Bordeaux INP
ENSEIRB
MATMECA

Rapport : Outils pour l'apprentissage

Mehdi HALA

1 Introduction :

Le but de ce projet est de mettre en oeuvre l'une des techniques vues en cours à savoir l'élagage (pruning).

2 Objectif initial :

L'objectif initial de ce projet était de construire une IA d'un jeu de réflexion comme le morpion ou puissance 4 par exemple en se basant sur un réseau de neurones déjà fait et en appliquant un élagage adéquat qui permettra d'avoir de bons résultats en de meilleurs temps.

3 Démarche de travail :

La première tâche à faire était de trouver un réseau de neurones sur lequel je pourrais travailler. Un CNN [1] m'a été proposé par Mme Shilova. Le dataset d'entraînement de celui-ci étant gigantesque, j'ai décidé de commencer par crée mes méthodes d'élagage sur un dataset moins grand et moins complexe puis essayer d'intégrer ceux-ci dans le CNN voulu.
Ainsi, j'ai décidé de travailler sur le dataset MNIST que j'ai utilisé dans d'autre modules et qui est aussi facilement accessible depuis la librairie **tf.keras.datasets**.

3.1 L'élagage :

Le prototype des méthodes d'élagage est le suivant :

```
pruning_method(model, pruning_fraction):
     return pruned_model
```

Pour pouvoir retourner un model, une fonction **create_model** est nécessaire. Le prototype de celle-ci sera le suivant :

```
def create_model(layer_sizes, n_classes, layer_fn, layer_kwargs_fn, weights)
```

avec :

- layer_sizes (list) : Nombre de neurones consécutifs dans une couche en excluant les couches de sortie.
- n_classes (int) : Nombre de classes de classification.
- layer_fn (fun) : Fonction créant des couches cachées.
- layer_kwargs_fn (fun) : Fonction transformants les poids en kwargs.
- weights (list or None) : Liste des poids pour initialiser le modèle. Random si None.

Et une sortie :

- Modèle élagué.

3.1.1 Élagage par poids (weight pruning) :

Sur la figure 1, un seuil a été déterminé et a permis d'enlever les poids non nécessaires. Pour appliquer cette méthodes dans notre cas. J'ai opter pour une approche un peu différente puisqu'on utilise des pourcentage d'élagage mais pas un seuil. Alors les étapes sont les suivantes :

1. Calculer le nombre k d'éléments à enlever. ($k = len(matrice) * fraction$)
2. Éliminer les k valeurs les moins pertinentes.
3. Créer un nouveau modèle avec les nouveaux poids.

3.1.2 Élagage par neurones (neuron/unit pruning) [3] :

Cet élagage n'est pas si différent du précédant. Au lieu que les éliminations se fassent par éléments, elles se fassent par neurones. Ainsi, les étapes sont les suivantes :

1. Calculer le nombre k de neurones à enlever. ($k = len(neuron_list) * fraction$)
2. Éliminer les k neurones les moins pertinents.
3. Créer un nouveau modèle avec les nouveaux poids.

3.2 Performances :

3.2.1 Évaluation d'un élagage :

Pour évaluer un modèle, nous avons créé une fonction prenant en arguments :

— Méthode d'élagage
— Modèle initial
— Dataset
— Fraction d'élagage

Cette fonction alors appliquera l'élagage au modèle donné en argument. Sur ce nouveau modèle, nous allons appliquer la méthode *tf.keras.evaluate* pour avoir la précision. Puis nous mesurons le temps d'inférence en exécutant à plusieurs reprises la prédiction sur l'ensemble de test.

3.2.2 Résultats :

La figure 2 présente la précision et le temps d'inférence des trois modèles en fonction de la fraction d'élagage.
On remarque que la précision est de 100% pour les modèles élagués jusqu'à 60% d'élagage ou même 95% pour l'élagage par poids.
D'autre part, on remarque que l'élagage par neurones et largement mieux au niveau du temps d'inférence. On remarque aussi que la version non élaguée a aussi un bon temps d'inférence, je n'arrive toujours pas à trouver une cause concrète qui pourrait causer ça. Après lecture de quelques articles [5], il se peut que cela est causé par le nombre de filtre contenu dans le raison de neurones.

3.3 Début des problèmes :

Ayant plus ou moins met en oeuvre les deux méthodes d'élagages, je voulais commencer à l'utiliser dans le code initial mais je n'y arrivais pas essentiellement parce que je n'avais pas assez de temps pour pouvoir comprendre de grands codes comme celui proposé avant. J'ai cherché d'autres dépôts avec des modèles déjà entraînés mais c'est toujours le même problème.

4 Conclusion :

Hormis la petite déviation de l'objectif initial, le projet a bien porté ses fruits en terme de compréhension plus profonde des méthodes d'élagage et de savoir l'état d'art de l'élagage des raison neurones [6].

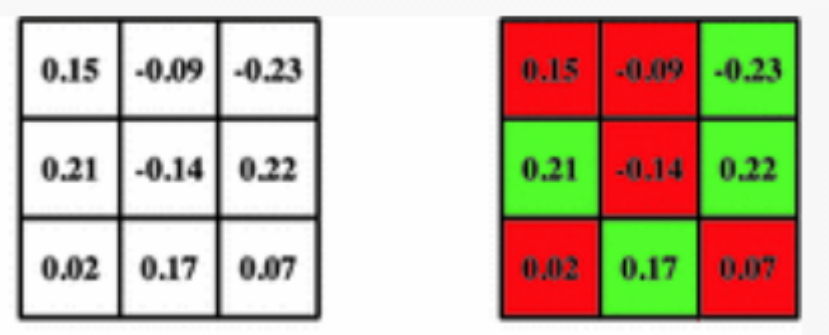

FIGURE 1 – Idée de l'élagage par poids [2]

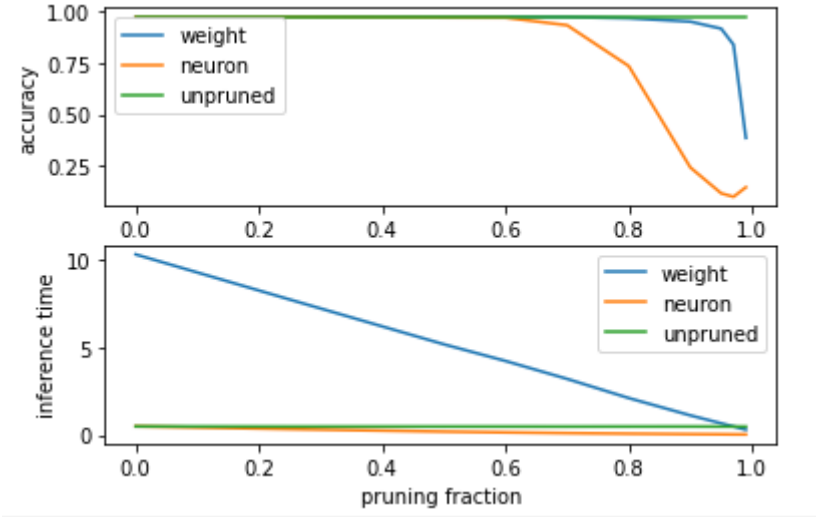

FIGURE 2 – Performance des deux nouveaux modèles élagués comparés au modèle non-élagué.

Références

[1] AlphaGo CNN :

`https://github.com/TheDuck314/go-NN`

[2] Fast CNN Pruning via Redundancy-Aware Training :

`https://link.springer.com/chapter/10.1007/978-3-030-01418-6_1`

[3] An Overview of Pruning Neural Networks using PyTorch :

`https://medium.com/@yasersakkaf123/pruning-neural-networks-using-pytorch-3bf03d16a76e`

[4] Choix des élagage inspiré par :

https://github.com/noelmat/Pruning-MNIST/blob/master/Pruning%20Notebook.ipynb

[5] PRUNING FILTERS FOR EFFICIENT CONVNETS

https://openreview.net/pdf?id=rJqFGTslg

[6] WHAT IS THE STATE OF NEURAL NETWORK PRUNING ?

https://arxiv.org/pdf/2003.03033.pdf