Legrand François

3IABD1

Rapport Projet Annuel 3IABD

Sommaire :

1 : Problématique

2 : Constitution du DataSet

3 : Implémentation et utilisation du Perceptron multicouches

4 : Expérimentations

1 : Problématique

On avait pour objectif de déterminer quel animal était en face de nous, notamment dans le cadre d’une visite au zoo ou d’un safari à l’aide d’une photo.

On s’est penché sur le machine learning, car les méthodes de programmations classiques ne permettent pas une classification d’image complexe. L’objectif était donc de créer des modèles de ML de réaliser des apprentissages en faisant des expérimentations afin d’avoir un modèle qui généralise au mieux possible la classification. Enfin nous devions sauvegarder ces modèles afin de les réutiliser dans une application web afin qu’ils soient utilisables facilement pour tous.

2 : Constitution du DataSet

Pour réaliser un apprentissage en Machine learning il faut récupérer une grande quantité d’images, et les étiqueter avec les noms de leur classe, le nombre d’image nécessaire va dépendre de la complexité des patterns des images, du nombre de classes, de la résolution de l’image et de l’algorithme utilisé. Dans notre, cas nous nous sommes limités à 3 classes afin de ne pas avoir besoin de trop d’images pour avoir des résultats, pour l’instant il y’a 550 images par classe environ. Nous avons créé un script de scrapping python utilisant selenium afin d’imiter un utilisateur et de télécharger des images depuis google image. Nous avons aussi un petit script pour dupliquer tout le dataset dans une résolution personnalisée, dans notre cas on va utiliser 32x32 et 16x16 pixels, garder une résolution trop importante pourrais rendre l’apprentissage trop long et trop complexe.

3 : Implémentation et utilisation du Perceptron multicouches

Nous devions lors de ce projet implémenter différents algorithmes de ML, un modèle linéaire, un MLP (Perceptron Multicouches), un RBF (Radial Basis Functions) et un SVM (Support Vector Machine).

Nous avons implémenté les algorithmes en C++ sous forme d’une librairie utilisable dans n’importe quel autre langage qui peut exécuter du C. les algorithmes sont ensuite utilisé dans du python afin de faciliter la manipulation de données.

Malheureusement, par manque de temps et étant seul dans le groupe, je n’ai pu implémenter que le modèle linéaire et le MLP. J’ai d’abord implémenté le modèle linéaire en l’appliquant au cas de test donné par notre professeur que nous devions valider, lorsqu’il a fonctionné pour toutes les classifications binaires simple, j’ai ensuite ajouté une fonction de classification multi classe en one vs one afin de réaliser les cas de test avec 3 classes.

Je suis ensuite passé au MLP, après avoir implémenter une première version je n’ai pas eu de mal à faire la plupart des cas de test de classification sauf le dernier qui était trop complexe pour mon MLP, j’ai ensuite commencé à implémenter les premières logs des entrainement avec l’accuracy le long des epochs, j’ai ensuite ajouter la loss avec un MSE (Mean Square Error), ces logs s’affichent dans la sortie python pendant l’entrainement et sont aussi noter dans un fichier csv pour tracer des graphiques et voir les courbes en direct avec tensorboard. J’ai par la suite construit une matrice de confusion afin d’avoir plus d’information sur l’efficacité du modèle.

Après avoir implémenté ces logs j’ai réussi à valider le dernier cas de test de classification, la multicross.

Une image contenant motif, capture d’écran, Caractère coloré, fleur

Le contenu généré par l’IA peut être incorrect.

Durant les premières expérimentations sur le dataset je me suis rendu compte de la différence de poids entre des points en 2d donc 2 paramètres et des images avec 32 pixels de résolutions avec RGB donc 32\*32\*3 = 3072 paramètres.

Les résultats d’apprentissage étaient très faibles malgré un MLP avec beaucoup de couches et de neurones par couche, j’avais donc de gros problème au niveau de la lenteur de l’apprentissage, j’ai donc ajouté la possibilité de multi-threader le MLP, afin que certaines opérations soient parallélisées et donc plus rapide. J’ai ensuite expérimenté beaucoup d’hyperparamètres différent de manière méthodique afin d’essayer d’optimiser la généralisation.

4: Expérimentations

layer\_sizes = [size\*size\*3, 128, 64, 32, 3)]

epochs = 5000

learning\_rate = 0.01

batch\_size = 32

size = 32\*32

Une image contenant capture d’écran, texte, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Rectangle, diagramme

Le contenu généré par l’IA peut être incorrect.

On est ici dans du underfitting, l’accuracy n’est pas satisfaisante et la loss continue de diminuer doucement, on va augmenter le nombre de neurones par couche pour voir si l’accuracy augmente et si on commence à observer de l’overfitting.

layer\_sizes = [size\*size\*3, 256, 128, 64, 3)]

epochs = 5000

learning\_rate = 0.01

batch\_size = 32

size = 32\*32

Une image contenant capture d’écran, texte, Tracé

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, diagramme, Rectangle

Le contenu généré par l’IA peut être incorrect.

L’augmentation du nombre de neurone par couche amène un overfitting tôt dans l’entraînement (seulement au bout de 1000 epochs), on va tester sur 2 couches cachée et un peu plus de neurones par couche pour le prochain entraînement.

epochs = 15000

learning\_rate = 0.005

batch\_size = 32

layer\_sizes = [size\*size\*3, 256, 128, 3)]

size = 32\*32

Une image contenant capture d’écran, texte

Le contenu généré par l’IA peut être incorrect.Une image contenant capture d’écran, texte, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, diagramme, nombre

Le contenu généré par l’IA peut être incorrect.

On commence à avoir une accuracy intéressante et toujours pas d’overfitting en vue, on va maintenant tester avec une seule couche caché mais beaucoup plus de neurone pour voir si le modèle gagne en accuracy

epochs = 20000

learning\_rate = 0.005

batch\_size = 32

layer\_sizes = [size\*size\*3, 512, 3)]

size = 32\*32

Une image contenant texte, capture d’écran, Caractère coloré

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, diagramme, Rectangle

Le contenu généré par l’IA peut être incorrect.

Une seule couche cachée dans le MLP malgré un nombre de neurones élevé semble rester dans du underfitting malgré un bon nombre d’epochs, il a du mal à généraliser et obtient de moins bon résultat que deux couches caché avec moins de neurones. On pourrait essayer de trouver un minimal de loss plus rapidement en augmentant légèrement le learning\_rate, mais on va pour l’instant retourner sur deux couches cachés car c’est ce qui fonctionne le mieux pour l’instant, on va augmenter le nombre de neurones par couche et on va augmenter le nombre d’epochs.

layer\_sizes = [size\*size\*3, 512, 256, 3)]

epochs = 20000

learning\_rate = 0.005

batch\_size = 32

size = 32\*32

Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, diagramme, Rectangle

Le contenu généré par l’IA peut être incorrect.

On a une légèrement moins bonne accuracy qu’avec moins de neurone et autant de couche, cependant on observe pas encore d’overfitting, on va relancer un test pour voir si l’entraînement avec ce réseau de neurones peut avoir de meilleurs résultats et on va augmenter légèrement le learning rate pour avoir une loss minimal plus rapidement.

layer\_sizes = [size\*size\*3, 512, 256, 3)]

epochs = 20000

learning\_rate = 0.0075

batch\_size = 32

size = 32\*32

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, diagramme, Rectangle

Le contenu généré par l’IA peut être incorrect.

L’augmentation du nombre de neurone et la légère augmentation du learning rate n’a pas été concluant, de l’overfitting est arrivé à peine au milieu de l’entrainement, et l’augmentation de la capacité du modèle avec un nombre élevé de neurone n’a pas permis une meilleurs accuracy au contraire, on va pour le prochain entrainement reprendre la structure de réseau de neurone qui a le mieux fonctionner en augmentant le nombre d’epoch et en repranant un learning rate plus faible, on va aussi faire le test avec une résolution d’image plus faible pour voir si moins de paramètre peut permettre de mieux généraliser dans notre cas et par la même occasion diminuer le cout en puissance de l’entrainement et avoir la possibilité de mettre plus d’epochs

layer\_sizes = [size\*size\*3, 256, 128, 3)]

epochs = 40000

learning\_rate = 0.005

batch\_size = 32

size = 16\*16

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Tracé, ligne

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, carré, Rectangle

Le contenu généré par l’IA peut être incorrect.

La diminution de la résolution des images a améliorer l’améliorer la généralisation du modèle, peut-être qu’en 32x32 les images avaient trop de pattern complexe et trop de bruit que le MLP ne pouvait pas traiter ou pour lesquels ils tombaient plus facilement en overfitting, par manque de temps cette expérimentation très positive est la dernière dans le cadre du projet annuel, cependant on pourrait réfléchir à ce qui pourrait encore plus améliorer la généralisation avec ce MLP. On pourrait déjà ajouter à la librairie C++ du MLP un learning rate dynamique qui diminue légèrement au fil des epochs, afin de réduire l’overfitting sur des longs apprentissages, on pourrait donc imaginer de plus longs apprentissages avec plus de 100K ou 200K epochs, on pourra aussi étendre le nombre d’image au sein du dataset cela pourra améliorer la généralisation en masquant le plus possible de « bruit » dans le dataset. On pourrait aussi essayer d’implémenter d’autres fonctions d’activation notamment la relu pour les couches cachées et softmax pour la couche de sortie que j’avais essayé sur le dataset et qui permettent des modèles beaucoup plus précis en très peu d’epochs.

Experimentation bonus :

layer\_sizes = [size\*size\*3, 3)]

epochs = 200000

learning\_rate = 0.005

batch\_size = 32

size = 16\*16

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Rectangle, carré

Le contenu généré par l’IA peut être incorrect.

Un perceptron multicouche sans couche cachée (représentant donc un modèle linéaire est finalement plus efficace que ce que je pensais pour ce type de donnée, plus efficace qu’un MLP avec trois couches cachées par exemple. On a un minimum de loss vers les 75K epochs et l’accuracy ne monte pas au-dessus de 60% après ces 75K epochs.

4 : Conclusion

Pendant les expérimentations je n’ai pas beaucoup utilisé les matrices de confusions que je générais, j’en ai surtout eu l’utilité lors des premiers debugs, elle pourrait servir à remarquer s’il y’a des problèmes d’équilibrage dans le dataset ou si pour certaines classes la complexité de classification fait qu’il y’a beaucoup plus d’erreur entre deux classes précises. N’utilisant que trois classes dans le dataset, ayant un nombre d’images par classe équilibrer et des animaux ne se ressemblant pas, nous n’avons pas eu ces problèmes.

Je me suis aussi rendu compte qu’il ne fallait pas forcément un MLP avec un réseau de neurones complexe pour avoir de bon résultat, c’est plutôt l’inverse dans notre cas, et un MLP sans couche cachée avec beaucoup d’epochs nous donnes de très bons résultats, il faudrait encore réaliser d’autre expérimentation pour voir si un MLP sans couche cachée peut être le plus efficace pour notre cas. S’il généralise autant que le MLP avec deux couches de 256 et 128 autant utiliser le modèle sans couche cachée car il a un coût économique et en temps beaucoup plus faible et est donc plus efficace. On pourrait ensuite essayer d’implémenter les autres algorithmes afin de voir si on peut mieux généraliser.

Durant ce projet annuel, j’ai pu découvrir les différentes étapes d’utilisation du machine learning en plus de l’implémentation d’algorithme, avec la constitution du dataset, les expérimentations et la création d’une application pour disposer les modèles à des utilisateurs.