Predicting CIFAR 10 data using neural networks

Zakaria IRAQI

29 septembre 2020

1 Construction du réseau de neurones, discussion sur le nombre de layers et l'architecture

Dans notre étude comparative pour effectuer la classification des images dans CIFAR 10, nous avons sélectionné trois modèles :

1.1 Modèle simple à trois layers :

20 canaux de convolution de noyau 5, suivi d'un max pooling de taille (2×2), qui réduira les images à des tailles (16×16), ensuite un second produit de convolution de noyau (5×5), et qui fait ressortir 15 canaux, suivi d'un triple flattening pour arriver à 10 outputs au final, auxquels on active finalement un softmax C'est le modèle le plus simple et le moins performant, il est à noter que, par expérience, le choix du nombre de canaux de sorties du premier produit de convolution augmente l'accuracy.(L'accuracy moyenne après 5 epochs reste limitée à 60 %).

1.2 Modèle complexe à trois layers :

La première layer prend les 3 canaux (RGB) en input, applique un produit de convolution de noyau (5×5) et de sortie de taille 108, à laquelle on applique un ReLu et un premier max pooling (2×2). Ce qui nous donne des images de taille réduite (16×16) Ensuite, nous appliquons une deuxième couche, constituée d'un produit de convolution de noyau (5×5), et faisant sortir 192 canaux, auquel on rajoute un ReLu et un max pooling(2×2), ce qui nous donne des images de taille (8×8). Finalement, une dernière layer qui permet d'aplatir le modèle, puis d'un dernier layer softmax. Il est aussi utile de remarquer que ce modèle applique un dropout des paramètres entre les layers 2 et 3. Ce modèle nous permet d'avoir une meilleure performance(70 % sous 10 epochs), et d'atteindre jusqu'à 80% sous 125 epochs.

1.3 Modèle "deep" à 5 layers :

La première layer prend les 3 canaux (RGB), applique une première convolution qui fait ressortir 32 canaux, de noyau (3×3), un ReLu et une renormalisation batch, puis une deuxième itération en reconvoluant les 32 canaux en 32 autres canaux avec un noyau (3×3), un ReLu et une renormalisation batch, puis finalement un max pooling qui réduira les images à des images de taille (16×16), tout ceci, en appliquant du dropout sur les paramètres. Le deuxième layer prend l'output du premier, applique une convolution de 64 canaux, et de noyau (3×3), suivie, d'un ReLu, une renormalisation batch, une deuxième convoluée de 64 canaux, de noyau (3×3), suivie d'un ReLu et d'une renormalisation batch, avant d'effectuer une nouvelle foix un max pooling, pour diviser la dimension de notre image. La troisième couche joue un rôle similaire à celui de la deuxième, sur 128 canaux. Ce qui donne au final des outputs de taille (4×4). Que l'on réaplatit dans la quatrième couche, puis dont on réduit la dimension pour n'avoir que 10 sorties. Avec ce modèle, nous avons pu avoir 72 % de bons résultats prédits sous 10 epochs, et 86 % sous 125 epochs.

1.4 Discussion sur le nombre de layers

Nous avons remarqué que rajouter des layers supplémentaires permettait d'arriver à de meilleurs résultats, avec un nombre d'epochs moindres, de plus, le fait d'empiler les layers donnait de meilleurs résultats sur les données testées.De plus, l'architecture composée d'une convolution + ReLu + renormalisation batch donnait de très bons résultats, si on la suivait d'un max pooling, qui permettait de réduire la dimension de travail, et donc de synthétiser les résultats.

1.5 Conclusion sur le choix du modèle

Nous avons choisi de faire la classification en utilisant un réseau de neurones multi-layer. Cela permet un apprentissage dans différents niveaux d'abstraction avec une précision assez importante. Afin de réaliser ce travail de classification, nous construisons un CNN avec 5 layers suivi d'un layer d'output. Ce dernier contient 10 nœuds correspondant aux 10 classes avec une fonction d'activation Softmax. Dans notre construction, nous avons un layer de convolution, un layer de pooling, un layer de flattening et puis un output layer.

2 Impact des choix entrepris

Nous allons choisir de travailler avec la base de données CIFAR10, pour laquelle on va étudier l'impact des paramètres suivants :

— L'augmentation des données
— Le nombre de couches (cf section 1)
— La méthode d'optimisation
— La fonction de coût
— Le Dropout

Mis à part ces paramètres-là, nous avons choisi, au lieu de travailler sur CPU, de travailler sous GPU, avec une NVIDIA 1050, qui permettait d'effectuer au maximum des calculs sous 1.4 Go, ce qui nous limite malheureusement le champ des possibilités. Nous travaillerons aussi avec un batch sur les données fixé à 64. Nous avions remarqué que sur CIFAR 10, augmenter le batch pouvait s'avérer intéressant.

Dans les sections suivantes, nous allons travailler avec le réseau de neuronnes "deep", en faisant varier paramètre par paramètre. Le "deep" standard incluera du dropout, de la data augmentation, du dropout et sera entrainé en utilisant unee descente de gradient stochastique comme optimiseur, et l'entropie croisée comme critère de loss.

2.1 Augmentation des données

Nous allons choisir deux différentes approches, une approche où l'on travaille uniquement avec les données CIFAR 10, sans introduire de modification, et une autre qui introduit une data augmentation composée de rotations, de crop, et de flips horizontaux. Tout ceci sera comparé sous le même nombre d'epochs (125 epochs) pour un classifieur 'deep', avec un batch de 64. Il est à noter que, à chaque epoch, les deux modèles utilisent le même nombre de données de départ, mais, pour celui qui est data-augmenté, les donnéees sont prééchantillonnées d'un plus large dataset (avec les rotations, ...).

Les deux modèles ont pris la même durée sous 125 epochs, à peu près 2h30. On remarque que le modèle data augmenté a de meilleurs résultats généraux, que ce soit en terme d'accuracy ou de loss sur les données train et test. Nous pouvons résumer ceci dans ce tableau synthétique, où la précision, et le test de précision sont donnés sur les 10000 images **test**. Chacun des deux modèles a pris à peu près 700 Mo de mémoire GPU.

Augmentation	Précision modèle	Test de précision	Avion	Voiture	Oiseau	Chat	Cerf	Chien	frog	cheval
Sans	77 %	80.1 %	82 %	88%	62%	54%	78%	67 %	87%	81%
Avec	81%	86%	85%	88%	81%	65%	80%	67%	87%	81%

2.2 Méthode d'optimisation

Nous allons dans cette section faire un comparatif entre plusieurs méthodes d'optimisation, notamment la descente de gradient stochastique, Adagrad (gradient adapatatif), l'optimiseur ADAM (adaptive moment estimation), et l'optimiseur Adadelta (n'utilisant que l'information de premier ordre).

Comme préindiqué, le modèle sera avec dropout, avec une cross entropy, il prendra aussi 2h30 à trainer sur 125 epochs. les résultats synthétiques sont les suivants :

Optim	Précision modèle	Test de précision	Avion	Voiture	Oiseau	Chat	Cerf	Chien	frog	cheval
SGD	81%	86%	85%	88%	81%	65%	80%	67%	87%	81%
Adagrad	61%	62.73%	69%	72%	43%	49%	50%	57%	66%	60%
Adam	77%	80.6%	87%	90%	62%	53%	81%	67%	72%	91%
Adadelta	53%	52.75%	62%	50%	34%	38%	32%	38%	58%	62%

Nous pouvons remarquer depuis ce tableau que les deux méthodes SGD et Adam, avec tous les autres paramètres fixés, donnent des résultats considérablement meilleurs que l'Adagrad. De plus, l'Adadelta ne permet pas du tout de faire une bonne prediction et n'arrive pas à un bon résultat(le modèle est precis à uniquement 50%.) Une étude plus détaillée permettrait de trancher entre le SGD et Adam, nous travaillerons dans le reste uniquement sous SGD, mais optimiser suivant Adam paraît être une très bonne idée aussi.

2.3 Fonction de coût

Dans cette section, nous allons étudier l'impact du choix de la fonction de coût. Nous aurons de ce fait à étudier plusieurs fonction de coût, avec tous les autres paramètres préfixés comme montré précedemment les critères à étudier seront :

- Entropie croisée : $-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$, avec M le nombre de classes et y, l'indicateur binaire, p la probabilité que l'objet soit effectivement de la classe indiquée
- Negative Log Likelihood Loss : $\boldsymbol{\mathcal{L}} = -\frac{1}{n}\sum_{i=1}^{n} \log(\hat{y}^{(i)})$ Il s'agit d'une mesure "douce" de la précision qui intègre l'idée de confiance probabiliste. Elle est intimement liée à la théorie de l'information. Et elle est similaire à l'entropie croisée (en classification binaire) ou à l'entropie croisée multi-classe (en multi-classification) mathématiquement.
- Norme deux : $\boldsymbol{\mathcal{L}} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$

Les résultats synthétiques sont les suivants :

Cost	Précision modèle	Test de précision	Avion	Voiture	Oiseau	Chat	Cerf	Chien	frog	cheval
CE	81%	86%	85%	88%	81%	65%	80%	67%	87%	81%
NLL	78%	81.38%	83%	94%	63%	63%	70%	67%	82%	81%
MSE	73%	74%	80%	81%	75%	60%	73%	59%	79%	73%

Nous pouvons en déduire de ce fait que le critère d'entropie croisée est un critère de perte qui conduit à un bon résultat, comparé à celui du NLL ou encore du MSE (ajusté). Ce sera le critère que l'on sélectionnera. Une combinaison de ces critères peut aussi conduire à des résultats intéressants.

2.4 Dropout

Dans ce paragraphe, nous allons étudier plusieurs modèles distincts, dont on ne fera varier que le paramètre de dropout, et que l'on fixera comme suit :

- Un modèle sans Dropout, dans toutes les couches, ce qui augmentera la complexité de notre back-propagation à chaque itération.
- Un modèle avec un dropout modéré ($0.2<\text{Dropout}<0.3$), qui nous permettra d'oublier à chaque itération à peu près 26% de nos paramètres.

— Un modèle avec de forts dropouts (0.5<Dropout<0.6), ce qui fera ignorer à chaque fois plus de la moitié des paramètres.
— Finalement un modèle à dropout très élevé (Dropout = 0.8), qui ne laissera que 20% de paramètres à chaque itération ...

Les résultats sont synthétisés dans le tableau suivant :

Dropout	Précision modèle	Test de précision	Avion	Voiture	Oiseau	Chat	Cerf	Chien	frog	cheval
0	80%	81.76%	83%	90%	75%	53%	76%	74%	89%	87%
0.2/0.3	81%	86%	85%	88%	81%	65%	80%	67%	87%	81%
0.5/0.6	69%	73.03%	78%	86%	60%	42%	61%	67%	75%	78%
0.8	56%	41%	57%	60%	40%	36%	58%	42%	66%	64%

Nous remarquons qu'inclure un léger Dropout (0.3) permettait une meilleure précision dans la classification qu'un modèle sans Dropout, de plus, les modèles à fort Dropout conduisent à des résultats moins intéressants, ce qui nous pousse à considérer un modèle à faible dropout comme classifieur.

2.5 Conclusion

D'après l'étude comparative précédente, nous pouvons déduire plusieurs paramètres qui, combinés, permettent de réaliser une meilleure classification de notre modèe à nombre d'epochs fixé. Ces paramètres seront : un réseau de neuronnes 'deep', avec une petite partie de Dropout, et une méthode de SGD, avec un critère de perte d'entropie croisée. Nous utiliserobs le même réseau de neuronnes(avec un différent output) pour les données CIFAR 10, pour analyser la reproductivité et l'extensivité des résultats.

3 Passage Cifar-10 à Cifar-100

Dans cette section, nous allons utiliser le modèle le plus efficace en CIFAR-10 sur CIFAR-100, à savoir un réseau de neuronnes "deep" avec un Dropout de 0.2 et 0.3 sur le modèle CIFAR 100, sur un batch de 64 images, avec comme méthode d'optimisation l'optimisation SGD, et la Cross Entropy Loss comme fonction de coût. Pour deux durées, 125 et 250 epochs. Nous effectureons ensuite la classification sur le modèle "deep" pour estimer l'efficacité de notre classifieur.

Dataset	Réseau utilisé	Nb Epochs	Précision modèle	Test de précision
CIFAR 10	Deep	125	81%	86%
CIFAR 10	Deep	250	79%	83.56%
CIFAR 100	Deep	125	54%	54.69%
CIFAR 100	Deep	250	50%	56.24%

Nous remarquons que l'estimateur "Deep" a une bonne efficacité à 125 epochs, et commence légèrement à overfitter quand on augmente le nombre d'epochs. D'un autre côté, sous CIFAR 100, les tests sont tout de même relativement bons, quand on sait le grand nombre de classes disponibles. Des modèles dans l'état de l'art permettraient sûrement d'aller jusqu'à 90 % de précision, mais prendraient plus de mémoire de calcul, et de temps de calcul, chose que l'on ne pouvait pas se permettre dans cette étude pour fournir un comparatif détaillé des paramètres qui donnent une bonne classification d'images. Une double classification pourrait aussi être utile dans le cas de CIFAR 100, qui utiliserait en même temps les labels et catégorie. De ce fait, on peut conclure que l'utilisation des nouvelles méthodes sur les réseaux de neuronnses, en ce qui concerne la classification, permet de donner de très bon résultats, sans pour autant nécessiter d'énormes ressources de calcul.

Annexe

Dans cette annexe, nous montrerons les différents résultats obtenus sur les différents modèles entraînés, en les comparant avec le modèle "idéal" : "deep", avec un léger dropout, avec une entropie croisée et une descente de gradient stochastique comme méthode d'optimisation, avec de la data augmentation.

Les codes envoyés jointement peuvent être modifiés pour tester chacun des paramètres précédents. Le fichier pour tester CIFAR 100 est aussi mis à part. Ils nécessitent d'avoir une préinstallation de pytorch sous cuda.

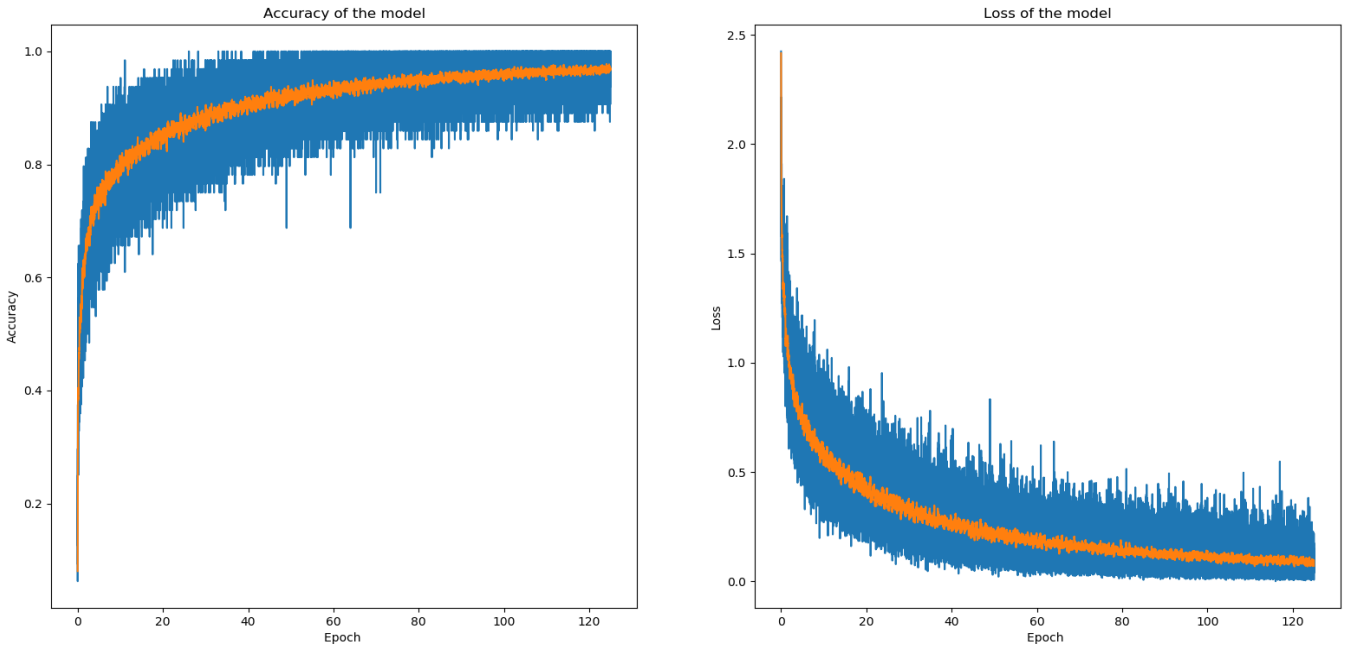

FIGURE 1 – Le modèle 'Deep' le plus performant : résultats sur les données test

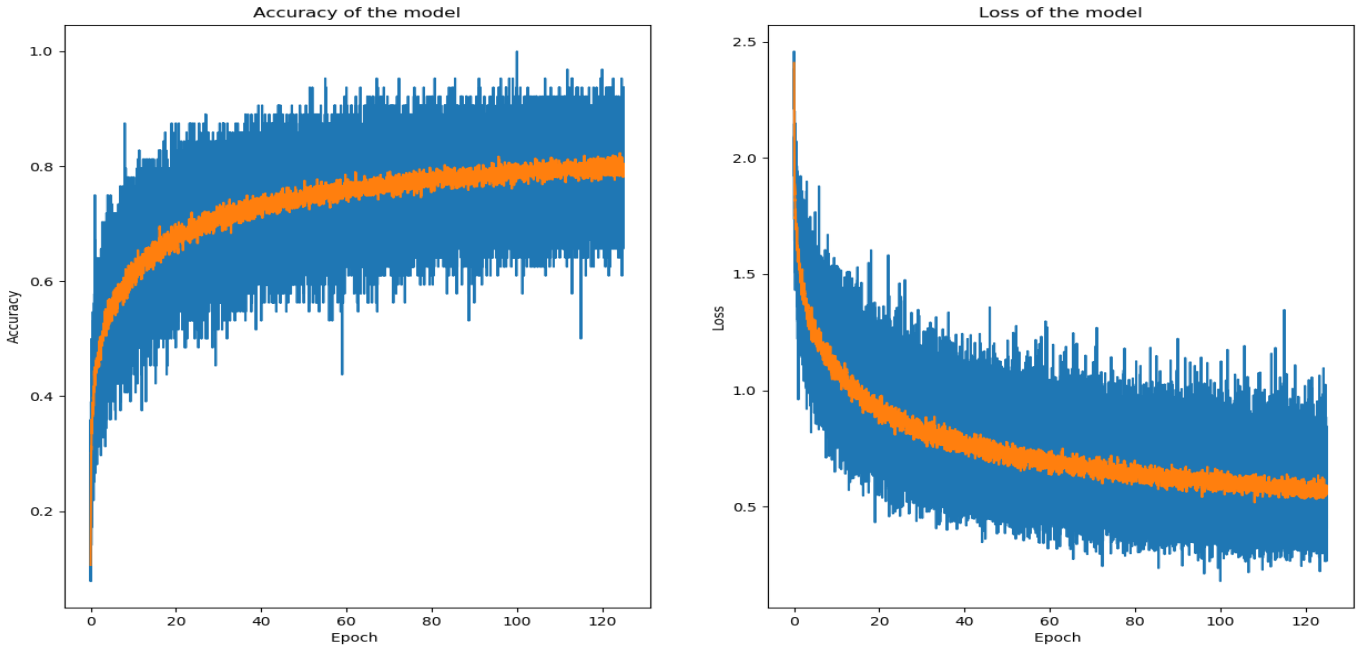

FIGURE 2 – Le modèle 'Deep' sans data augmentation : résultats sur les données test

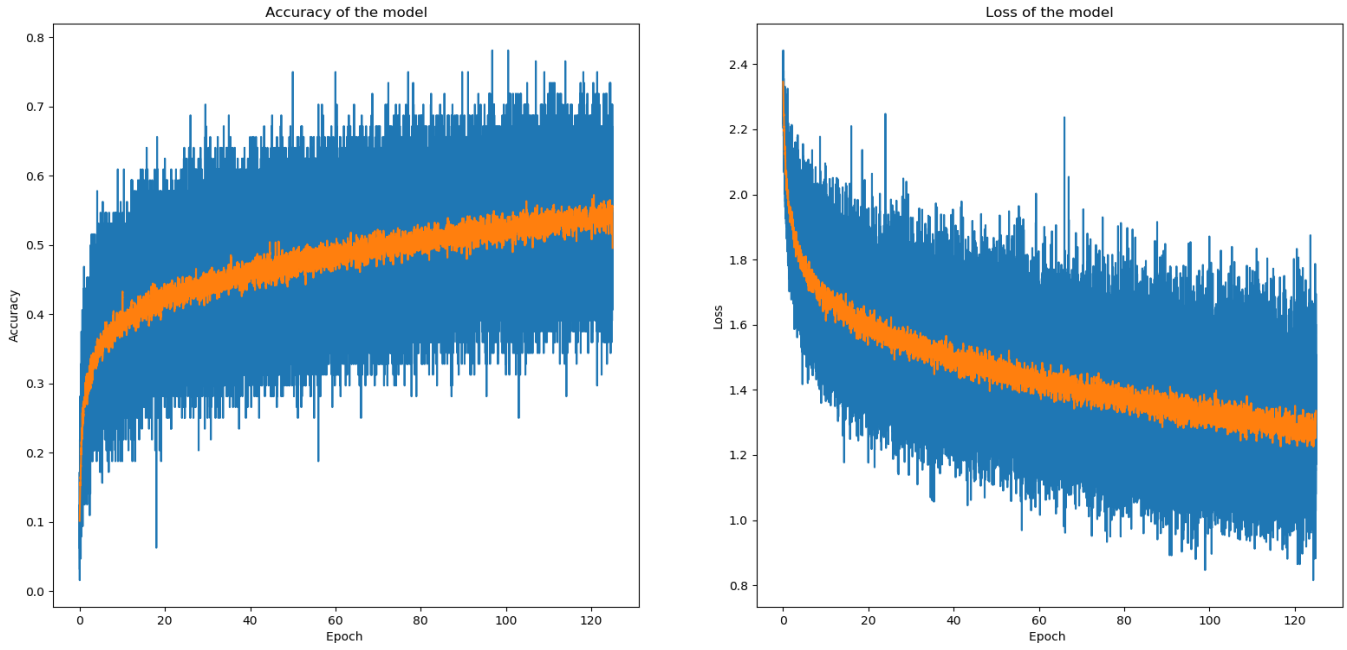

FIGURE 3 – Le modèle 'Deep' avec Adagrad comme méthode d'optimisation : résultats sur les données test

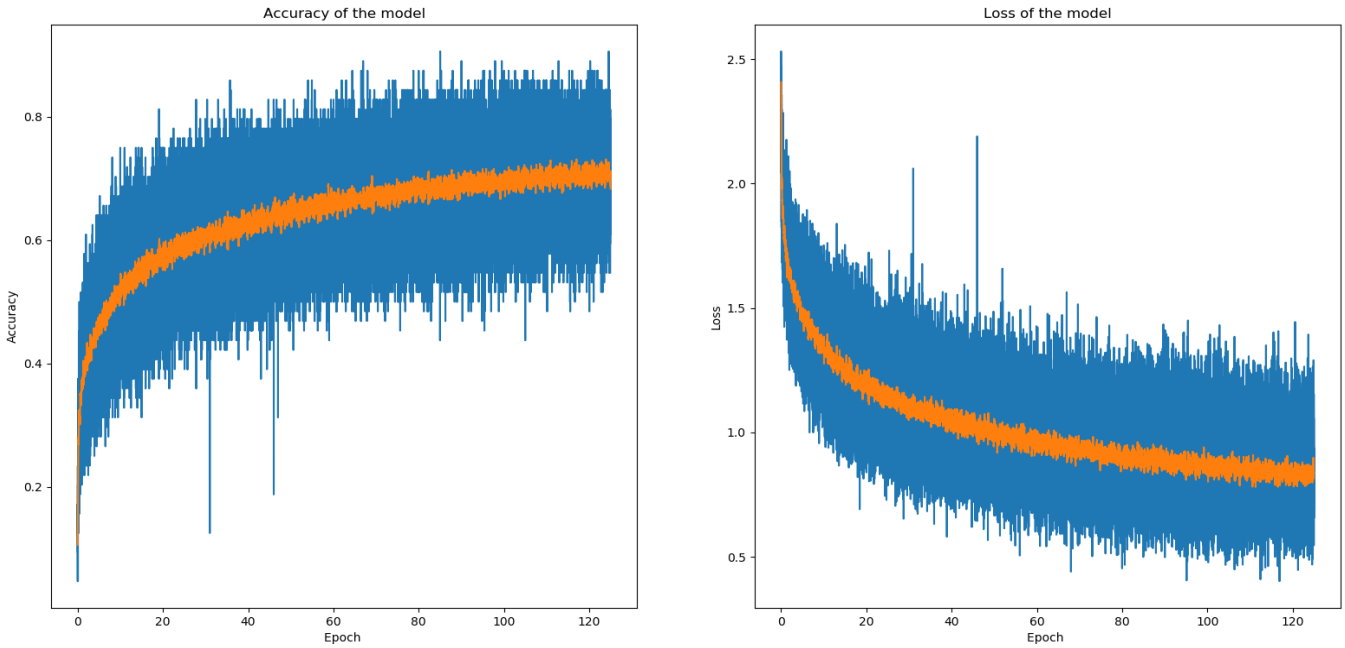

FIGURE 4 – Le modèle 'Deep'avec un Dropout de 0.5 : résultats sur les données test

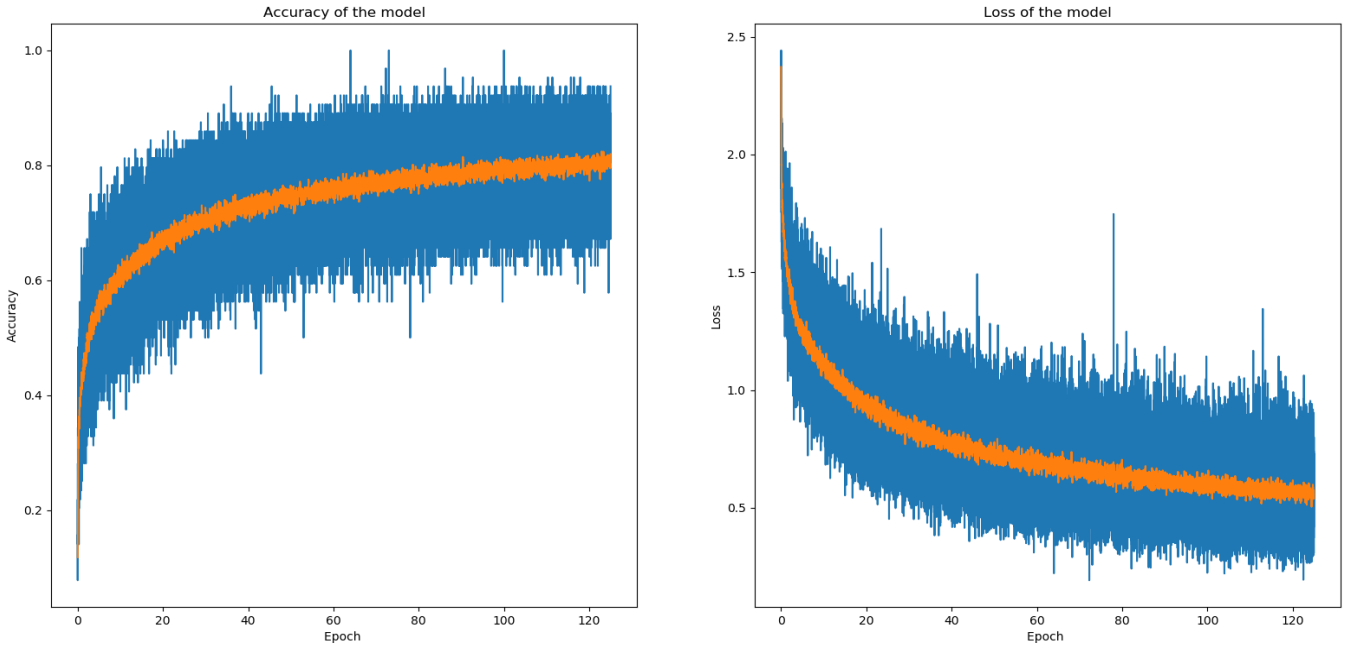

FIGURE 5 – Le modèle 'Deep'avec la vraisemblance log comme critère : résultats sur les données test

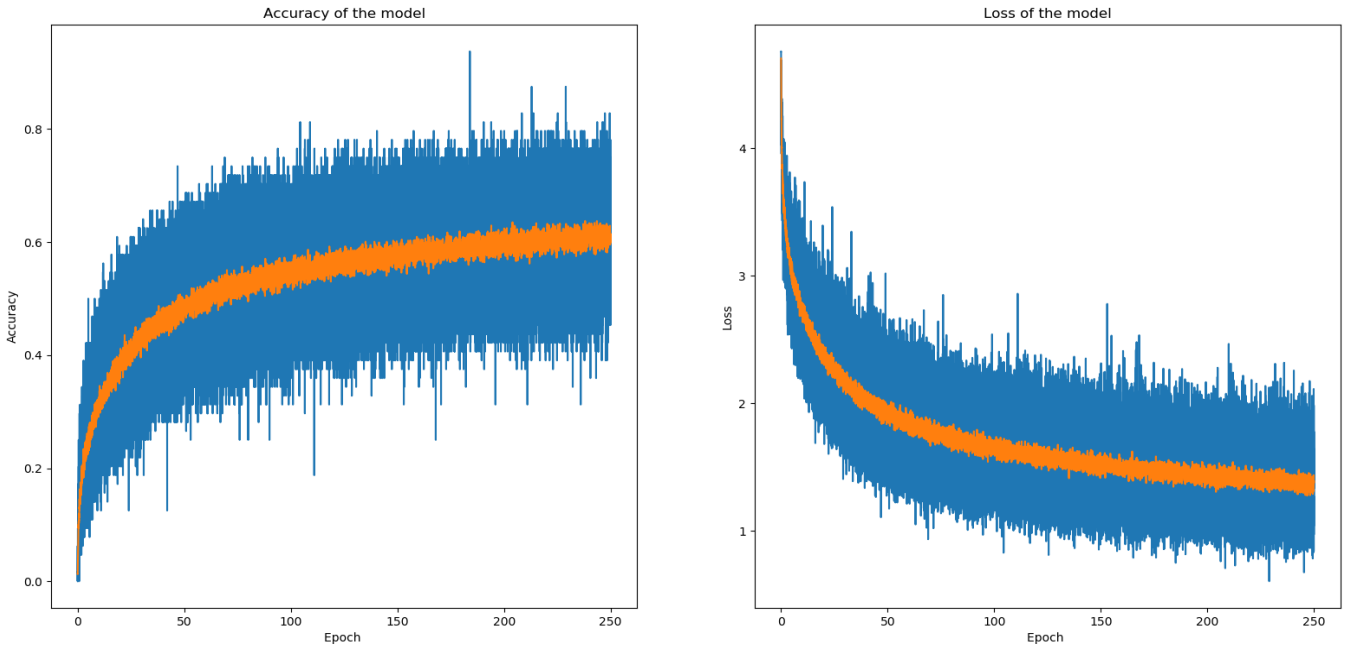

FIGURE 6 – Le modèle 'Deep' appliqué aux données CIFAR 100 : résultats sur les données test