DÉVELOPPEZ UNE PREUVE DE CONCEPT

PAR :

NAMA NYAM GUY ANTHONY

06 Mai 2020

Synthèse de travail

Le plan de travail suivant met en perspectives une veille thématique sur les avancées des architectures CNN et leurs fine-tuning pour la classification d'images à l'aide d'algorithme de Deep Learning.

- **Thématique :** elle porte sur les réseaux de neurones et du Deep Learning. Nous avons exploré un certain nombre de modèles de classification d'images de l'état de l'art sur les critères de performances ***Top-1 accuracy***, de complexité de l'architecture(***nombre de paramètres***) et de fine-tuning sur le jeu de données ImageNet.
- **Méthode mise en œuvre :** la méthode retenue après exploration est ***FixRes*** et l'usage des poids entraînés d'un modèle ***EfficientNet*** ou ***ResNext-101***. Le choix de cette méthode est due au fait qu'elle améliore systématique la Top-1 accuracy de l'ensemble des architectures CNN. Le choix du modèle est qu'il fournit les meilleurs capacité de compromis de performances ainsi que la disponibilité de ses poids. La méthode est développée par ***Facebook Research***. Nous utilisons le framework ***PyTorch*** pour implémenter la méthode.
- **dataset :** les performances de la méthode mise en œuvre sont évaluées sur le dataset ***Stanford Dogs***. Dataset avec 20,580 images de chiens pour 120 races. Les mêmes conditions de séparations des données sont reproduites c'est-à-dire même proportion train/validation/test et le contenu à l'identique que la méthode baseline.
- **Méthode baseline :** méthode utilisée lors du précédent projet 'Classez des images à l'aide d'algorithmes de Deep Learning'. La méthode portait sur les poids du modèle ***inceptionResNetV2*** implémentée avec le framework ***Keras*** et dont la performance Top-1 accuracy a atteint ***91%***.
- **Références bibliographiques.**

[1] Touvron, Hugo and Vedaldi, Andrea and Douze, Matthijs and Jégou, Hervé. Fixing the train-test resolution discrepancy. arXiv :1906.06423v3, 2020.

[2] Xie, Qizhe and Hovy, Eduard and Luong, Minh-Thang and Le, Quoc V. Self-training with Noisy Student improves ImageNet classification. arXiv preprint arXiv :1911.04252, 2019.

Table des matières

1 Introduction

Le domaine du machine learning, et plus généralement de la data science évolue très rapidement. Il est donc important pour l'ingénieur machine learning de pouvoir développer l'habitude à rester au courant des avancées dans le domaine en effectuant ***une veille thématique***. Cette dernière consiste aussi à pourvoir ***monter rapidement en compétences*** sur une nouvelle thématique en sachant effectuer une recherche et mettre en pratique ***un nouvel algorithme de façon autonome(POC)***.

La preuve de concept(Proof of concept(*POC*) en anglais) est une démonstration de principe visant à vérifier qu'un concept ou une théorie à un potentiel pratique. Elle est usuellement courte ou incomplète. La *POC* est située très en amont dans un processus de développement et considérée comme une étape importante sur la voie d'un prototype pleinement fonctionnel. Elle s'applique également à divers domaines(ingénierie logiciel, développement des affaires, sécurité informatique, etc.).

La *POC* commence par un tour d'horizon de l'état de l'art du domaine cible(machine learning dans notre cas) où un plan prévisionnel du travail à réaliser sera établi comprenant : l'algorithme envisagé, le dataset sur lequel vous pensez évaluer les performances, un ou deux arguments justifiant votre choix, deux ou trois références bibliographiques (post de blog ou article de recherche) vous permettant de présenter un état de l'art sur le problème étudié et sur lesquels votre travail futur s'appuiera.

Par la suite, nous présentons la thématique retenue, le dataset pour évaluer les performances, la méthode implémentée, comparaisons et discussions.

2 Thématique

Le travail réalisé porte sur le domaine de la vision par ordinateur(computer vision), domaine qui est une branche de l'intelligence artificielle dont le but principal est de permettre à une machine d'analyser, traiter et comprendre une image prise par un système d'acquisition(caméras par exemple).

Nous nous intéressons à la classification d'images, qui consiste à attribuer un label parmi un ensemble de labels prédéfinis à une image. L'approche moderne pour le faire est le ***Deep Learning***. L'apprentissage profond utilise les réseaux de neurones artificiels convolutionnels(CNN) adaptés pour le traitement de l'image.

On assiste à de nouveaux architectures qui repoussent les performances des précédentes, devenant par la même occasion l'état de l'art pour la classification d'images. Les performances d'un modèle dépend aussi fortement du fine-tuning de ses hyper-paramètres qui constitue aussi un élément essentiel dans la suite de notre travail.

Nous tirons pleinement profit de nouveaux modèles pour la classification d'images et une méthode de fine-tuning pendant l'étape de la data augmentation.

1. Les poids pré-entraînés des modèles ***EfficientNet***[3] et ***ResNext-101***[4].
2. La méthode ***FixRes***[2] pour le fine-tuning des hyper-paramètres.

3 État de l'art

Ses dernières années, on assiste à une amélioration significative des performances sur les nouveaux architectures des réseaux de neurones convolutifs(CNN) parmi lesquelles dominent l'architecture *efficientNet*. Nous récapitulons de manière graphique les performances(top-1 accuracy et nombre de paramètres) des différents modèles.

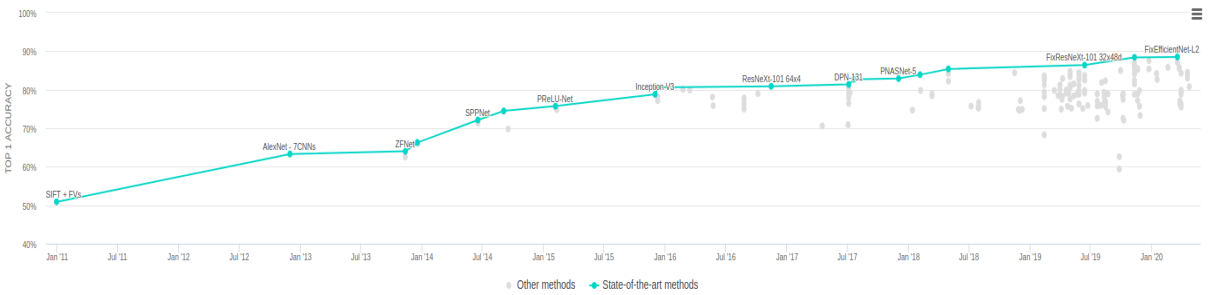

FIGURE 1 – Image Classification on ImageNet. - source : paperswithcode

Nous pouvons constater que le plus performant des modèles est ***FixEfficientNet-L2*** avec un ***top-1 accuracy de 88.5%*** pour ***480 millions de paramètres***. Néanmoins les poids pré-entrainés de ce modèle n'étant pas disponible et une des versions instables des prédécesseurs, nous utilisons les poids du modèle ***ResNeXt-101 32x48d*** seront utilisés, modèle à ***top-1 accuracy de 85.4%*** et ***829 millions de paramètres*** qui est l'un des meilleurs modèles disponibles et stables.

4 Dataset

Pour valider la nouvelle méthode élaborée, il est important de l'évaluer sur une méthode disponible(baseline) sur un dataset.

Le choix du dataset pour l'évaluation est ***Stanford Dogs*** qui contient 20,580 images de chiens pour 120 races. Ce jeu de données contient un nombre inégal de chiens par race comme répertorié sur la figure 2.

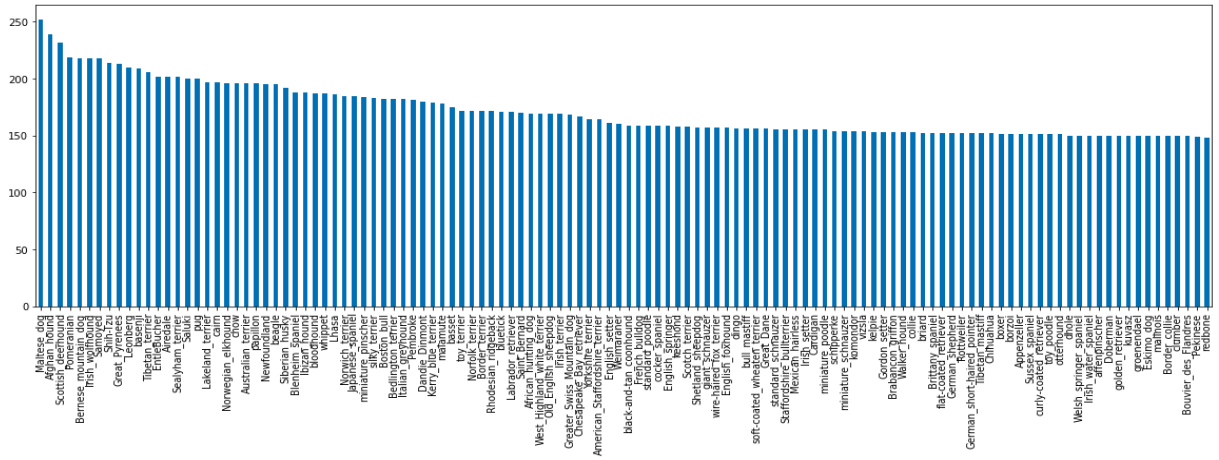

FIGURE 2 – Dataset Stanford Dogs - distribution race

Nous allons reprendre le contenu à l'identique de la séparation en *training set*, *validation set* et *test set* pour une comparaison équitable pour les différentes méthodes.
Autre information important dans le réglage dans la méthode implémentée est la taille des images(largeur et hauteur) du dataset. La figure 3 donne la distribution de ses paramètres.

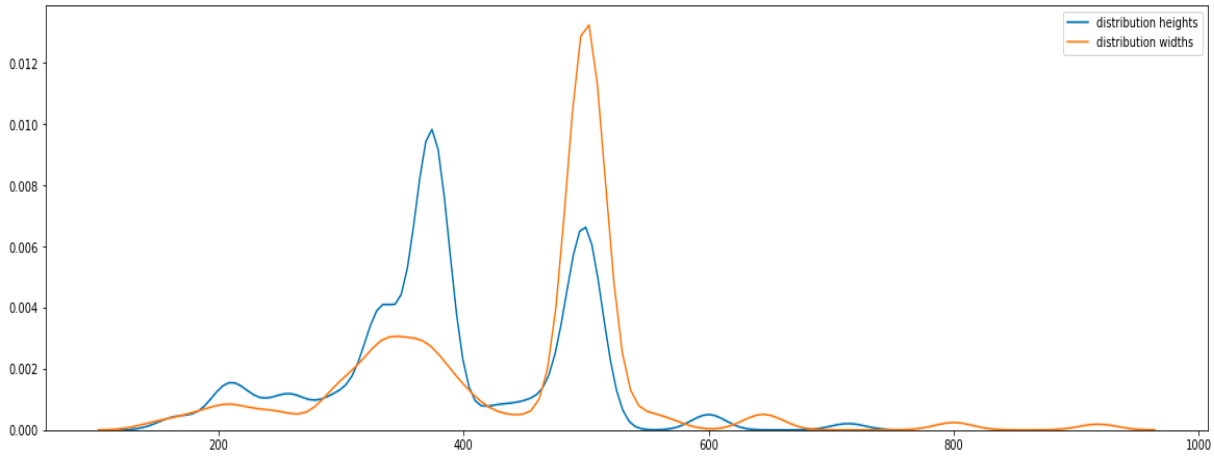

FIGURE 3 – Dataset Stanford Dogs - distribution image sizes

En ce qui concerne la méthode baseline, il s'agit de la méthode développée au cours du projet *'Classez des images à l'aide d'algorithmes de Deep Learning'*. La méthode utilisait les poids du modèle ***inceptionResNetV2*** et dont la performance ***Top-1 accuracy*** a atteint ***91%***.

5 Implémentation FixRes et ResNeXt-101 32x48d

FixRes[2] est une simple méthode pour corriger l'écart de résolution train-test. Elle peut améliorer les performances de toute architecture de réseau de neurone convolutif. La figure 4 permet d'illustrer le principe de la méthode.

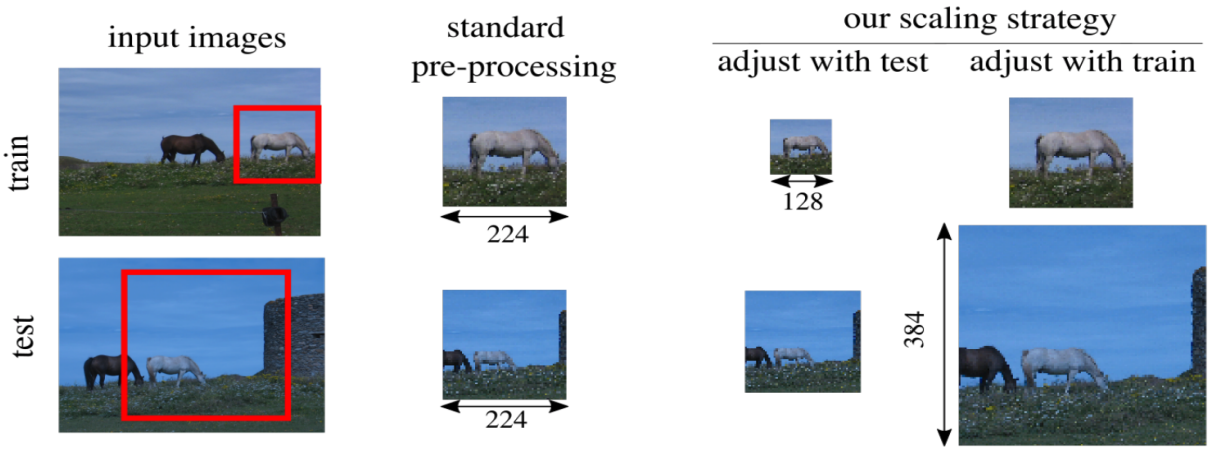

FIGURE 4 – FixRes principe. - source : facebookresearch/FixRes

Nous avons en rouge un exemple de sélection de régions d'images alimentant le réseau au moment d'entrainement(*RandomResizedCrop*) et de test(*CenterCrop*) avec une data augmentation classique. Ensuite ces régions sont recadrées(*Standard pre-processing*) pour alimenter les réseaux de neurones.

Problème : pour les objets de taille similaire dans l'image d'entrée comme le cheval blanc, la data augmentation standard le rend plus grand au moment de l'entraînement au détriment de la phase de test.

Solution : pour contrer cet effet, soit nous diminuons la résolution de l'image au moment de l'entraînement,soit nous augmentons la résolution de l'image au temps du test.

Le cheval blanc a donc la même taille au moment de l'entraînement et test ce qui entraîne une invariance d'échelle moindre pour le réseau de neurone. De plus la méthode nécessite juste un réglage fin et peu coûteux en terme de calculs.

Nous utilisons le module transforms_v2 pour la transformation du dataset mettant en œuvre la méthode FixRes dont nous devons renseigner la résolution des images d'entraînement(*input_size*) et la résolution des images de test(*test_size*) tel que sur la figure 5.

```
fixRes_image_transforms = get_transforms(input_size=320,test_size=380, kind='full', crop=True, need=('train', 'val'), backbone=None)
```

FIGURE 5 – fine tuning with FixRes method for fixing the train-test resolution discrepancy

Par la suite, utiliser la transformation *train* pour les données d'entraînement et la transformation *val_test* pour les données de validation et de test comme sur la figure 6. Les différentes transformations utilisent un unique recadrage(crop)

```
data_images = {
    'train': datasets.ImageFolder(root=dir_train, transform=fixRes_image_transforms['train']),
    'valid': datasets.ImageFolder(root=dir_valid, transform=fixRes_image_transforms['val_test'])
    'test': datasets.ImageFolder(root=dir_test, transform=fixRes_image_transforms['val_test'])
}
```

FIGURE 6 – Choix des transformations pour les données d'entraînement, validation et test

Nous chargeons le modèle *ResNeXt-101 32x48d* dont nous gelons les couches d'apprentissage et adaptons le classifier à 120 classes ce qui fera 245,880 paramètres à entraîner sur les 826,608,056 de l'ensemble du réseau. Le code est disponible dans [1].

6 Résultats et discussions

Pour obtenir les meilleurs résultats, il faut rechercher les hyper-paramètres (*train_size*, *test_size*). Plusieurs valeurs suivant les résultats de la littérature sur *FixRes* pour restreindre le champs d'exploration ont été balayées mais aussi la prise en compte des tailles des images du dataset *Stanford Dogs*. Les valeurs *accuracy* manquantes n'ont pas été évaluées dut à des mauvaises *accuracy* pendant la validation.

Train_size	***Test_size***	***Accuracy***
128	224	
224	128	89.30%
224	224	93.21%
224	320	93.48%
224	400	
320	224	94,73%
320	320	94.40%
320	380	CUDA out of memory

Performance : la première appréciation à faire est que la méthode développée permet un ***gain net*** par rapport à la méthode baseline de ***3.7%*** qui est considérable.

Résolution : plusieurs explorations sur la taille de résolution *Train_size* et *test_size* peut être faites, néanmoins la valeur fixe *Train_size* à 320 est un bon point de départ(peu de puissance de calcul) pour ce dataset.

Complexité de mise en oeuvre : la mise en œuvre de la méthode est simple.

Temps de calcul : le temps reste considérable sur un seul GPU et évolue faiblement avec la résolution des images d'entraînements et de tests. Temps évalué entre 4 et 6 heures en moyenne.

Framework Pytorch : la méthode baseline a été développée sur *Keras* ; je dirais que *Pytorch* est moins abstractif que *Keras* c'est à dire plus de code mais plus puissant que *Keras* pour le fine-tuning des méthodes. Je pense également que les temps d'exécutions d'entraînement et d'évaluation sont similaires.

7 Conclusion et perspectives

Au terme de notre expérience, force est de constater que la théorie initiale a un potentiel en pratique. Les nouveaux architectures permettent d'améliorer les résultats. Néanmoins, cela nécessite de peaufiner davantage la data augmentation. Dans ce projet, nous avons utilisée une méthode qui raffine le pré-traitement standard permettant de réduire l'écart de résolution ou invariance d'échelle *train-test*. Nous avons atteint *top-1 accuracy* de *94.73%* sur le dataset *Standford Dogs*. La mise en œuvre est simple et l'utilisation du *Framework PyTorch* apporte une plus grande flexibilité dans le pré-traitement des images.

Pour la suite de ce travail, il serait important d'avoir plus de ressources de calculs pour étendre l'espace de recherche de résolution. Il serait tout aussi important d'exploiter les poids pré-entraînés des modèles de meilleur *top-1 accuracy* que *ResNeXt-101 32x48d* mentionnés dans la littérature. Nous avons utilisé un seul recadrage(*Crop*) pendant la phase de la data augmentation, avec plus de puissance de calcul, on pourrait augmenter le nombre de recadrage(10) pour vérifier si l'impact est important sur les résultats.

Références

[1] Implementation. https://github.com/AnthonyNama/FixRes-method-fine-tuning-CNN-Pytorch-API

[2] Touvron, Hugo and Vedaldi, Andrea and Douze, Matthijs and Jégou, Hervé. Fixing the train-test resolution discrepancy. arXiv :1906.06423v3, 2020.

[3] Mingxing Tan 1 Quoc V. Le. EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks. arXiv :1905.11946v3, 2019.

[4] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. arXiv :1805.00932, 2018.