Projet-TP IRN

Rapport de projet IRN

Élèves :
Chloé Vidal
Matéo Bachelier

Enseignante :
Laurence Denneulin

07 Juin 2024

Table des matières

1 Introduction

La classification des galaxies est une tâche essentielle en astronomie qui permet de comprendre la formation et l'évolution de l'univers. Traditionnellement, cette classification était réalisée manuellement par des astronomes. Cependant, avec l'augmentation exponentielle des données astronomiques, il est devenu impraticable de traiter ces informations sans l'aide de méthodes automatisées. Les réseaux de neurones se sont donc révélés être des outils puissants pour la classification des images de galaxies.

Dans le cadre de notre cours d'introduction aux réseaux de neurones, nous souhaitons réaliser un programme capable de ranger des galaxies dans 10 catégories différentes. On peut donc se demander s'il est possible de classifier ces images efficacement. Pour cela, il existe déjà des solutions pour étudier des galaxies à partir d'image, cependant elles sont pour la plupart extrêmement poussées, et cherche par exemple à estimer leur taille ou encore leur température à partir uniquement de photo, en plus de pouvoir déterminer leur type. Pour ce projet, nous nous limiterons aux dix catégories proposées avec le dataset de nos galaxies. Le but de cette activité est de créer un réseau de neurones capable de classifier ces images.

2 État de l'art

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Plusieurs études ont démontré l'efficacité de ces réseaux pour la classification des galaxies. Par exemple, Dieleman et al. (2015) ont utilisé des CNN, de l'anglais Convolutional Neural Network, pour classer des galaxies issues du projet Galaxy Zoo, obtenant des résultats supérieurs aux méthodes traditionnelles. De même, Huertas-Company et al. (2018) ont appliqué des réseaux profonds aux données du Sloan Digital Sky Survey (SDSS), montrant une amélioration significative par rapport aux techniques manuelles.

Ils comportent deux parties bien distinctes :
La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Un modèle CNN classique comprend des couches de convolution pour l'extraction de caractéristiques, des couches de pooling pour réduire la dimensionnalité, et des couches entièrement connectées pour la classification Le résultat de la partie convolutive est ensuite envoyé vers la seconde partie, qui est constituée de couches entièrement connectées. Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories.

Dans notre projet, nous avons utilisé un CNN pour classifier les images du dataset Galaxy10 DECals. Notre modèle comprend plusieurs couches de convolution et de pooling, suivies de couches entièrement connectées. Nous avons expérimenté avec différents optimiseurs (SGD, Adagrad, Adam) et taux d'apprentissage pour identifier la meilleure combinaison. TensorBoard a été utilisé pour surveiller et ajuster le processus d'entraînement en temps réel.

3 Méthode

Dans cette étude, nous avons entraîné un réseau de neurones convolutionnel pour classifier des images de galaxies en utilisant le dataset Galaxy10 DECals. Nous avons exploré différents optimiseurs et taux d'apprentissage pour trouver la meilleure combinaison permettant de maximiser la précision de notre modèle.

3.1 Prétraitement des Données

Le prétraitement des données est une étape cruciale pour garantir que les images sont dans un format optimal pour l'entraînement du modèle. Nous avons effectué les opérations suivantes :

— **Redimensionnement des Images** : Toutes les images ont été redimensionnées à une taille uniforme de 64x64 pixels pour réduire l'utilisation de la mémoire et assurer une cohérence dans les dimensions des entrées du réseau.
— **Normalisation des Images** : Les valeurs des pixels des images ont été normalisées dans la plage [0, 1] afin de faciliter l'apprentissage du modèle et d'accélérer la convergence.
— **Division du Dataset** : Le dataset a été divisé en ensembles d'entraînement et de test, avec 80% des données utilisées pour l'entraînement et 20% pour le test. Cette division permet d'évaluer les performances du modèle sur des données non vues pendant l'entraînement.

3.2 Construction du Modèle

Nous avons construit un réseau de neurones convolutionnel en utilisant une architecture séquentielle comprenant les éléments suivants :
— **Couches de Convolution** : Trois couches de convolution avec des tailles de noyau spécifiées ont été utilisées pour extraire des caractéristiques spatiales des images. Chaque couche de convolution est suivie d'une couche de pooling pour réduire la dimensionnalité des données.
— **Couches de Pooling** : Les couches de pooling réduisent la dimensionnalité des caractéristiques extraites, diminuant ainsi le nombre de paramètres et le risque de surapprentissage.
— **Couches Denses** : Après la phase de convolution et de pooling, les données sont aplaties et passées à travers des couches denses entièrement connectées pour effectuer la classification. Une couche de dropout a été ajoutée pour la régularisation et pour prévenir le surapprentissage en désactivant aléatoirement une fraction des neurones durant l'entraînement.
— **Fonction d'Activation** : Nous avons utilisé la fonction d'activation ReLU (Rectified Linear Unit) pour les couches de convolution et denses afin d'introduire de la non-linéarité dans le modèle.
— **Fonction de Perte et Optimiseur** : Le modèle a été compilé avec la fonction de perte sparse categorical cross-entropy et divers optimiseurs (SGD, Adagrad, Adam) pour mettre à jour les poids durant l'entraînement. Chaque optimiseur a été testé avec différents taux d'apprentissage.

4 Expérimentation et résultats

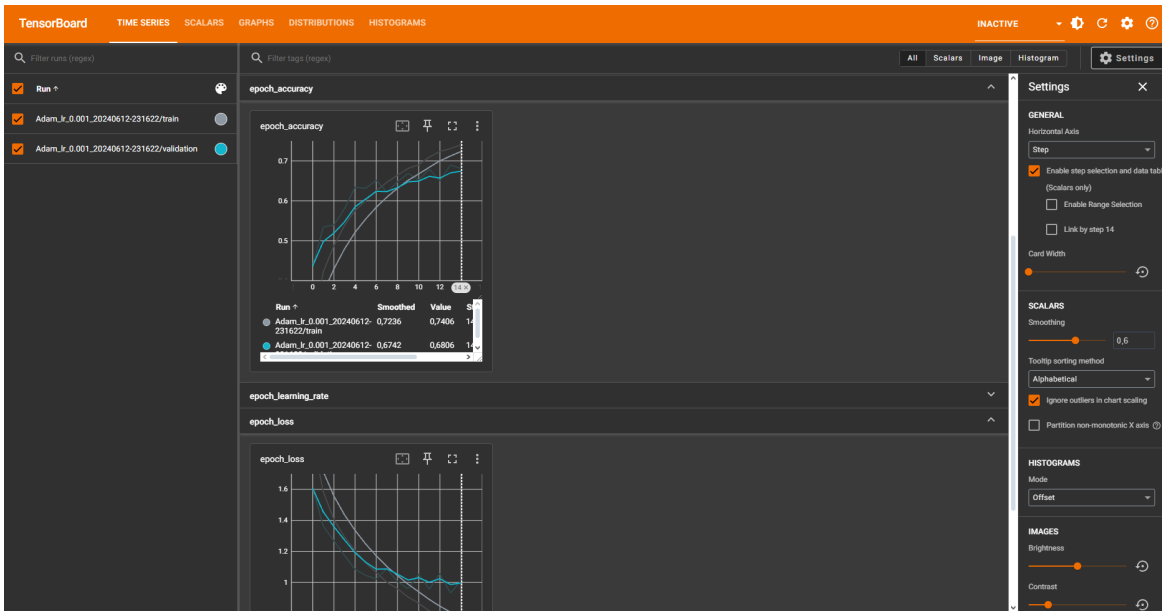
4.1 Entraînement et Évaluation

Nous avons créé un script python, qui pour chaque combinaison d'optimiseur et de taux d'apprentissage, entraîne le modèle sur les données d'entraînement et évalue ses performances sur les données de test. Il est bien sûr possible d'en rajouter à votre guise dans les encadrés prévus à cet effet (voir section "Paramètres" du notebook). Voici nos méthodes de comparaison :
— **Évaluation des Performances** : La précision du modèle sur l'ensemble de test a été mesurée pour chaque combinaison. Nous avons identifié la combinaison optimiseur-taux d'apprentissage qui donnait la meilleure précision. La meilleure combinaison trouvée lors de cette étude est celle qui a donné la précision la plus élevée sur l'ensemble de test. Cette combinaison optimale d'optimiseur et de taux d'apprentissage permet au modèle de généraliser efficacement sur des données non vues.

```
Epoch 4/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 15s 37ms/step - accuracy: 0.4538 - loss: 1.4829 - val_accuracy: 0.5152 - val_loss: 1.2899
Epoch 5/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 36ms/step - accuracy: 0.4821 - loss: 1.3732 - val_accuracy: 0.5171 - val_loss: 1.2818
Epoch 6/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 15s 38ms/step - accuracy: 0.5119 - loss: 1.3067 - val_accuracy: 0.5985 - val_loss: 1.1627
Epoch 7/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 15s 37ms/step - accuracy: 0.5300 - loss: 1.2825 - val_accuracy: 0.6135 - val_loss: 1.0983
Epoch 8/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 36ms/step - accuracy: 0.5486 - loss: 1.2224 - val_accuracy: 0.6210 - val_loss: 1.0917
Epoch 9/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 15s 38ms/step - accuracy: 0.5720 - loss: 1.1687 - val_accuracy: 0.6376 - val_loss: 1.0945
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 34ms/step - accuracy: 0.6032 - loss: 1.1050 - val_accuracy: 0.6408 - val_loss: 1.0478
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 34ms/step - accuracy: 0.6297 - loss: 1.0463 - val_accuracy: 0.6486 - val_loss: 1.0371
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 34ms/step - accuracy: 0.6392 - loss: 0.9971 - val_accuracy: 0.6345 - val_loss: 1.0655
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 35ms/step - accuracy: 0.6287 - loss: 1.0452 - val_accuracy: 0.6580 - val_loss: 1.0777
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 15s 36ms/step - accuracy: 0.6281 - loss: 1.0581 - val_accuracy: 0.6821 - val_loss: 0.9792
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 14s 35ms/step - accuracy: 0.6809 - loss: 0.8853 - val_accuracy: 0.6693 - val_loss: 1.0016
100/100 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.6838 - loss: 0.9612
Optimiseur: Adam, lr: 0.001 - Précision sur le test: 66.93%


Meilleure combinaison: (Optimiseur: Adam - Taux d'Apprentissage: 0.001 - Précision: 66.9277)
```

— **Comparaison des différents modèles à l'aide de courbes** : TensorBoard a été utilisé pour visualiser et surveiller le processus d'entraînement. Cela nous a permis d'observer en temps réel l'évolution de la perte et de la précision du modèle. En utilisant TensorBoard, nous avons pu ajuster différents hyperparamètres et observer leur impact sur les performances du modèle.



5 Conclusion

Lors de ce compte rendu, nous avons présenté notre travail sur la reconnaissance de galaxie à l'aide de réseaux de neurones par convolution. Après avoir reformulé le problème, à savoir classer ces images dans des catégories, nous avons fait le point sur les solutions déjà existantes. En effet, il existe plusieurs études sur la reconnaissance des galaxies, qui permettent d'obtenir de bien meilleurs résultats qu'avec une technique manuelle. Cependant, même si notre projet s'arrête à l'étude du type de galaxie, il permet de proposer plusieurs solution de CNN et de les comparer grâce à l'outil tensorboard, ce qui permet de mieux comprendre le comportement de ces réseaux de neurones. De plus, l'étude de galaxies demande une grande quantité d'image avec une résolution élevé, ce qui fait qu'avec les outils à notre disposition il a fallu réduire la qualité de ces image, et donc perdre un partie des informations. Cependant, nous avons pu obtenir près de 70% de précision avec notre modèle, dont l'implémentation est disponible dans le notebook Jupiter attaché.

6 Bibliographie

- Galaxy10 DECals Dataset Documentation

- Galaxy10 DECals Dataset repository

- Algorithme de classification : Définition et principaux modèles

- DataSet de l'étude : https://huggingface.co/datasets/matthieulel/galaxy10$_decals$

- Chapitre 1 - IRN

- Chapitre 2 - IRN