TP à rendre

Ce TP est un TP noté qui comptera dans votre évaluation finale. Vous devrez rendre un rapport PDF répondant aux questions qui seront posées, décrivant ce que l'on vous demande de décrire et illustrant ce que l'on vous demande d'illustrer. Pensez à justifier et argumenter vos réponses. Vous compléterez cela avec tous les codes que vous aurez écrits et rendrez le tout dans une archive zip. Ainsi, à chaque question où vous écrirez un code, vous aurez dans le document PDF les explications et une version d'un fichier source associé (dont vous indiquerez le nom dans le rapport).

Analyse de l'évolution de la *loss* pendant l'apprentissage

Nous allons nous intéresser aux variations de la *loss* (en fait des losses) permettant d'évaluer l'évolution de l'apprentissage d'un réseau de neurones. Pour rappel, c'est la *loss* qui guide l'apprentissage du réseau conjointement à la back propagation.

Lors d'un apprentissage, on divise le corpus en 2 ou 3 parties, nommée respectivement *train*, *validation* et *test*. La partie *train*, est la partie la plus importante et la plus grosse, généralement 80 à 90% du corpus. C'est la partie des données qui va servir à apprendre les paramètres du modèle. La partie *validation* est une partie de taille plus faible qui contient des données complètement disjointes de la partie apprentissage. Par exemple, il ne doit pas y avoir des images trop proches dans le *train* et dans la *validation*, ou encore pour la parole, une même personne ne peut pas être présente dans les parties *train* et *validation* sinon on apprendrait sa voix. L'intérêt de la partie *validation* est de montrer la performance du modèle sur des données non connues, et donc la capacité de généralisation de ce modèle. Enfin, quand il y a 3 parties au corpus, la partie *test* est une partie dont on a les données pour évaluer mais pas les labels. Cela permet d'éviter la triche lors de l'organisation de compétitions scientifiques où seulement l'organisateur de la compétition connait ces labels.

Lors d'un apprentissage, on utilise *train* et *validation* (ce dernier étant appelé *test* dans certains corpus comme MNIST où il n'y a que 2 parties) pour vérifier l'évolution conjointe de la *loss* sur l'apprentissage d'un côté et sur la validation de l'autre. L'évolution des *losses* devrait être la suivante pour un réseau de neurones lors de l'apprentissage :

- Si le réseau apprend correctement, la *loss* sur le *train* évolue à la baisse et tend vers 0 (même si elle oscille). Plus la valeur de la *loss* est faible, plus le réseau a appris des données d'entraînement.

- Si le réseau généralise correctement, la *loss* de validation (ou de *test* dans le cas de MNIST) suit la même tendance. Si cette *loss* (re)commence à augmenter, alors on dit que le réseau surapprend (« *overfit* ») : il a appris « par cœur » les données d'apprentissage et n'est pas capable de généraliser convenablement.

Dans cette partie, nous allons nous intéresser à mesurer les effets des modifications de l'apprentissage sur MNIST sur les *losses* de *train* et de *test*.

? Reprenez le code original qui se trouve sur Moodle (archive « mnist.zip »). En vous inspirant du calcul fait dans la fonction `test`, modifier la fonction `train` pour qu'elle calcule la *loss* totale sur les données d'apprentissage. Modifiez également les fonctions `test` et `train` afin de pouvoir récupérer pour chaque époque la *loss* de train et de test. Lancez l'apprentissage sur les 14 époques et tracez les 2 *losses* sous forme de courbe, chacune sur un graphique. Commentez les graphiques obtenus.

? Inverser les corpus *train* et *test* de MNIST dans le script pour apprendre sur le corpus de test (plus petit) et tester sur le corpus de train (beaucoup plus grand). Tracez les 2 courbes de *losses* comme précédemment et commentez.

? On garde l'inversion de corpus mais en plus, on supprime les fonctions de *dropout* dans la classe `Net` (classe contenant la définition du réseau de neurones). Lancez l'apprentissage, **renseigner-vous sur le dropout** pendant que l'apprentissage tourne, tracez les *losses* et commentez.

? Remettez les corpus de *test* et d'apprentissage comme à l'origine. Laissez les *dropout* inactifs. Lancez l'apprentissage, tracez les *losses* et commentez.

? Pour synthétiser les résultats de ces 4 analyses, réalisez 2 graphes. Le premier contiendra les 4 *losses* d'apprentissage, le second les 4 *losses* de test. Avez-vous obtenu du surapprentissage ou pas ? Pourquoi ? Si oui, dans quel(s) cas ?

Perception acoustique puis multimodale

Nous allons étudier un modèle neuronal pour Audio MNIST, une base de données de personnes qui énoncent en anglais des chiffres de 0 à 9 (comme MNIST). Nous nous basons sur le code fourni sur la plateforme Kaggle : https://www.kaggle.com/code/jokekling/pytorch-study-audio. Nous n'entraînerons pas ce modèle, nous allons juste l'analyser.

? Créez un environnement conda contenant les package suivant depuis le chanel pytorch : pip, pytorch, torchvision, torchaudio, pandas, numpy, matplotlib. Activez cet environnement. Si vous êtes sous Windows exécutez « pip install pySoundFile » sinon exécutez « pip install sox ».

? Téléchargez « AudioMNIST.zip ». Regardez et écoutez quelques exemples du répertoire « recordings ». Vous y trouverez des fichiers dont le nom est composé du chiffre à énoncer de 0 à 9, suivi du nom du locuteur, suivi de numéro de la répétition (de 0 à 50). Nous limitons Audio MNIST à 3000 fichiers pour ce TP.

? Utilisez le script « ShowAudioFile.py » pour visualiser quelques fichiers. L'axe des y représente les fréquences sur le spectrogram. D'après ce que nous avons vu en cours, quelles sont les fréquences représentées sur ce graphe ? Le script utilise (comme l'auteur) le MelSpectrogram. A quoi cela correspond-t-il ?

? Les scripts proposés par l'auteur sur Kaggle proposent de séparer le corpus en *train*/*test* automatiquement (bloc 1 du jupyter notebook en ligne). Je vous propose une version du script adapté à notre organisation des données : « SplitTrainTest_original.py ». Exécutez ce script, regardez le contenu des fichiers csv générés. Par rapport à ce que nous avons vu précédemment dans ce TP, que pouvez-vous dire de cette séparation en *train*/*test* ?

? Nous allons étudier certains blocs du jupyter Notebook. Les blocs 2 à 9 servent au chargement des données. Le bloc 10 sert à normaliser les données en utilisant un moyennage. Expliquez ce que font les blocs de 11 à 15. Quel modèle décrivent-ils ?

? Le bloc 16 présente les *losses* d'apprentissage et de *test* associées avec le score de reconnaissance des chiffres (de 0 à 100%). Commentez ce graphique.

? Nous souhaiterions utiliser les corpus MNIST et Audio MNIST pour réaliser un système multimodal de reconnaissance de caractère où l'on écrirait et énoncerait le chiffre en même temps, le but étant de palier les problèmes de reconnaissance des chiffres écrits et énoncés séparément pour se rapprocher d'une performance à 100%. Comment pourriez-vous procéder en utilisant le projet de Kaggle ? Décrivez toutes les étapes.

Analyse de la sensibilité des réseaux convolutifs

Nous allons maintenant analyser la sensibilité des réseaux convolutifs à leur entrée sur une tâche de classification d'image de type *ImageNet*. L'idée étant de visualiser, voire de comprendre à quoi, c'est-à-dire à quels pixels est sensible le réseau lors de prise de décision de classification.

? Regardez la page https://github.com/utkuozbulak/pytorch-cnn-visualizations/ et expliquez ce qu'il est possible de visualiser avec différents algorithmes de Class Activation Mapping (CAM).

Nous allons étudier les activations d'une classe sur des images. Pour cela, nous allons utiliser les modèles proposés en ligne sur https://huggingface.co/spaces/frgfm/torch-cam.

? Commencez par choisir un setup : un modèle *backbone*, une méthode de CAM et une classe que vous souhaitez étudier. **Ne prenez pas tous le même setup !** Ce setup est celui que vous garderez pour tous vos tests (indiquez ce setup dans votre rapport).

? Puisque vous avez choisi une classe, cherchez 3 images sur Internet comportant un seul « objet » de la classe choisie, plusieurs objets de la classe choisie et aucun objet de la classe choisie. Utiliser le site ci-dessus pour analyser les pixels « importants » de ces 3 images pour cette classe. Intégrez les images dans votre rapport et commentez comparativement les résultats obtenus.

? Lisez la section 4.2.3 et regardez la figure 5 de l'article https://hal.inria.fr/hal-02937871/document. Choisissez-en 2 transformations d'image uniquement parmi le changement de luminosité, le changement de contraste, le cisaillement, la distorsion élastique, la mise à l'échelle, la transformation de la perspective. En utilisant par exemple le module *imgaug* (https://github.com/aleju/imgaug) et l'image contenant 1 seul objet de votre classe, créez 10 images suffisamment différentes pour chaque transformation et regardez le résultat de votre CAM. Intégrez et commentez les résultats obtenus dans votre rapport.

? D'après vos expérimentations, que pouvez-vous dire de la sensibilité des réseaux de neurones convolutionnels ?