Rapport projet Kaggle

Kssim Aymane, Song Mickael

ENSEEIHT & INSA Toulouse - 4ModIA
2022-2023

1 Partie I : Introduction

Ce rapport présente notre projet de classification d'images utilisant le jeu de données ImageNet de 2012. L'objectif principal est d'obtenir une bonne précision de classification sur le jeu de données de test en utilisant un modèle de réseau de neurone existant dans la littérature. Nous avons décidé d'implémenter Resnet.

Le jeu de données d'entraînement comprend 4000 images de tailles variées (1000 images pour chacunes de ces classes : chou-fleur, tente de montagne, chou pommé et nid d'abeilles). Le jeu de données de test contient environ 1000 images à classifier.

2 Partie II : Resnet

2.1 Spécificité

La spécificité de ResNet réside dans son utilisation de blocs résiduels et de shortcuts afin de résoudre le problème de dégradation de l'accuracy lorsque la profondeur du réseau devient très élevé.

Contrairement aux architectures traditionnelles où chaque couche transforme complètement les données d'entrée, les shortcut permettent aux données non altérées de contourner certains blocs et d'être directement transmises à des couches ultérieures. En ajoutant les sorties d'une couche à l'entrée de la couche suivante, les shortcuts permettent aux données originales de se propager à travers le réseau sans subir de transformations excessives. Cela facilite la propagation des gradients lors de la rétropropagation de l'erreur et donc l'apprentissage de modèles plus profonds.

2.2 Implémentation de Resnet

Nous avons d'abord défini une classe représentant nos blocs résiduels (Residual Blocks). Chacun de ces blocs est composé de deux couches de convolution, suivies d'une couche de normalisation par lots (batch normalization) et de la fonction d'activation ReLU.

Nous avons ensuite défini notre modèle Resnet comme ceci :

1. Couche de convolution initiale : noyau de taille 7x7, un stride de 2 et un padding de 3, suivie d'une batch normalization et de la fonction d'activation ReLU (pour introduire la non-linéarité dans le réseau)

2. Couche de max pooling : noyau de taille 3x3, un stride de 2 et un padding de 1 pour réduire le nombre de paramètre tout en préservant les grandes valeurs

3. Quatre couches de blocs résiduels défini précédemment. A chaque sortie de bloc résiduel, on utilise l'opération d'identity mapping afin d'ajouter à cette sortie, l'entrée non transformée de ce bloc (par le shortcut). Si la dimension diffère entre l'entrée et la sortie du bloc, on ajuste les canaux de l'entrée pour correspondre aux canaux de sortie grâce à une projection linéaire implémentée avec une couche de convolution 1x1

4. Couche de average pooling : noyau de taille 8x8 et un stride de 1

5. Couche linéaire (fully-connected) : transforme le tenseur d'entrée en un tenseur de sortie de la taille du nombre de classes (ici 4). Cette couche est suivie d'une fonction de softmax pour obtenir des probabilités de classe.

3 Partie III : Entraînement

3.1 Prétraitement et augmentation des données

Nous avons réaliser de l'augmentation de données afin d'avoir un dataset d'entraînement plus riche.

Trois ensembles de données d'entraînement distincts sont créés en appliquant les transformations successives suivantes :

— Retournement horizontal aléatoire : Cette transformation flippe horizontalement les images de manière aléatoire

— Rotation aléatoire : Cette transformation effectue une rotation aléatoire des images dans une plage de 10 degrés

— ColorJitter : Cette transformation ajuste la luminosité, le contraste, la saturation et la teinte des images avec des valeurs aléatoires dans certaines plages spécifiées

Ensuite, ces 3 ensembles de données augmentées ainsi que l'ensemble de données d'entraînement initial subissent un prétraitement des données :

— Recadrage : Les images sont recadrées pour avoir une taille de 224x224 pixels en conservant le centre de l'image

— Conversion en tenseur : Les images sont converties en tenseurs

— Normalisation : Les tenseurs d'image sont normalisés en utilisant les moyennes et les écarts types spécifiés

Les 4 ensembles de données sont concaténé afin de donne notre jeu de données d'entraînement final.

3.2 Résultats

Le modèle ResNet utilisé dans notre projet a une profondeur de 50 couches, comme spécifié par les nombres de blocs [3, 4, 23, 3]. Il est optimisé avec la fonction de loss CrossEntropyLoss. L'optimisation est réalisée avec l'algorithme de descente de mini-batch gradient stochastique avec un taux d'apprentissage initial de 0,01, un poids de dégradation de 0,001, un batch_size de 64 et un momentum de 0,9. De plus, nous utilisons un scheduler pour ajuster le taux d'apprentissage avec une décroissance planifiée toutes les 5 époques.

Voici nos résultat que vous pouvez retrouver dans nos submits :

Modèle	Précision
ResNet 50 sans augmentation de données et sans normalisation	0.56931
ResNet 50 sans augmentation de données et avec normalisation	0.65619
ResNet 50 avec augmentation de données et avec normalisation	0.84103

TABLE 1 – Précisions des modèles ResNet 50 avec différentes configurations

4 Partie IV : Pouvons-nous faire confiance aux systèmes d'IA

La question de la confiance dans les systèmes d'IA devient d'autant plus importante aujourd'hui en raison de la multitude d'applications et de modèles reposant sur l'IA et de la vitesse à laquelle ce domaine de la recherche évolue. Il devient alors nécessaire de protéger non pas les systèmes d'IA eux-mêmes uniquement des attaques potentielles qu'ils peuvent subir, mais aussi de protéger les utilisateurs de ces systèmes des différents dangers liés à l'utilisation de l'IA dans l'aide à la décision.

Ces dangers reposent principalement dans les biais auquel peut être sujet un système d'IA, ce qui est moralement inadmissible. L'article donné comme référence donne plusieurs exemples ayant des

impacts non-négligeables sur la société dans différents domaines : juridique, financier et médical par exemple. Cet article insiste alors sur la présentation de différentes manières de mesurer et d'éviter l'apparition de biais discriminatoires dans les systèmes d'IA, malgré qu'il n'existe pas encore de critères standards ou de normes unifiées permettant la détection et contourner ce problème. Cependant, il est important de noter que l'utilisateur doit être mis au courant de l'existence potentielle ou certaine de biais dans le système qu'il s'apprête à utiliser, à travers la rédaction d'une documentation complète et transparente dans la mesure du possible.

Ceci étant dit, on peut proposer certaines mesures permettant la protection contre l'apparition de biais dans un système d'IA.
D'abord, une étape importante est d'identifier les biais potentiels qui peuvent influencer le système en question. En effet, si le concepteur ne prend pas en compte un certain critère lors du développement du modèle alors il ne pourra pas préparer le système à l'éviter. Et c'est ici qu'intervient le manque de standardisation et de normalisation de critères de discrimination contre lesquels il faut protéger les systèmes d'IA.
Ensuite, le point le plus important qui permet de minimiser les discriminations est de préparer l'ensemble d'entraînement. En effet un ensemble d'entraînement biaisé induit systématiquement à l'apparition de ces biais dans le modèle. Il faut donc être conscient des biais que comporte le jeu d'entraînement et essayer de l'équilibrer.

Enfin, afin de pouvoir faire confiance à un système d'IA, il faut être en mesure de l'expliquer et vérifier sa robustesse. L'explicabilité d'un modèle devient d'autant plus difficile quand le modèle devient complexe et possède un très grand nombre de degrés de libertés, et devient impossible quand le modèle n'est pas linéaire. Il faut alors bien choisir la classe de modèle de son système d'IA. On remarque par exemple que les modèles les plus explicables sont ceux basés sur les arbres de décision, et ceux basés sur les modèles linéaires. La robustesse d'un système d'IA quant à elle peut être vérifiée à travers le stress testing du système, l'analyse de sensibilité ou encore les tests adversariaux avec des outliers.

En conclusion, de notre temps il est très difficile de faire confiance aux système d'IA ; l'avancée de la recherche dans le domaine joue un rôle important dans ce phénomène, mais un plus grand contributeur reste la non acceptation de la société envers les erreurs commises par les systèmes d'IA, même si ces systèmes prouvent leur performance par rapport aux humains. Mais malgré tout, un problème encore plus grand réside dans la prise de responsabilité quand les systèmes d'IA se trompent. Devons-nous attribuer cette responsabilité au concepteur ou à l'usager ?