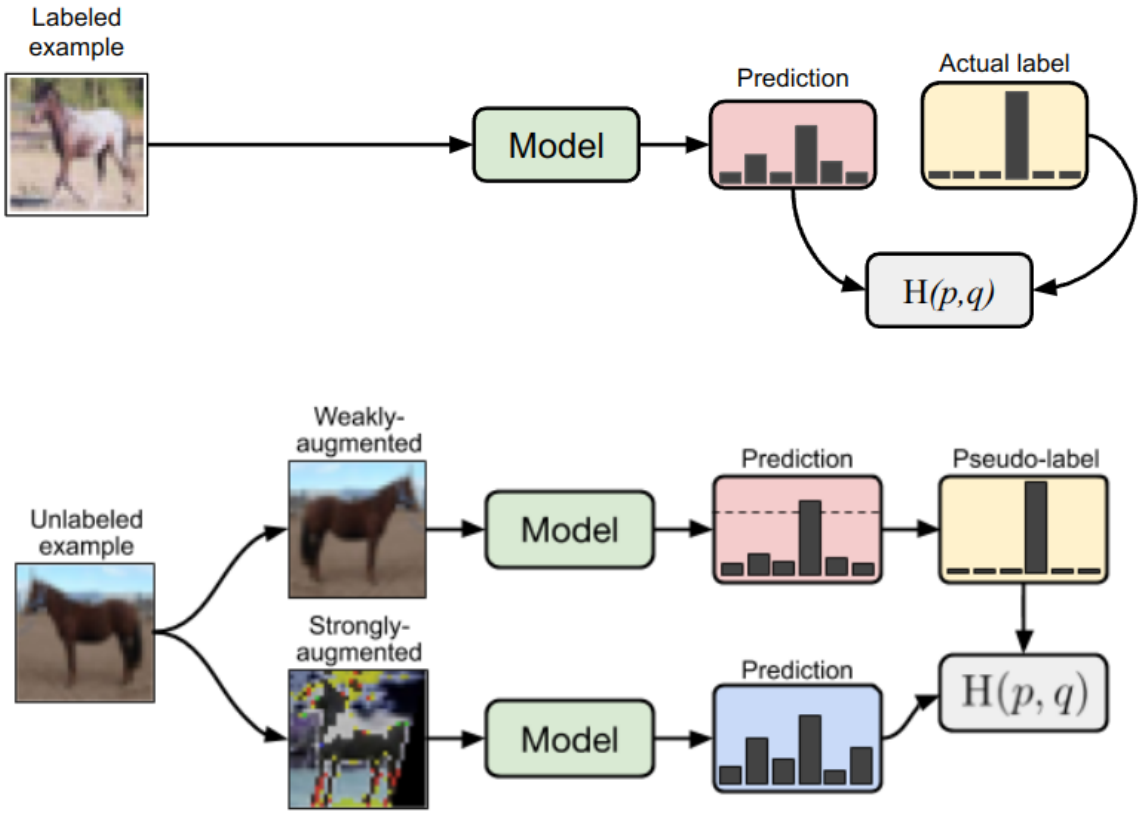
Projet d'apprentissage faiblement supervisé

July 5, 2023

Auteurs :
Alexandre Personnic
Thi Minh Ha Ho

Encadrant :
Axel Carlier



Contents

1 Introduction

Les réseaux de neurones profonds ont obtenu des performances remarquables sur un large spectre de tâches d'apprentissage supervisé, où les données sont labellisées. Cependant, c'est un défi lorsque la quantité de données annotées est limitée ou coûteuse à obtenir. De nombreuses recherches ont été menées pour trouver des méthodes permettant d'optimiser l'utilisation des données et de moins dépendre de la taille de l'ensemble des données labellisées. L'apprentissage semi-supervisé (SSL) est une approche puissante pour la conception de modèles sur une grande quantité de données dont la partie annotée est relativement faible (de 1 à 10%).

Dans ce projet, nous implémentons et étudions le modèle FixMatch introduit par [4] en 2020. FixMatch est une méthode d'apprentissage semi-supervisé non seulement précise mais aussi simple selon [2]. Notre objectif est d'observer la performance de FixMatch avec un petit pourcentage de données labellisées, respectivement 1%, 5% et 10%. Ce problème est souvent rencontré dans la vie réelle où le budget pour l'annotation des données est limité. Nous souhaitons donc évaluer l'impact de la quantité de données étiquetées à la précision et expérimentons de différents jeux de données.

Premièrement, nous rappelons les mécanismes du FixMatch dans 2. Ensuite, nous expliquons notre implémentation, notamment le choix de données labellisées dans 3. Enfin, la section 4 est consacrée aux commentaires et à l'analyse des résultats et la section 5 résume notre conclusion.

Notre code et nos modèles entraînés peuvent être trouvés dans ce répertoire de Github:

`https://github.com/tmhho/FixMatch`

2 Algorithme de FixMatch

Tout d'abord, nous expliquons les idées principales du FixMatch dans la sous-section 2.1. Nous dédions quelques sous-sections pour les composants importants qui contribuent à la bonne performance de l'algorithme FixMatch : la pseudo-labellisation (2.2), la régularisation de consistance (2.3) et l'augmentation des données (2.4).

2.1 Description de l'algorithme

Notre entrée contient deux ensembles $\mathcal{X}$ des images labellisées et $\mathcal{U}$ des images non-labellisées

$$\mathcal{X} = \{(x_i, \ell_i) \mid 1 \leq i \leq B\}, \qquad \mathcal{U} = \{u_i \mid 1 \leq i \leq \mu B\},$$

dont μ est l'hyper-paramètre qui détermine les tailles relatives entre $\mathcal{X}$ et $\mathcal{U}$.

L'objectif de l'algorithme de FixMatch est de bénificier des données non-annotées $\mathcal{U}$ pour la classification, un problème traditionnellement considéré comme l'apprentissage supervisé. Il se déroule en 3 étapes ci-dessous.

Étape 1: Apprentissage supervisé

Choisissons une augmentation faible $\boldsymbol{\alpha}$, c'est-à-dire une transformation qui conserve la plupart des caractéristiques des images. L'algorithme commence par concevoir un modèle initial en utilisant l'apprentissage supervisé sur $\mathcal{X}$ avec la fonction de perte d'entropie croisée :

$$L_s = \frac{1}{B} \sum_{i=1}^{B} H\left(\ell_i, \mathbb{P}_m\left(\ell \mid \boldsymbol{\alpha}\left(x_i\right)\right)\right). \tag{1}$$

Ce modèle nous donne des prédictions associées à une distribution $\mathbb{P}_m(\ell \mid x)$ pour une image x et une classe ℓ. Notons que l'opérande gauche de l'entropie constitue des classes dures ℓ_i.

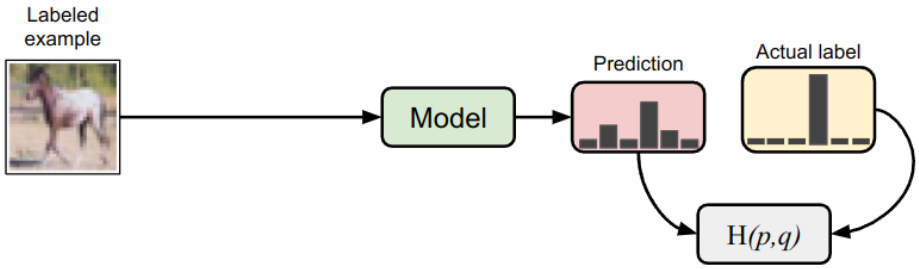


Figure 1: Partie supervisée de FixMatch

L'intuition de la partie non-supervisée se base sur cette remarque suivante : si un modèle est sûr de donner l'étiquette à une image, alors il devrait donner la même étiquette pour différentes augmentations de cette image. Alors, pour les deux prochaines étapes, nous générons les classes prédites des deux augmentations (faible et forte) de chaque $u_i \in \mathcal{U}$ et les «comparons» pour améliorer le modèle initial.

Étape 2 : Pseudo-labellisation

Fixons un seuil de confidence $\tau \in [0, 1]$.

Pour chaque image non annotée u_i, en appliquant le modèle initial à l'augmentation faible $\boldsymbol{\alpha}(u_i)$, nous obtenons la distribution de sa classe prédite :

$$q_i = \mathbb{P}_m(\ell \mid \boldsymbol{\alpha}(u_i)).$$

Si le modèle est suffisamment confiant pour prédire la classe de u_i, i.e., la plus grande probabilité de q_i dépass le seuil τ, nous acceptons pour l'étiquette dure de u_i le maximisateur

$$\hat{q}_i = \text{argmax}_\ell(q_i).$$

Étape 3 : Régularisation de consistance

L'augementation forte $\mathcal{A}(u_i)$ de u_i est utilisée pour calculer

$$Q_i = \mathbb{P}_m(\ell \mid \mathcal{A}(u_i)).$$

Cette distribution Q_i est comparée avec le pseudo-label $\hat{q}_i$ (s'il est retenu dans l'étape 2) par la fonction de perte non-supervisée

$$L_u = \frac{1}{\mu B} \sum_{i=1}^{\mu B} \mathbb{1} \left(\max \left(q_i \right) \geq \tau \right) H \left(\hat{q}_i, Q_i \right). \tag{2}$$

Alors, la perte totale minimisée par FixMatch est la somme pondérée de L_s et L_u

$$L_{\text{total}} = L_s + \lambda_u L_u \tag{3}$$

où λ_u est un hyper-paramètre scalaire qui représente le poids relatif de la perte non-supervisée.

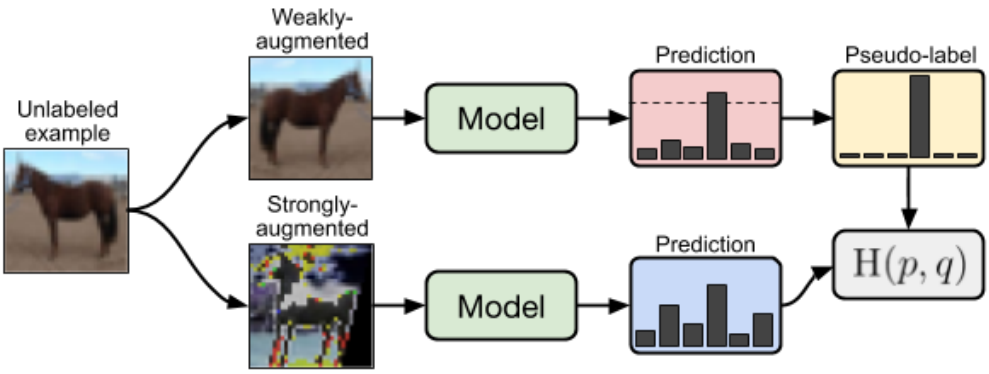


Figure 2: La pseudo-labellisation et la régularisation de consistance

2.2 Pseudo-labellisation

L'objectif de la pseudo-labellisation est d'utiliser le modèle lui-même pour obtenir des étiquettes artificielles pour des données non annotées. Seules les labels artificiels dont la plus grande probabilité de classe est supérieure à un seuil prédéfini τ sont retenues. Notons $q_i = \mathbb{P}_m(y \mid u_i)$, $\hat{q}_i = \text{argmax}_\ell(q_i)$, la pseudo-labellisation utilise la fonction de perte ci-dessous :

$$\frac{1}{\mu B} \sum_{i=1}^{\mu B} \mathbb{1} \left(\max \left(q_i \right) \geq \tau \right) H \left(\hat{q}_i, q_i \right)$$

L'argmax de la sortie du modèle retourne des *hard labels*, ce qui rend la pseudo-labellisation liée à la minimisation de l'entropie, où les prédictions du modèle sont encouragées à être à faible entropie ou à haute confiance sur des données non annotées.

2.3 Régularisation de consistance

L'hypothèse de base de la régularisation de consistance est que si une perturbation est appliquée aux images non labellisées u_b, la prédiction ne devrait pas changer de manière significative. Le modèle peut alors être entraîné pour obtenir une prédiction cohérente sur un exemple non labellisées u_i et sur sa version perturbée $\boldsymbol{\alpha}(u_i)$. Ainsi, le modèle est entraîné à la fois via une perte de classification supervisée standard et sur des données non étiquetées via la fonction de perte :

$$\sum_{i=1}^{\mu B} \| \mathbb{P}_m \left(\ell \mid \boldsymbol{\alpha} \left(u_i \right) \right) - \mathbb{P}_m \left(\ell \mid u_i \right) \|_2^2$$

2.4 Augmentation d'images

L'utilisation d'augmentations variées a été identifiée comme un élément clé de l'apprentissage semi-supervisé [2]. Comme elle permet d'accroître la diversité des données, d'améliorer la robustesse, de permettre une régularisation de la cohérence et d'éviter le overfitting. Ces avantages contribuent à la performance globale et aux capacités de généralisation de l'algorithme d'apprentissage semi-supervisé.

Pour **l'augmentation faible $\boldsymbol{\alpha}$**, on utilise une stratégie standard d'augmentation par retournement et décalage. Plus précisément, les images sont retournées horizontalement et les images sont translatées verticalement et horizontalement jusqu'à 12,5%.

Pour **l'augmentation forte** $\mathcal{A}$ (voir fig. 3), nous choisissons **RandAugment** [1] qui est une méthode d'augmentation automatisée. Tout d'abord, il génère une liste de transformations d'images possibles. Ensuite, il choisit aléatoirement N transformations de cette liste. Enfin, nous appliquons cette transformations avec une magnitude de M à l'image d'entrée. La valeur M contrôle l'intensité ou la sévérité de chaque transformation. Une valeur M plus élevée entraîne des modifications plus importantes de l'image.

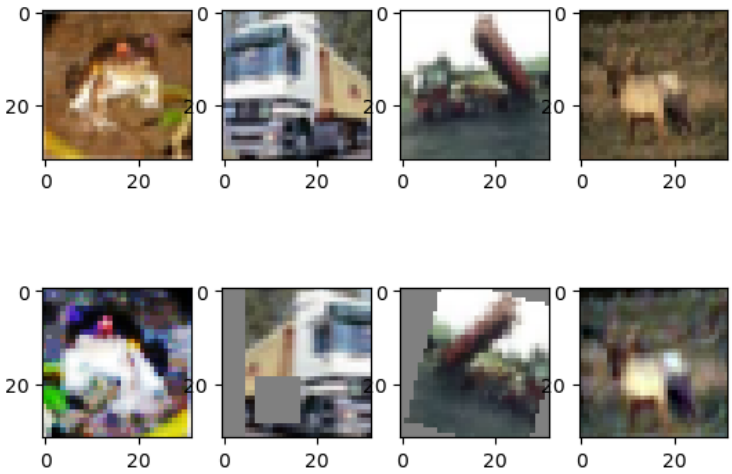


Figure 3: Exemples de l'augmentation forte des images

3 Implémentation

3.1 Modèle et hyperparamètres

Nous implémentons l'algorithme FixMatch en suivant la déscription de [4, Algo. 1]. Le code et les modèles réalisés dans ce projet se trouvent sur le répertoire GitHub :

`https://github.com/tmhho/FixMatch`

En outre, nous déployons le réseau de neurones (Fig. 3.1) composé de 4 paires de couches de convolution et de maxpooling, suivies de couches d'aplatissement et de densité. Le tableau 1 résume les valeurs des hyper-paramètres pour l'augmentation (N, M) (voir la sous-section 2.4) et celles du modèle FixMatch qui sont les mêmes dans [4]. Nous ferons varier le pourcentage de données étiquetées dans l'ensemble de données (1%, 5%, 10%).

Paramètres	Valeurs
B	64
μ	7
τ	0.95
λ_u	1
taux d'apprentissage	10^{-4}
N	2
M	15

Table 1: Hyper-paramètres choisis

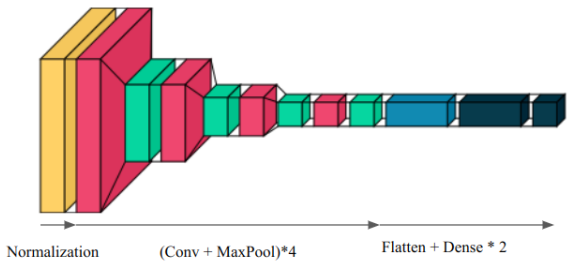


Figure 4: Réseau de neurones

3.2 Choix de l'ensemble de données

Dans ce projet, nous visons à classifier des images du jeu de données CIFAR-10. Ce jeu de données est composé de 60000 images couleur 32x32 réparties en 10 classes, dont 6000 images par classe. Il y a 50000 images d'entraînement et 10000 images de test.

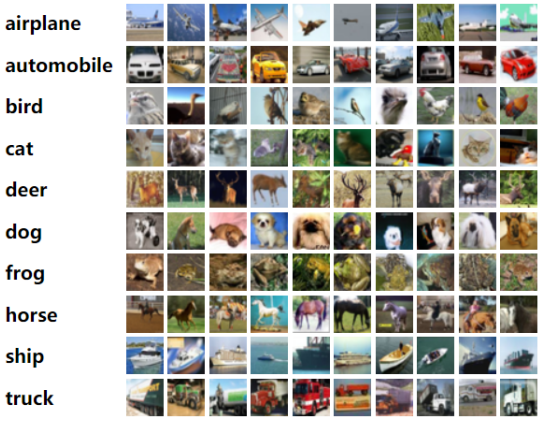


Figure 5: Exemples de chaque classe

Figure 6: Similarité cosinus de la première voiture par rapport aux autres

Parmi les 50000 images d'entraînement, nous choisissons respectivement 10%, 5%, 1% de données comme données labellisées et le reste est non labellisées. Dans un premier temps, nous nous concentrons sur l'étude des performances du modèle en choisissant aléatoirement les données labellisées. Après avoir obtenu des performances relativement précises, nous choisissons intentionnellement des images labellisées pour voir si cela peut améliorer la performance.

Notre idée est de sélectionner les images les plus uniques pour chaque classe afin d'assurer la diversité des données labellisées en trois étapes :

1. Extrayons un vecteur de caractéristiques pour chaque image d'entraînement à l'aide du pipeline **img2vec** [3] qui vectorise les images à l'aide d'un modèle pré-entraîné (**resnet-18**).

2. Calculons la similarité entre les images (Fig. 6) à l'aide de la similarité cosinus définie par :

$$\mathbf{sim}(X, Y) = \frac{\langle X, Y \rangle}{\|X\| \cdot \|Y\|}.$$

   Par exemple, Fig. 6 implique une forte similarité en cosinus entre les voitures de même couleur.

3. À l'aide d'une matrice de similarités entre les images, nous définissons l'unicité comme suit

$$\mathbf{uniqueness}(x_i) = 1 - \sum_{j=1, j\neq i}^{50000} \mathbf{sim}(x_i, x_j).$$

Enfin, nous choisissons respectivement les 10%, 5% et 1% d'images les plus uniques. L'avantage de cette approche est que nous n'avons besoin de calculer l'unicité qu'une seule fois. Cependant, cela signifie que l'ensemble de données étiquetées à 1% est inclus dans les ensembles de données à 5% et 10%.

4 Analyse des résultats

Comparaison aux modèles supervisés

Notre premier réflexe est de comparer le modèle implémenté sur des données étiquetées choisies de manière aléatoire et de manière intentionnelle. On regroupe les précisions atteintes sur le jeu de test des chaque modèles dans le tableau 2.

	100%	Données aléatoires			Données choisies		
		10%	5%	1%	10%	5%	1%
Supervisé	0.76	0.55	0.42	0.28	**0.61**	**0.48**	**0.44**
FixMatch	-	**0.75**	**0.61**	0.45	0.67	0.58	**0.47**

Table 2: Synthèse des précisions de test du modèle entraînné avec 500 epochs

Dans le cas de **l'apprentissage supervisé**, plus l'ensemble de données annotées est petit, plus la classification est mauvaise, ce qui est logique. En outre, nous pouvons remarquer qu'un bon choix de données étiquetées peut améliorer la précision de la classification supervisée de manière significative. Par exemple, selon Tab. 2, si nous choisissons les 5000 images les plus uniques pour entraîner le modèle, il peut prédire avec une précision de 0,61 alors qu'il a une précision de 0,55 avec des images étiquetées au hasard.

C'est l'inverse dans le cas de **l'apprentissage semi-supervisé** (FixMatch). Nous observons que FixMatch obtient de meilleurs résultats avec des images étiquetées au hasard qu'avec les images que nous avons choisies. Nous avons obtenu une précision de 0,75 pour 10% de données étiquetées choisies au hasard et de 0,61 pour des données choisies. Cela montre que FixMatch peut réaliser de bonnes performances même avec des images étiquetées choisies de manière totalement aléatoire.

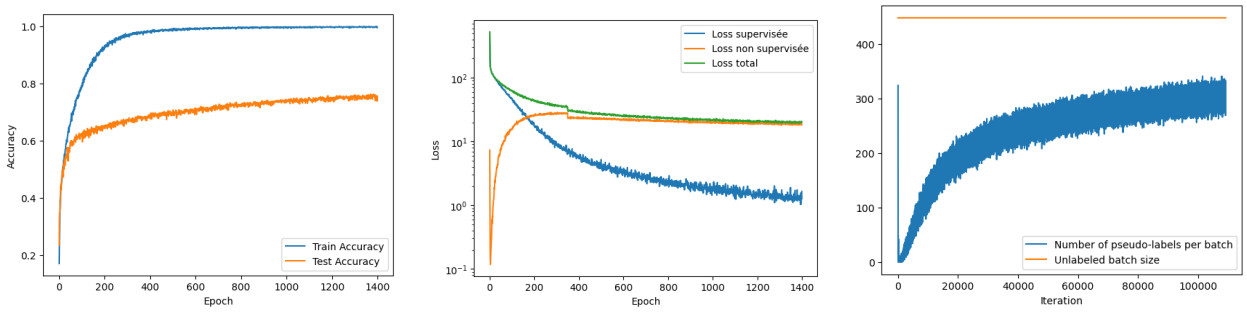
Il faut noter qu'un modèle de FixMatch entraîné avec seulement 10% des images annotées atteint la même précision que le modèle supervisé avec 100% d'étiquetage. Ceci est un résultat impressionnant pour un modèle assez simple comme FixMatch parmi les autres algorithmes de l'apprentissage semi-supervisé.

Comportement de la précision et de la fonction de perte

Ensuite, nous comparons la précision et la fonction de perte de Fixmatch entre différents pourcentages d'étiquetage : 10% (Fig. 7), 5% (Fig. 8) et 1% (Fig. 9). Pour chaque l'ensemble de données, nous rapportons le comportement des expériences séparément sur la partie supervisée, la partie non-supervisée et l'ensemble de l'algorithme.

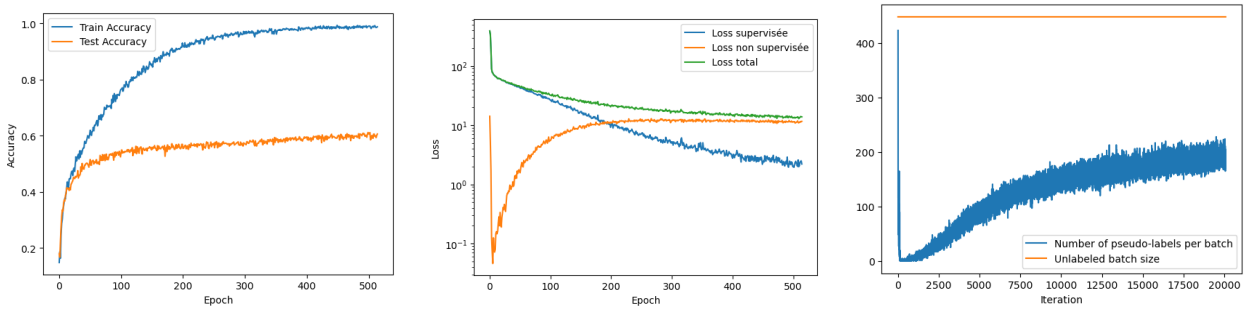
En général, les courbes de perte et de précision présentent les mêmes caractéristiques dans tous les cas de pourcentage d'étiquetage. La précision de l'entraînement atteint rapidement un score élevé d'environ 99% après 500 epochs. Les tests augmentent progressivement et restent relativement stables après 1000 epochs.

D'après les courbes de perte , la perte totale est la somme des pertes de la partie supervisée et de la partie non-supervisée, ce qui correspond au fait que λ_u est égal à 1. Au début, la diminution de la perte supervisée entraîne une diminution de la perte totale. La perte non supervisée augmente considérablement au début car le nombre d'images non étiquetées augmente alors que le modèle



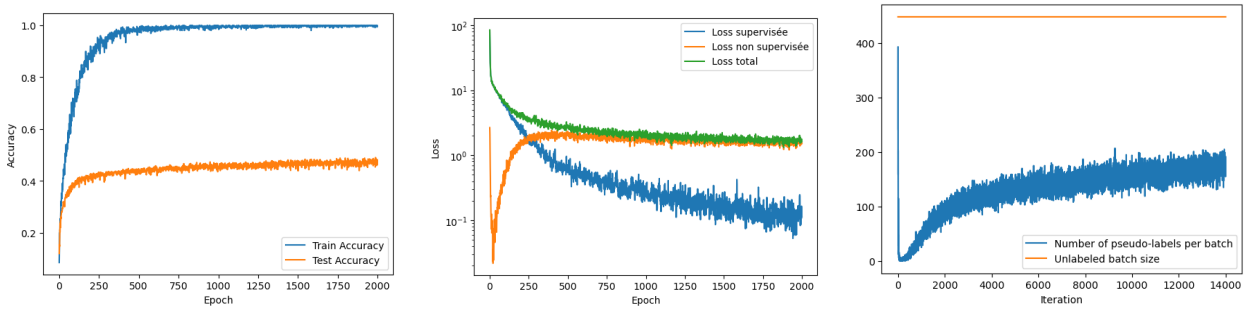
(a) Précision (b) Perte (c) Nombre de pseudo-labels

Figure 7: FixMatch avec 10% des données annotées



(a) Précision (b) Perte (c) Nombre de pseudo-labels

Figure 8: FixMatch avec 5% des données annotées



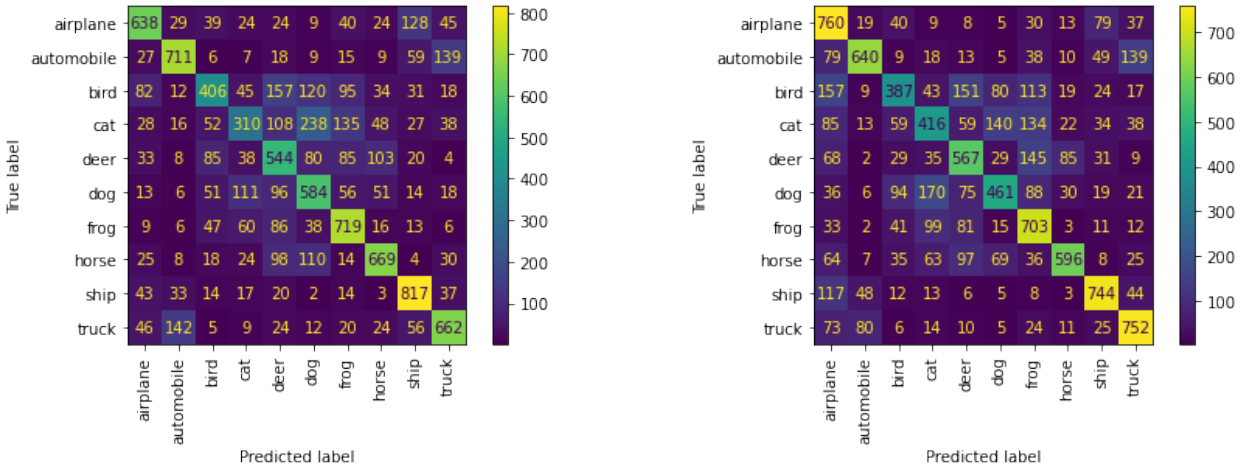
(a) Précision (b) Perte (c) Nombre de pseudo-labels

Figure 9: FixMatch avec 1% des données annotées

n'est pas encore suffisamment entraîné. Après 500 epochs, la perte totale est principalement due à la partie non supervisée et reste constante.

Concernant le nombre de pseudo-labels par batch, nous remarquons qu'il augmente après chaque epoch. Cela signifie que le modèle obtient une meilleure classification car de plus en plus de pseudo-labels passent le seuil de confiance.

Commentaire sur la misclassification.



(a) Modèle entraîné avec des images aléatoires

(b) Modèle entraîné avec des images uniques

Figure 10: Matrice de confusion de FixMatch sur des données de test (étiquetage à 10%)

D'après la matrice de confusion (Fig. 10), nous observons le défi typique de la classification d'images. Le modèle a du mal à classer les animaux à 4 pattes (chien, chat, cerf, cheval) ou à différencier les véhicules (automobile, camion). Il confond souvent les avions et les oiseaux, ce qui pourrait être dû au fait qu'ils ont tous deux des ailes. En outre, les objets les plus difficiles à détecter sont les oiseaux, seuls 40% d'entre eux sont correctement identifiés. Une perspective possible serait d'augmenter le nombre d'images d'oiseaux pour améliorer les performances du modèle.

5 Conclusion & perspective

Conclusion. Dans ce projet, nous avons réussi à implémenter FixMatch et analysé l'algorithme de FixMatch en variant la taille de données non-annotées (le hyper-paramètre μ).

Notre implémentation a obtenu de bons résultats pour le problème de classification avec 10% d'étiquetage : 75% de précision comparé à une précision de 76% obtenue sur les données entièrement annotées. Avec 1 et 5% données non-annotées, la précision n'est pas satisfaisant.

En plus, nous avons essayer une méthode de sélection des images représentatives avec un critère de l'unicité. Malheuresement, cette stratégie de sélection n'améliore pas la performance de FixMatch, voire est moins performante qu'une stratégie de sélection aléatoire.

Perspective. Pour avancer sur le sujet, une direction que nous voulons poursuivre est de concevoir une autre stratégie de sélection. Pour cela, il sera intéressant d'observer le nombre de pseudo-étiquettes par lot dans chaque classe. Cela peut nous aider à voir si le modèle a des difficultés à détecter certaines catégories d'images et donc si nous avons besoin de plus d'étiquettes pour cette classe par rapport aux autres.

References

[1] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[2] Y. Ouali, C. Hudelot, and M. Tami. An overview of deep semi-supervised learning, 2020.

[3] H. Pan, C. Liu, W. Wang, L. Yuan, H. Wang, Z. Li, and W. Liu. Img2vec: A teacher of high token-diversity helps masked autoencoders, 2023.

[4] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.