Note Technique : Segmentation Sémantique

Table des matières

1. État de l'art : segmentation d'images

La segmentation d'images est le processus de division d'une image en plusieurs segments. Dans ce processus, chaque pixel de l'image est associé à un type d'objet. Il existe deux grands types de segmentation d'images : la segmentation sémantique et la segmentation d'instance.

Dans la segmentation sémantique, tous les objets du même type sont marqués à l'aide d'une étiquette de classe tandis que dans la segmentation d'instance, les objets similaires reçoivent leurs propres étiquettes distinctes.

La segmentation sémantique intervient dans de nombreuses applications telles que la conduite autonome, l'imagerie médicale et les contrôles industriels. Par exemple, un véhicule autonome doit identifier des véhicules, des piétons, des panneaux de signalisation, des trottoirs et autres éléments de l'environnement routier. Des exemples d'utilisation de la segmentation sémantique sont :

- Conduite autonome : pour identifier un parcours conduisible pour les véhicules en distinguant la route des obstacles tels que les piétons, trottoirs, poteaux et autres véhicules.
- Contrôles industriels : pour détecter les défauts dans des matériaux, comme le contrôle des composants électroniques.

- Imagerie satellite : pour identifier les montagnes, les rivières, les déserts et autres types de terrains.
- Imagerie médicale : pour analyser et détecter les anomalies cancéreuses dans les cellules.
- Vision robotique : pour identifier les objets et le terrain et s'y déplacer.

Le processus d'apprentissage d'un réseau de segmentation sémantique suit les étapes suivantes : analyser un ensemble d'images aux pixels étiquetés, créer un réseau de segmentation sémantique, entrainer le réseau à classifier des images selon des catégories de pixels et évaluer la précision du réseau.

L'architecture de base de la segmentation d'images se compose d'un encodeur et d'un décodeur. L'encodeur, basé sur une architecture CNN, extrait les caractéristiques de l'image à travers des filtres. Le décodeur, implémentation inverse d'un CNN, est chargé de générer la sortie finale qui est généralement un masque de segmentation contenant le contour de l'objet. Le processus de suréchantillonnage est effectué le même nombre de fois que le processus de sous-échantillonnage pour s'assurer que l'image finale possède les mêmes dimensions que l'image d'entrée. La plupart des architectures ont cette architecture ou une variante de celle-ci.

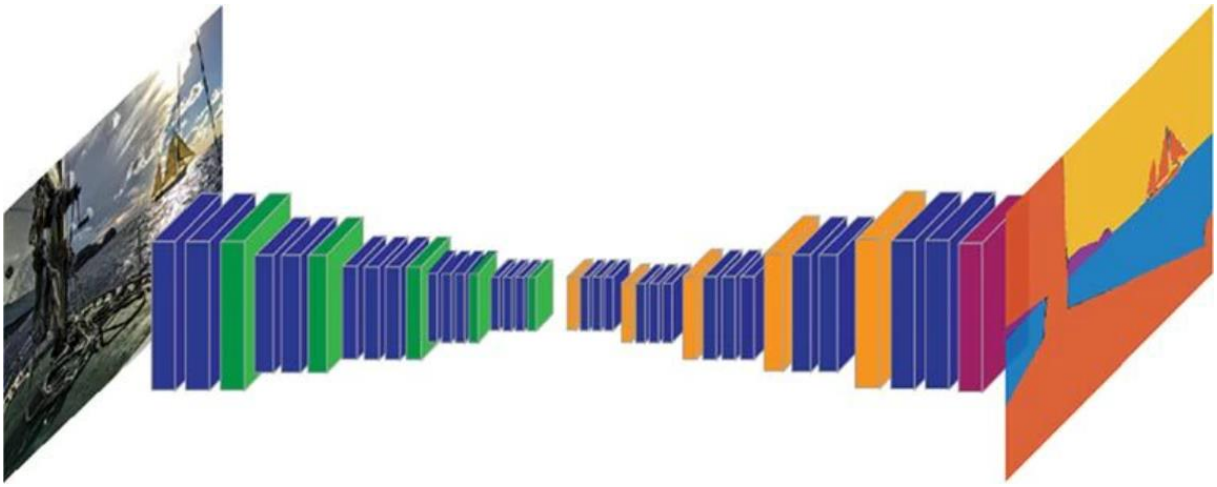

Figure 1 : Réseau CNN : sous-échantillonnage avec de pooling (en vert) et suréchantillonnage avec un nombre égal de couches de unpooling (en orange) @MATLAB.

1.1. U-NET

U-NET est un réseau de neurones convolutifs développé à l'origine pour segmenter des images biomédicales. Lorsqu'elle est visualisée, son architecture ressemble à la lettre U, d'où son nom : U-NET. Son architecture est composée de deux parties, la partie gauche, le chemin de contraction et la partie droite, le chemin expansif. Le but du chemin contractuel est de capturer le contexte tandis que le rôle du chemin expansif est d'aider à une localisation précise.

Le chemin contractuel est composé de couches de convolutions trois par trois. Les convolutions sont suivies d'une fonction ReLU pour éviter la saturation des unités et d'autre part, de diminuer le phénomène d'évanescence du gradient et d'un couche de pooling deux par deux, pour rééchantillonner les données.

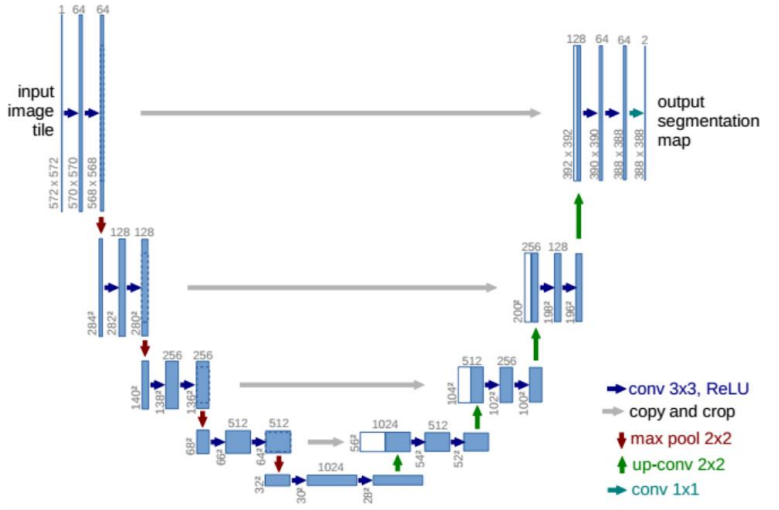

Figure 2 : Architecture U-NET.

1.2. PSPNet

H. Zhao et al. (2016) ont développé le Pyramid Scene Parsing Network (PSPNet) pour mieux apprendre la représentation globale du contexte d'une scène. Les motifs sont extraits de l'image d'entrée à l'aide d'un extracteur de caractéristiques (ResNet K. He et al. (2015)) avec une stratégie de réseau dilaté. Les cartes de caractéristiques alimentent un module de pooling pyramidal pour distinguer les modèles avec différentes échelles. Elles sont regroupées avec quatre échelles différentes correspondant chacune à un niveau de pyramide et traitées par une couche convolutive 1x1 pour réduire leurs dimensions. De cette façon, chaque niveau de pyramide analyse des sous-régions de l'image avec un emplacement différent. Les sorties des niveaux de la pyramide sont suréchantillonnées et concaténées aux cartes de caractéristiques initiales pour finalement contenir les informations contextuelles locales et globales. Ensuite, ils sont traités par une couche convolutive pour générer les prédictions au niveau des pixels.

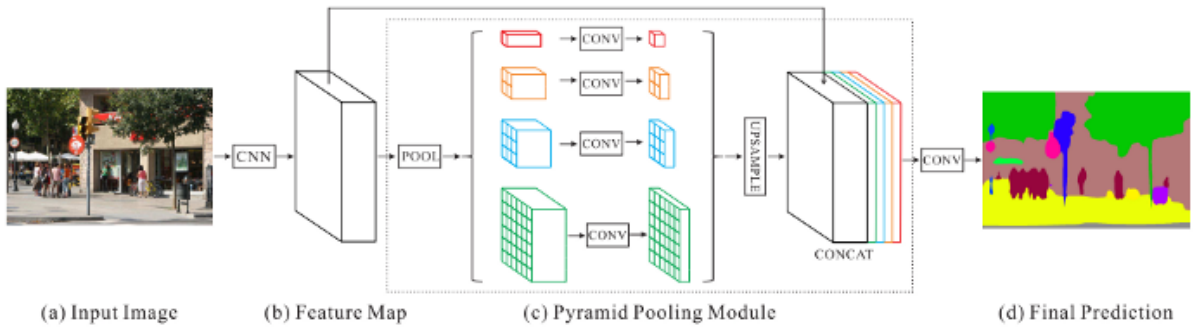

Figure 3 : Architecture PSPNet.

1.3. FPN

Le Feature Pyramid Network (FPN) a été développé par T.-Y. Lin et al (2016) et il est utilisé dans les cadres de détection d'objets ou de segmentation d'images. Son architecture est composée d'une voie ascendante, d'une voie descendante et de connexions latérales afin de joindre des éléments à basse et haute résolution.

La voie ascendante est un réseau convolutif typique pour l'extraction de caractéristique. La sortie de chaque module de convolution est utilisée dans la voie descendante.

La voie descendante consiste à suréchantillonner les dernières cartes de caractéristiques avec unpooling tout en les enrichissant de cartes de caractéristiques du même stade de la voie ascendante à l'aide de connexions latérales. Ces connexions consistent à fusionner les cartes de caractéristiques de la voie ascendante traitées avec une convolution 1x1 (pour réduire leurs dimensions) avec les cartes de caractéristiques de la voie descendante. Les cartes de caractéristiques concaténées sont ensuite traitées par une convolution 3x3 pour produire la sortie de la scène. Enfin, chaque étape de la voie descendante génère une prédiction pour détecter un objet.

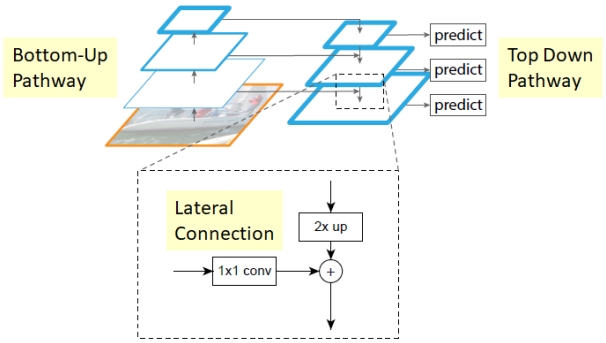

Figure 4 : Architecture FPN.

1.4. Les métriques et les fonctions loss

Une métrique d'évaluation quantifie la performance d'un modèle prédictif, le choix de la bonne métrique (ou bonnes métriques) est crucial lors de l'évaluation des modèles et la qualité d'un modèle dépend directement de la métrique utilisée pour l'évaluer. Il en existe plusieurs permettant d'évaluer un modèle de segmentation sémantique : Pixel Accuracy, Dice Coefficient (F1 Score), Intersection-Over-Union (IoU, Jaccard Index) et d'autres encore.

Les fonctions de perte sont utilisées pour déterminer l'erreur (alias "loss") entre la sortie de nos algorithmes et la valeur cible donnée. Elle exprime à quel point notre sortie calculée est loin de la cible. Il en existe plusieurs : Categorical Cross-Entropy (CCE), Dice Loss, JaccardLoss, Categorical Focal Loss, CCE + Dice Loss et bien d'autres encore.

1.4.1. Pixel Accuracy

Le pixel acccuracy correspond au pourcentage de pixels de l'image correctement classifié. Un pixel accuracy élevée n'implique pas toujours une capacité de segmentation supérieure. Ce problème est appelé déséquilibre de classe. Lorsque les classes sont extrêmement déséquilibrées, cela signifie qu'une classe ou certaines classes dominent l'image, tandis que d'autres classes ne constituent qu'une petite partie de l'image. Ce déséquilibre de classe ne peut pas être ignoré, car est répandu dans de nombreux ensembles de données du monde réel. Pour illustré ce problème, prenons la figure ci-dessous. Bien que le pixel accuracy soit de 95%, le modèle renvoie une prédiction mauvaise non utilisable.

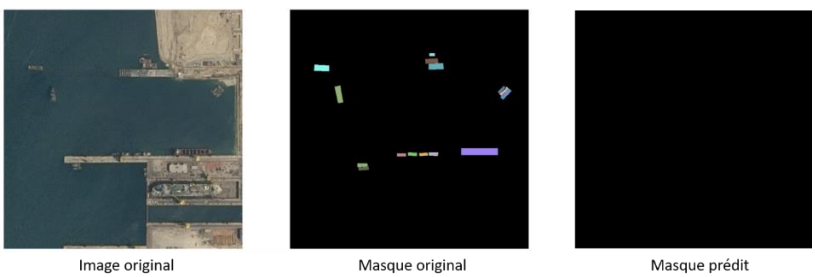

Figure 5 : Illustration du problème de classe déséquilibrée.

1.4.2. Intersection-Over-Union (IoU, Jaccard Index)

L'index Jaccard est l'une des métriques les plus utilisées dans la segmentation sémantique. C'est la zone de chevauchement entre la segmentation prédite et la vérité terrain divisée par la zone d'union entre la segmentation prédite et la vérité terrain. Prenant la figure de la section Pixel Accuracy et calculons l'IoU. Pour ce faire supposons que la surface totale de l'image est de 100 pixels et que 5 de ces pixels représentent les navires. Après avoir fait les calculs, l'IoU n'est que de 47,5%. Ce qui est inférieur à 95% obtenu par le pixel accuracy. IoU est donc une meilleure indication du succès de la segmentation.

$$IoU_{navire} = \frac{Intersection\ navires\ prédit\ et\ vérité\ terrain}{Union\ navires\ prédit\ et\ vérité\ terrain} = \frac{0}{(5+0)-0} = 0\%$$

$$IoU_{background} = \frac{Intersection\ background\ prédit\ et\ vérité\ terrain}{Union\ background\ prédit\ et\ vérité\ terrain} = \frac{95}{(95+100)-95} = 95\%$$

$$Mean\ IoU = \frac{IoU_{navire} + IoU_{background}}{2} = \frac{0+95}{2} = 47{,}5\%$$

1.4.3. Dice coefficient (F1 Score)

Le coefficient Dice est deux fois la zone d'intersection divisée par le nombre total de pixels dans la segmentation prédite et la vérité terrain.

$$Dice_{navire} = \frac{2\ \times Intersection\ navires\ prédit\ et\ vérité\ terrain}{Nombre\ total\ de\ pixels\ prédit\ et\ vérité\ terrain} = \frac{0}{200} = 0\%$$

$$Dice_{background} = \frac{2 \times Intersection\ background\ prédit\ et\ vérité\ terrain}{Nombre\ total\ de\ pixels\ prédit\ et\ vérité\ terrain} = \frac{2 \times 95}{200} = 95\%$$

$$Dice = \frac{Dice_{navire} + Dice_{background}}{2} = \frac{0+95}{2} = 47{,}5\%$$

1.5. Les backbones

Le « backbone » fait référence à l'extracteur de caractéristiques pour créer un modèle de segmentation. Cet extracteur de caractéristiques est utilisé pour coder l'entrée du réseau dans une certaine représentation de caractéristiques. Quelques exemples de « backbones » sont : VGG-16, Resnet50 et InceptionV3.

Backbone VGG-16 : Le réseau VGG-16 est composé de treize couches convolutives et de trois couches entièrement connectées avec activation ReLU. Le noyau de convolution 3x3, plus petit que d'autres noyau de convolution (exemple, AlexNet), permet de consommer moins de calcul. VGG utilise également un noyau de convolution 1x1 pour effectuer les traitements non linéaire via ReLU. Ce rend le modèle plus puissant en termes de capacité d'ajustement.

Backbone ResNet50 : Le ResNet introduit des connexions courtes aux réseaux de neurones, atténuant ainsi le problème de la disparition du gradient tout en obtenant des structures de réseau beaucoup plus profondes. Pendant la fonctionnalité procédure d'extraction, les connexions courtes permettent différentes combinaisons d'opérateurs convolutifs, résultant en un grand nombre d'échelles de caractéristiques équivalentes.

Backbone InceptionV3 : Avant la création du réseau Inception, les CNN les plus populaires empilaient des couches de convolution de plus en plus profondes, dans l'espoir d'obtenir de meilleures

performances. Le réseau Inception, en revanche, était complexe et utilise beaucoup d'astuces pour améliorer les performances à la fois en termes de vitesse et de précision. Son évolution constante a conduit à la création de plusieurs versions du réseau. Le réseau Inception est censé résoudre le problème suivant. Il y a une énorme variation dans l'emplacement des informations, le choix de la bonne taille de noyau de convolution devient difficile. Un noyau plus grand est préférable pour les informations distribuées plus globalement, et un noyau plus petit est préférable pour les informations distribuées plus localement. D'autre part, les réseaux très profonds sont sujets au surapprentissage et il est difficile de transmettre les mises à jour du gradient à travers l'ensemble du réseau. L'empilement des grandes opérations de convolution est coûteux en temps de calcul. Ainsi, le reseau Inception fait fonctionner des filtres de plusieurs tailles au même niveau. Le réseau devient plus large plutôt que profond et en ajoutant une couche de convolution 1x1 pour réduire la dimension, le réseau est peu onéreux en temps de calcul.

2. Modèle retenu

Différentes combinaisons d'architectures de modèle, de backbone, de fonction loss, de métrique et d'augmentation de données ont été testées. Ci-dessous se trouve un tableau récapitulatif des différentes combinaisons testés.

Modèle	Backbone	Fonction loss	Métriques	Augmentation de données
1) U-NET modèle de base		CCE	Accuracy Coefficient Dice IoU index	Non
2) U-NET		CCE	Accuracy Coefficient Dice IoU index	Oui
3) U-NET		Dice Loss	Accuracy Coefficient Dice IoU index	Non
4) U-NET		Categorical Focal Loss	Accuracy Coefficient Dice IoU index	Non
5) U-NET (décodeur)	VGG-16	Dice Loss	Accuracy Coefficient Dice IoU index	Non
6) U-NET (encodeur + décodeur)	VGG-16	Dice Loss	Accuracy Coefficient Dice IoU index	Non
7) U-NET (décodeur)	VGG-16	Jaccard Loss	Accuracy Coefficient Dice IoU index	Non
8) U-NET (décodeur)	VGG-16	CCE + Dice Loss	Accuracy Coefficient Dice IoU index	Non
9) PSPNet	Resnet101	Dice Loss	Accuracy Coefficient Dice IoU index	Non
10) FPN	Resnet50	Dice Loss	Accuracy Coefficient Dice IoU index	Non

11) FPN	InceptionV3	Dice Loss	Accuracy Coefficient Dice IoU index	Non

2.1. U-net VGG16

U-Net est une architecture de segmentation sémantique. Il se compose d'un chemin contractuel et d'un chemin expansif. Le chemin contractuel suit l'architecture typique d'un réseau convolutif. Il consiste en l'application répétée de deux convolutions 3x3, chacune suivie d'une fonction d'activation (ReLU) et d'une couche pooling 2x2. À chaque étape de sous-échantillonnage, le nombre de canaux de fonction est doublé. Chaque étape du chemin expansif consiste en un suréchantillonnage de la carte des caractéristiques suivi d'une convolution 2x2 qui divise par deux le nombre de canaux de caractéristiques, une concaténation avec la carte des caractéristiques rognée en conséquence du chemin de contraction, et deux 3x3 convolutions, chacune suivie d'une ReLU. Le recadrage est nécessaire en raison de la perte de pixels de bordure dans chaque convolution. Au niveau de la couche finale, une convolution 1x1 est utilisée pour mapper chaque vecteur d'entités à 64 composants au nombre de classes souhaité. Au total, le réseau compte 23 couches convolutives.

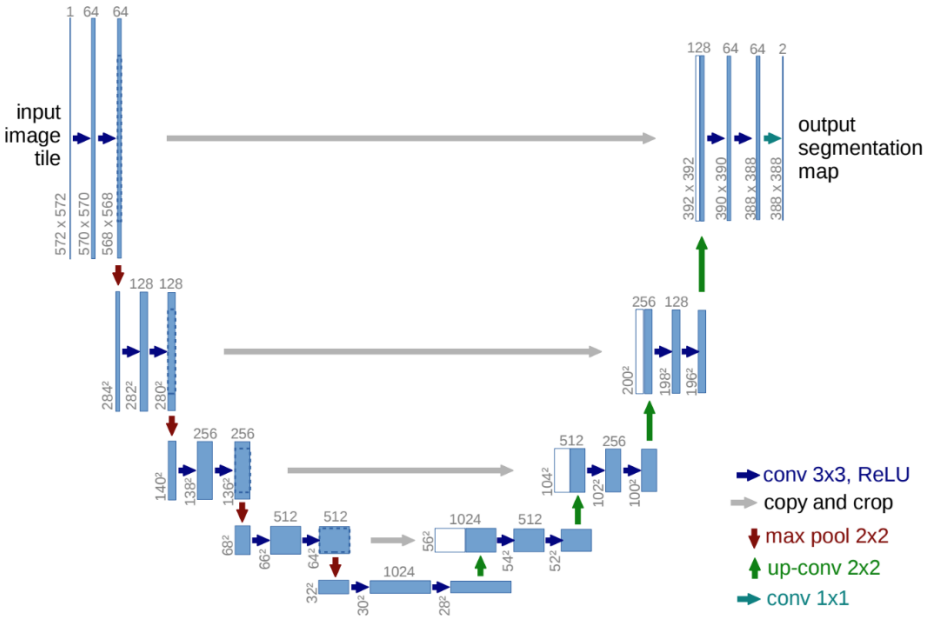

Figure 6 : Rappel architecture U-NET.

2.2. Choix de métriques et de fonction loss

Trois métriques sont suivies lors de l'entrainement du modèle et sont calculées pour évaluer les performances du modèle. Ces métriques sont : Accuracy, Dice Coefficient, IoU index. D'autre part, la fonction dice est utilisée comme fonction loss.

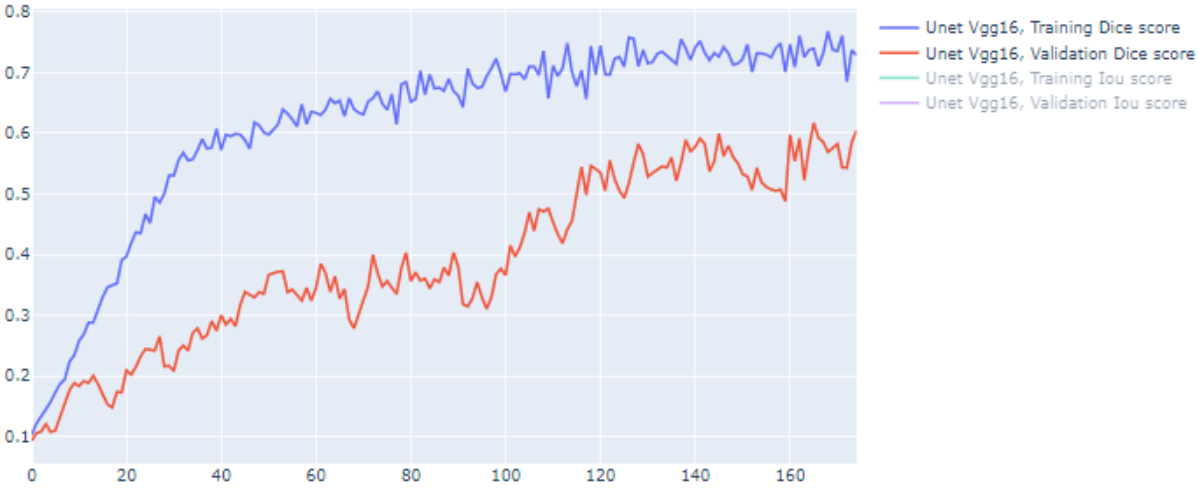

Figure 7 : Learning curves training et validation dice coefficient.

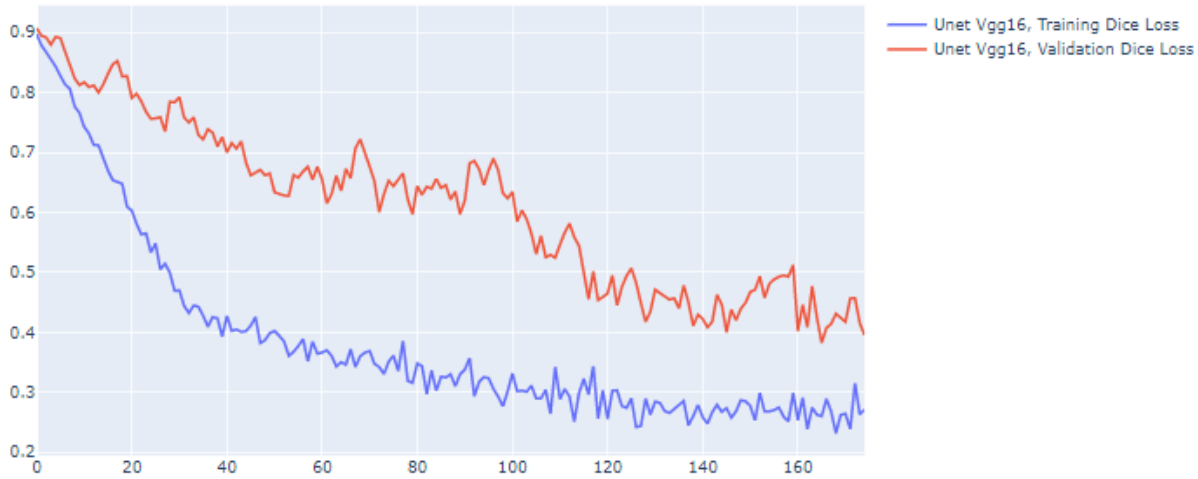

Figure 8 : Learning curves training and validation dice loss.

3. Synthèse des résultats

Dans un premier temps, on peut visuellement comparer les masques prédits par les différents modèles. Par la suite, on comparera les performances des modèles : les métriques accuracy, coefficient dice et jaccard index et le temps de calcul.

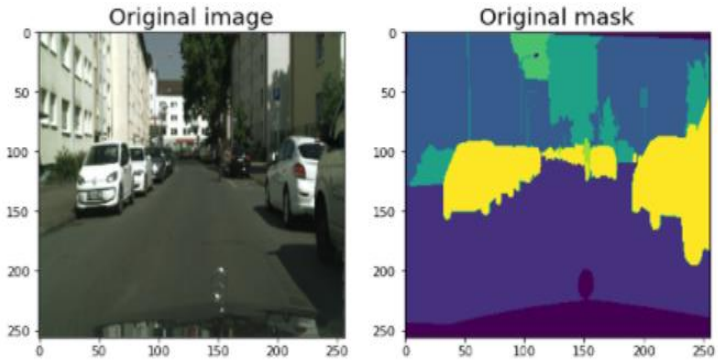

Figure 9 : Exemple d'une image originale et d'un masque original.

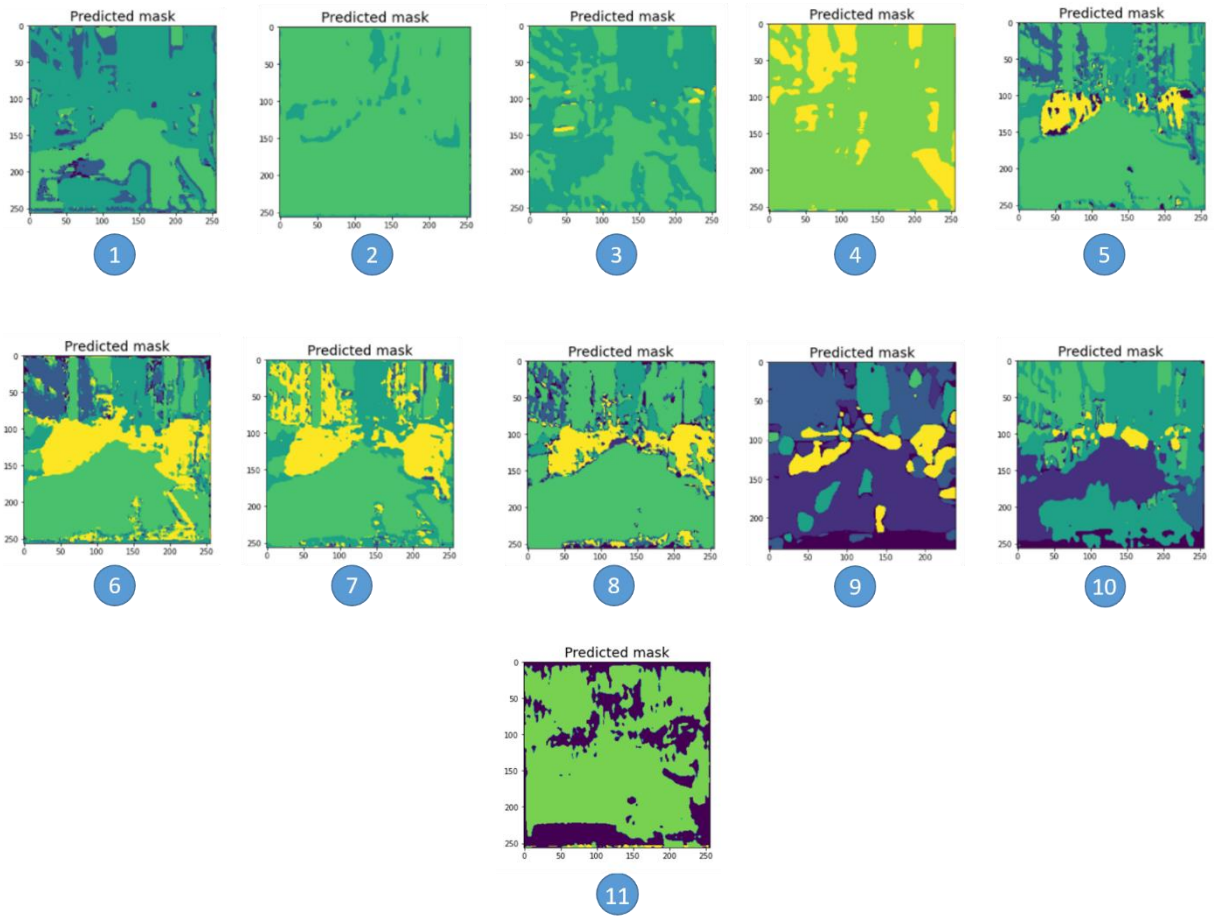

Figure 10 : Les masques prédits par les modèles allant de 1 à 11.

Les meilleures performances sont obtenues avec le modèle Unet avec un backbone VGG-16, en utilisant la dice loss comme fonction de perte/coût.

Model + Loss	Model	Loss Function	Loss	Accuracy	Dice metric	Jaccard index	Time	Time (HH:MM:SS)
Unet / Categorical crossentropy	Unet	Categorical crossentropy	0.843078	0.734229	0.433130	0.309372	359.849840	00:05:59
Unet Augmented Data / Categorical crossentropy	Unet Augmented Data	Categorical crossentropy	1.454532	0.510352	0.296561	0.194945	355.578558	00:05:55
Unet Dice / Dice	Unet Dice	Dice	0.513157	0.732046	0.486843	0.366852	350.740497	00:05:50
Unet Cat Focal / Categorical focal	Unet Cat Focal	Categorical focal	0.019611	0.592864	0.377452	0.258348	351.098519	00:05:51
Unet Vgg16 / Dice	Unet Vgg16	Dice	0.348191	0.796819	0.651809	0.515314	340.846538	00:05:40
Unet Vgg16 train all / Dice	Unet Vgg16 train all	Dice	0.370568	0.792849	0.629432	0.491777	361.786868	00:06:01
Unet Vgg16 / Jaccard Loss	Unet Vgg16	Jaccard Loss	0.512906	0.803262	0.615348	0.487094	346.819940	00:05:46
Unet Vgg16 / CCE + Dice Loss	Unet Vgg16	CCE + Dice Loss	0.453497	0.829607	0.629820	0.505657	344.793242	00:05:44
PSPNet / Dice	PSPNet	Dice	0.879011	0.396272	0.120989	0.075788	345.447343	00:05:45
FPN / Dice	FPN	Dice	0.824495	0.342309	0.175505	0.119207	381.604285	00:06:21
FPN Inception V3 / Dice	FPN Inception V3	Dice	0.277096	0.847340	0.722904	0.603756	396.883514	00:06:36

Figure 11 : Tableau comparatif des performances des modèles

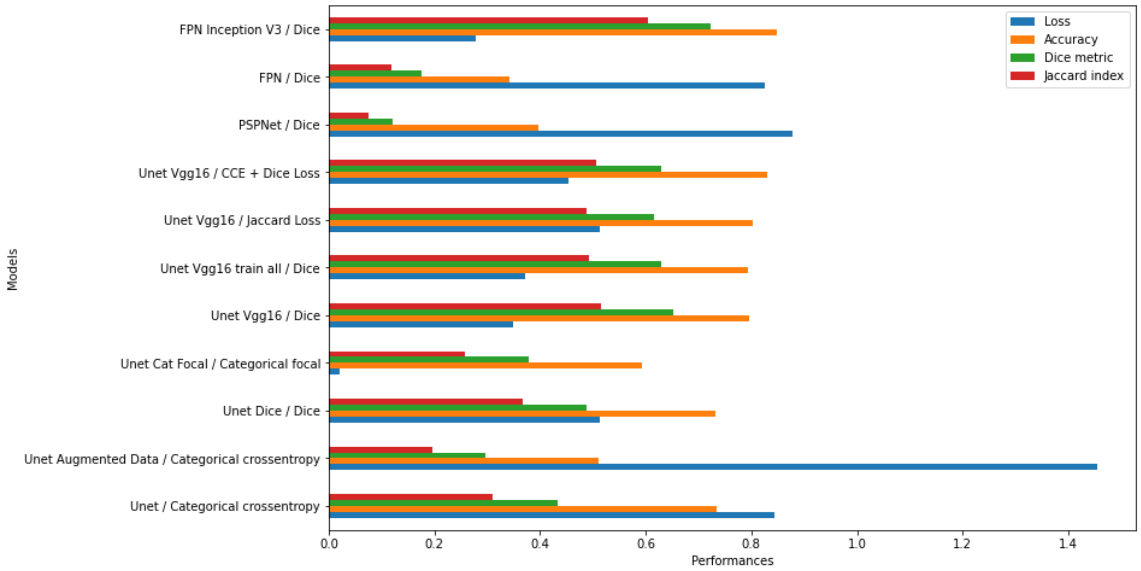

Figure 12 : Graphe comparatif des performances

4. Conclusion & Pistes d'amélioration

Cette note technique apporte des détails techniques sur les différentes architectures de segmentation sémantique testées dans le cadre du projet 8 du parcours Ingé IA. Les modèles U-NET, PSPNet et FPN sont comparés, testés avec différentes métriques, différentes fonctions loss et différents backbones.

Le modèle U-NET permet d'obtenir les meilleures performances comparé aux modèles PSPNet et FPN. Les métriques utilisées sont l'accruracy, dice coefficient, IoU index. Les fonctions loss testés sont Categorical Cross-Entropy (CCE), dice loss, Jaccard loss et CCE + dice loss. Les meilleures performances sont obtenues avec la fonction dice loss. Enfin, les différents backbones testés sont le VGG, le RESNet et l'Inception. Plus particulièrement, le backbone VGG16 permet d'obtenir les meilleures performances. En générale, le temps d'exécution (sur le GPU) des différents modèles testés sont très proches (environ 6 minutes pour 175 epochs), ce qui ne permet pas de les comparer sur ce critère.

L'augmentation de données a été choisie judicieusement pour fournir au modèle des images qu'il peut réellement rencontrer. Par exemple, pour un modèle dont le but est d'identifier le chat sur l'image, il peut être amené à voir des photos de chat avec la tête en bas (horizontal flip). Or, dans le cadre de la conduite autonome, le modèle ne verra pas les voitures ou la rue « tête en bas ». Dans ces cas, il faut plutôt appliquer un « vertical flip » afin d'effectuer l'augmentation de données. L'augmentation de données a été testée sur le modèle de base (cf modèle 1 et 2) et n'a pas permis d'améliorer les performances du modèle. Une piste d'amélioration serait de tester l'augmentation de données sur le modèle final U-NET avec un backbone VGG16. Ainsi, on peut optimiser et tester d'autres méthodes d'augmentation des données une fois le modèle et le backbone figé.

Même si plusieurs fonctions loss ont été testées, ils n'ont pas tous été testé sur le modèle finale U-NET avec un backbone VGG16. Dans le but d'optimiser le modèle final, il est possible de tester d'autres fonctions loss et d'autres combinaisons de fonctions loss.