Semi-Supervised Learning

Xie Antoine Dembélé Alex

January 25, 2023

Contents

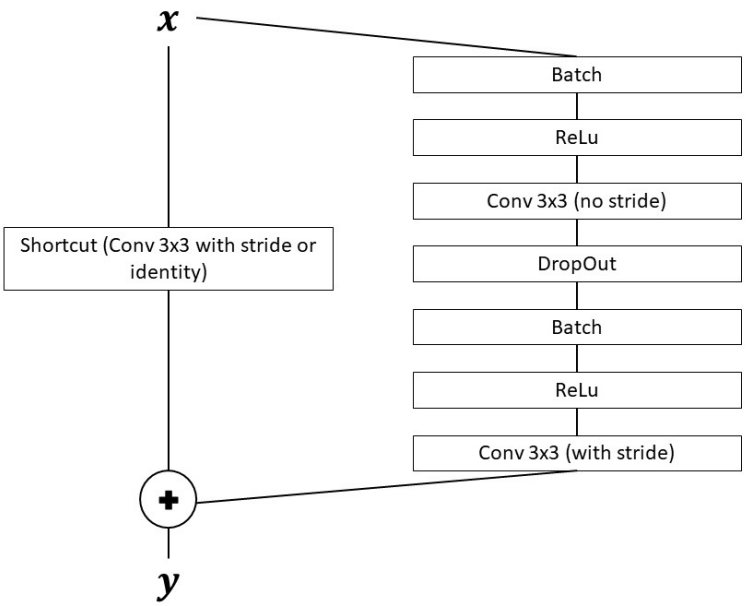
1 Introduction

Ce projet à pour objectif d'essayer plusieurs architecture d'apprentissage sur un jeu de données et de les comparer. Le but est de montrer la puissance du SSL sur les autres méthodes (notamment les réseaux de neurones en apprentissage supervisé)

2 Q0: WideResnet

WideResnet est une architecture de réseau de neurones se basant sur le modèle Resnet. Au lieu d'utiliser des raccords entre neurones, WideResnet augmente la largeur des couches de neurones (nombres de liaisons de sorties) plutôt que la profondeur des couches. Cela permet d'avoir moins de couches de neurones tout en conservant les performances.

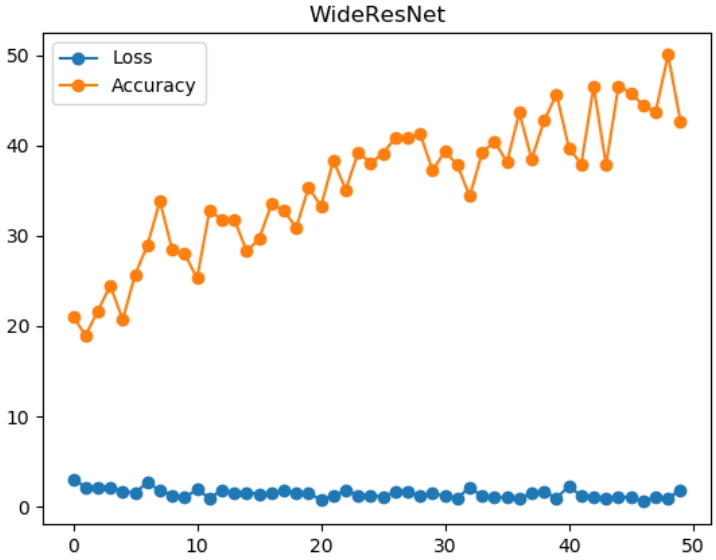
On dispose d'une couche de neurones appelée WideBasic sous cette forme



La particularité de cette couche est que l'entrée suit 2 chemins et y subit différentes transformations et que la sortie est donnée par une somme de ces 2 résultats. On crée ensuite des couches WideLayer qui consistent à mettre en série 4 couches WideBasic avec un stride différent sur la première couche WideBasic.

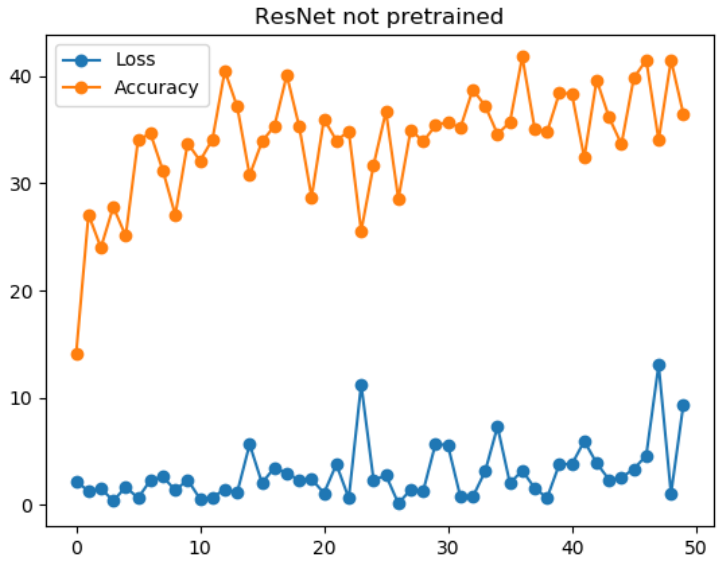
WideResNet consiste à construire un réseau avec la première couche comme une couche de convolution puis d'y mettre plusieurs couches WideLayer et de faire un pooling puis de créer une sortie linéaire (fully connected)

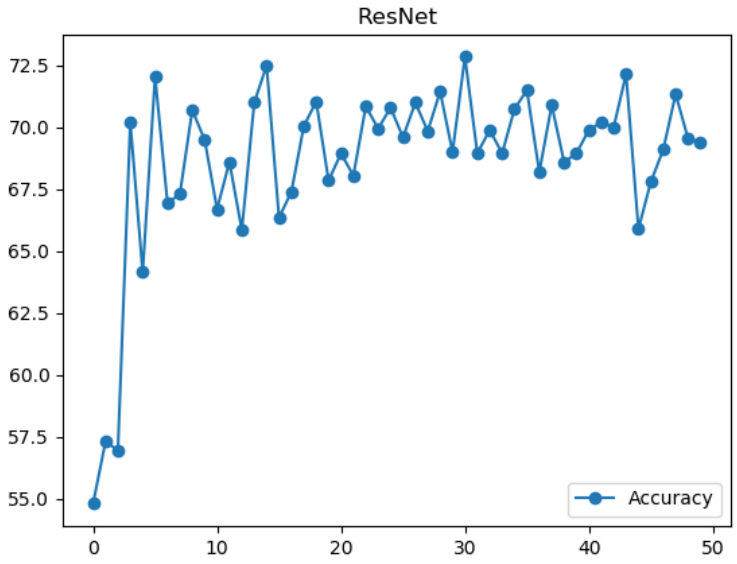
On constate que l'accuracy augmente bien au cours de l'entraînement (en moyenne) et que la Loss baisse

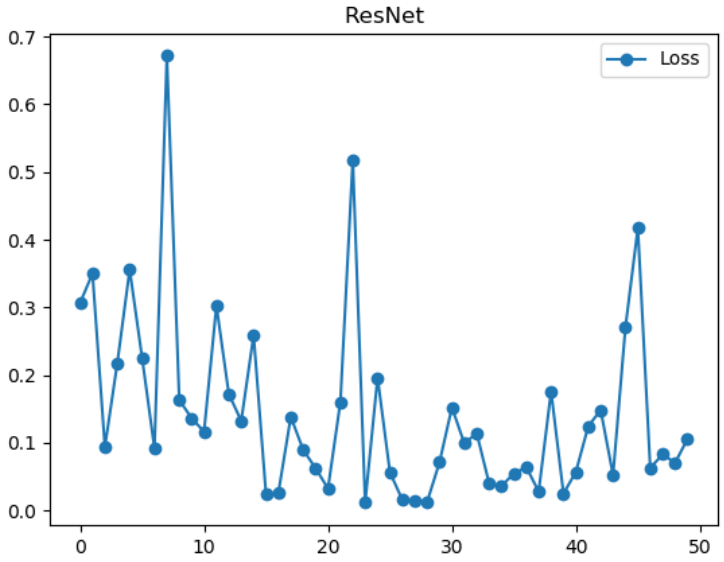

WideResNet
Loss
Accuracy
50
40
30
20
10
0
0
10
20
30
40
50

3 Q1: ResNet18

Pour le modèle pré-entraîné, on obtient une accuracy de 67.68%
Pour le modèle non pré-entraîné, on obtient une accuracy de 35.03% Il est donc intéressant de partir d'un modèle pré-entraîné pour l'entraînement d'un réseau de neurone ResNet.



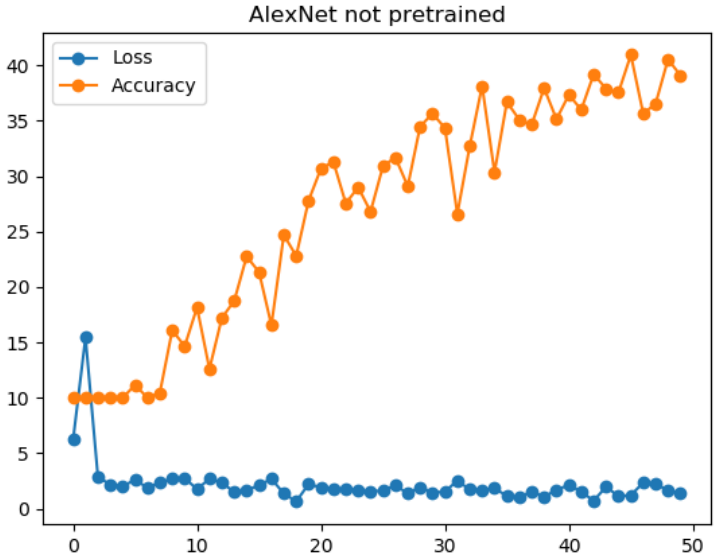


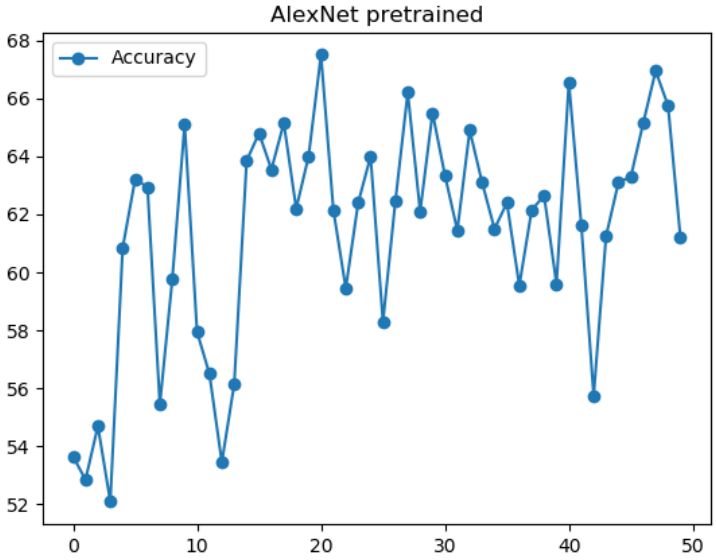


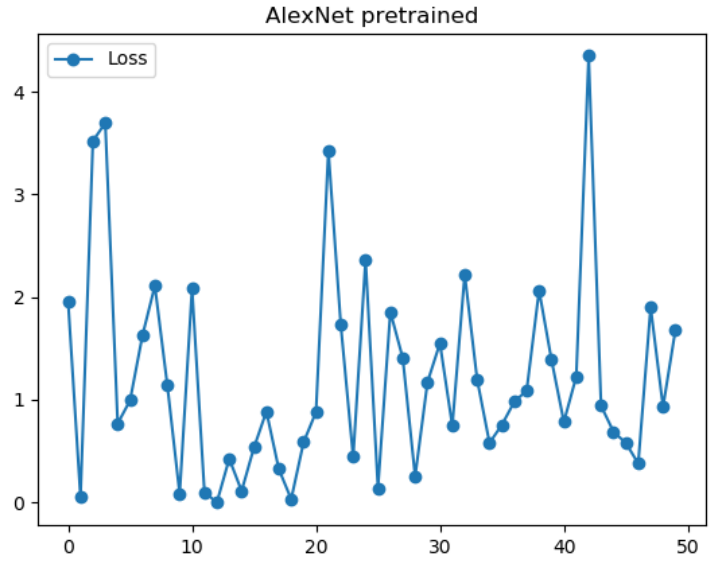
On constate que l'accuracy augmente bien au cours de l'entraînement (en moyenne) mais que la Loss se comporte bizarrement dans les deux cas. Elle ne diminue pas voir augmente spontanément dans le modèle non pré-entraîné, et elle baisse en moyenne mais augmente beaucoup spontanément dans le modèle pré-entraîné.

4 Q2: AlexNet

Pour le modèle pré-entraîné, on obtient une accuracy de 62.07%
Pour le modèle non pré-entraîné, on obtient une accuracy de 38.62% Il est donc intéressant de partir d'un modèle pré-entraîné pour l'entraînement d'un réseau de neurone AlexNet.







On constate un bon comportement pour le modèle non pré-entraîné. Cependant, la Loss du modèle pré-entraîné fait n'importe quoi.

Comme cela arrive pour les deux modèles pré-entraîné, le pré-entraînement sur un autre jeu de donnée doit affecter l'évolution de la Loss sur l'entraînement de nos réseaux. Les coefficients de certains neurones doivent totalement changer pour s'adapter au nouveeau jeu de données

5 Q3: SVM et Arbre

On arrive un peu à améliorer l'accuracy des arbres en limitant la profondeur maximale, cependant des fois cela n'a pas de sens.

Pour le svm, on arrive a une accuracy de 35% en décochant le gridsearch, mais il y a un temps de calcul très long pour gagner 1% d'accuracy. De plus, à partir de C=10, augmenter C n'a plus d'influence, ou de manière négligeable.

6 Q4: Comparaisons des différents modèles

On donne plusieurs tableaux comparatifs des modèles en fonctions de certains paramètres.

- On remarque que pour le SVM et le RandomForest, au mieux les résultats sont très semblables (autour de 34% de performance) pour une exécution en un temps quasi-similaire également.

SVM				
Kernel				
	linear	**poly**	**rbf**	**sigmoid**
Accuracy	26,16	25,42	34,41	25,46
Degree (pour kernel=poly)				
	2	**3**	**4**	**5**
Accuracy	25,59	25,42	18,49	18,16
Decision Function Shape				
	OvO	**OvR**		
Accuracy	34,41	34,41		
C				
	0,01	**0,1**	**1**	**10**
Accuracy	29,3	29,36	34,41	34,53

On peut monter à 35% pour les svm en faisant varier les paramètres avec GridSearch mais cela demande un gros temps de calcul !

Random Forest				
Criterion				
	gini	**entropy**		
Accuracy	31,79	30,97		
Profondeur max				
	None	**10**	**20**	**30**
Accuracy	31,79	31,98	31,71	31,79
Nombre d'arbres dans la forêt				
	10	**100**	**500**	**1000**
Accuracy	23	31,79	33,39	33,74

On constate que la profondeur optimale n'est pas la plus élevé! C'est la même chose pour un arbre simple où il y a des variations de 5% d'accuracy en limitant la profondeur à 6.

- Pour les réseaux de neurones, on a également crée des dataloaders avec des images de taille 256*256 pour faire la comparaison avec les 32*32 mais aussi car AlexNet ne fonctionne pas sur des images de trop petite taille. De plus on remarque que les réseaux préentraînés étaient entraînés sur des images de taille 256*256. On voit effectivement que changer la taille de l'image a un impact sur la performance des réseaux préentraînés mais ne change rien à ceux qui ne sont pas préentraînés. Enfin le learning rate a un impact sur la performance des réseaux et notamment sur AlexNet qui n'apprenait pas pour un learning rate au dessus de 0.01. On peut le baisser mais jusqu'à un certain seuil, en effet trop le baisser ne le fait pas apprendre assez vite. Enfin les réseaux de neurones avaient un temps d'exécution quasi-similaire et WideResNet 28-2 est celui qui obtenu les meilleurs résultats sur les différents tests (Il n'est pas pré-entraîné).

Neural Networks			
Learning rate			
	0,01	0,05	0,1
WideResNet	50,11	50,76	42,24
ResNet18 pretrained (32x32)	28,89	21,03	26,33
ResNet18 not pretrained (32x32)	39,66	37,65	33,82
ResNet18 pretrained (256x256)	67,68	60,65	61,72
ResNet18 not pretrained (256x256)	35,03	38,18	35,25
AlexNet not pretrained (256x256)	38,62		
AlexNet pretrained (256x256)	62,07	65,4	54,91

Résumé:

Méthode	**Accuracy**	**Commentaire**
WideResNet	50.67%	On a pas utilisé de réseau pré-entraîné
ResNet	67.68%	Réseau pré-entraîné
AlexNet	62.07%	Réseau pré-entraîné
SVM	35%	Long temps de calcul pour avoir les bons paramètres
Arbre	20.46%	Avec une profondeur de 6
RandomForest	33.74%	Avec 1000 arbres et pas de profondeur définie

On conclut que les réseaux de neurones avec convolutions sont les outils les plus adaptés pour notre problème. De plus, il vaut mieux utiliser des réseaux pré-entraînés car les résultats sont bien meilleurs (jusqu'à 2x supérieur)

7 Q5: Explication de Fixmatch

FixMatch est une méthode d'apprentissage qui a été pensé par Google Reasearch en 2020. Il a été pensé car en DL il est nécessaire d'avoir une grande base de données pour pouvoir effectuer l'apprentissage or créer cette base de données (et surtout poser des labels sur chaque élément) est très coûteux.

FixMatch a été originellement pensé sur la classification d'images à partir de réseaux de neurones tel que ResNet. Imaginons qu'on possède des images avec un label et d'autres non. Le réseau de neurones traditionnel ne peut apprendre que sur les images avec un label or FixMatch est une méthode qui vise également à apprendre sur les données sans label d'où le fait que c'est une méthode d'apprentissage semi-supervisé car il apprend également sur des données non annotées.

FixMatch utilise la data augmentation, technique utilisée pour augmenter la taille d'une base de données en utilisant des propriétés sur les objets sur lesquels on travaille. Par exemple si on souhaite classifier des images d'animaux, alors effectuer une symétrie horizontale, une faible rotation ou une faible translation ne change que très peu l'image et le réseau de neurones doit être capable de toujours donner le même animal en sortie peu importe le changement sur l'image. On distingue alors deux types de data augmentation :

- Weak augmentation : On retrouve des des techniques telle que la symétrie horizontale, la translation, le zoom/dezoom. Ces techniques sont catégorisées comme weak car on ne change qu'assez peu l'image et qu'un bon réseau de neurones entraîné serait capable de classifier l'image avec une accuracy qui varierait assez peu. Dans FixMatch, ils ont choisi de n'utiliser que la symétrie horizontale et des translations (qui ne vont que jusqu'à 12.5% de la taille initiale)

- Strong augmentation : Les techniques utilisées effectuent des changements plus drastiques sur l'image et il est bien plus compliqué de classifier l'image, parmi ces techniques on retrouve des changements sur la couleur, le contraste, la luminosité, etc... Parmi les techniques de strong augmentation, FixMatch utilise le CutOut qui consiste à enlever un carré (de taille raisonnable) de l'image. FixMatch effectue également un AutoAugment qui consiste à choisir un certain nombre de transformations parmi un set prédéfini, de choisir une magnitude qui définit la "puissance" de la transformation et d'effectuer ces transformations sur l'image. FixMatch utilise le RandAugment ou le CTAugment pour effectuer l'AutoAugment

L'apprentissage de FixMatch s'effectue en plusieurs étapes :

- Il faut tout d'abord créer la base de données avec des données avec label et des données sans label avec μ fois plus de données sans label que données avec label (il a été prouvé que plus μ est grand, meilleurs sont les résultats). Il faut également choisir une architecture pour les réseaux de neurones.

- On commence à travailler uniquement sur les données ayant un label, la fonction de perte à minimiser est l_s qu'on calcule à l'aide de l'entropie croisée

- Après avoir fait cette étape, on va effectuer une weak augmentation sur les données sans label. Et on va demander au réseau de neurones de classifier les images. Si la classification est donnée avec une certaine confiance supérieuse à un seuil (pour éviter d'avoir des pseudo label aberrant), on va poser un pseudo-label sur l'image en fonction de ce qu'a donné le réseau de neurones.

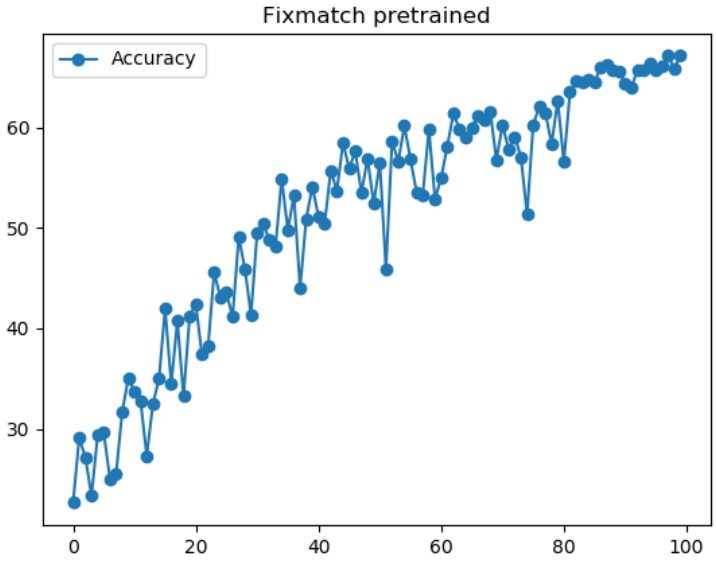
- On va maintenant effectuer une strong augmentation sur les données avec un pseudo-label et demander au réseau de neurones de classifier les images qui ont subi cette augmentation. A partir du pseudo label et de la sortie, on détermine l'entropie croisée et la fonction de perte l_u uniquement sur les données ayant un pseudo label

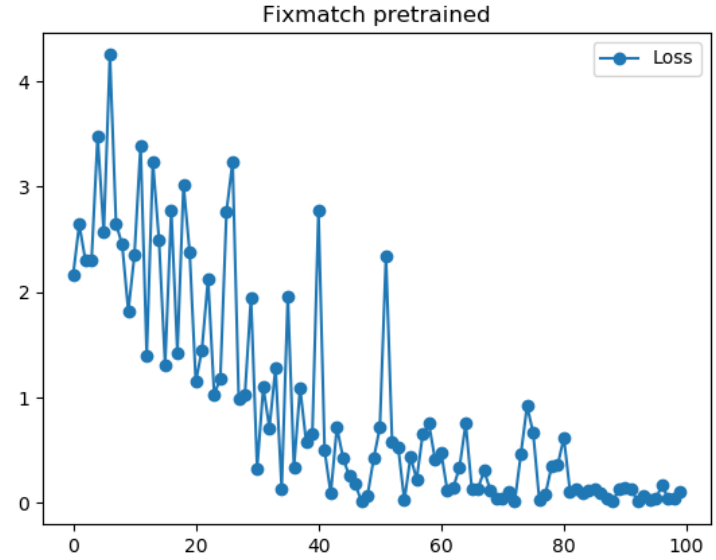
- On calcule la fonction de perte totale $l_s + \lambda_u l_u$ avec λ_u un paramètre de régularisation (qui sera fixé contrairement à d'autres algorithmes de SSL). A partir de cette fonction, on va remettre à jour les poids

Le fait d'introduire un seuil pour créer le pseudo label fait qu'au début le réseau de neurones ne pourra mettre aucun pseudo label sur les images. Donc au début l'entraînement s'effectuera comme si on entrainait traditionnellement un réseau de neurones sur des des données avec label, puis petit à petit il va être capable de créer quelques pseudo label et ainsi les données sans label vont commencer à avoir de plus en plus d'impact sur les poids du réseau

8 Q6 : Implémentation de FixMatch

On entraîne notre réseau sur 1000 images de validation, 47000 images non-laebelisées et 2000 images de test.
On obtient une accuracy de 65.94% ce qui est assez élevé. C'est au même niveau que les réseaux pré-entraîné sauf qu'ici le réseau n'est pas pré-entraîné.





On constate une évolution totalement normale de la Loss et de l'accuracy lors de l'entraînement avec Fixmatch.

9 Q7: Comment éviter le sur-apprentissage

Pour éviter le sur-apprentissage en Deep Learning on peut limiter la taille de nos architectures utilisés en fonction de la taille des données d'entraînement : pour les arbres de décisions on limite la profondeur des arbres, pour les réseaux de neurones on limite le nombre de neurones. Avec une architecture trop grande, on peut apprendre par coeur une base de données. On peut aussi utiliser un paramètre de régularisation pour contrôler le sur-apprentissage. Une dernière possibilité serait d'utiliser des données sans label pendant l'entraînement comme ça a été fait dans FixMatch, le réseau de neurones ne peut apprendre ces données par coeur étant donné qu'il ne sait pas quelle sortie il doit obtenir sur ces données.

10 Conclusion

A travers nos tests, on a constaté que différentes architectures d'apprentissage peuvent-être mise en oeuvre pour résoudre un problème, mais certaines sont préférables à d'autres selon le contexte. Ici, les réseaux de neurones sont plus efficaces. De plus, il est plus intéressant de partir d'un réseau de neurones pré-entraîné (même sur un jeu de donnée différent) plutôt que de repartir de zéro. Nous avons vu la puissance du SSL avec l'algoritme FixMatch, ses résultats sont meilleurs que ceux des autres (sans pré-entraînement)