ENSTA 2e année - Cours CSC_4MI04
TP3 : Classification d'images par CNN

Dans ce TP nous allons expérimenter les réseaux de neurones convolutifs (CNN) pour la classification d'images. Pour cela, nous utiliserons l'interface de programmation (API) Keras [1] qui permet de créer très simplement des architectures neuronales, de les entraîner et de les tester.

Pour pouvoir utiliser nos propres machines et leur puissance limitée (notamment en l'absence de GPU), nous allons travailler sur un nombre adapté de petites images issues de la base CIFAR10 (Figure 1), et avec des réseaux de petite taille qui ne sont que partiellement représentatifs des capacités de l'apprentissage profond. L'objectif est principalement de comprendre la structure des réseaux de convolution, la dynamique de l'apprentissage, et l'influence des différents paramètres sur les performance d'un modèle.

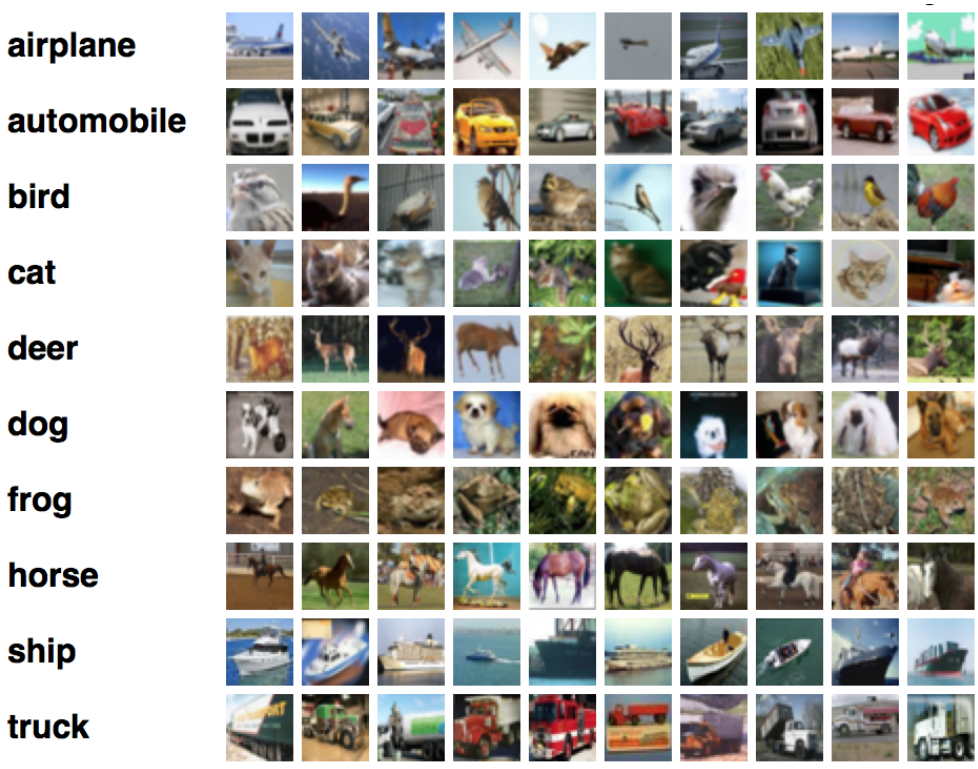

FIGURE 1 – Quelques images de la base de données CIFAR 10

1. https://keras.io/guides/

1 Préliminaire informatiques

Keras est une API de haut niveau (front-end) basée sur Python, qui permet à un non-développeur de concevoir, entraîner, et utiliser des réseaux profonds, dans une perspective de compromis entre facilité et flexibilité. Keras s'appuie sur des bibliothèques Python de plus bas niveau (back-ends), qui implémentent de façon efficace les opérations intensives d'inférence et de rétro-propagation sur les réseaux, en particulier grâce à une extension des tableaux multidimensionnels de NumPy à des structures de *tenseurs*, qui incluent des mécanismes d'*auto-différentiation*. Keras peut s'appuyer sur les back-ends TensorFlow [2] et/ou Theano [3].

- **Étape 1 - option 1 :** Sous Linux, avec votre gestionnaire de packages :
  - sudo apt-get install python3-keras
  - sudo apt-get install jupyter
- **Étape 1 - option 2 :** En commande Python, avec pip :
  - pip install tensorflow
  - pip install keras
  - pip install jupyter
- **Étape 2 :** Téléchargez le notebook Jupyter à partir duquel vous allez travailler sur `https://perso.ensta-paris.fr/~manzaner/Cours/MI204/TP3_CNN.ipynb`.

Attention, sauf exceptions explicitement indiquées (comme la section II.5), les cellules du notebook doivent être exécutées de façons séquentielle et ordonnée.

2 Structuration des données

Dans la partie II.1 du notebook, après avoir téléchargé la base CIFAR10 (suite à un premier téléchargement, la base est conservée dans votre répertoire `~/.keras/datasets/`), on structure les données en vue de l'apprentissage. Vous pourrez ajuster la taille de vos bases en fonction de la puissance de votre machine... et de votre patience !

1. Pourquoi divise-t-on la base en trois sous-ensembles {*training, validation, test*} ?
2. Que réalise la fonction *standardize* ? Quel est son intérêt ?

3 Architecture du réseau

Une architecture basique de réseau est définie dans la partie II.2 du notebook. Reconnaissez les éléments essentiels de cette structure en identifiant :

- La fonction de chacune des couches du réseau
- Le rôle des différents paramètres de chaque couche
- La taille de chacune des couches (hauteur, largeur, épaisseur [4])
- La manière dont la sortie (la classe) est codée

1. Dans le résumé de l'architecture du réseau qui s'affiche dans la partie II.3, que sont les "trainable params" ? Comment leur nombre est-il calculé ?

2. `https://www.tensorflow.org/`
3. `https://pypi.org/project/Theano/`
4. Nombre de caractéristiques

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 8)	224
dropout (Dropout)	(None, 32, 32, 8)	0
max_pooling2d (MaxPooling2D)	(None, 16, 16, 8)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 64)	131,136
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 10)	650
dropout_2 (Dropout)	(None, 10)	0

Total params: 132,010 (515.66 KB)
Trainable params: 132,010 (515.66 KB)
Non-trainable params: 0 (0.00 B)

4 Apprentissage

Dans la partie II.4 du notebook est réalisé l'entraînement du modèle défini dans les parties II.2 et II.3.

1. A quoi correspondent une époque ("epoch"), une étape ("step") et un lot ("batch") ?
2. Essayez d'ajuster la taille de lot ("batchsize"), et analysez son effet sur le temps de calcul d'une étape, d'une époque, ainsi que sur les courbes d'apprentissage.
3. Comparez les différentes fonctions d'optimisation : "SGD", "SGD avec Momentum", et "Adam".

5 Hyperparamètres

1. Identifiez dans l'ensemble du code les hyperparamètres du modèle, précisez pour chacun sa nature, et l'influence qu'il peut avoir dans le processus d'apprentissage.

6 Approfondissement du modèle

1. Essayez d'améliorer les performances de votre modèle en approfondissant le réseau à plusieurs couches de convolution. Vous devrez respecter les contraintes de taille (i.e. nombre de neurones) entre les tenseurs d'entrée et ceux de sortie.
2. Lorsque vous êtes satisfait de votre modèle, représentez graphiquement son réseau multi-couches final, récapitulez ses hyperparamètres, affichez les courbes et les matrices de confusion correspondantes.
3. Critiquez votre modèle en soulignant ses avantages et limitations.

7 Sur-apprentissage

1. Comment pouvez-vous observer le phénomène de sur-apprentissage (overfitting) sur les courbes montrant les fonctions de coût ? Montrez et commentez un exemple de courbes mettant en évidence un sur-apprentissage.
2. Quels mécanismes peuvent influer favorablement sur ce phénomène ? Montrez quelques exemples montrant les effets de leur utilisation.

8 Cartes d'activation

La partie IV du notebook propose une technique de visualisation des "cartes d'activation".

1. Comment pouvez-vous interpréter ces cartes d'activation pour les features de la première couche ?
2. Une fois que vous aurez obtenu un meilleur modèle avec un réseau plus profond, essayez de visualiser des cartes d'activation correspondant à des couches plus profondes, et d'interpréter ces cartes.