Projet Deep

Master 2 informatique, Université Angers

Au cours de ce projet, l'objectif est de ré-utiliser ce que l'on a fait ensemble lors des TPs sur l'apprentissage profond pour la computre vision, afin de l'appliquer à un nouveau jeu de données et tester les performances de différents modèles de classification.

1 Chargement des données

Nous utiliserons le corpus Caltech101*. Ce jeu de données contient environ 10000 images 300 x 200 pixels d'objets rangés dans 101 catégories (retirer le répertoire BACKGROUND_Google des données extraites de l'archive).

Pour le charger, il s'agit d'utiliser ImageFolder (documentation disponible ici: `https://pytorch.org/docs/stable/torchvision/datasets.html#imagefolder`). On utilisera 90% des données pour l'apprentissage et 10% des données pour le test. La procédure pour créer les dataloader est la suivante:

1. Charger le dataset entier avec ImageFolder utilisant les trasformations prévues pour l'apprentissage (comportant éventuellement des modifications aléatoires pour l'augmentation des données d'apprentissage comme vu en TP);
2. Choisir aléatoirement 90% des indices d'images du jeu chargé pour constituer le jeu d'apprentissage, réserver les indices restants pour le test;
3. Utiliser torch.utils.data.Subset (doc: `https://pytorch.org/docs/stable/data.html#torch.utils.data.Subset`) pour former le jeu d'apprentissage avec les indices d'apprentissage
4. Recharger le jeu entier avec les transformateurs prévus pour le test (avec seulement Resize, transformation en tenseur et normalisation comme effectué au cours du TP pour l'utilisation des modèles pré-entraînés sur ImageNet), puis ré-utiliser torch.utils.data.Subset pour former le jeu de test en fonction des indices d'images sélectionnés pour le jeu de test.

*`https://data.caltech.edu/records/mzrjq-6wc02`

Pour travailler sur colab, attention à avoir bien spécifié que l'on souhaite utiliser le runtime GPU. Il faut aussi lancer les commandes suivantes avant le chargement des données:

```
!pip install Pillow==4.0.0
!pip install PIL
!pip install image
from PIL import Image
```

2 Comparaison des performances de différents modèles

L'idée est de comparer les différents modèles pré-entraînés de pytorch:

- resnet18
- alexnet
- squeezenet1_0
- vgg16
- densenet161
- inception_v3

Il s'agit de charger la version pré-entraînée de chacun de ces modèles et de les adapter pour traiter les images du corpus Caltech101. Penser à:

1. Mettre les images au même format que celles utilisées lors du pré-apprentissage (224*224 pour tous les modèles sauf inception qui attend des images 299*299);
2. Normaliser les images de la même façon qu'elles l'étaient lors de l'apprentissage (i.e., utiliser transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]));
3. Adapter la sortie du modèle pour que le nombre de sorties corresponde au nombre de classes attendues (101);
4. Définir un optimiseur ne modifiant que les paramètres non-pré-appris.

3 Evaluation des modèles

L'évaluation des modèles se fera selon le taux de bonne classification en test. Nous souhaitons avoir un score moyen ne dépendant pas du découpage en train et test. Pour cela, nous adopterons une approche Cross-Validation:

1. Refaire pendant k iterations (par exemple k=10):
   (a) Construire les jeux de données de train et test selon un découpage aléatoire (comme défini dans la section chargement des données)

   (b) Apprendre les modèles sur le jeu de train
   (c) Enregistrer le taux de bonne classification en test pour chaque modèle
2. Retourner les taux moyens de bonne classification en test pour chaque modèle

On pourra également éventuellement enregistrer des courbes moyennes de loss en train et en test pour chaque modèle sur l'ensemble des folds.