Projet “Uncertainty exploration“

L’apprentissage profond est un sous-domaine de l’apprentissage automatique qui utilise des réseaux de neurones artificiels pour apprendre à partir de données. Bien que l’apprentissage profond ait connu un grand succès dans diverses applications, il est souvent confronté à des problèmes d’incertitude. L’incertitude peut être causée par des données bruyantes, des erreurs de mesure, des données manquantes, des erreurs de modélisation, etc. L’incertitude peut affecter la qualité des prédictions et la fiabilité des décisions prises par les modèles d’apprentissage profond. Il existe plusieurs façons de traiter l’incertitude en apprentissage profond, notamment l’utilisation de modèles probabilistes, l’ajout de régularisation, l’utilisation de techniques de détection d’anomalies, etc.

Nous vous conseillons la lecture de l’article de blog “A Comprehensive Introduction to Uncertainty in Machine Learning“ [1].

Le but du projet est d’explorer la voie des ensembles pour avoir une première notion d’incertitude dans les résultats des réseaux de neurones.

Pour le rendu, merci de vous référer aux consignes indiquées sur le Moodle (la dernière version publiée étant seule à prendre en compte).

Les tâches à réaliser pour ce projet sont :

- Revue de l’article de B. L Lakshminarayanan et al. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensemble” [2]
    - Analyse de la problématique, de l’approche et des résultats des auteurs
- Réaliser différentes expériences visant à évaluer la sensibilité des résultats d’un réseau de neurones a des donnes corrompues ou hors domaine.
- Expérience 1 :
    - Entrainez un classificateur (resnet18 de torchvision) sur MNIST (https://pytorch.org/vision/stable/generated/torchvision.datasets.MNIST.html#torchvision.datasets.MNIST)
        - Version 1 : utilisez un réseau sans pré-entrainements (poids initiaux aléatoires).
        - Version 2 : utilisez un réseau avec pré-entrainements (poids initiaux obtenus à partir d’un entrainement sur ImageNet)
            - Quel est l’intérêt d’utiliser un réseau pré-entrainé ? Comment cela se manifeste-t-il sur vos données d’entrainement ? Notamment sur l’évolution de fonction de cout au cours des epochs/itérations.

Écrire un script qui permet d’entrainer un réseau resnet18 sur MNIST et qui enregistre le modèle en sortie. Attention si vous utilisez un random seed pour la phase de développement, ce dernier devra pouvoir être retirer pendant l’exécution du script afin de ne pas reproduire systématiquement le même entrainement.

- Expérience 2 :
    - Utilisez votre script d’entrainement d’un resnet18 sur MNIST afin de pouvoir réutiliser le modèle.
    - Avec ce modèle, observez pour un ensemble de 20 images de MNIST (test), prises aléatoirement, les résultats de classification de votre meilleur modèle entrainent.

    - Appliquez maintenant du bruit blanc sur ces images à différents niveaux d'intensité (image <- image + bruit blanc * niveau, pour différentes valeurs de niveau).
        - Que constatez-vous ? Une dégradation visuelle notable (pour l'œil humain) est-elle nécessaire pour observer une dégradation significative des résultats de classification ?
- Expérience 3 :
    - Utilisez votre script d'entraînement d'un resnet18 sur MNIST afin de pouvoir réutiliser le modèle. Réitérez l'opération pour avoir un total de 7 modèles différents entrainés sur MNIST.
    - Avec ces modèles, observez pour un ensemble de 20 images de MNIST (test), prises aléatoirement, les résultats de classification de votre meilleur modèle entrainent.
    - Appliquez maintenant du bruit blanc sur ces images à différents niveaux d'intensité (image <- image + bruit blanc * niveau, pour différentes valeurs de niveau) et refaites la classification avec vos 7 modèles.
        - Que constatez-vous ? Une dégradation visuelle notable (pour l'œil humain) est-elle nécessaire pour observer une dégradation significative des résultats de classification ?
        - Quel est l'intérêt d'utiliser 7 modèles ici ?
- Expérience 4 :
    - Ré-utilisez vos 7 modèles différents entraines sur MNIST.
    - Avec ces modèles, observez pour un ensemble de 20 images de MNIST (test) ainsi que 20 images de KMNIST (https://pytorch.org/vision/stable/generated/torchvision.datasets.KMNIST.html#torchvision.datasets.KMNIST), prises aléatoirement, les résultats de classification de votre meilleur modèle entrainent.
        - Quelles seraient vos conclusions si vous n'utilisiez qu'un seul modèle dans cette expérience ? Quel est l'intérêt d'utiliser plusieurs modèles ?
- Expérience 5 :
    - Ré-utilisez vos 7 modèles différents entrainés sur MNIST.
    - Avec ces modèles, observez pour un ensemble de 20 images de MNIST (test) et observés les résultats de classification en fonction de l'angle de rotation appliqué à l'image initiale (entre 0 et 180 degrés, voire expérience illustrée en Fig.1)
        - Qu'observez-vous ?
        - Quelles seraient vos conclusions si vous n'utilisiez qu'un seul modèle dans cette expérience ? Quel est l'intérêt d'utiliser plusieurs modèles ?

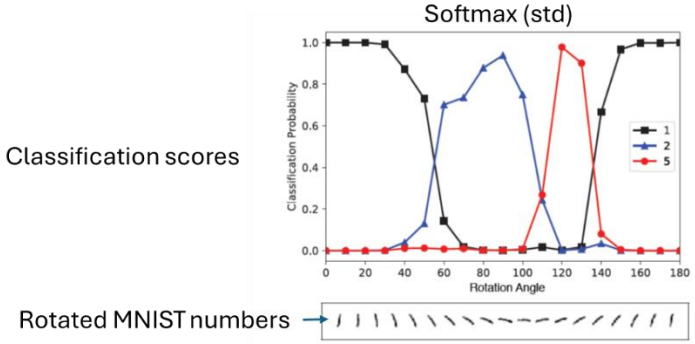

Figure 1. Schema du résultat typique attendu pour l'expérience #5 de rotation MNIST (ici cas avec un seul modèle). D'après [3].

Références

[1] https://imerit.net/resources/blog/a-comprehensive-introduction-to-uncertainty-in-machine-learning-all-una/
[2] https://doi.org/10.48550/arXiv.1612.01474
[3] https://doi.org/10.48550/arXiv.1806.01768