Synthèse technique projet segmentation image au sein de Future Vision Transport

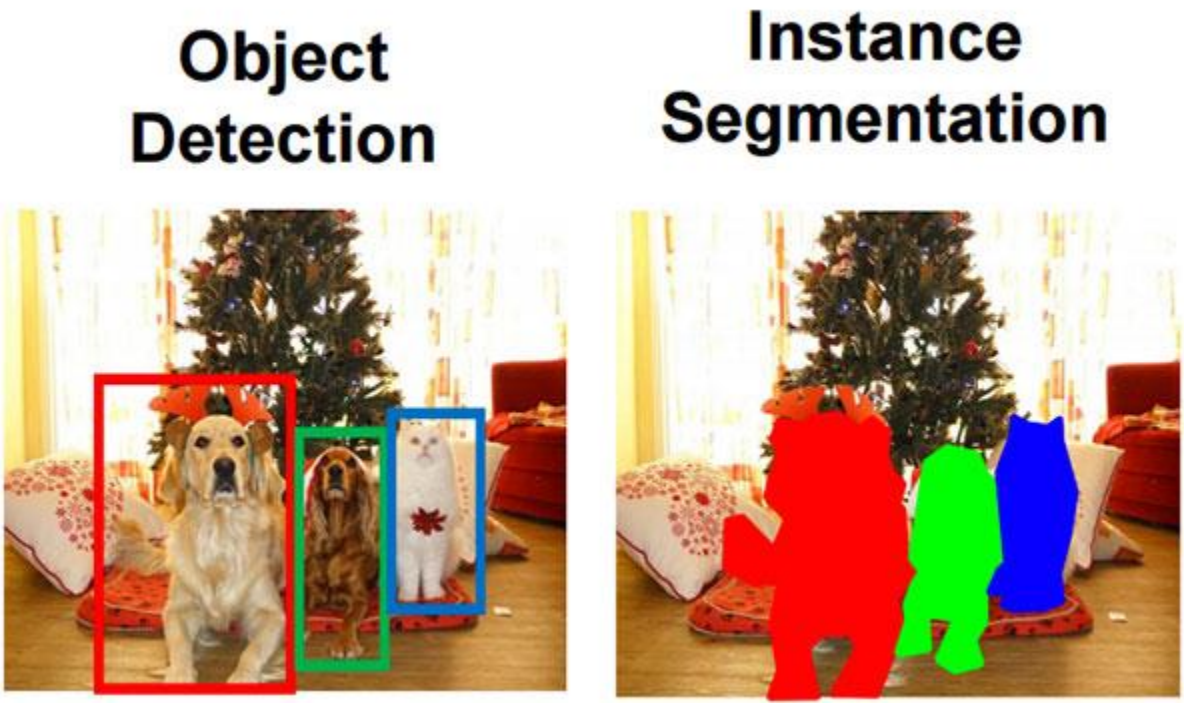
Bar Augustin, le 2022/10/12

Formation Ingénieur IA par OpenClassrooms, projet 8

Introduction au concept

Le projet demandé par le client concernait une tâche de segmentation sémantique d'une image. Cette discipline fait partie des problèmes de vision assistée par ordinateur des données en plusieurs régions, effectuée ici sur des images.

Ici, nous ne sommes pas sur une tâche de **détection** mais de **segmentation** : on cherche à découvrir la forme de l'objet et non sa position.



Source : cs231n.stanford.edu

De plus, nous sommes sur un problème de segmentation **sémantique** et non **instanciée** car nous ne souhaitons pas dénombrer les instances des classes mais juste leur nature.

Image 1

Image 2

Dans le cas de la demande client, le but est : à partir d'une image source, produisez une image segmentée. L'utilisation pratique sera de permettre à une voiture autonome de prendre des décisions de conduite en filmant la route, analysant ces images via notre modèle de segmentation et utilisant les résultats pour décider de sa conduite.

Nous disposons ici de données d'entrainement labellisées selon des normes définies préalablement. Vous pourrez retrouver le jeu de donnée utilisé sur ce site.

Les différents types de régions à segmenter sont au nombre de 8 et peuvent apparaître plusieurs fois de façon indépendantes dans l'image. Le jeu de données fait une distinction entre plusieurs sous-classes au sein même de chaque classe mais comme preuve de concept, notre client nous a demandé de ne faire une segmentation que sur les 8 classes principales (désigné dans le jeu de donnée sous la dénomination « group »).

Les différentes classes sont :

1. Sol : la route, les trottoirs, les places de parking ainsi que les rails
2. Humain : piétons ou humain dans un véhicule. Cependant, les passagers de voitures avec un toit sont labellisés comme le véhicule dont ils font partie
3. Véhicule : les voitures, les camions, les bus, les trains, les motos, les vélos, les caravanes ainsi que les remorques
4. Construction : les immeubles, maisons, murs, barrières, les barrières piétonnes, les ponts ainsi que les tunnels
5. Objet : Les poteaux, les groupes de poteaux, les panneaux de signalisation, et les feux de circulation
6. Nature : la végétation ainsi que les terrains qui ne sont pas praticables (pelouse, sable, gravier, etc…)
7. Ciel : concerne le ciel
8. Vide : Ce qui est plat et ne rentre pas dans les catégories précédentes, comme les ronds-points ou autres obstacles horizontaux. Concerne aussi les objets dits « dynamiques », dont leur présence variera selon les horaires de la journée (poubelles, animal, sac, etc..) ainsi que les objets « statiques » qui font partie de la voiture sur laquelle est montée la caméra (si on voit le capot par exemple).

Etat de l'art

La tâche de segmentation sémantique d'une image est divisée en deux sous-tâches qui peuvent être effectuées en parallèle ou à la suite : la détection des contours et la classification.

Jusqu'à il y a quelques années, la plupart des méthodes de détection des formes se faisait via des techniques de VAO (vision assistée par ordinateur), comme la détection des contours, la détection de pixels similaires ou des techniques de clustering. Souvent, la tâche de classification venait après. L'entrainement de ces modèles se faisait de façon non supervisée.

Grâce à la publication d'U-net en 2015, au départ pour une utilisation médicale, la recherche s'est transformée et s'est concentrée sur les réseaux de neurones à convolution (CNN). Ceux-ci ont un temps de calcul raisonnable tout en prouvant des performances étonnantes.

Depuis, la recherche a énormément avancé en proposant un nombre d'architecture impressionnant dont les connus FCN (Fully Convolutional Network) et PSPNet (Pyramid Scene Parsing Network).

De plus, il semblerait qu'à ce jour, les réseaux les plus performants utilisent des techniques basées sur les transformers, une autre philosophie d'architecture.

Méthodologie de sélection des modèles

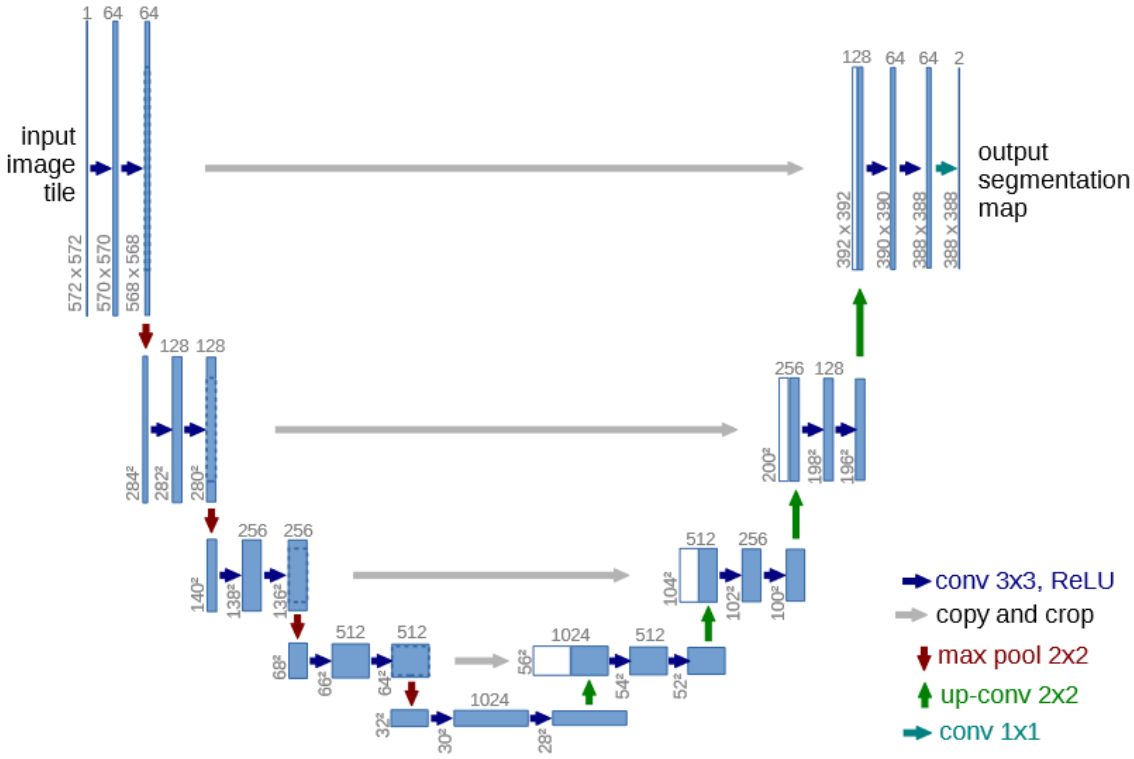
Pour des raisons de performance mais surtout car nous avions besoin de l'identification des régions reconnues, nous avons décidé de ne pas implémenter les modèles basés sur des techniques de VAO classiques comme la détection de contours. Nous avons décidé de même de ne pas nous pencher sur l'entrainement d'un modèle basé sur une architecture de transformer car les nombres de poids à entrainer sont trop gros pour une preuve de concept.

Nous avons alors décidé de nous pencher sur deux réseaux, en premier U-Net car il a été le déclencheur d'une nouvelle méthode de recherche et d'architecture et PSP-Net car c'est un des plus populaires encore aujourd'hui.

Présentation approfondie des modèles retenus

U-net

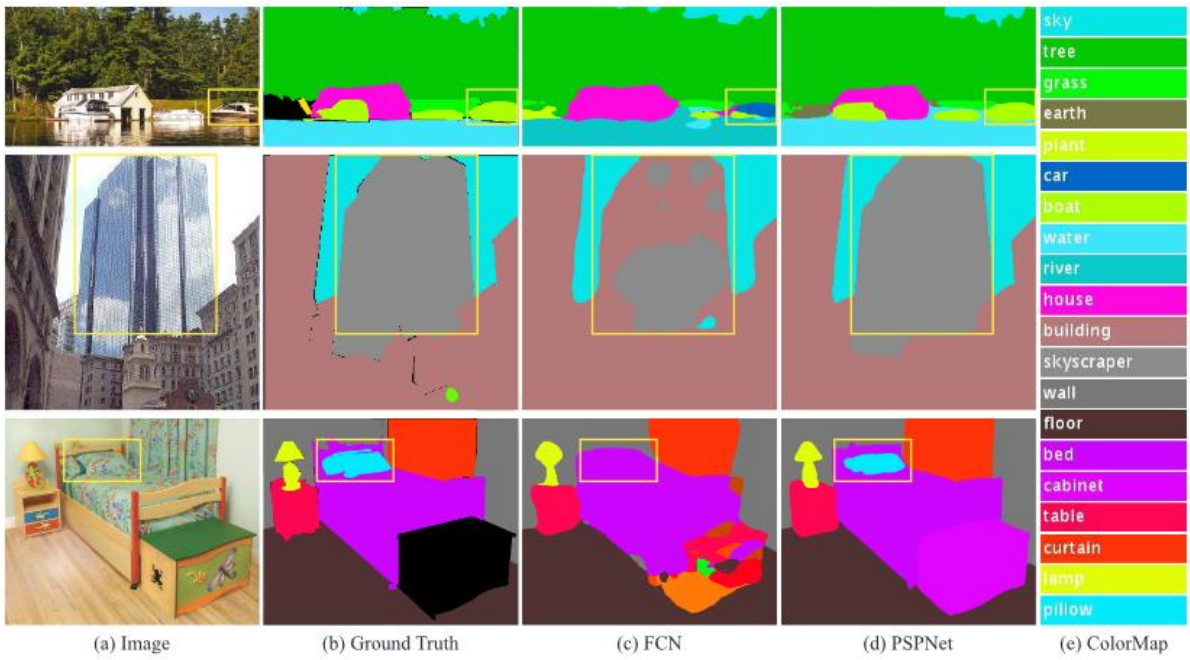
Ce modèle est basé sur un réseau de neurones à convolution, c'est-à-dire un empilement de couches de convolution, de pooling, de correction, de perte ainsi que de couches entièrement connectées. L'architecture Unet est composée d'une partie contractante (aussi appelée encoder) et d'une partie expansive (aussi appelée decoder). Ces deux parties sont mises à suite l'une de l'autre mais aussi reliés par des liens « raccourcis ». Sa forme caractéristique en U lui a donné son nom :



Source : https://arxiv.org/pdf/1505.04597.pdf

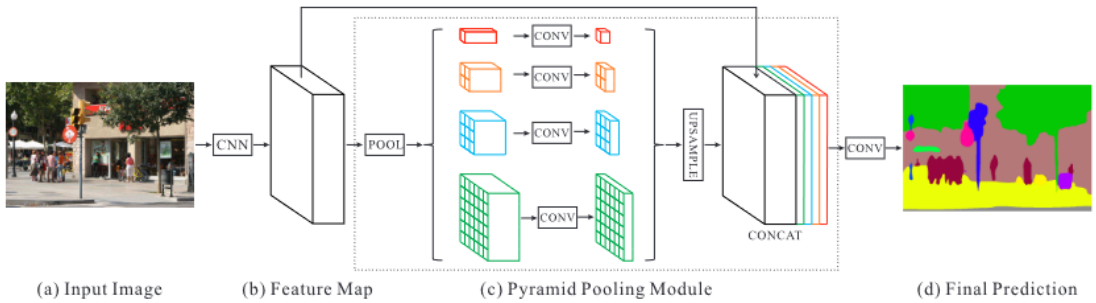
PspNet

PspNet est très fort pour prendre en compte le contexte de son environnement, c'est ce qui le différencie le plus de son concurrent. Dans le papier de publication, il est mentionné qu'il ne fait plus, comme ses prédécesseurs, l'erreur de classifier comme une voiture un bateau car il reconnait que la supposée voiture est sur l'eau, ni l'erreur de classifier des objets de plusieurs labels (ex : mélanger buildings et skyscraper), et enfin l'erreur de mal classifier un oreiller car il est de la même texture que le drap.



Source : https://arxiv.org/pdf/1612.01105v2.pdf

La structure du PSPNet est dite en pyramide et elle contient en son sein un modèle de CNN (possiblement déjà entrainé). L'image sortante du CNN rentre alors dans la structure pyramidale et subit des opérations de pooling, de convolution et d'upsampling avant d'être concaténés. A la toute fin, une couche de convolution se charge de faire la classification.



Source : https://arxiv.org/pdf/1612.01105v2.pdf

Le modèle choisi utilise le CNN nommé VGG16 qui est déjà pré-entrainé.

Processus d'entrainement

Un prétraitement est nécessaire sur les données afin de les préparer à être exploitées. En premier, il faut réduire leur taille afin de limiter le temps de calcul et les adapter aux architectures des modèles choisis. Certains modèles ont des architectures variables mais nous avons préféré de redimensionner les images en 384x576 et afin de correspondre aux deux architectures.

Ensuite, il a fallu modifier les annotations des images car celles-ci désignaient les sous-classes et non les classes (comme discuté précédemment). Cette opération de mapping de valeur à valeur prend surprenamment un temps non négligeable.

Afin de pouvoir ensuite mesurer l'efficacité des deux modèles sur les mêmes critères, nous avons entrainé les modèles sur 5 cycles avec un batch size de 2. L'entrainement d'un modèle prend de 2h30 à 3h00.

Résultats préliminaires

Lors de l'entrainement de nos modèles, nous avons enregistré les performances au cours des différents cycles ainsi que les modèles intermédiaires associés.

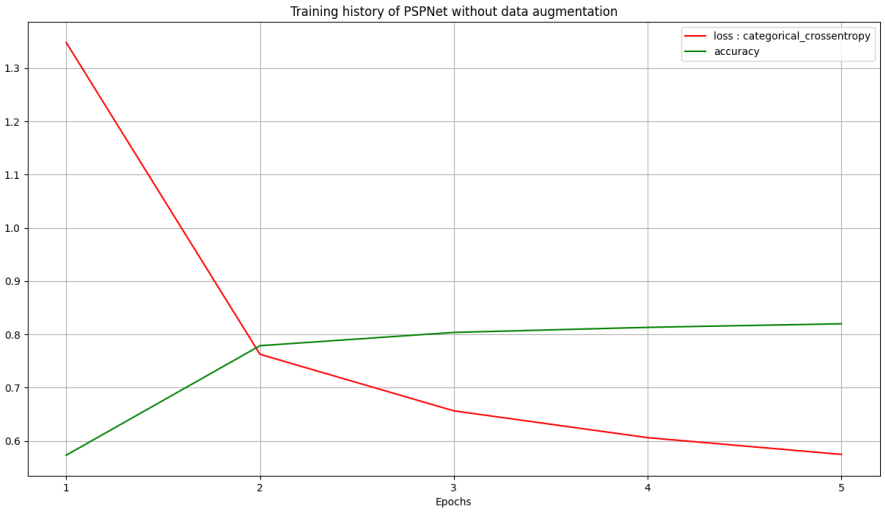
Avec l'historique des entrainements, nous avons pu tracer l'évolution des performances au cours de l'apprentissage pour les deux modèles sélectionnés. Le calcul des performances d'accuracy et de mean IoU final est réalisé sur un jeu de validation indépendant.

Accuracy : 0.76558703

Mean IoU : 0.4136017

IoU for each class :

- Class flat : 51.3%
- Class human : 76.3%
- Class vehicle : 38.9%
- Class construction : 2.5%
- Class object : 65.0%
- Class nature : 58.7%
- Class sky : 3.2%
- Class void : 34.9%



Accuracy : 0.8153718

Mean IoU : 0.428192

IoU for each class :

- Class flat : 45.9%
- Class human : 80.5%
- Class vehicle : 56.7%
- Class construction : 0.0%
- Class object : 64.1%
- Class nature : 45.4%
- Class sky : 5.5%
- Class void : 42.2%

On voit une bonne amélioration des performances, l'accuracy est toujours croissante et le loss toujours décroissant donc pas de trace d'overfitting. Des cycles d'entrainement supplémentaires sont envisageables.

D'ailleurs, dans la publication de PSPNet, entrainé sur le même dataset, on arrive à une accuracy de 80.88% et Mean IoU de 34.81% (Table 3). https://arxiv.org/pdf/1612.01105v2.pdf

Ce sont des résultats très cohérents avec ceux qu'on a trouvé.

Processus d'augmentation des données

Afin d'améliorer les performances de nos modèles et d'éviter l'overfitting, on va introduire un processus d'augmentation de données via la librairie imgaug.

On a arbitrairement choisi d'augmenter 10% de nos données. Ce qui, avec 849 images nous donne une création de 85 images augmentées.

Parmi les augmentations, on retrouver 4 types de modification :

- Le rognage, ou on va venir recadrer l'image de 0px à 50px
- La rotation horizontale, qui est appliquée dans 50% des augmentations
- Le flou gaussien, appliqué de 0 à 3
- La rotation, appliquée de -45° à 45°

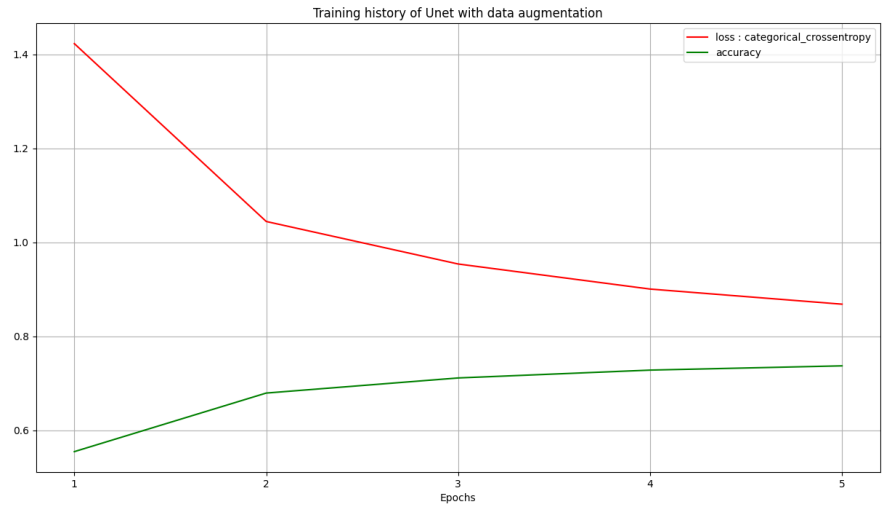
Bien évidemment, on augmente **seulement** le jeu d'entrainement et non pas le jeu de validation (ou de test) pour ne pas influencer la partialité de l'évaluation.

Dans le jeu d'entrainement, les images ainsi que les annotations subissent les mêmes transformations.

Les modèles ayant été entrainés sur un jeu de modèle augmenté seront nommés respectivement « Unet augmenté » et « PSPNet augmenté ».

Résultats de l'entrainement augmenté

Après entrainement des modèles et récupération des métriques, on peut constituer des graphiques pour analyser le processus d'entrainement et les performances de nos modèles.



Accuracy : 0.74567604

Mean IoU : 0.40094718

IoU for each class :

Class flat : 48.5%

Class human : 75.0%

Class vehicle : 38.3%

Class construction : 2.6%

Class object : 62.1%

Class nature : 56.8%

Class sky : 2.8%

Class void : 34.7%

Accuracy : 0.80709636

Mean IoU : 0.4134327

IoU for each class :

Class flat : 45.9%

Class human : 80.1%

Class vehicle : 53.6%

Class construction : 0.0%

Class object : 62.8%

Class nature : 45.8%

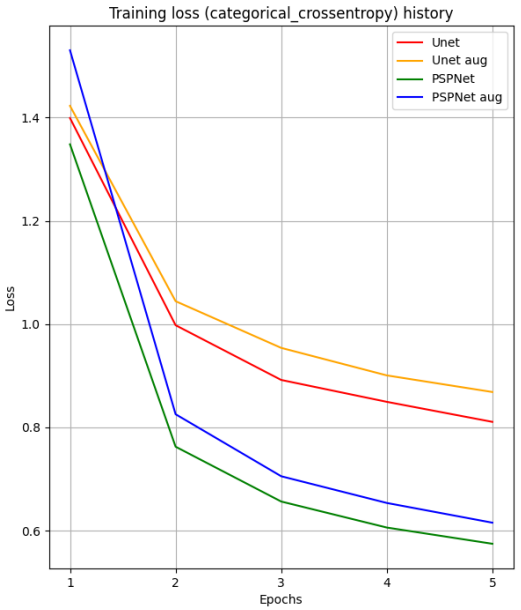
Class sky : 2.9%

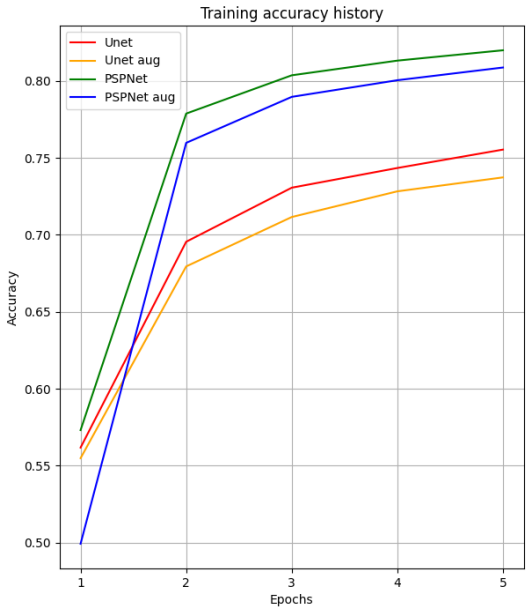
Class void : 37.8%

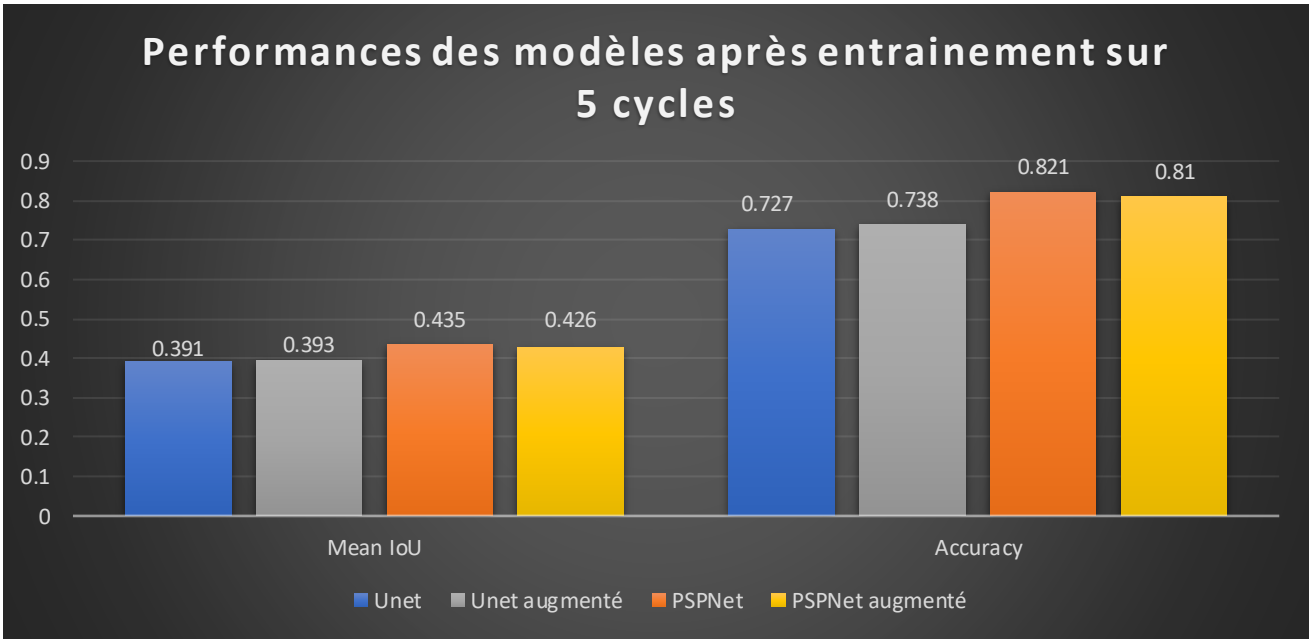
De la même façon que pour l'entrainement sans data augmentation, on voit de nettes améliorations dès le second cycle puis qui ralentissent. L'entrainement s'est bien passé et les performances sont très correctes.

Analyse des résultats finaux

L'entrainement s'est fait sur 5 cycles donc toutes les images d'entrainement ont été vues par le modèle 5 fois durant l'entrainement. On peut regarder pour chaque modèle individuellement sa performance au long de l'entrainement ainsi que les performances des modèles ensembles durant l'entrainement.







Les résultats sont satisfaisants et cohérents avec nos hypothèses.

En effet, le modèles U-net est moins performant que PSPNet mais reste à des performances assez impressionnantes surtout avec seulement 5 cycles d'entrainement (sur environ 800 photos).

Nous n'avons pas rencontré d'overfitting durant l'entrainement car notre accuracy reste croissante sur notre jeu de validation, ce qui est un bon point.

Au final, on se rend compte que l'augmentation des données est situationnelle car elle a été bénéfique pour le loss mais pas pour l'accuracy.

Les performances me paraissent convenables pour être utilisées pour notre projet.

Pistes d'amélioration

Afin d'obtenir des résultats meilleurs, voici quelques pistes d'améliorations qui pourraient être mis en place :

- Mettre en place un système de métriques personnalisées. Nous avions voulu utiliser MeanIoU comme métrique de performance mais ça a été difficile à mettre en place car la librairie utilisée ne le permettait pas. Au final, à force de parcourir les fichiers de la librairie, j'aurai été capable de m'en affranchir et de recréer mes fonctions personnalisées mais ça aurait pris trop de temps.
- Implémenter un modèle se basant sur une architecture de transformer aurait pu être une alternative prometteuse. Plus de détails ici.
- Effectuer un entrainement plus long, avec plus de cycles d'entrainement afin de déterminer le meilleur modèle
- Effectuer des entrainements sur plusieurs taux de data augmentation. Pour l'instant, nous avons modifiés 10% des images présentes dans le jeu de données mais il aurait été intéressant de faire varier ce taux pour déterminer la meilleure proportion d'image à augmenter. De plus, on a bien vu que selon le modèle, l'augmentation à 10% pouvait être bénéfique ou non.
- Faire des recherches sur la co-segmentation qui est un type de segmentation sur vidéo, qui aurait permis une cohérence entre les frames analysées.