IA 201

Pedro Coutinho – pedro.coutinho@ims-bordeaux.fr

Projet Deep Learning

Pour ce projet , il vous est demandé d'utiliser des réseaux de neurones pour faire de la classification sur différentes bases de données. Pour cela, vous devrez utiliser le langage Python et la bibliothèque PyTorch (normalement déjà installés sur les machines de l'école).

Les objectifs de ce projet sont (liste non exhaustive) :

- Se familiariser avec la bibliothèque PyTorch, largement utilisée dans la communauté *deep learning.*
- Apprendre à gérer différents aspects d'un code (chargement de données, définition d'une architecture de réseau, définition de la fonction coût, boucle d'apprentissage).
- Comprendre l'impact de différents choix (taille et profondeur du réseau, taux d'apprentissage, type d'optimiseur, type de fonction d'activation, type de normalisation).
- Comparer les performances d'un CNN appris *from scratch* par rapport à des utilisations de techniques de *transfer learnin*g.
- Analyser l'impact des techniques de *data augmentation.*

Activités demandées

Les activités proposées dans ce projet sont :

1. Construction d'un classifieur de type CNN appris *from scratch* sur deux différentes bases de données disponibles sur Kaggle :
   a. *New plants disease dataset*
   b. *Melanoma cancer dataset*
2. Pour les deux datasets, construire un classifieur en utilisant des techniques de *transfer learning* :
   a. Utiliser un réseau pré-entraîné sur ImageNet figé qui fera l'extraction de features des images. Ces features seront ensuite utilisées comme l'entrée d'un réseau *fully connected* qui devra être appris.
   b. Utiliser un réseau pré-entraîné sur ImageNet, mais avec du *finetuning*. Cette fois, on cherche à modifier légèrement les poids de la partie convolutive du réseau, pour mieux adapter l'extraction des features à nos bases de données.

À la fin des séances, il vous sera demandé de rendre les codes utilisés, et aussi un rapport contenant les éléments suivants :

- Explication du choix de la fonction coût et de l'optimiseur.
- Schéma des architectures utilisées.
- Pour le premier cas (réseau *from scratch*) :

- Analyse de l'impact des différents choix dans la définition de l'architecture. Quels sont les paramètres qui ont eu plus d'influence ? Quelles sont les possibles explications pour ces constatations ?

Suggestions de paramètres à étudier :

- Taille du réseau (nombre de filtres par couche)
- Profondeur du réseau (nombre de couches)
- Impact du batch normalization
- Impact du taux d'apprentissage (*learning rate*)
- Si possible, quel est l'impact d'utiliser des blocs residuels ?

- Pour le deuxième cas (*transfer learning*) :
  - Quelle technique de *transfer learning* a mieux fonctionné ? L'utilisation d'un réseau convolutif figé pour l'extraction de features ou quand on fait du *finetuning* ? Quels sont les avantages de chaque méthode ? Justifiez les réponses aux questions avec vos résultats.
  - Comment les choix de l'architecture *fully connected* impacte les performances ?
- Comparer les performances du réseau appris *from scracth* avec celui qui utilise le *transfer learning*
  - Le *transfer learning* présente des performances similaires pour les deux datasets ? Si non, dans quel cas il fonctionne mieux ? Pourquoi ?
- Un cas de surapprentissage a été constaté ? Si oui, comment pourrait-on le prévenir ?

Remarques:

- **Toutes les analyses devront être faites pour les deux datasets proposés.**
- **Les performances du réseau doivent être mesurées sur un dataset de test, qui ne doit pas contenir d'échantillons déjà utilisés dans la phase d'apprentissage.**

Evaluation de la performance des réseaux

Pour évaluer la performance des réseaux, il faudra définir des métriques. Etant dans un contexte de classification, vous pouvez vous baser sur l'*accuracy*, et aussi sur l'observation des matrices de confusion. Les mesures de *precision* et *recall* peuvent aussi être regardées, étant donné leur intérêt dans le cas des bases de données non équilibrées.

Vu que la performance des réseaux dépend aussi de la réalisation, il vous est demandé de faire plusieurs réalisations de chaque configuration (au moins 5 pour chaque réseau, par exemple), avant de montrer vos résultats. Assurez-vous aussi de définir un nombre d'époques suffisant pour que le réseau converge, pour cela il sera nécessaire de regarder les courbes d'apprentissage pour un dataset d'entraînement et un autre de validation.

Workflow suggéré

Afin de garantir un bon déroulement du projet, il est demandé de réaliser le projet en validant une progression par parties.

1. D'abord, il faut s'assurer d'avoir bien défini un *data loader* qui alimentera le réseau pendant l'étape d'apprentissage.
2. Ensuite, il faudra définir le modèle qui sera entraîné.
    a. Pour le cas du modèle appris *from scratch*, toutes les couches doivent être définies. Plusieurs configurations peuvent être considérées, de la simple concaténation de convolutions, activations, pooling et normalisation, jusqu'à l'utilisation de blocs plus complexes, comme les blocs résiduels.
    b. Pour le *transfer learning*, il faudra charger d'abord le modèle pré-entraîné, et aller modifier les couches *fully connected* après l'extraction de features à partir des couches convolutives.

Remarque : dans cette étape, il faut faire attention aux dimensions de chaque couche, pour que ce soit compatible avec la taille des images d'entrée.

3. Une fois l'architecture validée, il faudra définir une fonction coût pour apprendre le réseau, ainsi qu'un optimiseur qui sera responsable de la mise à jour des poids du réseaux.
4. Finalement, il sera nécessaire de mettre en place une boucle d'entraînement qui, à chaque itération, recevra des données venant du *data loader* pour calculer l'erreur faite par le réseau, et optimisera ses poids à partir de la rétropropagation du gradient.

Un exemple de projet de classification sur la base de données CIFAR10 est disponible dans les références.

Références

- Tutorials PyTorch :
    - https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
    - https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- Datasets Kaggle:
    - https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset
    - https://www.kaggle.com/datasets/bhaveshmittal/melanoma-cancer-dataset