Compte-Rendu

Deep Learning TP 5-6
Réseaux convolutionnels pour l'image

Yining Bao **Hanshuo WANG** **Yongtao WEI**

1. Partie 1 – Introduction aux réseaux convolutionnels

1. Considérant un seul filtre de convolution de padding p, de stride s et de taille de kernel k, pour une entrée de taille x × y × z quelle sera la taille de sortie ?

La formule pour calculer la taille de l'image de sortie est :

$$x' = (x+2p-k)/s + 1$$
$$y' = (y+2p-k)/s + 1$$

La taille de l'image de sortie est donc ((x+2p-k)/s + 1) * ((y+2p-k)/s + 1)* nb_kernel

Comme il n'y a qu'un seul noyau de convolution, la profondeur est de 1.

La taille de l'image de sortie est donc ((x+2p-k)/s + 1) * ((y+2p-k)/s + 1)* 1

Combien y a-t-il de poids à apprendre ?

L'objet à entraîner est le noyau de convolution, par conséquent, le nombre de paramètres avec entraînement est k*k*z.

Combien de poids aurait-il fallu apprendre si une couche fully-connected devait produire une sortie de la même taille ?

Comme la taille de l'image de sortie est x' * y' * 1, la couche entièrement connectée étend cette feature map en un vecteur unidimensionnel 1 * (x' * y'* 1). En fait, on peut comprendre que le fonctionnement de la couche entièrement connectée consiste à utiliser x' * y' * 1 filtres de taille x' * y' * 1 pour faire la convolution. Par conséquent, le nombre de poids à apprendre de la couche entièrement connectée est (x' * y' * 1)^2.

2. Quels avantages apporte la convolution par rapport à des couches fully-connected ? Quelle est sa limite principale ?

Avantages : Les couches entièrement connectées (FC) jouent le rôle de "classificateur" dans l'ensemble du réseau de neurones convolutifs, utilisé pour synthétiser l'extraction de caractéristiques faite par la couche convolutive précédente.

Limite principle : D'après le calcul du nombre de paramètres dans la question précédente, nous pouvons constater que la couche entièrement connectée aura des problèmes de redondance des paramètres (seuls les paramètres de la couche entièrement connectée peuvent représenter environ 80% de l'ensemble des paramètres du réseau).

3. Quel intérêt voyez-vous à l'usage du pooling spatial ?

Le goût de la couche de pooling est de réduire et compresser la taille de l'image, ce qui peut réduire la difficulté de l'apprentissage et réduire de manière appropriée les paramètres d'apprentissage. Nous pouvons augmenter les informations de profondeur de l'image pour compenser la perte d'informations de l'image qui rétrécit après l'utilisation de la couche pooling.

4. Supposons qu'on essaye de calculer la sortie d'un réseau convolutionnel classique (par exemple celui en Figure 2) pour une image d'entrée plus grande que la taille initialement prévue (224×224 dans l'exemple). Peut-on (sans modifier l'image) calculer tout ou une partie des couches du réseau ?

Nous pouvons calculer toutes les couches du réseau. La conception du modèle de réseau n'est pas limitée au nombre d'images d'entrée d'une taille spécifique. Par conséquent, lorsque la taille de l'image d'entrée est supérieure à 224, toutes les couches réseau peuvent également être générées.

5. Montrer que l'on peut voir les couches fully-connected comme des convolutions particulières.

Supposons que nous obtenions la taille de la dernière couche convolutive comme x*y*z. Selon l'idée générale, il est entièrement développé en un vecteur unidimensionnel puis connecté à la couche entièrement connectée (en supposant que la couche entièrement connectée a k éléments).Pour chaque élément de la couche entièrement connectée, c'est le produit de une couche d'expansion unidimensionnelle et une matrice de coefficients. Mais lorsque nous considérons ce problème comme une convolution, la couche entièrement connectée peut être considérée comme une forme de 1*1*k. C'est-à-dire que pour la dernière couche convolutive, nous utilisons k noyaux de convolution de taille x*y*z pour obtenir la couche convolutive de sortie. Chaque noyau de convolution x*y*z est équivalent à une matrice de coefficients, qui est multipliée par chaque élément, et la somme correspond à l'un des bits de sortie.

6. Supposons que l'on remplace donc les fully-connected par leur équivalent en convolutions, répondre à nouveau à la question 4. Si on peut calculer la sortie, quelle est sa forme et son intérêt ?

Si on remplace la couche Fully-connected 1x1x4096 par leur équivalent en convolution, c'est à dire on peut créer une couche de convolution avec neurones de 7x7 stride est 1 et pas de padding et 4096 de channels, et les couches Fully-connected suivant remplacent par les neurones de 1x1. Donc en général, on configure la taille de neurone comme la taille des entrées, l'intérêt est de la taille de la neurone n'a pas de limite avec l'entrée.

7. On appelle champ récepteur (receptive field) d'un neurone l'ensemble des pixels de l'image dont la sortie de ce neurone dépend. Quelles sont les tailles des receptive fields des neurones de la première et de la deuxième couche de convolution ? Pouvez-vous imaginer ce qu'il se passe pour les couches plus profondes ? Comment l'interpréter ?

On suppose que la taille de la couche convolutive est toujours k*k, et la couche de pooling est de 2*2 Max-Pooling, puis pour la première couche, champ récepteur = k*k, pour la deuxième couche,

champ récepteur = 4*k* k. Pour les couches plus profondes, en supposant que le nombre de couches est n, alors le RF de cette couche = 4^n *k*k.

Nous pouvons constater qu'à mesure que le nombre de couches augmente, le champ récepteur du noyau de convolution augmente de manière exponentielle, ce qui signifie que l'extraction de caractéristiques est progressivement intégrée des pixels minuscules aux grandes zones.

2. Partie 2 – Apprentissage from scratch du modèle

2.1. Architecture du réseau

8. Pour les convolutions, on veut conserver en sortie les mêmes dimensions spatiales qu'en entrée. Quelles valeurs de padding et de stride va-t-on choisir ?

D'après l'analyse de la partie précédente, la taille de la sortie de convolution est : x'= (x+2p-k)/s + 1. Si vous souhaitez conserver la même taille de l'image de sortie et de l'image d'entrée, alors x'= x. Lorsque nous déterminons la taille du noyau de convolution, le stride et le padding peuvent être ajustés dynamiquement. Par exemple, lorsque la taille du noyau de convolution est de 5*5, nous pouvons choisir le padding à 2 et le stride à 1.

9. Pour les max poolings, on veut réduire les dimensions spatiales d'un facteur 2. Quelles valeurs de padding et de stride va-t-on choisir ?

En utilisant max pooling pour réduire la dimension spatiale d'un facteur de deux, nous pouvons définir le stride sur 2 et le padding sur 0.

10. Pour chaque couche, indiquer la taille de sortie et le nombre de poids à apprendre. Commentez cette répartition.

conv1 : Le nombre de paramètres à apprendre est de 32*5*5*3 =2400,
et la taille de l'image sortie à travers cette couche convolutive est de 32*32*32.
pool1 :la taille de l'image de sortie après cette couche est de 16*16*32.
conv2 : Le nombre de paramètres à apprendre est de 64*5*5*32=51200,
et la taille de l'image sortie à travers cette couche convolutive est de 16*16*64.
pool2 : la taille de l'image de sortie après cette couche est de 8*8*64.
conv3 : le nombre de paramètres à apprendre est de 64*5*5*64=102400,
et la taille de l'image sortie à travers cette couche convolutive est de 8*8*64.
pool3 : la taille de l'image de sortie après cette couche est de 4*4*64.
fc4 : Le nombre de paramètres à apprendre est de 1000* (4*4*64+1) = 1025000
la taille de l'image de sortie après cette couche est de 1*1*1000
fc5 : Le nombre de paramètres à apprendre est de 10*(1000+1)=10010.
la taille de l'image de sortie après cette couche est de 1*1*10.

11. Quel est donc le nombre total de poids à apprendre ? Comparer cela au nombre d'exemples.

2400+51200+102400+1025000+10010=1191010

Comparé au modèle VGG16, le modèle VGG16 a plus de paramètres entraînables car le noyau de convolution VGG16 a un plus grand nombre et plus de couches convolutives.

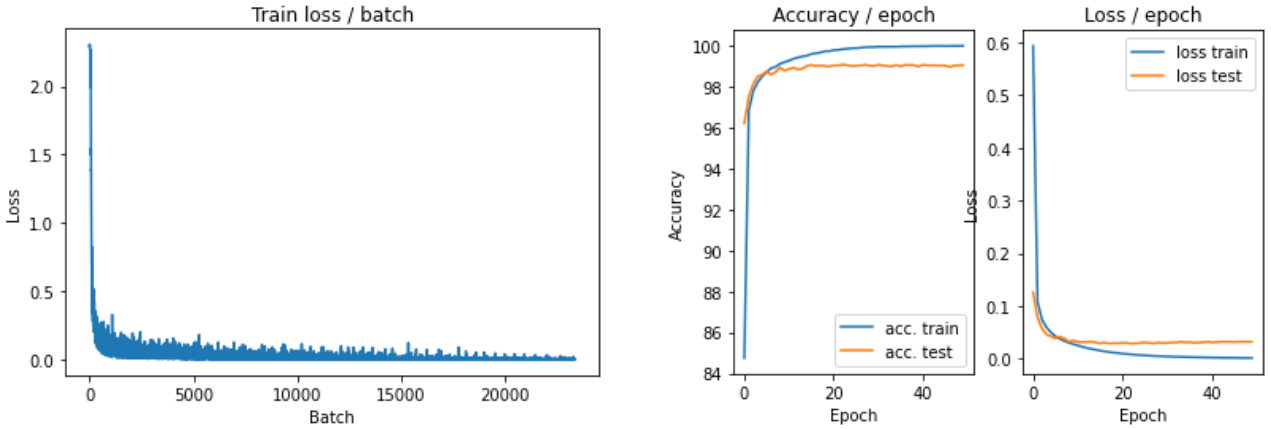
12. Comparer le nombre de paramètres à apprendre avec celui de l'approche BoW et SVM.

Le nombre de paramètres d'apprentissage du réseau de neurones convolutifs est bien supérieur à celui de BOW et SVM.

2.2. Apprentissage du réseau

13. Lisez et testez le code fourni. Vous pouvez lancer l'entrainement par cette commande : main(batch_size, lr, epochs, cuda=True)

main(128, 0.1, 50, cuda=True)



14. Dans le code fourni, quelle différence importante y a-t-il entre la façon de calculer la loss et l'accuracy en train et en test (autre que les données sont différentes) ?

Dans le code fourni, nous avons effectué les calculs de loss et de l'accuracy en train et en test aux endroits suivants de la fonction main :

```
# Phase de train
top1_acc, avg_top5_acc, loss = epoch(train, model, criterion, optimizer, cuda)
# Phase d'evaluation
top1_acc_test, top5_acc_test, loss_test = epoch(test, model, criterion,
cuda=cuda)
```

Nous pouvons constater que la différence entre eux est d'utiliser ou non le paramètre "optimizer" dans la fonction "epoch", nous arrivons donc à la fonction d'époque et pouvons voir l'instruction suivante:

```
model.eval() if optimizer is None else model.train()
```

Donc la différence est qu'ils utilisent des fonctions différentes. En consultant les documents officiels de Pytorch, on sait :

La fonction model.train() active le “Batch Normalization” et le “Dropout” s'il existe les couches de Batch Normalization ou les couches de Dropout dans le réseau de neurones, contrairement à model.eval().

Le dropout fait référence à la suppression aléatoire d'une partie des données pour éviter le surapprentissage.

Le fonctionnement de BN est :

1. Normaliser le Batch : x -> x^ ;
2. effectuer une transformation linéaire : y = ax^ + b .

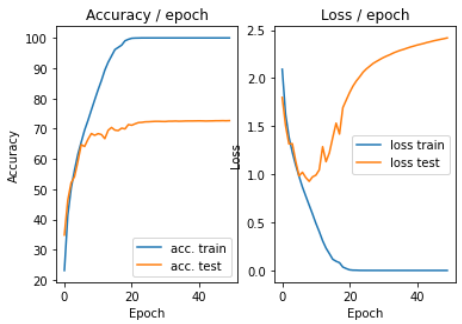
Ici, a et b sont également des paramètres qui doivent être entraînés pendant le phase d'entraînement.

Pour résumer, la différence est que nous activons le ”Dropout” et le “Batch Normalization” dans la phase d'entraînement, alors que nous n'avons pas besoin de les utiliser dans la phase d'évaluation.

15. Modifiez le code pour utiliser la base CIFAR-10 et implémenter l'architecture demandée ci-dessus. (la classe est datasets.CIFAR10). Attention à bien faire suffisamment d'époques pour que le modèle ait fini de converger.

```python
class ConvNet(nn.Module):
    """
    Cette classe contient la structure du réseau de neurones
    """

    def __init__(self):
        super(ConvNet, self).__init__()
        # On défini d'abord les couches de convolution et de pooling comme un
        # groupe de couches `self.features`
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, (5, 5), stride=1, padding=2), #conv1
            nn.ReLU(),
            nn.MaxPool2d((2, 2), stride=2, padding=0), #pool1
            nn.Conv2d(32, 64, (5, 5), stride=1, padding=2), #conv2
            nn.ReLU(),
            nn.MaxPool2d((2, 2), stride=2, padding=0), #pool2
            nn.Conv2d(64, 64, (5, 5), stride=1, padding=2), #conv3
            nn.ReLU(),
            nn.MaxPool2d((2, 2), stride=2, padding=0), #pool3
        )
        # On défini les couches fully connected comme un groupe de couches
        # `self.classifier`
        self.classifier = nn.Sequential(
            nn.Linear(1024, 1000), #fc4
            nn.ReLU(),
            nn.Linear(1000, 10), #fc5
            # Rappel : Le softmax est inclus dans la loss, ne pas le mettre ici
        )
```



16. Quels sont les effets du pas d'apprentissage (learning rate) et de la taille de mini-batch ?

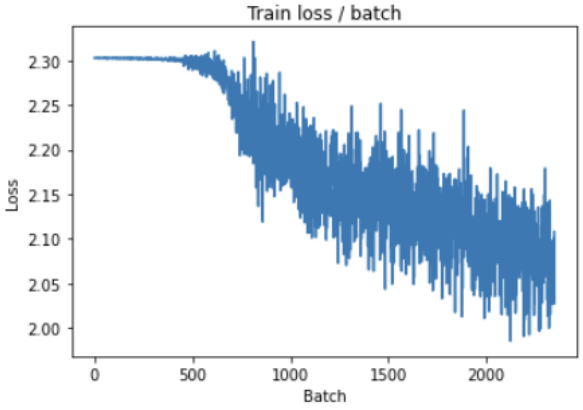
Le taux d'apprentissage peut contrôler le taux de mise à jour de chaque paramètre. Un taux d'apprentissage trop élevé entraînera une fluctuation trop importante de la mise à jour des paramètres et un échec de convergence, et un taux d'apprentissage trop faible entraînera une vitesse de convergence trop lente et prendra plus de temps.

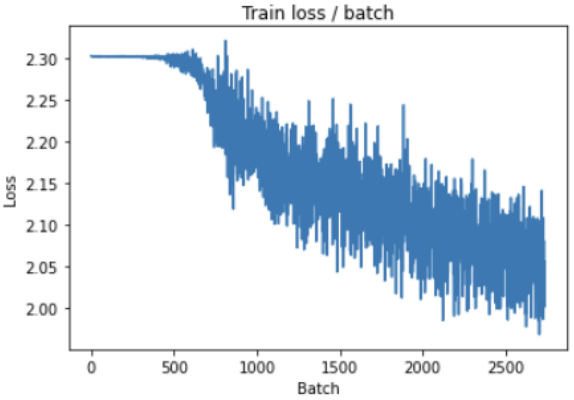
Lorsque le mini-batch est important, le temps d'entraînement peut être réduit et la stabilité peut être améliorée. Mais cela entraînera également une détérioration de la capacité de généralisation des résultats de la formation. Essentiellement, le nombre de sessions de formation a diminué.

17. A quoi correspond l'erreur au début de la première époque ? Comment s'interprète-t-elle ?

La perte de la première époque dépend de la qualité de l'initialisation des paramètres. Une bonne initialisation des paramètres aidera le modèle à converger, tandis qu'une mauvaise initialisation des paramètres affecte les résultats de l'apprentissage. Les méthodes d'initialisation de paramètres couramment utilisées incluent la "valeur aléatoire uniforme d'intervalle", etc.

18. Interpréter les résultats. Qu'est-ce qui ne va pas ? Quel est ce phénomène ?





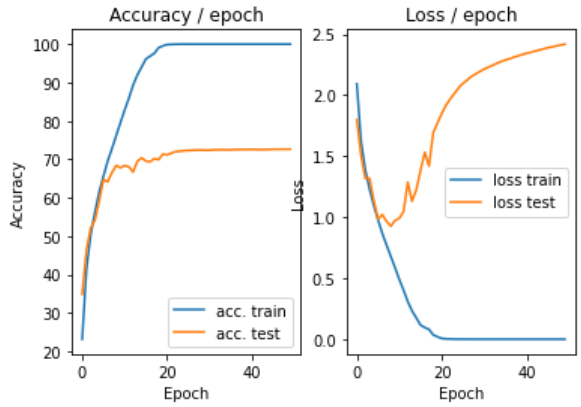
Grâce aux résultats de la formation, nous avons constaté que la courbe de la fonction de perte a une forte oscillation, que les résultats de la formation n'ont pas tendance à converger et que la perte est toujours importante.

C'est parce que nous utilisons la méthode d'entraînement par mini-batch, mais pas de normalisation. Parce que la distribution des données de chaque batch n'est pas nécessairement la même, cela augmentera la difficulté de l'apprentissage et rendra la convergence des paramètres instable.

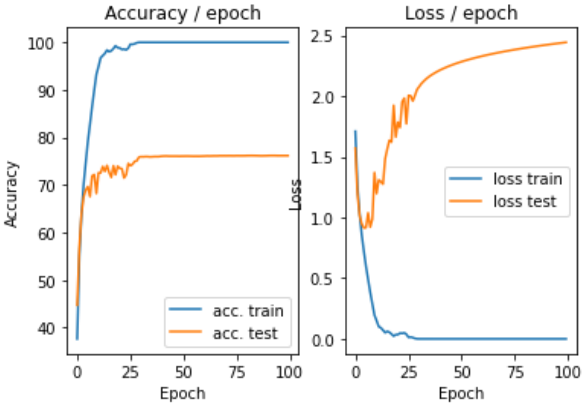
3. Partie 3 – Améliorations des résultats

3.1. Normalisation des exemples

19. Décrire vos résultats expérimentaux.



Avant normalisation



Apres normalisation

Par rapport à l'opération avant la normalisation, loss_test_normalisé et précision_test_normalisé deviennent plus oscillants, mais la valeur finale de précision_test_normalisé est légèrement plus élevée qu'avant :

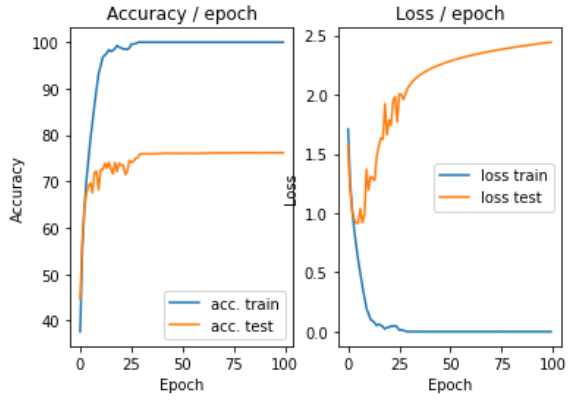
loss_test_avant -> 2.5	précision_test_avant -> 70%
loss_test_après -> 2.5	précision_test_après -> 75%

20. Pourquoi ne calculer l'image moyenne que sur les exemples d'apprentissage et normaliser les exemples de validation avec la même image ?

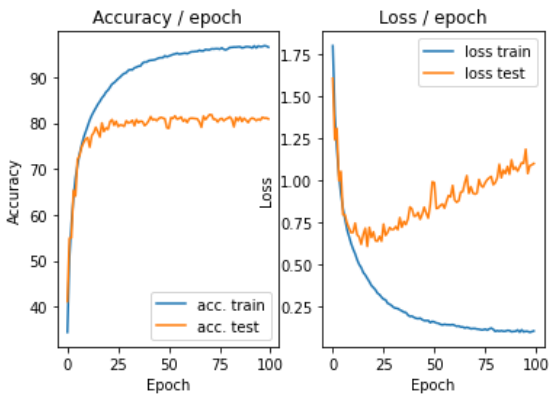
Le calcul de l'image moyenne de l'ensemble d'apprentissage peut réduire la complexité des données d'apprentissage et normaliser les données. L'utilisation de l'image moyenne de l'ensemble d'apprentissage dans l'ensemble de validation peut garantir que les données sont cohérentes avec la même distribution.

3.2. Augmentation du nombre d'exemples d'apprentissage par data augmentation

22. Décrire vos résultats expérimentaux et les comparer aux résultats précédents.



normalisation



normalisation + data augmentation

Par rapport aux résultats des seules opérations de normalisation, après amélioration des données, la précision de test est encore améliorée et le loss de test n'augmente plus, mais conserve une tendance à la baisse :

loss_test_avant -> 2.5	précision_test_avant -> 75%
loss_test_après -> 1.1	précision_test_après -> 80%

Par conséquent, l'amélioration des données peut réduire le surapprentissage.

23. Est-ce que cette approche par symétrie horizontale vous semble utilisable sur tout type d'images ? Dans quels cas peut-elle l'être ou ne pas l'être ?

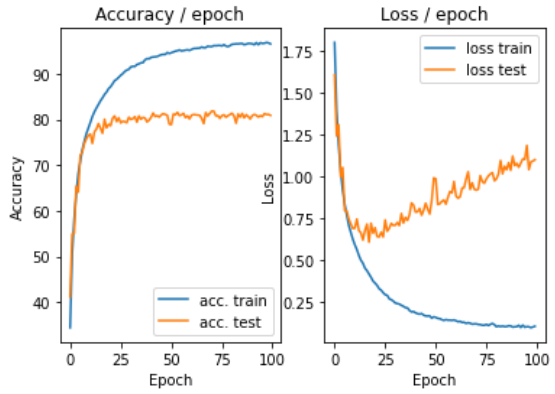
L'opération d'inversion de symétrie est limitée aux ensembles de données d'images qui ne sont pas sensibles à la direction, tels que les vêtements et les objets réels. Mais cette opération ne fonctionne pas pour les images directionnelles, telles que les nombres et le texte.

24. Quelles limites voyez-vous à ce type de data augmentation par transformation du dataset ?

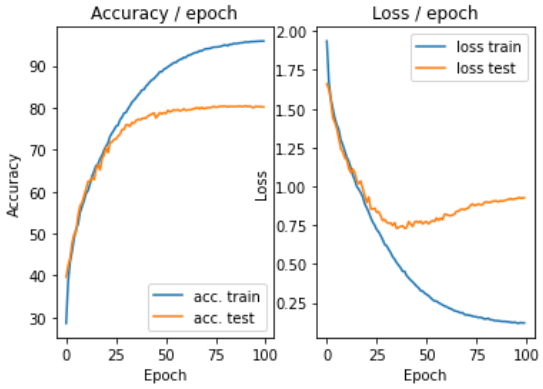
L'augmentation du nombre d'ensembles de données grâce à l'amélioration de l'image peut aider à améliorer la précision de la formation, mais comme le contenu de l'ensemble de données ne change pas beaucoup (par exemple, l'arrière-plan, les informations contenues), cela n'améliore pas la capacité de généralisation du modèle.

3.3. Variantes sur l'algorithme d'optimisation

26. Décrire vos résultats expérimentaux et les comparer aux résultats précédents, notamment la stabilité de l'apprentissage.



normalisation + data augmentation



normalisation + data augmentation + SGD

On peut voir que le phénomène de surapprentissage est encore réduit, l'amplitude de l'oscillation devient très faible, et la courbe de train et la courbe de test se chevauchent de plus en plus :

loss_test_avant -> 1.1	précision_test_avant -> 80%
loss_test_après -> 0.85	précision_test_après -> 80%

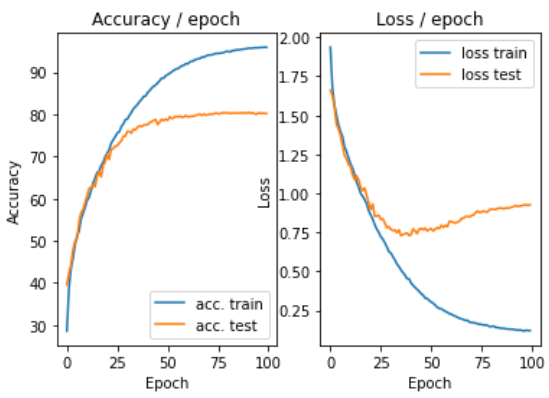
27. Pourquoi ces deux méthodes améliorent l'apprentissage ?

Le concept de méthode de quantité de mouvement est utilisé dans l'algorithme d'optimisation SGD. Lors de la mise à jour des informations de gradient, les informations de gradient précédemment formées sont utilisées. Lorsque la pente actuelle est en sens inverse de la pente précédente, cela signifie qu'il peut être en choc ou dans un endroit avec une pente douce.A ce moment, il est nécessaire de réduire au maximum le choc. La méthode de la quantité de mouvement peut le faire.Lorsque le gradient actuel et la dernière direction du gradient ont une amplitude similaire, le gradient approchera de zéro après soustraction, réduisant ainsi le choc.

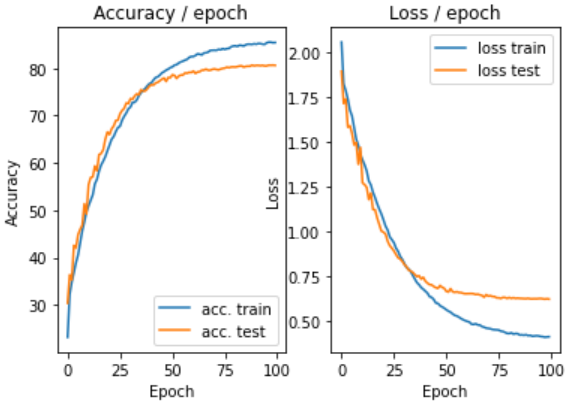
L'utilisation de la méthode de décroissance exponentielle du taux d'apprentissage peut également améliorer la vitesse et la précision de l'entraînement. Nous devrions avoir un taux d'apprentissage plus élevé au début de la formation, ce qui aide à sortir du minimum local. Dans la phase ultérieure de la formation, nous avons besoin d'un taux d'apprentissage plus faible pour que l'apprentissage tende vers le point le plus bas. À ce stade, un taux d'apprentissage plus élevé provoquera un choc.

3.4. Régularisation du réseau par dropout

29. Décrire vos résultats expérimentaux et les comparer aux résultats précédents.



normalisation + data augmentation + SGD



normalisation + data augmentation + SGD+dropout

Après le dropout, la perte de loss diminue encore, mais accuracy_train et loss_train ne sont pas aussi bons qu'avant :

loss_test_avant -> 0.90 précision_test_avant -> 80%

loss_test_après -> 0.65 précision_test_après -> 80%

30. Qu'est-ce que la régularisation de manière générale ?

La régularisation consiste à ajouter des termes réguliers à votre modèle pour re-contraindre l'espace des paramètres à une plage plus petite à résoudre, en évitant le sur-apprentissage.

31. Cherchez et "discutez" des possibles interprétations de l'effet du dropout sur le comportement d'un réseau l'utilisant ?

« Dropout » signifie une inactivation aléatoire. Lors de l'entraînement, il sélectionne au hasard des neurones, leur fonction d'activation est 0, de sorte que les neurones sont inactivés. Cette méthode permet de réduire efficacement le risque de surapprentissage, et elle peut également éviter que les paramètres d'un certain neurone ne soient trop déformés (car il est susceptible d'être inactivé pendant l'entraînement).

32. Quelle est l'influence de l'hyper-paramètre de cette couche ?

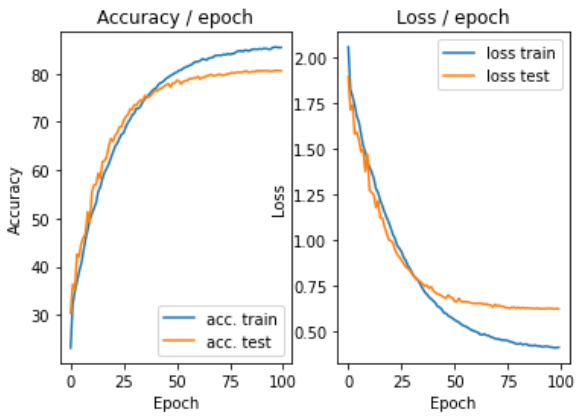
Il existe les deux hyperparamètres dans la couche de Dropout, un est le p, l'autre est inplace, cela signifie s'il faut écraser la valeur précédente avec la valeur calculée et il peut économiser la consommation de la mémoire. L'hyper-parametre p signifie que la probabilité qu'un élément soit mis à zéro, pendant le training, les sorties sont échelonnées d'un facteur 1/(1-p). Donc si un p plus grand, les sorties sont échelonnées par un facteur plus grand.

33. Quelle est la différence de comportement de la couche de dropout entre l'apprentissage et l'évaluation ?

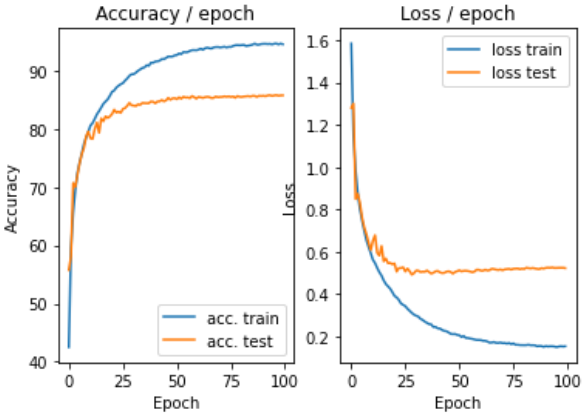
Dropout ne sera utilisé que pendant la formation, pas pendant les tests et la vérification.

3.5. Utilisation de batch normalization

34. Décrire vos résultats expérimentaux et les comparer aux résultats précédents.



...+ SGD+dropout



...+ SGD+dropout+BN

Après le batch normalization, la perte de loss diminue encore, et accuracy_train et loss_train sont aussi bons qu'avant, et notre taux de précision des prévisions est également devenu plus élevé:

loss_test_avant -> 0.65	précision_test_avant -> 80%
loss_test_après -> 0.55	précision_test_après -> 85%