1}^{K} p_k^\alpha q_k^{1-\alpha}\right) \tag{4.1}$$

où le logarithme-$\alpha$ (Cichocki & Amari (2010)) est défini par : $\log_\alpha(x) = \frac{1}{1-\alpha}\left(x^{1-\alpha} - 1\right)$, pour $x > 0$. Le terme d'entropie marginale est généralisé vers la forme $\alpha$ de Tsallis par :

$$\mathcal{H}_\alpha(\boldsymbol{p}) = \log_\alpha(K) - K^{1-\alpha}\,\mathcal{D}_\alpha(\boldsymbol{p}\|\boldsymbol{u}_K) = \frac{1}{\alpha-1}\left(1 - \sum_{k} p_k^\alpha\right) \tag{4.2}$$

(Cichocki & Amari (2010)) explique que la divergence KL est généralisée en utilisant le logarithme-$\alpha$, dont la limite lorsqu'$\alpha$ tend vers 1 est la fonction logarithme : $\lim_{\alpha\to 1} \log_\alpha(x) = \log(x)$. Cette propriété permet de prouver que la divergence-$\alpha$ ainsi que l'entropie-$\alpha$ découlent bien de la divergence KL et de l'entropie de Shannon :

$$\lim_{\alpha\to 1} \mathcal{D}_\alpha(\boldsymbol{p}\|\boldsymbol{q}) = \mathcal{D}_{\mathrm{KL}}(\boldsymbol{p}\|\boldsymbol{q}) \quad \text{et} \quad \lim_{\alpha\to 1} \mathcal{H}_\alpha(\boldsymbol{p}) = \mathcal{H}(\boldsymbol{p}) = -\sum_{k=1}^{K} \widehat{p}_k \log\left(\widehat{p}_k\right) \tag{4.3}$$

Comme la divergence-$\alpha$ est une généralisation de la divergence KL, ces dernières partagent les propriétés suivantes (Cichocki & Amari (2010)) :

1. Convexité par rapport à $\boldsymbol{p}$ et $\boldsymbol{q}$, mais sans s'y limiter
2. La divergence est strictement positive : $\mathcal{D}_\alpha(\boldsymbol{p}\|\boldsymbol{q}) \geq 0$ et est égale à 0 : $\mathcal{D}_\alpha(\boldsymbol{p}\|\boldsymbol{q}) = 0$, si $\boldsymbol{p} = \boldsymbol{q}$

De plus la manipulation de la valeur du paramètre $\alpha$ est ce qui rend divergence-$\alpha$ flexible. Nous retrouvons : l'inverse de la divergence KL $\mathcal{D}_{KL}(\boldsymbol{q}\|\boldsymbol{p})$ lorsqu'$\alpha$ tend vers 0, la distance

d'Hellinger $\mathcal{D}_H(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{2}\sum_{k=1}^{K}\left(\sqrt{p_k} - \sqrt{q_k}\right)^2$ (Amari & Nagaoka (2000)) lorsqu'$\alpha$ tend vers 0.5 et la distance Pearson Chi-square $\mathcal{D}_{P_{C-s}}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{2}\sum_{k=1}^{K}\frac{\left(p_k^2 - q_k^2\right)^2}{q_k}$ (Pearson (1900)) lorsqu'$\alpha$ tend vers 2.

L'entropie-$\alpha$ telle que définie à l'équation 4.2 introduit toujours le biais de l'uniformité de la solution, par contre permet de l'atténuer grâce à sa forme modulable. La figure 4.1 présente un exemple de régression logistique binaire de la forme de l'entropie-$\alpha$ $\mathcal{H}_\alpha$ et de celle de son gradient $\partial\mathcal{H}_\alpha/\partial l$ en fonction de prédictions $\boldsymbol{p} = \{p, 1-p\}$, où $p$ est le résultat de la fonction d'activation sigmoïde $\sigma$ appliquée sur les logits : $l \in \mathbb{R}$, $p = \sigma(l)$. Nous pouvons observer que plus la valeur d'$\alpha$ augmente, plus les gradients des prédictions confiantes sont considérés. En effet, l'adaptation des poids du modèle par la rétropropagation du gradient s'effectuera en considérant les prédictions les plus fiables. L'entropie de Shannon ($\alpha \to 1$) génère de grands gradients pour des prédictions qui sont encore peu confiantes. Par exemple, lorsque $p = 0.6$, une grande valeur de gradient sera rétropropagée pour la mise à jour des poids du modèle. L'ampleur d'importance accordée aux prédictions peu confiantes lors de la mise à jour des poids peut mener à des solutions sous-optimales surtout dans un contexte où les données $\boldsymbol{Q}$ sont aléatoirement distribuées. L'entropie-$\alpha$ permet de réduire l'ampleur du gradient accordé aux prédictions encore trop peu confiantes. Prenons comme exemple lorsqu'$\alpha$ égal à 11. L'entropie a une forme aplatie pour les prédictions encore peu confiantes ($p = [0.2; 0.8]$), mais présente une forme fortement abrupte lorsque les prédictions approchent l'un des sommets du simplexe. La forme de cette entropie permet d'effectuer la mise à jour des poids en rétropropageant des gradients imposants uniquement pour les prédictions confiantes.

L'entropie-$\alpha$ où $\alpha > 1$ est adéquate pour le contexte de test où les données $\boldsymbol{Q}$ sont aléatoirement distribuée. Cette entropie offre la flexibilité de diverger de la solution uniforme ($p = 0.5$) tout en agissant comme régularisateur pour éviter les solutions triviales à classe unique.

Pour ces raisons, nous proposons de redéfinir le terme de l'information mutuelle (équation 1.56) en utilisant l'entropie-$\alpha$ (équation 4.2) afin d'offrir plus de flexibilité face au déséquilibre des classes de $\boldsymbol{Q}$.

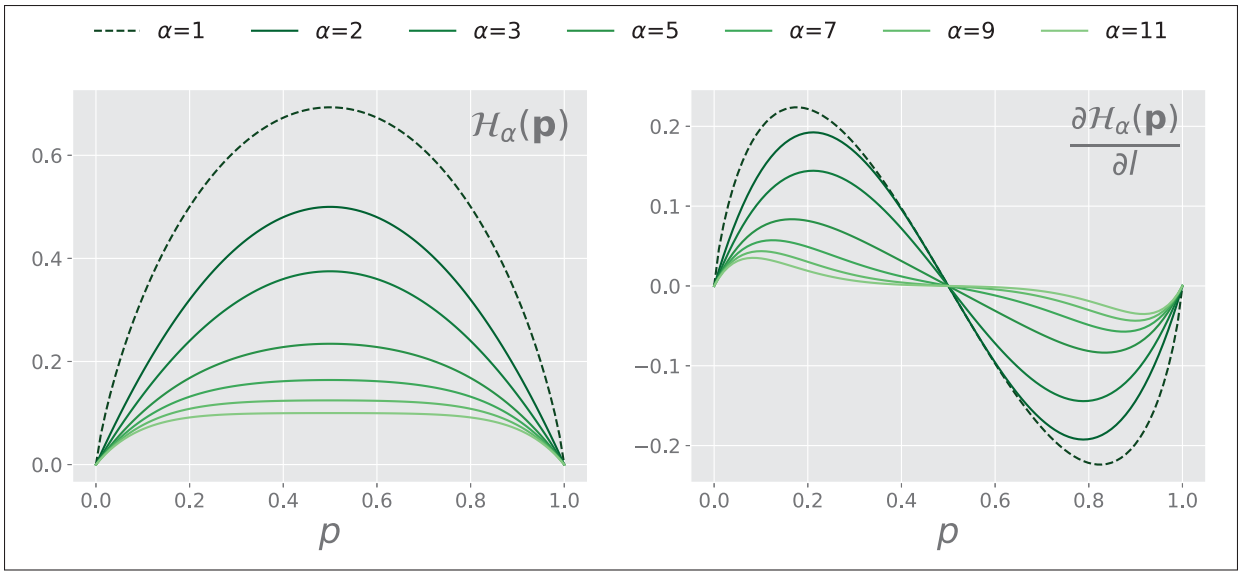


FIGURE 4.1 (Gauche) L'entropie-$\alpha$ $\mathcal{H}_\alpha(\boldsymbol{p})$ en fonction des probabilités $\boldsymbol{p}$. (Droite) Le gradient de l'entropie-$\alpha$ qui est la dérivé $\partial\mathcal{H}_\alpha(\boldsymbol{p})/\partial l$ en fonction des logits $l$. Chacune des mesures sont présentées pour différentes valeurs d'$\alpha$. Dans les deux cas, $p = \sigma(l)$ et $l \in \mathbb{R}$
Tirée de (Veilleux *et al.* (2021))

$$\begin{aligned}\mathcal{I}_\alpha(X_Q;Y_Q) &= \mathcal{H}_\alpha(Y_Q) - \mathcal{H}_\alpha(Y_Q|X_Q)\\ &= \frac{1}{\alpha-1}\left(\underbrace{\frac{1}{|I_Q|}\sum_{i\in I_Q}\sum_{k=1}^{K} p_{ik}^{\alpha}}_{-\mathcal{H}_\alpha(Y_Q|X_Q)} - \underbrace{\sum_{k=1}^{K}\widehat{p}_k^{\,\alpha}}_{\mathcal{H}_\alpha(Y_Q)}\right)\end{aligned} \tag{4.4}$$

La fonction de coût de notre méthode $\alpha$-TIM est dénotée :

$$\mathcal{L}_{\alpha\text{-TIM}} = \text{CE} - \mathcal{I}_\alpha(X_Q;Y_Q) \tag{4.5}$$

L'évaluation de notre méthode est effectuée au chapitre 5 et ses performances sont comparées à celles des méthodes de l'état de l'art.

# CHAPITRE 5

# EXPÉRIMENTATIONS ET RÉSULTATS

Ce chapitre présente les procédures et résultats expérimentaux. L'évaluation de toutes les méthodes a été effectuée dans un *framework* commun, le code est disponible dans le répertoire GitHub suivant : https://github.com/oveilleux/Realistic_Transductive_Few_Shot. Seulement les méthodes SIB [1] (Hu *et al.* (2020a)) et LR+ICI [2] (Wang *et al.* (2020)) ont été évaluées en utilisant le code fourni dans leur répertoire GitHub respectif.

## 5.1 Bases de données

Les méthodes de l'état de l'art ainsi que notre méthode proposée sont évaluées sur trois bases de données de référence pour la classification *few-shot* : *mini*-Imagenet (Russakovsky *et al.* (2015)), *tiered*-Imagenet (Ren *et al.* (2018)) et *Caltech-UCSD Birds 200* (CUB) (Wah, Branson, Welinder, Perona & Belongie (2011)).

### 5.1.1 *mini*-Imagenet

La base *mini*-Imagenet est un sous-ensemble de la base ILSVRC-12 (Russakovsky *et al.* (2015)). *mini*-Imagenet est composé de 60000 images couleur ayant des dimensions de 84 par 84 pixels (Vinyals *et al.* (2016)). Cette base comprend 100 classes où 600 images appartiennent à chaque classe. La base de données est divisée en sous-ensemble selon les divisions standard proposées par (Ravi & Larochelle (2017); Wang *et al.* (2019)), 64 classes sont utilisées pour le préentraînement du modèle, 16 pour la validation et 20 classes pour la phase de test. Les classes des bases de préentraînement, de validation et de test sont toutes différents ($\mathcal{Y}_{base} \cap \mathcal{Y}_{val} \cap \mathcal{Y}_{test} = \emptyset$), cela est également appliqué lors de la division de *tiered*-Imagenet et CUB.

[1] https://github.com/hushell/sib_meta_learn

[2] https://github.com/Yikai-Wang/ICI-FSL

### 5.1.2 *tiered*-Imagenet

La base *tiered*-Imagenet est également un sous-ensemble de ILSVRC-12. Cette dernière est par contre plus imposante que *mini*-Imagenet, elle est composée de 779165 images appartenant à 608 classes. Ces images sont aussi en couleur et ont des dimensions de 84 par 84 pixels. Les divisions standards (Ren *et al.* (2018)) utilisées comprennent 351 classes pour le préentraînement du modèle, 97 classes pour la validation et 160 classes pour la phase de test.

### 5.1.3 *Caltech-UCSD Birds 200* (CUB)

La base *Caltech-UCSD Birds 200* (CUB) est constituée de 11788 images appartenant à 200 classes. Ces images couleur ont des dimensions de 84 par 84 pixels. Les divisions proposées par (Chen *et al.* (2019)) sont utilisées, soient 100 classes pour le préentraînement du modèle, 50 classes pour la validation et 50 classes pour la phase de test.

## 5.2 Réseaux de neurones

Les méthodes sont évaluées en utilisant deux architectures de réseaux de neurones fortement utilisés par la communauté FSL. Les CNN considérés sont le ResNet-18 et le WRN28-10.

### 5.2.1 ResNet-18

Ce CNN introduit par (He *et al.* (2016)) a une architecture résiduelle profonde. L'un des problèmes rencontrés lors de l'implémentation de modèles profonds (avec beaucoup de couches) est la disparition du gradient. Les auteurs ont proposé d'utiliser des blocs résiduels afin maintenir un gradient significatif, ce malgré le nombre de couches considérées. Tel qu'illustré à la figure 5.1 le raccourci permet d'additionner les valeurs d'entrées aux résidus (valeurs de sorties) d'un bloc. La minimisation de perte d'information permet l'implémentation de modèles plus profonds.

Le modèle ResNet-18 considéré est composé de huit blocs résiduels. Tel qu'il a été fait par (Ziko *et al.* (2020); Boudiaf *et al.* (2020)), nous retirons les deux premières couches de sous-

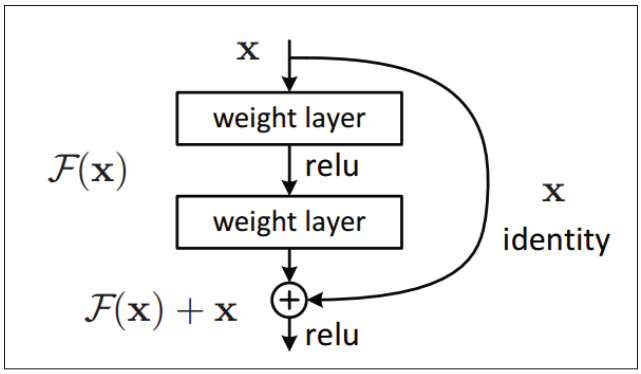


FIGURE 5.1 Bloc résiduel
Tirée de (He *et al.* (2016)))

échantillonnage, le pas (*stride*) de la première couche convolutionnelle est fixé à un et la première couche de *max-pooling* est retirée. À la sortie du modèle, les caractéristiques des données ont une dimension de 512.

### 5.2.2 WRN28-10

Les réseaux résiduels larges (*Wide Residual Networks* (WRN)) (Zagoruyko & Komodakis (2016)) sont inspirés des réseaux résiduels. Les auteurs ont proposé d'élargir les blocs résiduels en leur ajoutant plus de couches convolutionnelle. Cette manipulation permet de réduire la profondeur du modèle en compensant par une plus grande largeur. Le WRN28-10 utilisé comporte 28 couches convolutionnelles et des blocs résiduels ayant une largeur de 10. À la sortie du modèle, les caractéristiques des données ont une dimension de 640.

### 5.2.3 Procédure de pré-entraînement

Afin d'effectuer une comparaison juste, l'évaluation de toutes les méthodes par transfert d'apprentissage est effectuée en utilisant les mêmes extracteurs de caractéristiques (ResNet-18 et WRN28-10). L'entraînement de ces modèles a été effectué en suivant les mêmes procédures employées par (Ziko *et al.* (2020); Boudiaf *et al.* (2020)). Les modèles sont entraînés de manière supervisée par la minimisation de l'entropie croisée sur $\mathcal{D}_{base}$ et en appliquant un lissage des

étiquettes (*label smoothing*) dont l'hyperparamètre utilisé est : 0.1. La minimisation de CE est assuré par l'algorithme de la descente du gradient stochastique (SGD) pour 90 epochs avec un taux d'apprentissage initialisé à 0.1 et divisé par 10 aux epochs 45 et 66. L'entraînement du ResNet-18 a été effectué en lui présentant des mini-batch de taille 256 et celui du WRN28-10 avec des mini-batchs de taille 128. De plus, les données de $\mathcal{D}_{base}$ ont été augmentés par l'application de recadrages aléatoires, des *flips* horizontaux aléatoires et de l'altercation de couleurs (*color jittering*).

Nous avons utilisé les modèles préentraînés fournis dans les répertoires GitHub propres à chaque méthode par méta-apprentissage évaluée. Nous avons opté pour cette option dû au fait les méthodes par méta-apprentissage ont une procédure d'entraînement qui leur est spécifique. L'utilisation des modèles préentraînés fournis permettait de s'assurer que l'entraînement ait correctement été effectué .

## 5.3 Échantillonnage des tâches

Les méthodes ont été évaluées selon les scénarios de test suivants, 1-*shot*, 5-*shot*, 10-*shot*, 20-*shot* et ce toujours pour une classification à 5 classes possibles : 5-*way*. Pour chaque tâche, le nombre total d'exemples dans la base de *query* $|Q|$ est de 75. Les classes des données de la base de *query* $Q$ sont aléatoirement distribuées en suivant une distribution de Dirichlet. Durant la phase de test, la distribution de classes de $Q$ est échantillonnée en utilisant le paramètre $\boldsymbol{a} = 2 \cdot \mathbb{1}_K$. La distribution produite en utilisant ce paramètre permet de générer des tâches qui sont aléatoirement distribuées sur le simplexe (voir figure 5.2), tout en évitant les cas de déséquilibre extrême, par exemple que tous les exemples de $Q$ soient échantillonnés d'une seule et même classe.

## 5.4 Hyperparamètres

Durant la phase de validation, la distribution des classes de $Q$ est échantillonnée en utilisant le paramètre $\boldsymbol{a} = \mathbb{1}_K$. Ce paramètre permet de générer une distribution aléatoire pouvant se

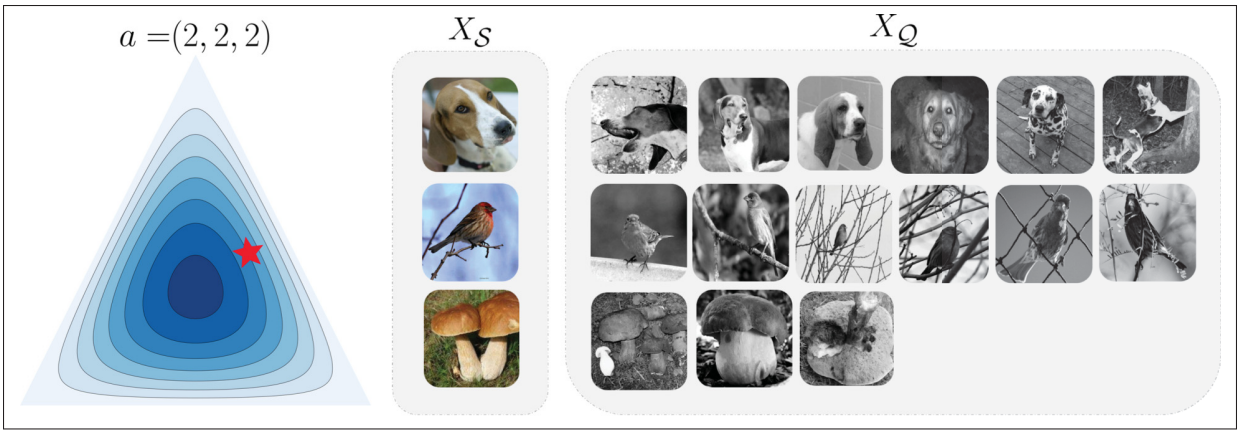


FIGURE 5.2 Exemple d'échantillonnage d'une tâche à 3 classes ($K = 3$) représentée par l'étoile rouge sur le simplexe. Pour cet exemple, $|Q| = 15$

trouver partout sur le simplexe. Nous avons opté pour l'utilisation d'un paramètre différent lors de la phase de validation et de test afin d'éviter d'introduire quelconque information a priori par rapport à la distribution des classes de $Q$.

Les hyperparamètres des méthodes évaluées ont été fixés aux valeurs recommandées dans leur papier respectif. Seulement les hyperparamètres influant sur le balancement de classes ont été validés et optimisés. Dans le cas de la méthode TIM (Boudiaf *et al.* (2020)), tel qu'expliqué à la sous-section 2.2.1, le terme de l'entropie marginal peut forcer le modèle à produire des solutions qui tendent vers le balancement de classes. Le poids $\lambda$ de ce terme a donc été optimisé afin que ce dernier n'impacte pas négativement les résultats de la méthode.

L'hyperparamètre optimisé pour notre méthode proposée $\alpha$-TIM est $\alpha$. Ce paramètre contrôle la forme de la divergence. Nous utilisons la même valeur optimisée $\lambda$ que celle de la méthode TIM dans notre généralisation $\alpha$-TIM.

Les résultats de validation des hyperparamètres de TIM et d'$\alpha$-TIM sont présentés aux figures 5.3, 5.4, 5.5 et 5.6.

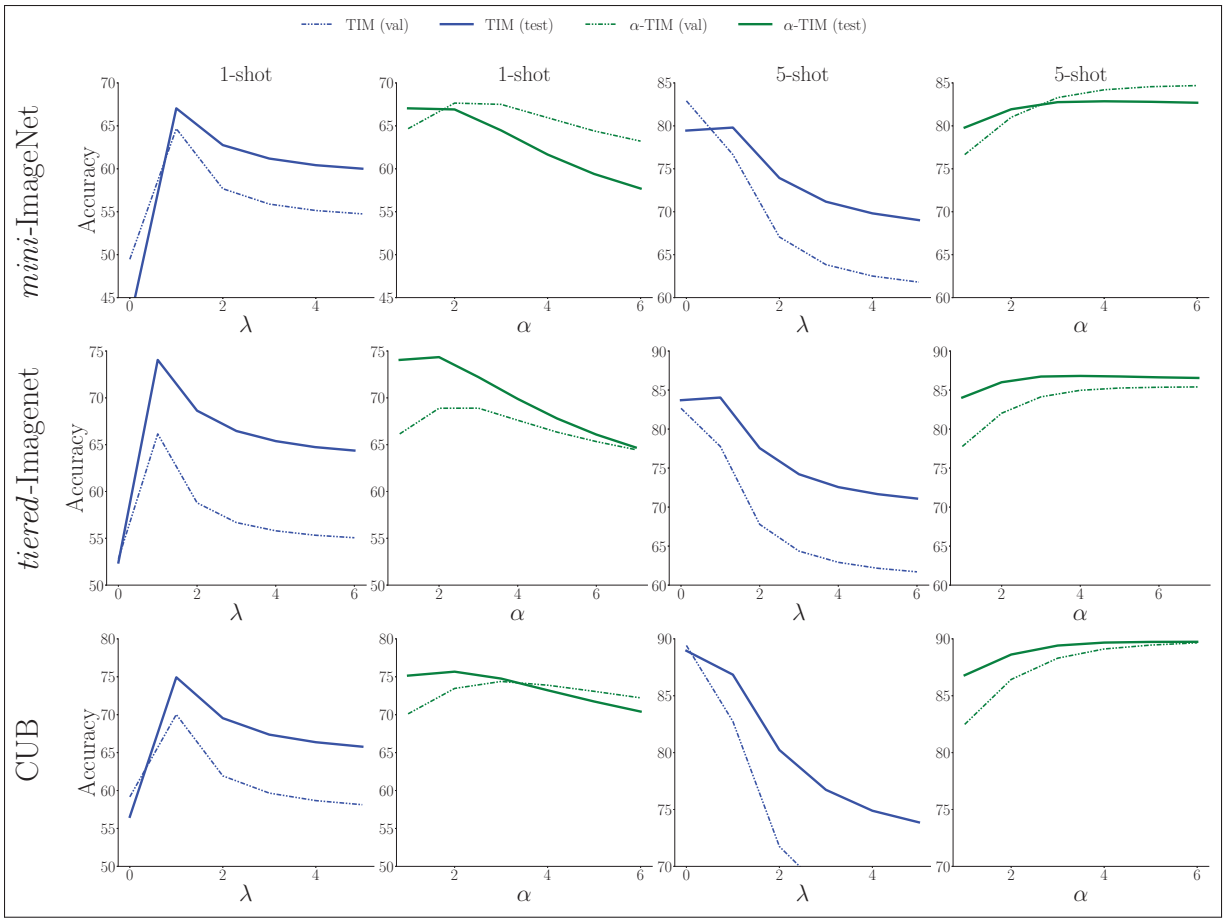


FIGURE 5.3 Comparaison du taux de classification (*accuracy*) de validation et de test en fonction de $\lambda$ pour TIM (Boudiaf *et al.* (2020)) et de $\alpha$ pour $\alpha$-TIM. Les résultats ont été obtenus dans des scénarios 1-*shot* et 5-*shot* en utilisant le ResNet-18 comme extracteur de caractéristiques
Tirée de (Veilleux *et al.* (2021))

## 5.5 Résultats

Les tableaux 5.1 et 5.2 présentent les résultats des méthodes de l'état de l'art. Les bases de *query* ont été échantillonnées en suivant la procédure décrite à la section 5.3. Toutes les méthodes ont été évaluées sur plus de 10000 tâches et les taux de précision présentés sont les moyennes de ces résultats. Les flèches rouges (↓) indiquent une perte de performance entre l'environnement où les classes de $Q$ sont artificiellement équilibrées et notre environnement de test réaliste. Les flèches bleues (↑) indiquent au contraire un gain de performance. Le fait qu'une méthode ne présente

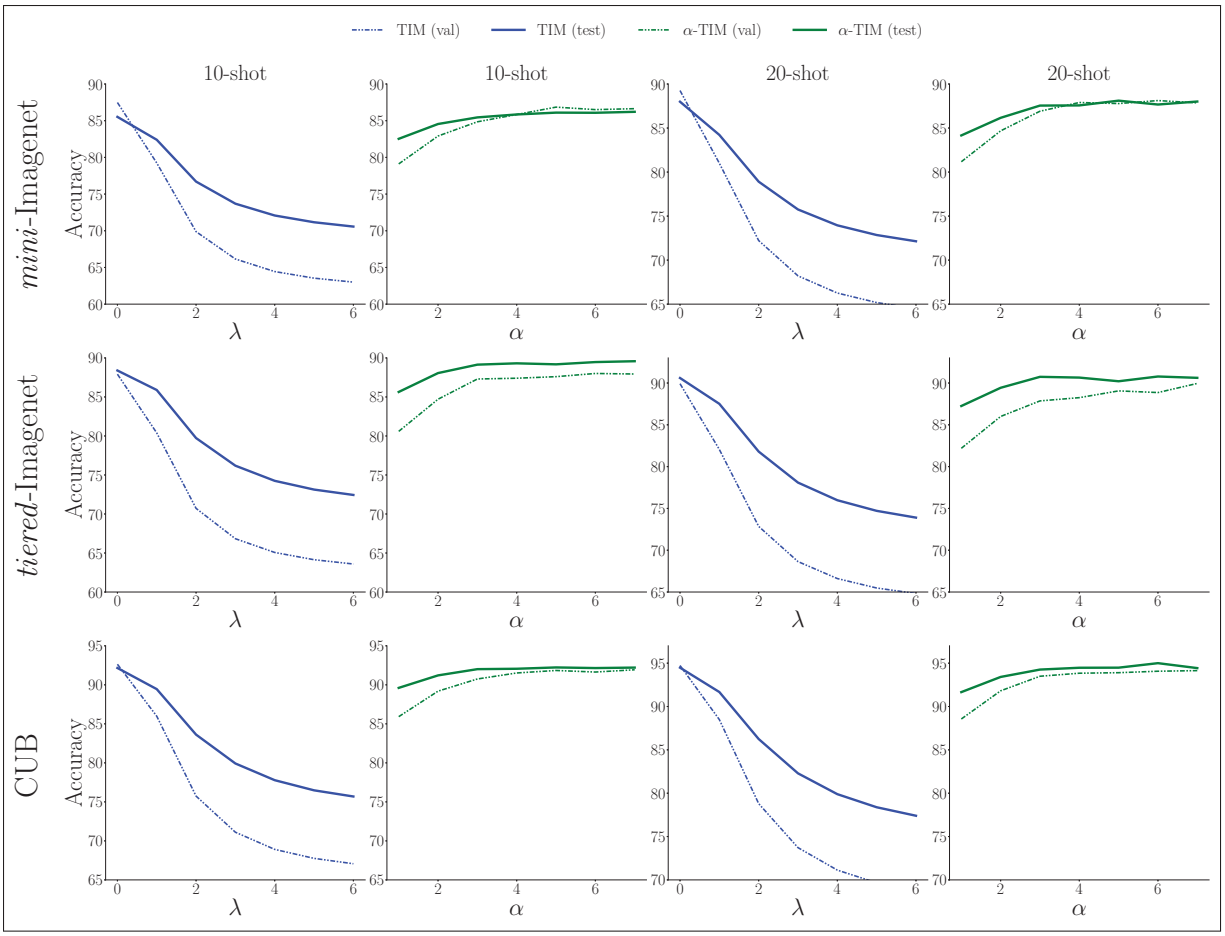


FIGURE 5.4 Comparaison du taux de classification (*accuracy*) de validation et de test en fonction de $\lambda$ pour TIM (Boudiaf *et al.* (2020)) et de $\alpha$ pour $\alpha$-TIM. Les résultats ont été obtenus dans des scénarios 10-*shot* et 20-*shot* en utilisant le ResNet-18 comme extracteur de caractéristiques
Tirée de (Veilleux *et al.* (2021))

aucune flèche signifie que ses performances restent constantes, ce peu importe l'environnement de test.

Les performances des méthodes inductives ne sont pas affectées dans notre environnement de test. Tel qu'expliqué à la sous-section 1.3.1, les méthodes inductives n'exploitent par les statistiques de $Q$ lors de la construction de leur classificateur, cela fait en sorte que la distribution de $Q$ n'a pas d'impact sur leurs performances. Les expérimentations confirment le fait que ces

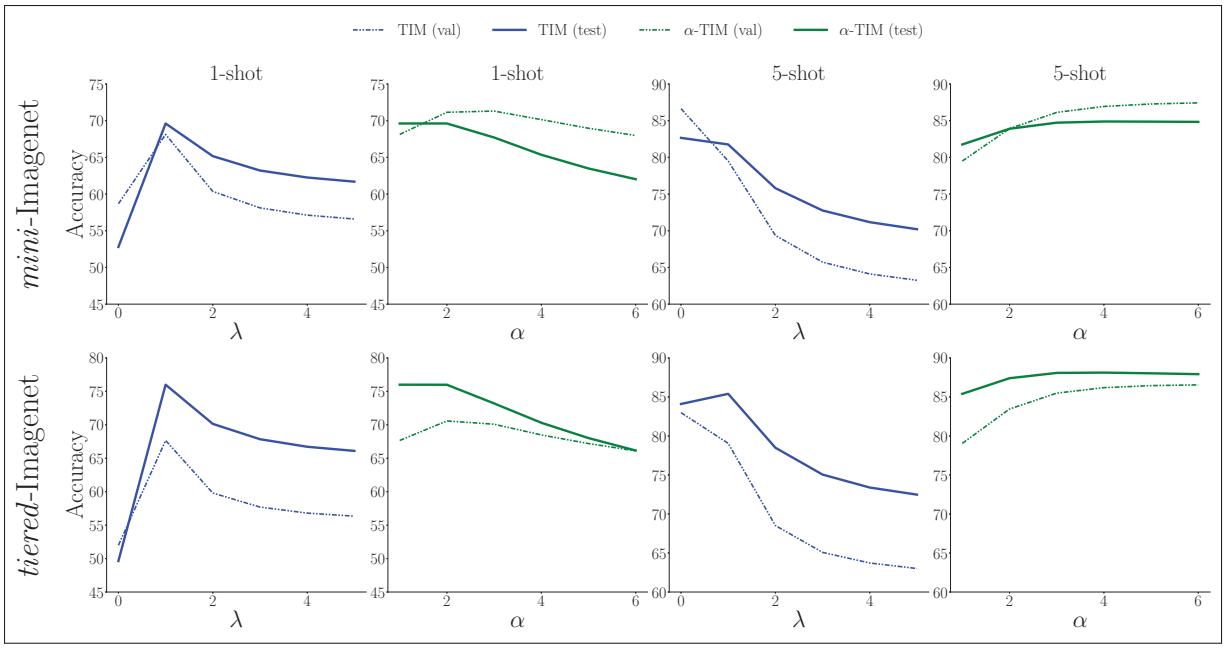


FIGURE 5.5 Comparaison du taux de classification (*accuracy*) de validation et de test en fonction de $\lambda$ pour TIM (Boudiaf *et al.* (2020)) et de $\alpha$ pour $\alpha$-TIM. Les résultats ont été obtenus dans des scénarios 1-*shot* et 5-*shot* en utilisant le WRN28-10 comme extracteur de caractéristiques
Tirée de (Veilleux *et al.* (2021))

méthodes sont robustes à faire face à des scénarios de classification où la distribution des classes est aléatoire.

Les expérimentations ont également permis de confirmer que les méthodes transductives dépendent fortement de la distribution de $Q$. Toutes les méthodes sans exception ont subi une perte de performances lorsqu'évaluées dans l'environnement où les classes de $Q$ sont aléatoirement distribuées. Dans la présente littérature FSL, beaucoup d'intérêt est porté aux méthodes transductive dû au fait que l'exploitation des statistiques de $Q$ mène à de forts gains en performance comparativement aux méthodes inductives. On peut observer que dans notre environnement de test réaliste, plus de la moitié des méthodes transductives ont des performances inférieures que celles des méthodes inductives.

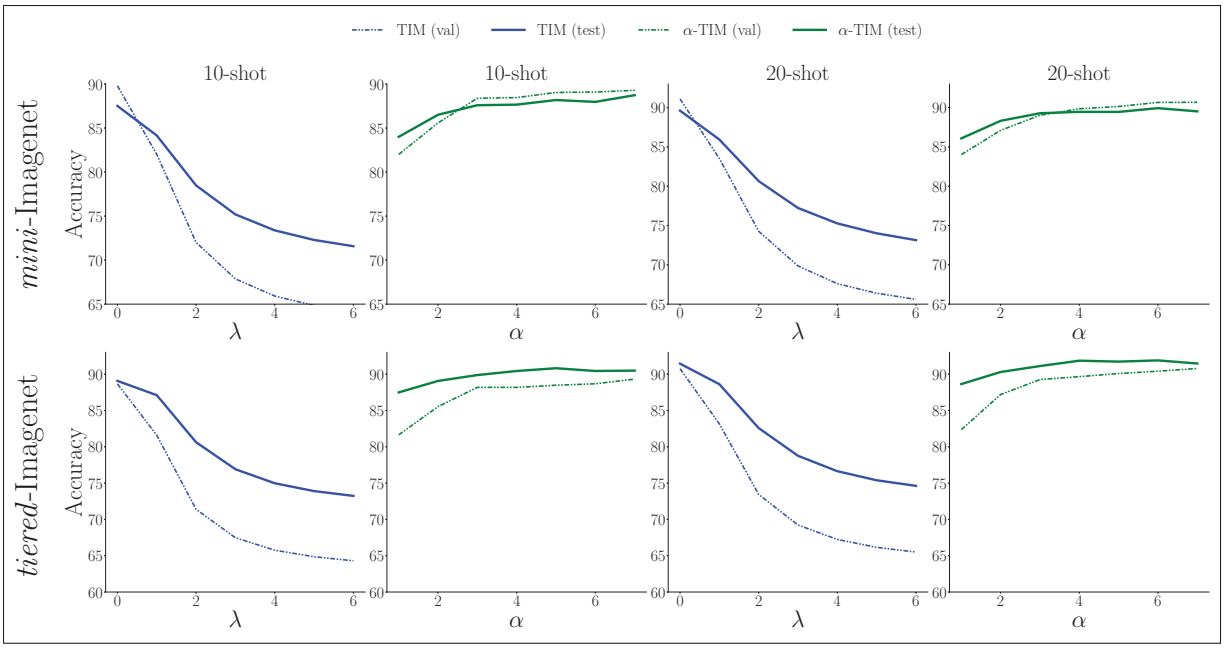


FIGURE 5.6 Comparaison du taux de classification (*accuracy*) de validation et de test en fonction de $\lambda$ pour TIM (Boudiaf *et al.* (2020)) et de $\alpha$ pour $\alpha$-TIM. Les résultats ont été obtenus dans des scénarios 10-*shot* et 20-*shot* en utilisant le WRN28-10 comme extracteur de caractéristiques
Tirée de (Veilleux *et al.* (2021))

De plus, deux des méthodes transductives les plus compétitives de la littérature FSL soient PT-MAP et TIM présentent une perte de performance imposante. Parmi les expérimentations sur les trois bases de données et en utilisant les deux architectures de CNN, PT-MAP perd en moyenne 18% lorsqu'évaluée de manière réaliste. Malgré l'optimisation de l'hyperparamètre $\lambda$ pouvant influencé négativement les performances de TIM, la méthode perd en moyenne 4%.

Les expérimentations confirment la remise en question de l'environnement de test utilisé dans la présente littérature FSL. Les résultats prouvent que les méthodes transductives dépendent fortement de la distribution de $Q$ et que l'implémentation de certaines d'entre elles s'appuie sur le biais de l'environnement de test où la distribution des classes de $Q$ est artificiellement uniforme.

Notre méthode $\alpha$-TIM obtient dans presque tous les scénarios de test les meilleurs résultats. Cette dernière peut être considérée comme un bon point de départ pour les futures recherches du FSL transductif évalué de manière réaliste.

## 5.6 Étude d'ablation

Afin de présenter une analyse plus approfondie, d'autres expérimentations ont été conduites. Nous avons effectué une étude de la corrélation entre la sévérité du déséquilibre des classes de $Q$ et les performances des méthodes transductives. Pour se faire, les méthodes ont été évaluées selon les scénarios de test 5-*shot*, 5-*way* sur les trois bases de données et en utilisant le ResNet-18 comme extracteur de caractéristiques. Le niveau de déséquilibre de la distribution a été contrôlé en faisant varier le paramètre $\boldsymbol{a}$, où plus $\boldsymbol{a}$ est petit, plus la distribution est déséquilibrée (voir figure 3.1). Les hyperparamètres sont maintenus aux valeurs déterminées par le protocole présenté à la section 5.4. Les résultats de la figure 5.7 démontrent que la majorité des méthodes transductives subissent de fortes pertes en performances lors de l'augmentation de la sévérité du déséquilibre de la distribution de $Q$. De plus, à partir d'un certain seuil de déséquilibre les performances de la méthode inductive SimpleShot (Wang *et al.* (2019)) surpassent celles de la majorité des méthodes transductives évaluées. Tel qu'escompté, SimpleShot est robuste face au changement de déséquilibre de la distribution $Q$. En effet, les performances de la méthode inductive sont stables et ne sont pas affectées par la distribution de classes de $Q$. Notre méthode $\alpha$-TIM offre les meilleurs résultats dans un scénario où la distribution $Q$ est fortement déséquilibrée. Les performances de cette dernière diminuent un peu et se maintiennent stables lorsque $Q$ tend vers une distribution uniforme.

Les résultats de la figure 5.7 démontrent également la flexibilité d'$\alpha$-TIM dans l'environnement de test conventionnel ($Q$ uniforme). Notre méthode $\alpha$-TIM est une généralisation de TIM (Boudiaf *et al.* (2020)) qui permet de moduler la forme de la divergence considérée. Lorsque $\alpha$ tend vers 1, la forme de la divergence revient à celle de Shannon (figure 4.1) qui est en fait celle utilisée dans TIM. Afin d'évaluer et de comparer les performances d'$\alpha$-TIM dans l'environnement de test artificiellement uniforme de manière juste, il est est tout fait acceptable

d'effectuer l'exercice de validation du paramètre $\alpha$ qui module la forme de la divergence. Une fois $\alpha$ optimisé dans l'environnement de test conventionnel, la fonction de coût d'$\alpha$-TIM devient identique à celle de TIM, les deux méthodes partagent donc les mêmes performances de classification.

Le tableau 5.3 présente une étude de l'impact qu'ont les termes des fonctions de coût de TIM et $\alpha$-TIM. Les tâches évaluées ont été échantillonnées en suivant la procédure décrite à la section 5.3. Dans un contexte très peu supervisé (1-*shot*), l'entropie marginale joue un rôle de régularisation important. L'omission du terme mène à des pertes en performance du régime de 17 à 26,2% pour TIM et de 12,5 à 19,2% pour $\alpha$-TIM. La flexibilité de la forme de la divergence d'$\alpha$-TIM fait ses preuves dans des contextes où la supervision est plus grande (5-*shot* et 10-*shot*). Dans ces scénarios de test, l'inclusion du terme de l'entropie marginale n'amène presqu'aucun changement en termes de performances. Par contre, le terme d'entropie marginale affecte négativement les performances de TIM. Dans les scénarios de test à 10-*shot*, TIM subit des pertes de performances entre 1,8 et 3,2% lorsque l'entropie marginale est considérée, et ce malgré la validation de son poids $\lambda$. Les points importants à tirer de cette étude sont que le terme d'entropie marginale amène une régularisation importante dans des contextes très peu supervisés et que la forme de la divergence-$\alpha$ est flexible par rapport à l'importance de la supervision.

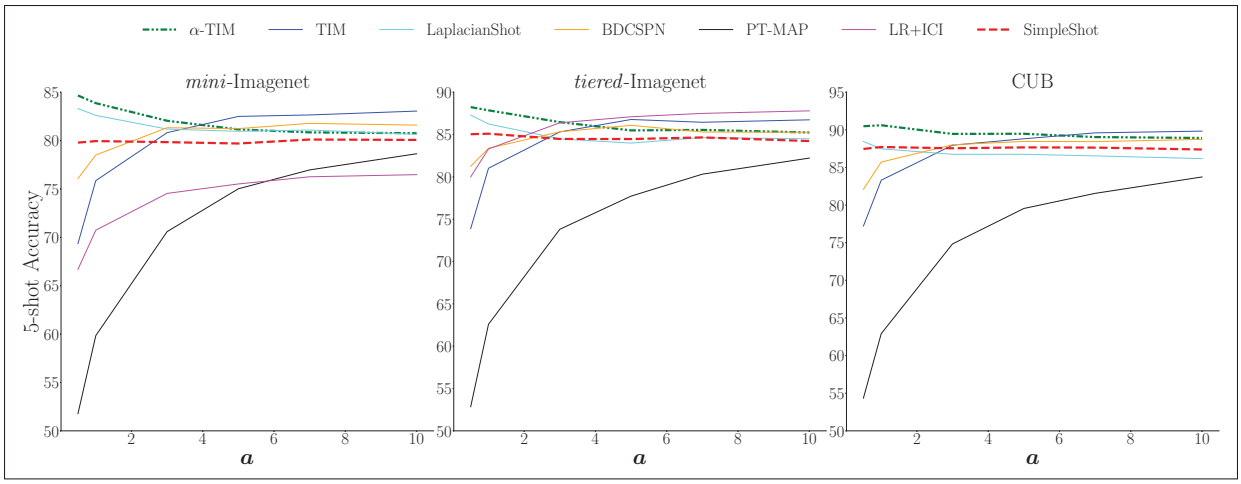


FIGURE 5.7 Comparaison du taux de précision de classification (*accuracy*) en fonction de différentes sévérités de déséquilibre des classes
Tirée de (Veilleux *et al.* (2021))

TABLEAU 5.1 Comparaison des méthodes de l'état de l'art dans l'environnement de test réaliste. Les expérimentations ont été conduites sur les bases *mini*-ImageNet et *tiered*-ImageNet

| | | | ***mini*-ImageNet** | | | |
|---|---|---|---|---|---|---|
| | **Méthode** | **Arch.** | **1-shot** | **5-shot** | **10-shot** | **20-shot** |
| Induct. | Protonet (NeurIPS'17 Snell *et al.* (2017)) | RN-18 | 53.4 | 74.2 | 79.2 | 82.4 |
| | Baseline (ICLR'19 Chen *et al.* (2019)) | | 56.0 | 78.9 | 83.2 | 85.9 |
| | Baseline++ (ICLR'19 Chen *et al.* (2019)) | | 60.4 | 79.7 | 83.8 | 86.3 |
| | Simpleshot (arXiv Wang *et al.* (2019)) | | 63.0 | 80.1 | 84.0 | 86.1 |
| Transduct. | MAML (ICML'17 Finn *et al.* (2017)) | RN-18 | 47.6 (↓ 3.8) | 64.5 (↓ 5.0) | 66.2 (↓ 5.7) | 67.2 (↓ 3.6) |
| | Versa (ICLR'19 Gordon *et al.* (2019)) | | 47.8 (↓ 2.2) | 61.9 (↓ 3.7) | 65.6 (↓ 3.6) | 67.3 (↓ 4.0) |
| | Entropy-min (ICLR'20 Dhillon *et al.* (2020)) | | 58.5 (↓ 5.1) | 74.8 (↓ 7.3) | 77.2 (↓ 8.0) | 79.3 (↓ 7.9) |
| | LR+ICI (CVPR'2020 Wang *et al.* (2020)) | | 58.7 (↓ 8.1) | 73.5 (↓ 5.7) | 78.4 (↓ 2.7) | 82.1 (↓ 1.7) |
| | PT-MAP (arXiv Hu *et al.* (2020b)) | | 60.1 (↓ 16.8) | 67.1 (↓ 18.2) | 68.8 (↓ 18.0) | 70.4 (↓ 17.4) |
| | LaplacianShot (ICML'20 Ziko *et al.* (2020)) | | 65.4 (↓ 4.7) | 81.6 (↓ 0.5) | 84.1 (↓ 0.2) | 86.0 (↑ 0.5) |
| | BD-CSPN (ECCV'20 Liu *et al.* (2020a)) | | 67.0 (↓ 2.4) | 80.2(↓ 1.8) | 82.9 (↓ 1.4) | 84.6 (↓ 1.1) |
| | TIM (NeurIPS'20 Boudiaf *et al.* (2020)) | | 67.3 (↓ 4.5) | 79.8 (↓ 4.1) | 82.3 (↓ 3.8) | 84.2 (↓ 3.7) |
| | $\alpha$-TIM (ours) | | **67.4** | **82.5** | **85.9** | **87.9** |
| Induct. | Baseline (ICLR'19 Chen *et al.* (2019)) | WRN | 62.2 | 81.9 | 85.5 | 87.9 |
| | Baseline++ (ICLR'19 Chen *et al.* (2019)) | | 64.5 | 82.1 | 85.7 | 87.9 |
| | Simpleshot (arXiv Wang *et al.* (2019)) | | 66.2 | 82.4 | 85.6 | 87.4 |
| Transduct. | Entropy-min (ICLR'20 Dhillon *et al.* (2020)) | WRN | 60.4 (↓ 5.7) | 76.2 (↓ 8.0) | – | – |
| | PT-MAP (arXiv Hu *et al.* (2020b)) | | 60.6 (↓ 18.3) | 66.8 (↓ 19.8) | 68.5 (↓ 19.3) | 69.9 (↓ 19.0) |
| | SIB (ICLR'20 Hu *et al.* (2020a)) | | 64.7 (↓ 5.3) | 72.5 (↓ 6.7) | 73.6 (↓ 8.4) | 74.2 (↓ 8.7) |
| | LaplacianShot (ICML'20 Ziko *et al.* (2020)) | | 68.1 (↓ 4.8) | 83.2 (↓ 0.6) | 85.9 (↑ 0.4) | 87.2 (↑ 0.6) |
| | TIM (NeurIPS'20 Boudiaf *et al.* (2020)) | | 69.8 (↓ 4.8) | 81.6 (↓ 4.3) | 84.2 (↓ 3.9) | 85.9 (↓ 3.7) |
| | BD-CSPN (ECCV'20 Liu *et al.* (2020a)) | | **70.4** (↓ 2.1) | 82.3(↓ 1.4) | 84.5 (↓ 1.4) | 85.7 (↓ 1.1) |
| | $\alpha$-TIM (ours) | | 69.8 | **84.8** | **87.9** | **89.7** |
| | | | ***tiered*-ImageNet** | | | |
| | | | **1-shot** | **5-shot** | **10-shot** | **20-shot** |
| Induct. | Baseline (ICLR'19 Chen *et al.* (2019)) | RN-18 | 63.5 | 83.8 | 87.3 | 89.0 |
| | Baseline++ (ICLR'19 Chen *et al.* (2019)) | | 68.0 | 84.2 | 87.4 | 89.2 |
| | Simpleshot (arXiv Wang *et al.* (2019)) | | 69.6 | 84.7 | 87.5 | 89.1 |
| Transduct. | Entropy-min (ICLR'20 Dhillon *et al.* (2020)) | RN-18 | 61.2 (↓ 5.8) | 75.5 (↓ 7.6) | 78.0 (↓ 7.9) | 79.8 (↓ 7.9) |
| | PT-MAP (arXiv Hu *et al.* (2020b)) | | 64.1 (↓ 18.8) | 70.0 (↓ 18.8) | 71.9 (↓ 17.8) | 73.4 (↓ 17.1) |
| | LaplacianShot (ICML'20 Ziko *et al.* (2020)) | | 72.3 (↓ 4.8) | 85.7 (↓ 0.5) | 87.9 (↓ 0.1) | 89.0 (↑ 0.3) |
| | BD-CSPN (ECCV'20 Liu *et al.* (2020a)) | | 74.1 (↓ 2.2) | 84.8 (↓ 1.4) | 86.7 (↓ 1.1) | 87.9 (↓ 0.8) |
| | TIM (NeurIPS'20 Boudiaf *et al.* (2020)) | | 74.1 (↓ 4.5) | 84.1 (↓ 3.6) | 86.0 (↓ 3.3) | 87.4 (↓ 3.1) |
| | LR+ICI (CVPR'20 Wang *et al.* (2020)) | | **74.6** (↓ 6.2) | 85.1 (↓ 2.8) | 88.0 (↓ 2.1) | 90.2 (↓ 1.2) |
| | $\alpha$-TIM (ours) | | 74.4 | **86.6** | **89.3** | **90.9** |
| Induct. | Baseline (ICLR'19 Chen *et al.* (2019)) | WRN | 64.6 | 84.9 | 88.2 | 89.9 |
| | Baseline++ (ICLR'19 Chen *et al.* (2019)) | | 68.7 | 85.4 | 88.4 | 90.1 |
| | Simpleshot (arXiv Wang *et al.* (2019)) | | 70.7 | 85.9 | 88.7 | 90.1 |
| Transduct. | Entropy-min (ICLR'20 Dhillon *et al.* (2020)) | WRN | 62.9 (↓ 6.0) | 77.3 (↓ 7.5) | – | – |
| | PT-MAP (arXiv Hu *et al.* (2020b)) | | 65.1 (↓ 19.5) | 71.0 (↓ 19.0) | 72.5 (↓ 18.3) | 74.0 (↓ 17.7) |
| | LaplacianShot (ICML'20 Ziko *et al.* (2020)) | | 73.5 (↓ 5.3) | 86.8 (↓ 0.5) | 88.6 (↓ 0.4) | 89.6 (↓ 0.2) |
| | BD-CSPN (ECCV'20 Liu *et al.* (2020a)) | | 75.4 (↓ 2.3) | 85.9 (↓ 1.5) | 87.8 (↓ 1.0) | 89.1 (↓ 0.6) |
| | TIM (NeurIPS'20 Boudiaf *et al.* (2020)) | | 75.8 (↓ 4.5) | 85.4 (↓ 3.5) | 87.3 (↓ 3.2) | 88.7 (↓ 2.9) |
| | $\alpha$-TIM (ours) | | **76.0** | **87.8** | **90.4** | **91.9** |

TABLEAU 5.2 Comparaison des méthodes de l'état de l'art dans l'environnement de test réaliste. Les expérimentations ont été conduites sur la base CUB

| | Méthode | Arch. | CUB 1-shot | CUB 5-shot | CUB 10-shot | CUB 20-shot |
|---|---|---|---|---|---|---|
| Induct. | Baseline (ICLR'19 Chen *et al.* (2019)) | RN-18 | 64.6 | 86.9 | 90.6 | 92.7 |
| | Baseline++ (ICLR'19 Chen *et al.* (2019)) | | 69.4 | 87.5 | 91.0 | 93.2 |
| | Simpleshot (arXiv Wang *et al.* (2019)) | | 70.6 | 87.5 | 90.6 | 92.2 |
| Transduct. | PT-MAP (arXiv Hu *et al.* (2020b)) | RN-18 | 65.1 (↓ 20.4) | 71.3 (↓ 20.0) | 73.0 (↓ 19.2) | 72.2 (↓ 18.9) |
| | Entropy-min (ICLR'20 Dhillon *et al.* (2020)) | | 67.5 (↓ 5.3) | 82.9 (↓ 6.0) | 85.5 (↓ 5.6) | 86.8 (↓ 5.7) |
| | LaplacianShot (ICML'20 Ziko *et al.* (2020)) | | 73.7 (↓ 5.2) | 87.7 (↓ 1.1) | 89.8 (↓ 0.7) | 90.6 (↓ 0.5) |
| | BD-CSPN (ECCV'20 Liu *et al.* (2020a)) | | 74.5 (↓ 3.4) | 87.1 (↓ 1.8) | 89.3 (↓ 1.3) | 90.3 (↓ 1.1) |
| | TIM (NeurIPS'20 Boudiaf *et al.* (2020)) | | 74.8 (↓ 5.5) | 86.9 (↓ 3.6) | 89.5 (↓ 2.9) | 91.7 (↓ 2.8) |
| | $\alpha$-TIM (ours) | | **75.7** | **89.8** | **92.3** | **94.6** |

TABLEAU 5.3 Étude de l'ablation du terme de l'entropie marginale des fonctions de coût de TIM (Boudiaf *et al.* (2020)) et d'$\alpha$-TIM (Veilleux *et al.* (2021)) et comparaison des effets selon la forme d'entropie considérée (Shannon ou entropie-$\alpha$)

| Fonction de coût | Base de données | Arch. | Méthode | 1-shot | 5-shot | 10-shot |
|---|---|---|---|---|---|---|
| $\text{CE} + \mathcal{H}(Y_Q \mid X_Q)$ | *mini*-Imagenet | RN-18 | TIM | 42.2 | 79.5 | 85.5 |
| | | | $\alpha$-TIM | **48.4** | **82.4** | **86.0** |
| | | WRN | TIM | 52.8 | 82.7 | 87.5 |
| | | | $\alpha$-TIM | **57.3** | **84.6** | **88.0** |
| | *tiered*-Imagenet | RN-18 | TIM | 52.4 | 83.7 | 88.4 |
| | | | $\alpha$-TIM | **59.0** | **86.3** | **89.2** |
| | | WRN | TIM | 49.6 | 84.1 | 89.1 |
| | | | $\alpha$-TIM | **56.8** | **87.6** | **90.4** |
| | CUB | RN-18 | TIM | 56.4 | 89.0 | 92.2 |
| | | | $\alpha$-TIM | **63.2** | **89.8** | **92.3** |
| $\text{CE} + \mathcal{H}(Y_Q \mid X_Q) - \mathcal{H}(Y_Q)$ | *mini*-Imagenet | RN-18 | TIM | 67.3 | 79.8 | 82.3 |
| | | | $\alpha$-TIM | **67.4** | **82.5** | **85.9** |
| | | WRN | TIM | 69.8 | 82.3 | 84.5 |
| | | | $\alpha$-TIM | 69.8 | **84.8** | **87.9** |
| | *tiered*-Imagenet | RN-18 | TIM | 74.1 | 84.1 | 86.0 |
| | | | $\alpha$-TIM | **74.4** | **86.6** | **89.3** |
| | | WRN | TIM | 75.8 | 85.4 | 87.3 |
| | | | $\alpha$-TIM | **76.0** | **87.8** | **90.4** |
| | CUB | RN-18 | TIM | 74.8 | 86.9 | 89.5 |
| | | | $\alpha$-TIM | **75.7** | **89.8** | **92.3** |

## CONCLUSION ET RECOMMANDATIONS

Dans ce projet de recherche, nous avons pointé le biais par rapport à l'uniformité de la distribution des classes des données à classifier dans l'environnement de test FSL. Afin d'évaluer les méthodes de manière plus réaliste, nous avons proposé une procédure d'échantillonnage des proportions de classes suivant la distribution de Dirichlet. Cette méthode permet de générer les proportions de classes de la base $\boldsymbol{Q}$, ce de manière aléatoire dans le simplexe. De plus, cette méthode permet de facilement moduler la concentration des distributions sur le simplexe en ajustant la valeur du paramètre $\boldsymbol{a}$.

Nous avons ensuite évalué les méthodes de l'état de l'art dans notre environnement de test réaliste en utilisant deux architectures de réseaux de neurones (RN-18 et WRN) et sur trois bases de données largement utilisées dans la littérature (*mini*-Imagenet, *tiered*-Imagenet et CUB). Suite à l'analyse des résultats expérimentaux, nous constatons que les performances des méthodes transductives souffrent lorsque les proportions de classes dans $\boldsymbol{Q}$ sont aléatoires. Les performances des méthodes inductives restent sans surprise stables lorsqu'évaluées de manière réaliste. Ces observations confirment notre hypothèse que les performances des méthodes transductives sont en partie dépendantes de la fausse assomption de l'uniformité des classes, tenue en compte et exploitée par ces dernières.

Pour attaquer ce nouveau problème de classification, nous avons proposé la méthode $\alpha$-TIM qui est une généralisation de la méthode TIM. La divergence-$\alpha$ s'avère à être efficace et offre une flexibilité pour l'obtention de solutions satisfaisantes pour des problèmes de classification dont la distribution des classes est aléatoire. De plus, notre méthode se trouve à être plus performante que toutes les méthodes de l'état de l'art évaluées. Elle permet d'obtenir les meilleurs résultats de classification, et ce dans tous les scénarios de test.

Pour faire suite à ce projet de recherche, il serait intéressant d'évaluer les méthodes FSL dans un scénario de test où la distribution de classes est aléatoire à la fois dans la base de support $\mathcal{S}$

et de *query* $Q$. De plus, la divergence-$\alpha$ semble être efficace dans les contextes présentant un grand déséquilibre de classes. Ce principe pourrait être appliqué pour résoudre des problèmes présentant cet obstacle, par exemple pour la segmentation sémantique.

## BIBLIOGRAPHIE


Amari, S.-I. (2000). $\alpha$-Divergence Is Unique, Belonging to Both $f$-Divergence and Bregman Divergence Classes. *IEEE Transactions on Information Theory*, 55(11), 4925–4931.

Amari, S. & Nagaoka, H. (Éds.). (2000). *Methods of information geometry*. Oxford University Press : New York, USA : American Mathematical Society.

Antoniou, A., Edwards, H. & Storkey, A. (2019). How to train your MAML. *International Conference on Learning Representations*. Repéré à https://openreview.net/forum?id=HJGven05Y7.

Arimoto, S. (1977). Information measures and capacity of order $\alpha$ for discrete memoryless channels. *Topics in information theory*.

Ba, J., Kiros, J. R. & Hinton, G. E. (2016). Layer Normalization. *ArXiv*, abs/1607.06450.

Ben Ayed, I. (2019). SYS843 : Réseaux de neurones et systèmes flous. École de technologie supérieure.

Boudiaf, M., Ziko, I. M., Rony, J., Dolz, J., Piantanida, P. & Ben Ayed, I. (2020). Transductive Information Maximization For Few-Shot Learning. *Neural Information Processing Systems (NeurIPS)*.

Bronskill, J., Gordon, J., Requeima, J., Nowozin, S. & Turner, R. E. (2020). TASKNORM : Rethinking Batch Normalization for Meta-Learning. *International Conference on Machine Learning (ICML)*.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F. & Huang, J.-B. (2019). A Closer Look at Few-shot Classification. *International Conference on Learning Representations (ICLR)*.

Chen, X., Dai, H., Li, Y., Gao, X. & Song, L. (2020). Learning to stop while learning to predict. *International Conference on Machine Learning (ICML)*.

Chernoff, H. et al. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4), 493–507.

Cichocki, A. & Amari, S.-I. (2010). Families of alpha-beta-and gamma-divergences : Flexible and robust measures of similarities. *Entropy*, 12(6), 1532–1568.

Cui, W. & Guo, Y. (2021, 18–24 Jul). Parameterless Transductive Feature Re-representation for Few-Shot Learning. *Proceedings of the 38th International Conference on Machine Learning*, 139(Proceedings of Machine Learning Research), 2212–2221. Repéré à https://proceedings.mlr.press/v139/cui21a.html.

Cuturi, M. (2013). Sinkhorn distances : Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2292–2300.

Dengyong, Z., Bousquet, O., Lal, T. N., Weston, J. & Schölkopf, B. (2004). Learning with local and global consistency. *Neural Information Processing Systems (NeurIPS)*.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer.

Dhillon, G. S., Chaudhari, P., Ravichandran, A. & Soatto, S. (2020). A Baseline for Few-Shot Image Classification. *International Conference on Learning Representations (ICLR)*.

Du, Y., Zhen, X., Shao, L. & Snoek, C. G. M. (2021). MetaNorm : Learning to Normalize Few-Shot Batches Across Domains. *International Conference on Learning Representations*. Repéré à https://openreview.net/forum?id=9z_dNsC4B5t.

Fei-Fei, L., Fergus, R. & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 594–611.

Finn, C., Abbeel, P. & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*.

Finn, C., Xu, K. & Levine, S. (2018). Probabilistic model-agnostic meta-learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Gidaris, S. & Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P. & Cord, M. (2019). Boosting Few-Shot Visual Learning with Self-Supervision. *Proceedings of the IEEE International Conference on Computer Vision*.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S. & Turner, R. (2019). Meta-Learning Probabilistic Inference for Prediction. *International Conference on Learning Representations (ICLR)*.

Grandvalet, Y. & Bengio, Y. (2005). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems (NeurIPS)*.

Grant, E., Finn, C., Levine, S., Darrell, T. & Griffiths, T. (2018). Recasting Gradient-Based Meta-Learning as Hierarchical Bayes. *International Conference on Learning Representations*.

Guo, Y. & Cheung, N.-M. (2020). Attentive Weights Generation for Few Shot Learning via Information Maximization. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T. & Feris, R. (2020). A broader study of cross-domain few-shot learning.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. doi : 10.1038/s41586-020-2649-2.

Havrda, J. & Charvát, F. (1967). Quantification method of classification processes – Concept of structural $a$-entropy. *Kybernetika*, 3(1), 30–35.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Trischler, A. & Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *ICLR*.

Hou, R., Chang, H., Bingpeng, M., Shan, S. & Chen, X. (2019). Cross Attention Network for Few-shot Classification. *Neural Information Processing Systems (NeurIPS)*.

Hu, S. X., Moreno, P. G., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N. D. & Damianou, A. (2020a). Empirical Bayes Transductive Meta-Learning with Synthetic Gradients. *International Conference on Learning Representations (ICLR)*.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E. & Sugiyama, M. (2017, 06–11 Aug). Learning Discrete Representations via Information Maximizing Self-Augmented Training. *Proceedings of the 34th International Conference on Machine Learning*, 70(Proceedings of Machine Learning Research), 1558–1567. Repéré à http://proceedings.mlr.press/v70/hu17b.html.

Hu, Y., Gripon, V. & Pateux, S. (2020b). Leveraging the feature distribution in transfer-based few-shot learning. *arXiv preprint :2006.03806*.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Huang, H., Zhang, J., Zhang, J., Wu, Q. & Xu, C. (2021). PTN : A Poisson Transfer Network for Semi-supervised Few-shot Learning. *AAAI*.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Joachims, T. (1999). Transductive Inference for Text Classification Using Support Vector Machines. *International Conference on Machine Learning (ICML)*.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, pp. 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. doi : 10.1109/5.726791.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

Lee, D. H. & Chung, S.-Y. (2021, 18–24 Jul). Unsupervised Embedding Adaptation via Early-Stage Feature Reconstruction for Few-Shot Classification. *Proceedings of the 38th International Conference on Machine Learning*, 139(Proceedings of Machine Learning Research), 6098–6108. Repéré à https://proceedings.mlr.press/v139/lee21d.html.

Lee, H. B., Lee, H., Na, D., Kim, S., Park, M., Yang, E. & Hwang, S. J. (2019). Learning to balance : Bayesian meta-learning for imbalanced and out-of-distribution tasks. *arXiv preprint :1905.12917*.

Lee, Y. & Choi, S. (2018). Gradient-based meta-learning with learned layerwise metric and subspace. *International Conference on Machine Learning*, pp. 2933–2942.

Li, Z., Zhou, F., Chen, F. & Li, H. (2017). Meta-SGD : Learning to Learn Quickly for Few Shot Learning. *CoRR*, abs/1707.09835.

Lichtenstein, M., Sattigeri, P., Feris, R., Giryes, R. & Karlinsky, L. (2020). TAFSSL : Task-Adaptive Feature Sub-Space Learning for Few-Shot Classification. *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII*, 12352, 522-539. doi : 10.1007/978-3-030-58571-6_31.

Liu, J., Song, L. & Qin, Y. (2020a). Prototype Rectification for Few-Shot Learning. *European Conference on Computer Vision (ECCV)*.

Liu, Y., Schiele, B. & Sun, Q. (2020b). An ensemble of epoch-wise empirical bayes for few-shot learning. *European Conference on Computer Vision (ECCV)*.

Mallawaarachchi, V. (2020). Inductive vs. Transductive Learning [misc]. Repéré à https://towardsdatascience.com/inductive-vs-transductive-learning-e608e786f7d.

Miller, E., Matsakis, N. & Viola, P. (2000). Learning from One Example through Shared Densities on Transforms. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mishra, N., Rohaninejad, M., Chen, X. & Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. *International Conference on Learning Representations (ICLR)*.

Nichol, A., Achiam, J. & Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *ArXiv*, abs/1803.02999.

Ochal, M., Patacchiola, M., Storkey, A., Vazquez, J. & Wang, S. (2021). Few-Shot Learning with Class Imbalance. *arXiv preprint :2101.02523*.

Oreshkin, B., López, P. R. & Lacoste, A. (2018). Tadam : Task dependent adaptive metric for improved few-shot learning. *Neural Information Processing Systems (NeurIPS)*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019). PyTorch : An Imperative Style, High-Performance Deep Learning Library. Dans *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. Repéré à http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302), 157–175. doi : 10.1080/14786440009463897.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Qiao, L., Shi, Y., Li, J., Wang, Y., Huang, T. & Tian, Y. (2019). Transductive Episodic-Wise Adaptive Metric for Few-Shot Learning. *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Qiao, S., Liu, C., Shen, W. & Yuille, A. L. (2018). Few-shot image recognition by predicting parameters from activations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7229–7238.

Ravi, S. & Larochelle, H. (2017). Optimization as a Model for Few-Shot Learning. *International Conference on Learning Representations (ICLR)*.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H. & Zemel, R. S. (2018). Meta-Learning for Semi-Supervised Few-Shot Classification. *International Conference on Learning Representations (ICLR)*.

Rodriguez, P. (2020). Understanding Transductive Few-shot Learning [misc]. Repéré à https://opencv.org/understanding-transductive-few-shot-learning/.

Rodriguez, P., Laradji, I. H., Drouin, A. & Lacoste, A. (2020). Embedding Propagation : Smoother Manifold for Few-Shot Classification. *ECCV*.

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. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211-252. doi : 10.1007/s11263-015-0816-y.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S. & Hadsell, R. (2019). Meta-Learning with Latent Embedding Optimization. *International Conference on Learning Representations (ICLR)*.

Satorras, V. G. & Estrach, J. B. (2018). Few-Shot Learning with Graph Neural Networks. *International Conference on Learning Representations*.

Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

Snell, J., Swersky, K. & Zemel, R. (2017). Prototypical Networks for Few-shot Learning. *Neural Information Processing Systems (NeurIPS)*.

Sun, Q., Liu, Y., Chua, T. & Schiele, B. (2019, June). Meta-Transfer Learning for Few-Shot Learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. & Hospedales, T. M. (2018). Learning to compare : Relation network for few-shot learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B. & Isola, P. (2020). Rethinking Few-Shot Image Classification : a Good Embedding Is All You Need ? *European Conference on Computer Vision (ECCV)*.

Triantafillou, E., Zemel, R. & Urtasun, R. (2017). Few-shot learning through an information retrieval lens. *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2255–2265.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A. et al. (2019). Meta-dataset : A dataset of datasets for learning to learn from few examples. *arXiv preprint :1903.03096*.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1), 479–487.

Tseng, H.-Y., Lee, H.-Y., Huang, J.-B. & Yang, M.-H. (2020). Cross-domain few-shot classification via learned feature-wise transformation. *International Conference on Learning Representations (ICLR)*.

Ulyanov, D., Vedaldi, A. & Lempitsky, V. S. (2016). Instance Normalization : The Missing Ingredient for Fast Stylization. *ArXiv*, abs/1607.08022.

van Erven, T. & Harremos, P. (2014). Rényi Divergence and Kullback-Leibler Divergence. *IEEE Transactions on Information Theory*, 60(7), 3797-3820. doi : 10.1109/TIT.2014.2320500.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks (TNN)*, 10(5), 988-999.

Veilleux, O., Boudiaf, M., Piantanida, P. & Ben Ayed, I. (2021). Realistic Evaluation of Transductive Few-Shot Learning. *Neural Information Processing Systems (NeurIPS)*.

Vinyals, O., Blundell, C., Lillicrap, T. P., Kavukcuoglu, K. & Wierstra, D. (2016). Matching Networks for One Shot Learning. *Neural Information Processing Systems (NeurIPS)*.

Wah, C., Branson, S., Welinder, P., Perona, P. & Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset.

Wang, Y., Chao, W.-L., Weinberger, K. Q. & van der Maaten, L. (2019). SimpleShot : Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv preprint :1911.04623*.

Wang, Y., Xu, C., Liu, C., Zhang, L. & Fu, Y. (2020). Instance Credibility Inference for Few-Shot Learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wertheimer, D. & Hariharan, B. (2019). Few-shot learning with localization in realistic settings. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567.

Beta distribution. (2022a). Dans *Wikipedia*. Repéré le 2022-03-09 à https://en.wikipedia.org/wiki/Beta_distribution.

Dirichlet distribution. (2022b). Dans *Wikipedia*. Repéré le 2022-02-07 à https://en.wikipedia.org/wiki/Dirichlet_distribution.

Gamma function. (2022c). Dans *Wikipedia*. Repéré le 2022-03-11 à https://en.wikipedia.org/wiki/Gamma_function.

Kullback–Leibler divergence. (2022d). Dans *Wikipedia*. Repéré le 2022-02-24 à https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence.

Rényi entropy. (2022e). Dans *Wikipedia*. Repéré le 2022-02-02 à https://en.wikipedia.org/wiki/R%C3%A9nyi_entropy.

Wu, Y. & He, K. (2018, September). Group Normalization. *Proceedings of the European Conference on Computer Vision (ECCV)*.

Yanbin, L., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S. & Yang, Y. (2019). Learning to Propagate Labels : Transductive Propagation Network for Few-shot Learning. *International Conference on Learning Representations (ICLR)*.

Yang, L., Li, L., Zhang, Z., Zhou, X., Zhou, E. & Liu, Y. (2020). DPGN : Distribution Propagation Graph Network for Few-shot Learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ye, H.-J., Hu, H., Zhan, D.-C. & Sha, F. (2020). Few-Shot Learning via Embedding Adaptation with Set-to-Set Functions. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yue, Z., Zhang, H., Sun, Q. & Hua, X.-S. (2020). Interventional few-shot learning. *Neural Information Processing System*.

Zagoruyko, S. & Komodakis, N. (2016, September). Wide Residual Networks. *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1-87.12. doi : 10.5244/C.30.87.

Zhang, C., Cai, Y., Lin, G. & Shen, C. (2020). DeepEMD : Few-Shot Image Classification With Differentiable Earth Mover's Distance and Structured Classifiers. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.

Zhen, X., Sun, H., Du, Y., Xu, J., Yin, Y., Shao, L. & Snoek, C. (2020). Learning to learn kernels with variational random features. *International Conference on Machine Learning (ICML)*.

Ziko, I., Granger, E. & Ayed, I. B. (2018). Scalable Laplacian K-modes. *Advances in Neural Information processing Systems*, pp. 10041–10051.

Ziko, I. M. (2020). *Flexible and scalable models for clustering and few-shot learning*. (Thèse de doctorat, École de technologie supérieure, 1100 Rue Notre Dame O, Montréal, QC H3C 1K3). Repéré à https://espace.etsmtl.ca/id/eprint/2613.

Ziko, I. M., Dolz, J., Granger, E. & Ben Ayed, I. (2020). Laplacian Regularized Few-Shot Learning. *International Conference on Machine Learning (ICML)*.