Rapport de challenge

Land cover predictive modeling from satellite images

Par :
Homer Durand

Encadrant :
Maxime Sangnier

Novembre 2021

Contents

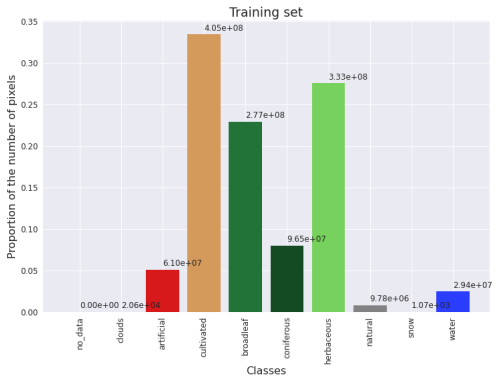

Figure 1: Distribution of the classes on the lancover masks

1 Analyse et pré-traitement des données

Le jeu de données proposé pour ce challenge est composé d'un ensemble de 18491 images satellites pour l'entraînement et de 5043 images de test, toutes de taille 256×256 et composées de 4 canaux (les trois premier étant les canaux RGB et le quatrième représentant les proches infrarouges). L'objectif étant de prédire la distribution des classes associées à chacun des pixels sur chacune des images. Nous disposons pour cela de masque qui sont des images de taille 256×256 dont l'unique canal correspond à la classe du pixel se trouvant à la même position sur l'image satellite. Comme on peut le voir (fig [1]) la distribution des différentes classes est particulièrement déséquilibrée, cette information sera importante à prendre en compte lors du choix de la fonction de coût utilisée pour entraîner le modèle.

Les temps d'entraînement des modèles étant relativement élevés (de l'ordre de plusieurs heures) il ne sera pas possible de mettre en place une évaluation du type validation croisée car celle-ci serait bien trop coûteuse en temps, nous séparerons simplement les données en un jeu d'entraînement (80% des données d'entraînement) et un jeu de validation (20% des données d'entraînement) afin de sélectionner le modèle le plus performant et d'optimiser les hyper-paramètres du modèle.

2 Sélection de modèle

Les Réseaux de neurones convolutionels (CNN) sont particulièrement performant pour un grand nombre de tâche de *computer vision* et particulièrement pour la segmentation sémantique. Ils sont aujourd'hui considérés comme l'état de l'art pour ce type de tâche lorsque le nombre de données est suffisamment important et c'est pourquoi j'ai décidé de me concentrer sur ce type de modèle. La première approche mise en place pour ce challenge a donc été d'utiliser un simple réseau de convolution afin de réduire la dimensionalité des images et de finalement prédire les vecteurs des distributions des classes avec une couche de neurones dense. Ce type de modèle semblait poser un problème du fait du nombre de données relativement réduit qui limité ainsi le degré de liberté du modèle et donc sa capacité de généralisation. J'ai donc choisi d'utiliser un modèle pré-entraîné (sur le jeu de données ImageNet) comme *backbone* du CNN afin d'utiliser les informations acquises sur ce large jeu de données et ainsi d'augmenter la capacité de généralisation du modèle (mobilenetV2 semblait être un choix cohérent du fait de ses performances pour la classification sur ImageNet et du fait qu'il est relativement rapide à entraîner comparativement à des modèles plus lourd du type ResNet).

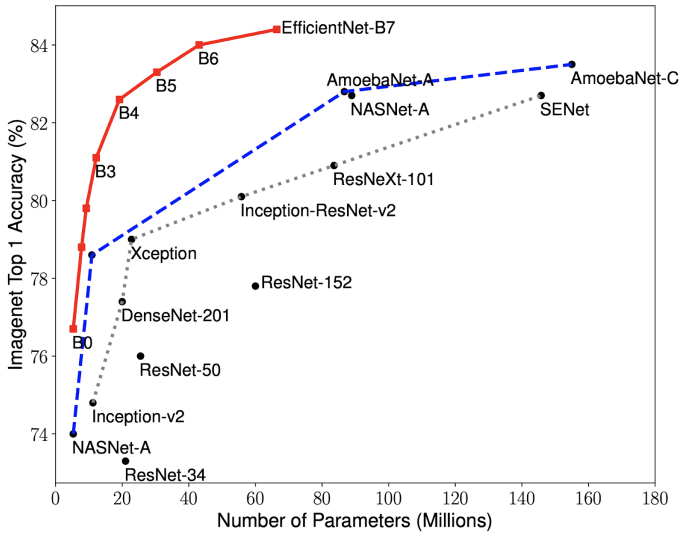

Figure 2: EfficientNet performance compared to others benchmarks models

Malgré que cela ait amélioré les résultats obtenus ce type de modèle semblait atteindre un plateau dans les performances qu'il permettait (une divergence de kullback-leibler d'environ 0.2). Je me suis donc tourné vers les modèles de type Unet qui semblent faire parti de l'état de l'art en ce qui concerne la segmentation sémantique d'image (voir [1]). L'utilisation de la librairie python **segmentation models** (`https://github.com/qubvel/segmentation_models`) m'a permis de tester plusieurs *backbones* comme structure du Unet sans avoir à le ré-implémenter complètement.

Il semblait alors que les modèles du type EfficientNet serait particulièrement intéressant puisqu'il présentent un bon compromis entre leur performance sur le jeu de données ImageNet et leur nombre de paramètres (voir 2). Après plusieurs tests, le modèle EfficientNet-B3 semblait permettre les meilleurs performances tout en gardant un temps d'entraînement raisonnable et en évitant le sur-apprentissage dû à un trop grand degré de liberté du modèle.

Après plusieurs tests il s'est avéré que l'architecture LinkNet était plus performante et plus rapide a entraîné que l'architecture Unet, c'est donc cette architecture que j'ai finalement choisi.

Afin d'entraîner le modèle il était également nécessaire de choisir une fonction de coût à optimiser. Dans un premier temps tout les modèles ont été entraîné avec la *categorical cross entropy* qui permettait d'optimiser l'accuracy du modèle. Il m'a dans un second temps parût plus pertinent de chercher à optimiser la mesure de performance *Mean Intersection Over Union* (mIoU) et j'ai donc pour cela utilisé la fonction de coût jaccard. En effet cette loss prendra ainsi mieux en compte l'aspect très déséquilibré des classes des masques.

$$
\begin{aligned}
IoU_i(A, B) &= \frac{|A_i \cap B_i|}{|A_i \cup B_i|} \\
mIoU &= \frac{1}{n} \sum_{i=1}^{n} IoU_i(A, B) \\
jaccard(A, B) &= 1 - mIoU(A, B)
\end{aligned}
$$

Où i représente le nombre de classe des masques.

3 LinkNet

Comme cela est décrit dans [2] le modèle LinkNet est une variante des modèles en forme de U du type Unet qui possède deux principales différence, à savoir que les bloques de convolution sont remplacés par des bloques résiduelles (*res-block*) et que la synthèse des bloques ne se fait plus par empilement mais par ajout.

L'architecture m'ayant permis d'obtenir les meilleurs résultats est un LinkNet dont le *backbone* est le modèle EfficientNet-B3. Seules les couches de déconvolution sont entraîné. La fonction de coût utilisé est la *jaccard loss* est l'optimiseur Adam. L'entraînement est arrêté par *early-stopping* lorsque la fonction de coût commence à augmenter (après 6 epochs dans notre cas).

Le modèle entraîné sur 80% des données d'entraînement (le reste servant à évaluer le modèle) atteint ainsi une divergence kullback-leibler de 0.0917 sur le jeu de test.

Il semble qu'une piste d'amélioration intéressante serait l'utilisation de data augmentation (*horizontal et vertical flip*) qui permettrait d'utiliser un modèle ayant un plus grand degré de liberté et ainsi une meilleure capacité de généralisation sur les données de test.

4 Réussites et difficultés

La première difficulté rencontrée durant ce challenge a été le taille du jeu de données qui était trop grand pour être stocké complètement dans la RAM de mon ordinateur, il a donc fallut utiliser un générateur de données afin de résoudre ce problème.

La seconde difficultés vient du manque de puissance de calcul pour entraîner les modèles rapidement. J'ai dans un premier temps utilisé google colab qui permet l'utilisation de GPUs qui accélèrent significativement le temps d'entraînement des modèles mais ait ensuite abandonné cette option car le temps d'utilisation des GPUs est trop limité pour que cela permette un gain de temps significatif pour l'entraînement des modèles.

Une autre difficulté à laquelle j'ai été confronté durant ce challenge est l'implémentation des modèles avec les librairies python *keras* et *tensorflow*. La librairie **Segmentation Models** a donc été d'une grande aide pour cela.

Je suis en revanche heureux d'avoir réussi a atteindre de bonnes performances sur une tâche à laquelle je n'avais jamais été confronté. Plus particulièrement je suis satisfait d'avoir réussit à trouver des approches et modèles qui m'ont permis de constamment améliorer les performances sur cette tâche de segmentation sémantique.

5 Bibliographie

References

[1] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 02 2021.

[2] Ming Wu, Chuang Zhang, Jiaming Liu, Lichen Zhou, and Xiaoqi Li. Towards accurate high resolution satellite image semantic segmentation. *IEEE Access*, PP:1–1, 04 2019.