Mémoire technique
Projet 9 : segmentation d'images

Parcours OpenClassrooms Ingénieur IA - Cédric Dietzi

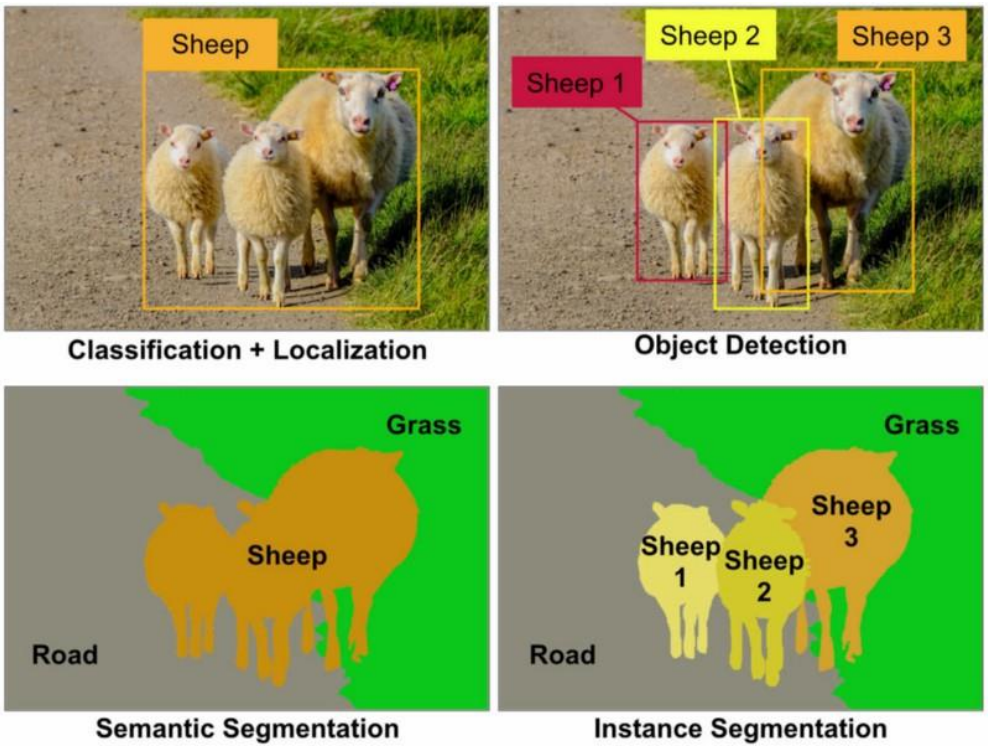
L'objectif du projet est de développé un prototype de segmentation sémantique d'images de scènes urbaines.

1 Etat de l'art

1.1 Segmentation sémantique d'images

La segmentation sémantique est une tâche de vision par ordinateur. Elle est un des outils de compréhension d'une scène avec la classification d'images, la détection d'objet et la segmentation d'instances :

- La **classification d'images** associe une classe à une image,
- **La détection d'objets** identifie la présence d'objets dans l'image, les associe à une classe et les localise grossièrement dans un cadre. Elle permet de dénombrer les objets présents dans l'image.
- La **segmentation sémantique** associe chaque pixel de l'image à une classe. Elle localise plus finement que la détection les contours des classes mais ne permet plus de compter les objets.
- La **segmentation d'instances** a les avantages de la détection d'objets et de la segmentation sémantique. Elle associe chaque pixel à une classe mais aussi une sous-classe en différenciant les objets d'une même classe les uns des autres.



Différences entre classification, détection, segmentation sémantique et segmentation d'instances (image par Aurélien Géron).

1.2 Mesures de performance associées à la segmentation sémantique

Note : par soucis de clarté, nous désignons les mesures par leur terme anglais.

Les mesures classiques utilisées pour la segmentation sémantique sont l'accuracy, l'intersection-over-union (IoU), le coefficient Dice (Dice). Nous expliquons ces trois mesures ci-dessous.

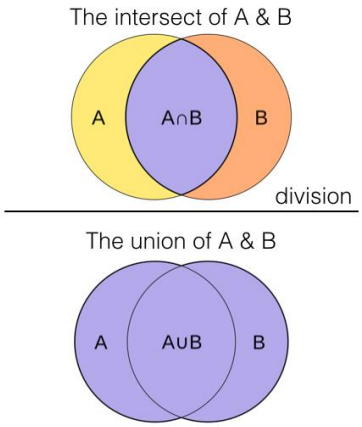
L'accuracy est le ratio du nombre de pixels bien labellisés sur le nombre de pixels total de l'image. L'accuracy est une mesure globale de performance qui peut masquer des performances locales médiocres. On peut par exemple obtenir une très bonne accuracy alors que des classes de petite taille sur l'image sont intégralement mal labellisées. L'intersection-over-union (IoU) et le coefficient Dice répondent à ce problème.

1.2.1 Mesures IoU et Dice

Pour comprendre ces mesures, cherchons d'abord à définir la performance de la segmentation pour une classe donnée K. Par exemple, on cherche à savoir si, dans une image de trafic routier, la segmentation identifie correctement les panneaux de signalisation dont les pixels sont labellisés K.

A est l'ensemble des pixels de l'image dont le label réel est K. B est l'ensemble des pixels de l'image dont le label prédit est K.

Les mesures IoU et Dice s'expriment alors comme suit :



$IoU = \frac{|A \cap B|}{|A \cup B|}$, IoU est le ratio du nombre de pixels correctement prédits et du nombre de pixels de label réel ou prédit K.

$Dice = \frac{2*|A \cap B|}{|A|+|B|}$, le coefficient Dice est le ratio du nombre de pixels correctement prédits et de la somme des pixels de label réel K d'une part et de label prédit K d'autre part. Le facteur 2 compense le double comptage de l'intersection au dénominateur.

Nous venons de décrire les mesures pour une classe donnée K. Pour connaitre la performance de la segmentation d'une image par rapport à l'ensemble des classes, on moyenne les mesures individuelles. On peut éventuellement raffiner en calculant une moyenne pondérée dans laquelle les classes ont plus ou moins d'importance en fonction de leur criticité métier.

Les deux mesures paraissent très similaires. Leur différence apparait lorsqu'on évalue la performance de la segmentation sur un ensemble d'images plutôt qu'une image isolée. IoU pénalise plus fortement les mauvaises prédictions. Par conséquent, IoU aura tendance à favoriser un algorithme qui limite les pires cas quitte à dégrader la moyenne, à l'inverse de Dice.

1.3 Fonctions de perte

Les fonctions de perte classiques utilisées sont l'entropie croisée, (1 – IoU) et (1 – Dice).

L'entropie croisée est une (pseudo) mesure de distance entre la distribution statistique des labels réels et la distribution statistique des labels prédits.

1.4 Principes algorithmiques

(Inspiré de https://divamgupta.com/)

Les algorithmes de vision par ordinateur reposent sur des réseaux de neurones convolutifs. Lors de la phase de prédiction, ces modèles identifient dans l'image, à différentes échelles, des motifs typiques qui sont ensuite interprétés comme caractéristiques de telle ou telle classe.

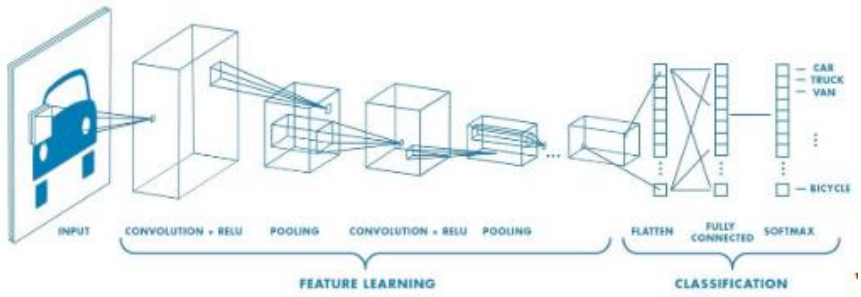
La phase d'apprentissage a pour objectif de déterminer les motifs les plus pertinents pour discriminer les classes. Concrètement, ces motifs sont les filtres utilisés dans les convolutions.

1.4.1 Réseau convolutif pour la classification d'image : architecture encodeur - perceptron

Un réseau convolutif pour classifier des images est composé d'une succession de blocs typiquement construits avec une couche de convolution, puis d'activation, de normalisation et enfin de pooling.

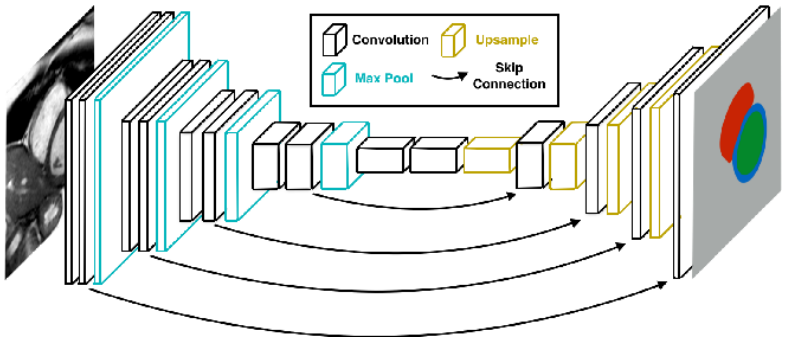
Les couches de convolution en amont détectent des motifs fins dans l'image et les couches en aval des motifs plus globaux.

Un dernier bloc dense (un bloc perceptron) suivi d'une fonction softmax permet de déterminer la probabilité de chaque classe.



1.4.2 Réseau convolutif pour la segmentation : architecture encodeur - decodeur

Pour la segmentation, on ne peut pas utiliser de perceptron qui détruirait l'information de position des labels dans l'image (lors de l'étape 'Flatten' sur le schéma ci-dessus). On utilise plutôt un réseau convolutif où les couches de pooling sont remplacées par des interpolateurs pour reconstituer une sortie qui a la dimension de l'entrée.



Pour bénéficier de l'information calculée par les couches de convolution en amont du réseau, l'architecture prévoit des connexions directes entre l'encodeur et le décodeur appelées 'Skip Connections'.

1.4.3 Transfer learning en segmentation sémantique d'image

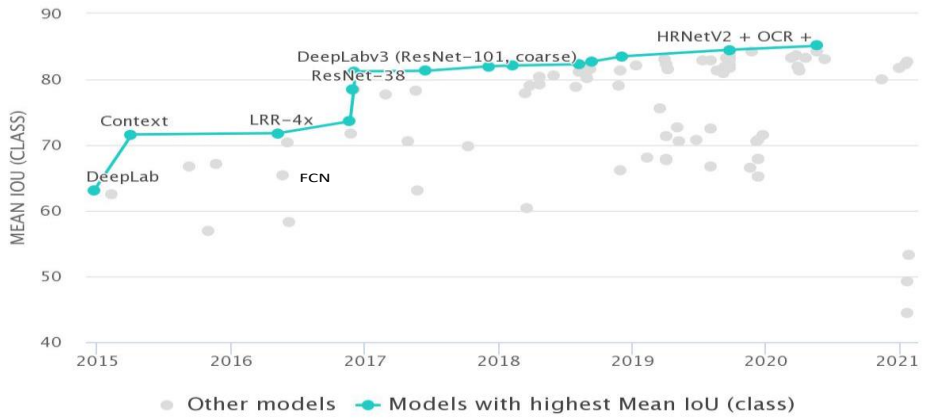
On peut utiliser des encodeurs entrainés sur des tâches de classification comme ImageNet pour faire du transfer-learning.

1.5 Algorithmes existants

Notre recherche sur l'état de l'art s'appuie sur le site paperswithcode.com. Ce site centralise, pour chaque tâche de machine learning et chaque jeu de données de référence, un benchmark des algorithmes, les papiers décrivant les algorithmes et le code permettant de les reproduire.

paperswithcode.com propose par exemple un benchmark pour la segmentation sémantique appliquée au jeu de données CityScapes sur lequel nous travaillons.

1.5.1 Courbe SOTA (State Of The Art)



En 2015 les meilleures performances sont obtenues par l'algorithme DeepLab de Google. DeepLab améliore la segmentation en utilisant un perceptron CRF (Conditional Random Field) en bout de chaine. Le CRF prend en compte dans sa prédiction le contexte, c'est-à-dire les autres prédictions.

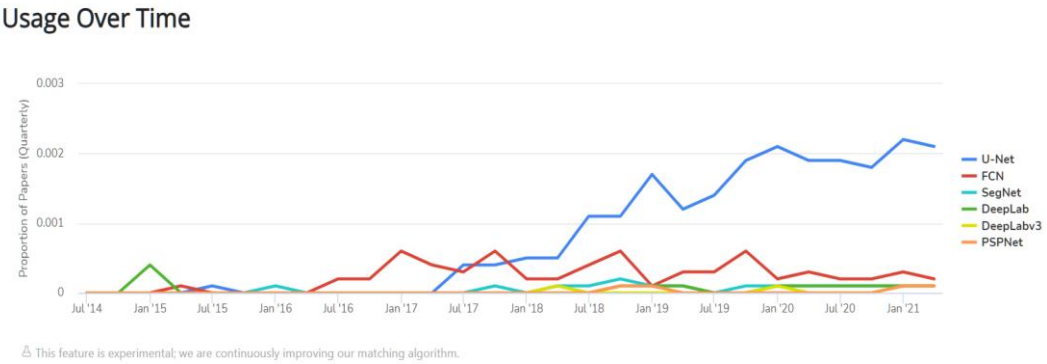
En 2017, le modèle PSPnet permet d'améliorer le benchmark de 7.6 points. L'architecture s'appuie sur un réseau ResNet-101 et un module de 'pooling pyramidale' qui aide le modèle à prendre en compte le contexte global de l'image dans la classification d'un pixel.

Après 2017, le benchmark est amélioré incrémentalement en particulier par des évolutions de DeepLab.

Note sur ResNet : avant 2015, la profondeur croissante des réseaux de neurones avait permis des résultats toujours meilleurs. Cependant ces réseaux profonds se heurtaient à un phénomène de saturation des performance non dû à un surapprentissage. Par ailleurs, les erreurs d'apprentissage (le biais) s'accentuait même avec l'ajout de couches. En 2015, l'implémentation proposée par ResNet permet de dépasser cette limite et de développer des solutions de machine learning en s'appuyant sur des réseaux très profonds.

La courbe SOTA nous donne un benchmark mais la course à la performance sur les benchmarks peut donner lieu à des algorithmes trop spécifiques / complexes. Par conséquent, pour sélectionner notre algorithme, nous proposons de vérifier quels sont ceux qui sont le plus souvent mis en œuvre.

1.5.2 Taux d'utilisation des algorithmes dans la littérature



paperswithcode.com fournit un graphique de l'évolution de l'utilisation des algorithmes de segmentation sémantique dans le temps.

L'algorithme U-Net se dégage nettement du lot. Son absence du benchmark CityScapes plus haut est peut-être liée à son application originelle en biomédecine.

FCN est un des premiers modèles proposés pour la segmentation sémantique. La performance d'une des implémentations de FCN est indiquée sur le benchmark ci-dessus et tombe largement sous celui-ci.

Segnet est une architecture de type encodeur-décodeur basée sur un réseau convolutif VGG-16. Sa particularité est d'utiliser des informations de pooling pour opérer les interpolations sur la partie encodeur.

On retrouve DeepLab et PSPNet parmi les algorithmes les plus utilisés.

1.6 Implémentations testées

On retient les implémentations suivantes :

Modèle	**Transfer learning**	**Code source de base**
Unet	Non	https://keras.io/examples/vision/oxford_pets_image_segmentation
VGG16 + Unet	ImageNet	https://github.com/divamgupta/image-segmentation-keras
Resnet + Unet	ImageNet	https://github.com/divamgupta/image-segmentation-keras
Resnet + Pspnet	ImageNet	https://github.com/divamgupta/image-segmentation-keras

L'implémentation de type Unet pourra être utilisée comme modèle de base. Nous testerons ensuite l'effet d'un tranfer-learning en utilisant comme encodeurs deux réseaux convolutifs classiques, VGG16 puis le réseau ResNet. Enfin nous testerons Resnet associé à Pspnet.

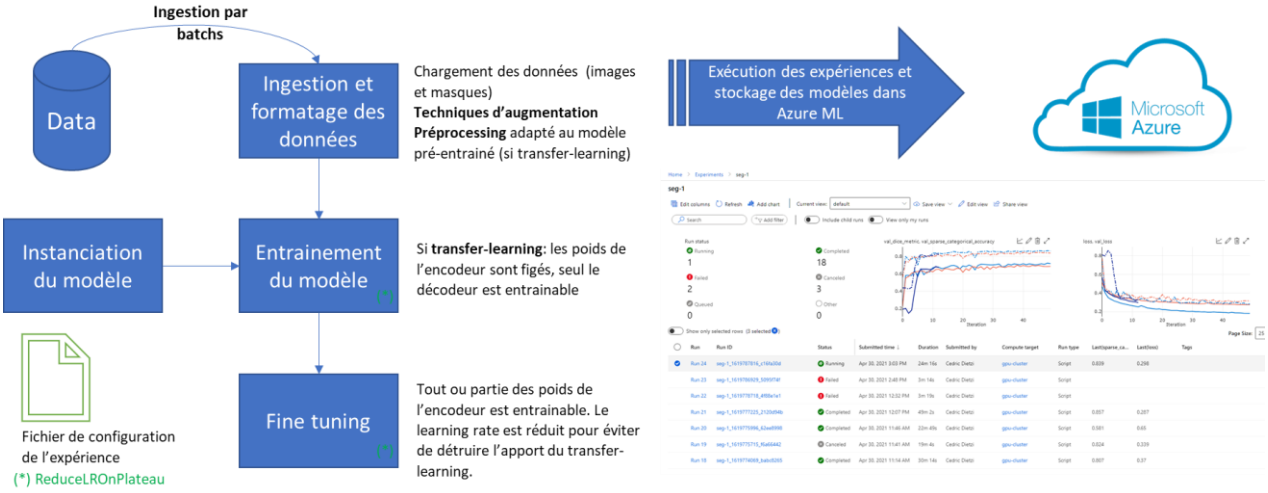
2 Entrainement des modèles

2.1 Jeu de données: CityScapes

Le jeu de données utilisée est 'CityScapes'. Il contient 5000 images de scènes urbaines (50 villes allemandes) annotées avec 30 classes et 8 catégories.

2.2 Processus d'entrainement

Nous décrivons ci-dessous les principales étapes de l'entrainement :



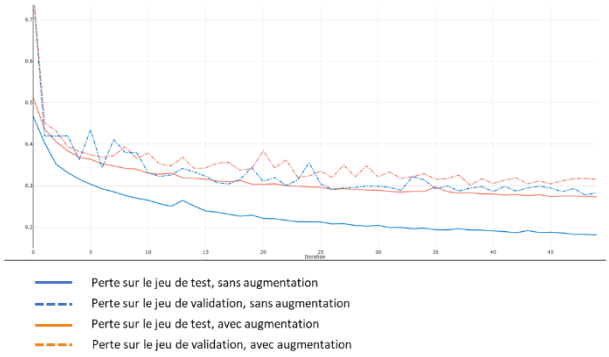
Note : le code d'entrainement mais aussi d'inférence et le code de déploiement dans l'application web a d'abord été testé sur des jeux de données réduits en local avant d'être déployé sur Azure. Cette approche de test est disponible dans un unique notebook. Celui-ci permet de suivre et comprendre linéairement toutes les étapes du processus mis en œuvre du formatage des données à la mise en production de l'application web.

2.3 Effet des techniques d'augmentation de données

Les techniques d'augmentation de données permettent de réduire le biais du modèle. L'enrichissement du nombre d'exemples tempère la tendance du modèle au 'par cœur' et limite le sur-apprentissage.

Nous avons testé différentes combinaisons d'augmentation. Les techniques d'augmentation utilisées finalement sont la rotation (+/-15° max), des modifications de couleur (contraste, luminosité, saturation) et la symétrie horizontale.

Impact de l'augmentation de données sur le modèle U-Net



Dans le cas où les techniques d'augmentation sont appliquées systématiquement, on constate bien que le surapprentissage diminue. On passe des courbes bleues à oranges sur le graphique avec un espace test-validation réduit.

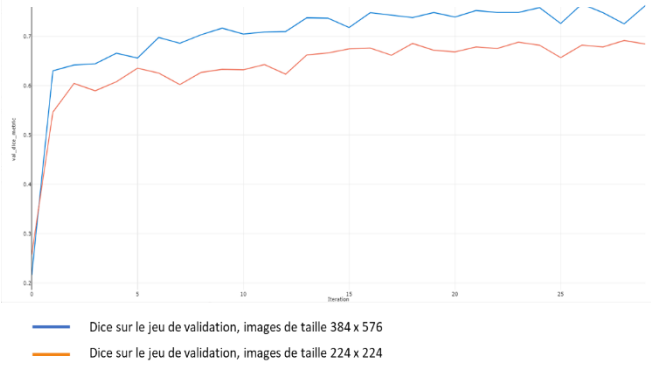
Toutefois, le biais augmente suffisamment pour dégrader la performance sur le jeu de validation.

Nous utilisons donc ces techniques de façon plus légère. Chaque technique a 35% de chance de s'appliquer sauf la symétrie que nous appliquons avec une probabilité de 50%.

2.4 Effet de la taille de l'image en entrée de modèle

Nous avons pris soin de comparer les modèles en prenant des images de tailles identiques chaque fois que c'était possible. En effet, la taille de l'image influence largement la précision des résultats obtenus.

Impact de la taille de l'image sur le modèle U-Net



Cet exemple illustre l'impact de la taille de l'image sur les performances du modèle. Ici le seul paramètre qui diffère entre les deux expériences est la taille de l'image.

Au bout de 30 epochs, la différence de performance Dice est de ~ 6-8 %.

Le tableau ci-dessous résume les tailles des images en entrée des modèles testés :

Modèle	**Taille de l'image en entrée du modèle**
Type Unet	224 x 244
VGG16 + Unet	224 x 244
Resnet + Unet	224 x 244
Resnet + Pspnet	384 x 576 (contrainte de multiplicité de 192)

2.5 Synthèse des résultats

Le jeu d'entrainement comporte 2975 couples image-masque et le jeu de validation comporte 500 couples image-masque.

Un batch comporte 32 exemples et les entrainements sont réalisés sur 50 epochs au total. Lorsque nous utilisons du transfer-learning, 15 epochs sur les 50 au total sont dédiées au fine-tuning.

Pour le modèle Resnet + Pspnet, nous sommes passés à 8 exemples par batch pour libérer des ressources mémoire nécessaires à la phase de fine-tuning.

2.5.1 Fonction de perte entropie croisée

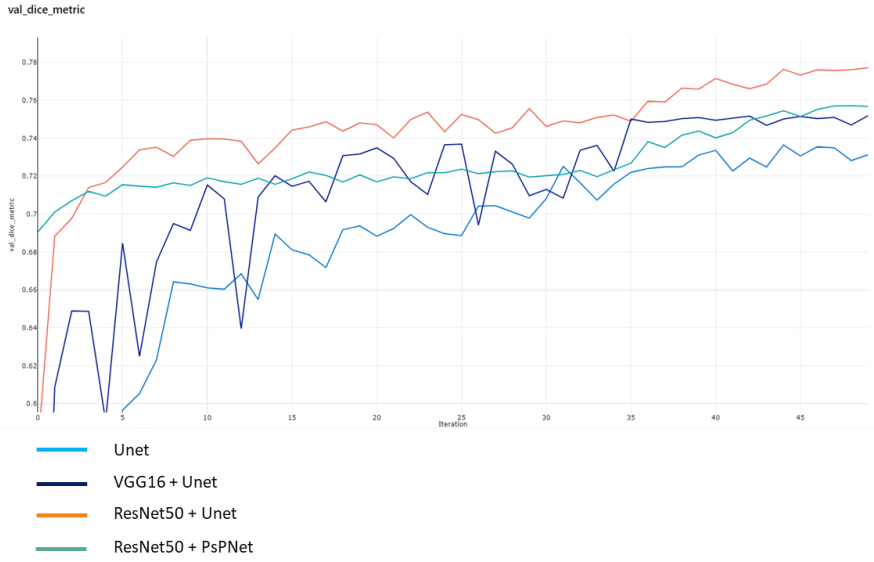
Modèle	Coefficient Dice	Pixel accuracy	Temps d'entrainement
Type Unet	0.653	0.866	1h16
VGG16 + Unet	0.717	0.888	1h28
Resnet + Unet	0.721	0.886	1h30
Resnet + Pspnet	0.677	0.872	3h05

2.5.2 Fonction de perte (1-Dice)

Modèle	Coefficient Dice	Pixel accuracy	Temps d'entrainement
Type Unet	0.736	0.860	1h12
VGG16 + Unet	0.752	0.868	1h17
Resnet + Unet	0.777	0.884	1h36
Resnet + Pspnet	0.757	0.871	3h28

Les résultats confirment que pour optimiser le coefficient Dice il vaut mieux utiliser la fonction de perte (1 – Dice).

2.5.3 Convergence des modèles



Note : sur les modèles avec transfer-learning, on remarque bien l'effet du fine-tuning qui démarrage à l'epoch 35.

3 Modèle retenu: ResNet50 + Unet

Le modèle retenu pour la mise en production est le modèle ResNet50 + Unet qui présente le meilleur compromis performance / temps d'entrainement.

Architecture ResNet

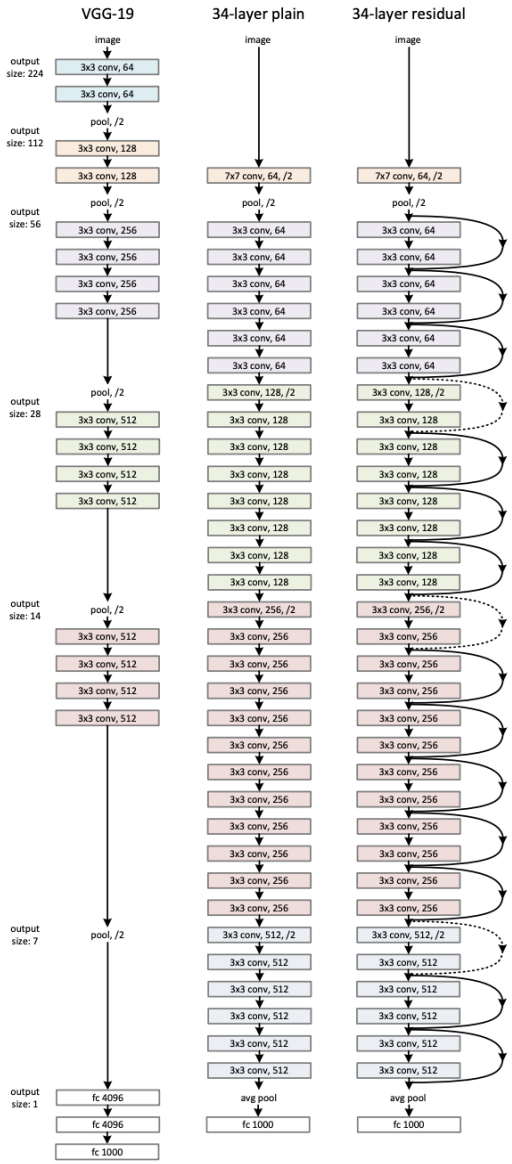
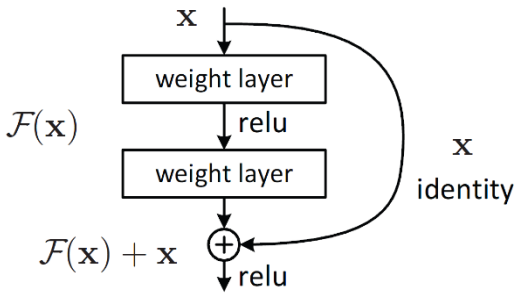


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.



Bloc résiduel d'un modèle ResNet

L'encodeur ResNet-50 est une architecture de réseau convolutif ayant permis l'empilement d'un très grand nombre de couches sans qu'il y ait saturation de la performance comme c'était le cas des réseaux de neurones très profonds avant l'introduction de cette architecture.

Le modèle est constitué de blocs résiduels. Chacun des blocs du modèle a la particularité d'apprendre une fonction résiduelle F(x) par rapport à une fonction de référence x en entrée. L'intuition de ce modèle est qu'il est plus facile d'optimiser la fonction résiduelle qu'une fonction quelconque (sans référence à l'entrée).

Le décodeur correspond à la partie expansive du modèle U-Net, encadrée en bleu sur le schéma ci-contre.

Chacun de ses blocs est constitué d'un interpolateur qui agrandit la 'feature map', d'une 'up-convolution' qui divise par deux le nombre de features et une concaténation avec la 'feature map' appropriée en provenance de l'encodeur (correspond aux flèches grises mais qui proviendrait dans notre cas du ResNet50).

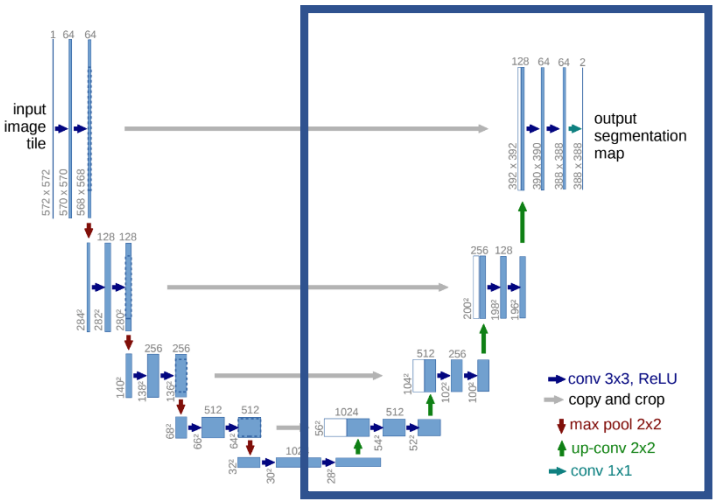


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

4 Pistes d'amélioration

Nous proposons les pistes d'amélioration suivantes :

Fonction de perte	=> Explorer l'impact d'une pondération des différentes classes dans le calcul du Dice moyen. Par exemple donner plus de poids aux éléments critiques d'un point de vue métier comme les piétons ou les panneaux de signalisation.
Fonction de perte	=> Tester une fonction d'entropie croisée pondérée par l'importance des classes.
ResNet50-PSPNet	Ce modèle dans sa version pré-entrainé donne des résultats décevant par rapport au benchmark. => Explorer la littérature pour comprendre et écarter la possibilité d'un bug dans le code.
Batch size	Lors de nos essais-erreurs, nous avons constaté qu'un batch de 32 exemples donne de meilleures performances qu'un batch de 8 exemples. => Explorer des batchs plus grands.
Learning rate	Nos simulations utilisent une réduction automatique du learning rate en cas de stagnation de la performance. Cependant nous n'avons pas monitoré son évolution. => Monitorer et ajuster la stratégie de 'learning rate decay'.
Taille des images	Comme mentionné plus haut, la taille des images impacte la performance. => Etablir une relation entre la taille des images et le couple performance / temps d'entrainement, d'inférence.
Fine-tuning ResNet50-Unet et ResNet50-PspNet	Le fine-tuning sur les modèles ResNet50-Unet et ResNet50-PspNet s'applique à l'ensemble des poids des modèles. Nous avons fait ce choix car nous ne maitrisons pas l'impact d'un fine-tuning partiel du modèle. => Clarifier ce point.
Fine-tuning	=> Explorer des fine-tuning sur différentes profondeurs des encodeurs.