

APPLICATION DE SEGMENTATION SEMANTIQUE D'IMAGES



Note Technique

SOMMAIRE

I - CONTEXTE

II - SEGMENTATION SEMANTIQUE

II.1 - Principe de segmentation

III - Mesures de performances

III.1 - Interception over Union (IoU)

III.2 - Tversky coefficient

III.3 - Dice coefficient

IV - Fonctions pertes

V - Modèles d'apprentissage

V.1 - Modèle Unet

V.1.1 - Type d'architecture

V.1.2 - Résultats et performances

V.1.3 - augmentation d'images

V.2 - Modèle LinkNet

V.1.1 - Type d'architecture

V.1.2 - Résultats et performances

V.3 - Modèle PspNet

V.1.1 - Type d'architecture

V.1.2 - Résultats et performances

VI - Discussion et Perspectives d'améliorations

VII - Annexe

I - CONTEXTE

Cette fiche technique sur la réalisation d'une application de segmentation d'image s'inscrit dans le cadre d'un projet plus global sur la conception d'une voiture autonome. Le projet en lui même est découpé en plusieurs parties :

- **Une partie sur l'acquisition de données** : qui consiste essentiellement à acquérir des images à l'aide de caméras et qui seront plus tard utilisées pour l'entraînement du modèle de prédiction ;
- Une partie sur le **traitement des images** : le but à terme ici étant de transformer les images et les traiter de manière à les rendre utilisables par un modèle;
- Une partie sur l'**apprentissage d'un modèle de segmentation des images** par un modèle qui saura découper les images en plusieurs zones d'intérêt ayant des caractéristiques communes;
- Une dernière partie sur **le développement du système de décision** de la voiture autonome. Elle met en relation la sortie du modèle de segmentation avec le circuit électronique décisionnel qui est responsable de l'ensemble de toutes les décisions sur l'état de la voiture en fonction de l'environnement dans lequel se trouve la voiture selon la segmentation fournie par le modèle.

La partie sur laquelle l'on va se concentrer ici est celle portant sur la conception du modèle de segmentation des images. Plus généralement, dans les problèmes de vision par ordinateur entrent souvent en jeu :

II - Segmentation sémantique

Plus généralement, dans les problèmes de vision par ordinateur, les images sont très souvent abordées sous 3 écoles :

- La **classification d'images** qui va consister à associer une catégorie à chaque images; par exemple une image peut correspondre à un chien, un cheval, un restaurant, une piscine ou un téléphone ;

Is this a dog?



What is there in image and where?



Which pixels belong to which object?

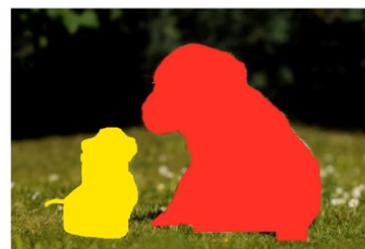


Image Classification

Object Detection

Image Segmentation

Figure 1

- La **détection d'objets** qui consiste à délimiter dans l'image, une ou plusieurs zones qui correspondent à des objets différents. Par exemple sur une même image on peut avoir un chat, un chien, une voiture et une maison et la tâche consistera à encadrer chacun de ces objets par un quadrilatère convexe tel un carré ou un rectangle.

- La **segmentation sémantique** consistant à associer plusieurs pixels appartenant à un même objets tout en les attribuant une coloration unique et donc uniforme.

Dans le cas présent qui nous intéresse nous utilisons la segmentation sémantique afin que l'ordinateur de bord de la voiture autonome puisse être consciente de

qui l'entourent et en fonction de cette information, prendre des décisions objectives qui lui permettront de naviguer dans cet environnement. Par exemple, on sait tous instinctivement que lorsqu'on veut traverser la route on doit regarder la couleur du feu piéton : s'il est rouge on doit s'arrêter et s'il est vert on peut traverser la route mais une sécurité supplémentaire serait de s'assurer de tous les véhicules sont à l'arrêt car il peut arriver un disfonctionnement du dispositif ou qu'un conducteur de véhicule ne soit pas suffisamment attentif et poursuit sa course sans s'arrêter. Aussi, lorsque je me promène dans la rue et qu'un autre personne circule en sens opposé sur le même passage piéton que moi je dois changer de couloir pour éviter une collision ou même encore si je parcourt le même chemin qu'un autre piéton et que je me déplace plus vite que lui à un moment lorsque je suis suffisamment près de ce piéton je dois soit ralentir et adapter ma vitesse à la sienne ou prendre un autre chemin si cela est possible si la densité de piétons dans mon voisinage est suffisamment faible.

C'est ce même principe de fonctionnement qu'on essaie d'inculquer à notre voiture autonome, et pour cela il a besoin d'être suffisamment conscient des objets qui se trouvent dans son environnement grâce à des caméras embarquées et l'application qui permet de segmenter chaque images afin de faire ressortir les types d'objets avec lesquels il interagit.

II.1 - Principe de segmentation

Une image est une assemblée de plus petites images monochromes appelées pixels. En effet pour numériser une image on est obligé de découper l'image à travers une grille/tamis; ce qui aura pour but de discréteriser l'image en de petites portions ayant une couleur uniforme (voir figure 2). Pour sauvegarder l'image il suffira alors d'enregistrer les différents position des différents morceaux ainsi que la couleur qui leur correspond. Dans ce projet nous avons utilisés des images issues de la base de données **cityscapes** : www.cityscapes-dataset.com



Figure 2

Les images obtenues de Cityscapes sont de deux types :

- Les images brutes: correspondant aux images obtenues à partir de caméras placées sur la voiture. Sur chaque image on peut nettement distinguer chaque objet situé dans le champ de vision de la caméra comme les phares de voitures, des passants, leurs vêtements, l'éclairage public, des magasins, des habitations, des chiens, chats , vélos . . . ce qui constitue une grande quantité d'objets que le modèle devra reconnaître.



- Les masques: qui sont une représentation plus simplifiée de l'image de base consistant à regrouper plusieurs objets de différentes nature en un seul objet ayant une seule couleur ou niveau de gris.



Chacune des couleurs ou niveau de gris du masque étant codé par un entier naturel compris entre 0 et 255. Ici on se focalise essentiellement sur 8 catégories d'objets à savoir: véhicules, humains, ciel, végétation, signalétique routière, constructions, route et le reste

Chacune de ces 8 classes est ainsi représentée par un entier compris entre 0 et 7, voir l'exemple de la figure 3. Chacun des objets étant considéré comme un label et chaque label correspondant à un niveau de gris bien défini qui à son tour est associé à un entier bien précis. Par "classe" j'entends ici toute région de l'image qui contient toute ou partie d'une collection d'objets donné



Figure 3

On peut ainsi observer une transformation de l'histogramme des pixels sur une collection d'une trentaine d'images.

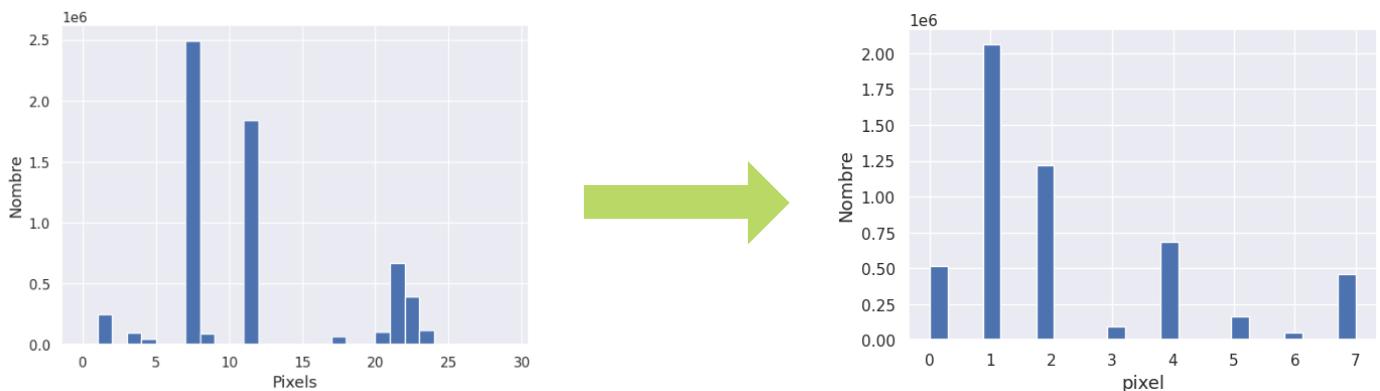


Figure 4

Une fois arrivé à ce niveau, la tâche consiste simplement à élaborer un modèle qui sera capable de reproduire le masque exacte associé à une image dès lors qu'on l'aura fourni l'image brute. La situation peut être résumée par la figure ci-dessous.

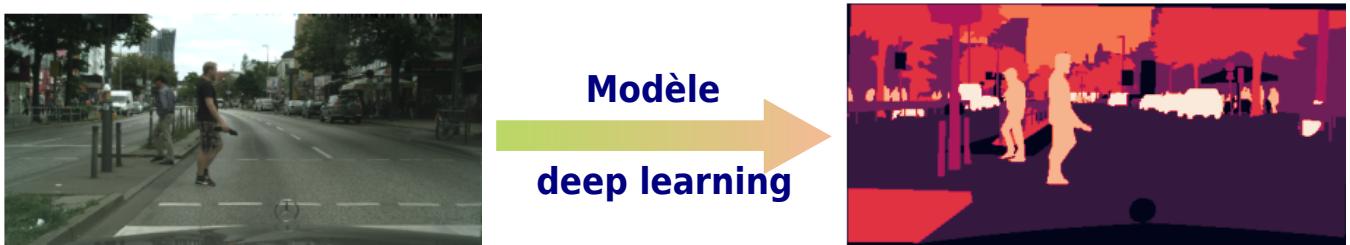


Figure 5

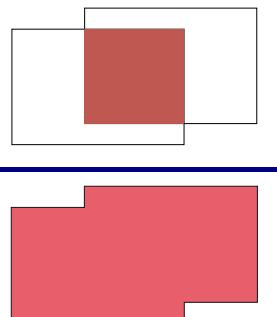
III - Mesures de performances

Très généralement lorsqu'on entraîne un modèle à faire des prédictions, il arrive que le résultat diffère de ce qu'on attendait de lui. Il est donc important de définir des critères qui vont permettre d'évaluer les performances du modèle et de savoir à quel dégré les prédictions du modèles sont proches de la réalité. Un peu à l'image d'un enseignant qui corrige les copies de ses étudiants au sortir d'un contrôle de connaissances. 3 métriques sont employées ici à savoir **Intersection over union**, le **Dice coefficient** et le **Coefficient de Tversky**. Toutes ces métriques faisant appel à des notions de théorie des ensemble car une image numérisée peut être perçue comme une collection d'objets appelés pixels. Il est important de raisonner en termes de cardinal des ensembles et pas d'éléments de l'ensemble pour bien comprendre les différentes schématisations ensemblistes des métriques ci-dessous présentées : La mesure employée est le cardinal.

III.1 - Intersection over union (IoU)

Considérant ici le masque prédit par le modèle et le masque réellement attendu, la métrique ci après compare le nombre de paires de pixels qui se correspondent au nombre total de pixels unique dans les deux masques. Considérant **P** le masque prédit et **R** le masque réel attendu, la métrique s'exprime comme suit :

$$\text{IoU} = \frac{|\mathbf{P} \cap \mathbf{R}|}{|\mathbf{P} \cup \mathbf{R}|}$$

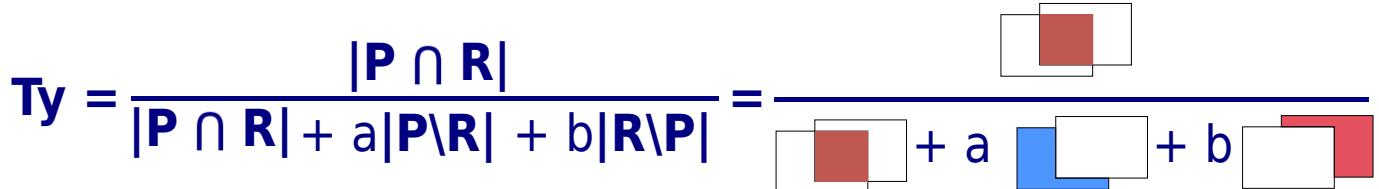


Lorsq'aucune paire de pixel ne se correspondent sur les deux masques , l'intersection est vide et la métrique IoU est donc nulle, d'un autre côté lorsque le modèle prédit un masque qui est de plus en plus similaire au masque réellement attendu, l'intersection se rapproche de plus en plus de l'union et la métrique tend de plus en plus vers la valeur unitaire 1.

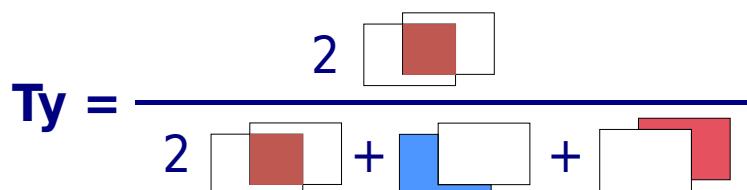
III.2 - Tversky coefficient

le coefficient de Tversky à l'origine est introduit pour la mesure de l'assymétrie entre deux ensemble contrairement à la métrique précédante qui elle est symétrique : $\text{IoU}(P ; R) = \text{IoU}(R ; P)$

L'assymétrie du coefficient de Tversky est introduite par l'utilisation des complémentaires d'un ensemble dans un autre dans sa formulation

$$Ty = \frac{|\mathbf{P} \cap \mathbf{R}|}{|\mathbf{P} \cap \mathbf{R}| + a|\mathbf{P} \setminus \mathbf{R}| + b|\mathbf{R} \setminus \mathbf{P}|} = \frac{2}{2 + a + b}$$


Les coefficients a et b étant à valeurs réelles; dans notre projet nous avons utilisé la valeur 0.5 pour a et b ; ce qui corresponds au Coefficient de Dice habituellement rencontré dans la littérature :

$$Ty = \frac{2}{2 + |\mathbf{P} \setminus \mathbf{R}| + |\mathbf{R} \setminus \mathbf{P}|}$$


Dans la situation la moins désirée où le masque prédit ne correspond en aucun pixel au masque attendu, l'intersection des deux masques est vide et le coefficient de Tversky est nul. À l'opposée lorsque les deux masques se correspondent exactement, l'intersection des deux masques se confond à leur union en conséquence de quoi le coefficient prends la valeur 1. À noter cependant qu'il ne s'agit pas du seul cas où le coefficient prends cette valeur. En effet une résolution de l'équation $Ty = 1$ nous donne une solution triviale pour laquelle les complémentaires de chaque masque l'un dans l'autre possèdent le même cardinal; ce qui ne corresponds pas forcément à la solution recherchée sauf si c'est un cardinal nul. C'est une raison pour laquelle il est nécessaire de combiner l'utilisation de cette métrique en complément à au moins une autre permettant afinide monitorer la convergencence du modèle et lever l'indétermination sur le fait que la solution soit celle attendu ou non.

III.3 - Dice coefficient

Ici nous utilisons un coefficient de Dice amélioré

$$\text{Dice} = \frac{2 \begin{array}{c} \square \\ \square \end{array} + 1}{2 \begin{array}{c} \square \\ \square \end{array} + \begin{array}{c} \square \\ \square \end{array} + \begin{array}{c} \square \\ \square \end{array} + 1}$$

IV - Fonctions pertes

Les fonctions de perte ou loss functions sont régulièrement employées pour définir une mesure sur le système qui doit atteindre un minimum lorsque le système atteint sa convergence voulue. Elles peuvent être définies à partir des métriques de performances comme celles précédemment définies. Par exemple la quantité **1 - IoU** représente une fonction de perte, que l'on peut nommer **IoU loss**, pour laquelle l'algorithme d'optimisation s'attardera à miniser la valeur de manière à avoir une **IoU** maximale : C'est une fonction qui permet d'obtenir une valeur maximale pour une métrique; l'écart quadratique moyen en est un autre exemple. Il est aussi possible de combiner plusieurs loss fonctions différentes en utilisant des multiplicateurs de Lagrange avec des différentes valeurs pour donner plus de poids sur le travail de plus maximiser une métrique plutôt qu'une autre. Par exemple si on veut maximiser à la fois l'**IoU** et le coefficient de Tversky en mettant 3 fois plus d'effort sur la maximisation de l'**IoU**, on peut définir une fonction perte, que l'on pourrait appeler **IoU-Tversky Loss**, comme suit :

$$\text{Loss} = 3x(1 - \text{IoU}) + 1 - \text{Ty}$$

Dans ce projet nous avons effectués des tests avec 5 types de fonctions pertes à savoir :

- **jaccard loss**
- **dice loss**
- **categorical focal loss**
- **categorical focal jaccard loss**
- **categorical focal dice loss**

V - Modèles d'apprentissage

3 modèles de deep learning à couches de convolution ont été employées pour réaliser la segmentation des images. Pour chacun de ces modèles les différentes fonctions pertes énumérées précédemment ont été employés donnant des résultats de convergence différents pour chacun d'eux. Les modèles utilisés sont : **Unet**, **LinkNet** et **PspNet**

V.1 - Modèle Unet

V.1.1 -Architecture

Le modèle tient son nom de la forme en **U** de son architecture de réseau :

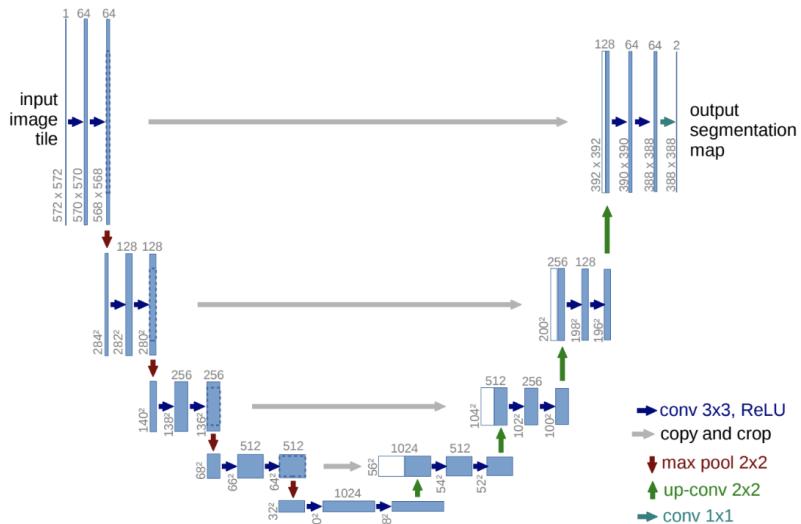


Figure 6

Le réseau est composé de 2 grandes partie: un **encodeur (descente)** qui construis des "copies" distinctes de l'image avec des dimensions plus réduites et un **décodeur (montée)** qui construit progressivement des images de dimensions de plus en plus grandes à partir de sorties de l'encodeur. Quand le modèle reçoit une image, une série de convolutions est effectuées avec un nombre de filtres que l'utilisateur peu définir. L'opération permet d'extraire différents types d'informations ou features sur l'image à partir des filtres. Entre plusieurs séries de convolutions des **Max Pooling** permettent de concentrer l'information transportée par des pixels voisins en un pixel central; l'opération permet ainsi de réduire la taille de l'image par compression de l'information transportée.

À la suite de la compression de l'image par l'encodeur vient ensuite le décodeur qui réalise les même convolutions avec la différence qu'à la place de couches de Max pooling on aura des couches de **up-conv** qui réalisent l'opération inverse à savoir agrandir la taille de l'image. Entre les séries de convolution et les up-conv, une **concaténation** est réalisée entre les sorties de la couche up-conv et des couches de convolution équivalentes dans la partie encodeur. L'opération de concaténation agit ici comme un couche mémoire où en reconstruisant l'image (up-conv) on effectue un "rappel mémoire" sur les localisations des features obtenues des convolutions dans l'encodeur.

V.1.2 -Résultats et performances

Après entraînement des paramètres du modèle en utilisant progressivement les différentes fonctions de perte, un récapitulatif des performances optimales du modèle est fait à la figure ci-dessous

	mean_IoU	dice_coeff	Tversky_coef	val_mean_IoU	val_dice_coeff	val_Tversky_coef	modèle	Loss
0	0.803224	0.890870	0.269754	0.546726	0.706025	0.253375	Transfert_learning_UNET	jaccard_loss
1	0.464027	0.633378	0.246937	0.383669	0.548988	0.239680	Transfert_learning_UNET	dice_loss
2	0.519803	0.684039	0.252582	0.369610	0.538498	0.239779	Transfert_learning_UNET	categorical_focal_loss
3	0.844026	0.915395	0.271937	0.707761	0.828775	0.264345	Transfert_learning_UNET	categorical_focal_jaccard_loss
4	0.835610	0.910435	0.271543	0.639473	0.779913	0.259874	Transfert_learning_UNET	categorical_focal_dice_loss

Figure 7

Le modèle présentant les meilleures performances est construit avec la fonction perte **categorical focal jaccard**. Cet entraînement a été réalisé avec un modèle qui utilise successivement sur chaque série de couches de convolution 256 filtres, 128, 64, 32 et 16. Un autre essai a été réalisé qui utilise un peu plus de filtres à savoir successivement 512, 256, 128, 64 et 32 filtres sur les couches de convolution. Le résultat est présenté à la figure suivante :

	mean_IoU	dice_coeff	Tversky_coef	val_mean_IoU	val_dice_coeff	val_Tversky_coef	modèle	Loss
0	0.842273	0.914383	0.271828	0.699934	0.823430	0.263800	Transfert_learning_UNET+512-32	jaccard_loss
1	0.568702	0.724629	0.255021	0.459231	0.627754	0.246580	Transfert_learning_UNET+512-32	dice_loss
2	0.516508	0.681144	0.252429	0.304611	0.465737	0.233218	Transfert_learning_UNET+512-32	categorical_focal_loss
3	0.880238	0.936301	0.273713	0.686438	0.812918	0.262808	Transfert_learning_UNET+512-32	categorical_focal_jaccard_loss
4	0.865076	0.927636	0.273050	0.640489	0.779926	0.259966	Transfert_learning_UNET+512-32	categorical_focal_dice_loss

Figure 8

V.1.3 - augmentation d'images

Des tests ont aussi été réalisés avec ce modèle en utilisant des images augmentées. Le principe des images augmentées est de se dire que nous disposons d'un modèle qui est gourmand en données et que nous n'en avons pas suffisamment pour le nourrir. Le dilemme peu être résolu si on construit de nouvelles images à partir d'images disponible en y réalisant des modifications comme des opérations de symétrie, modification de l'éclairage, de luminosité, de contraste, ajout de bruit gaussien, effet spéciaux comme de la pluie, de la neige . . . De ce fait il existe une infinité de combinaisons sur les types de changements réalisables. Un exemple est illustré à la figure ci-dessous :



Figure 9

À noter cependant que des tests d'augmentation d'images ont été réalisés uniquement avec ce modèle à cause des contraintes liées au temps disponible pour entraîner les différents modèles. Les résultats de performances obtenus avec des augmentations sur les images sont consignés dans le tableau suivant :

	mean_IoU	dice_coeff	Tversky_coef	val_mean_IoU	val_dice_coeff	val_Tversky_coef	modèle	Loss
0	0.773399	0.872222	0.268073	0.462624	0.632595	0.246863	Transfert_learning_UNET+augmentation	jaccard_loss
1	0.466391	0.636108	0.246872	0.354483	0.523422	0.237347	Transfert_learning_UNET+augmentation	dice_loss
2	0.338915	0.506253	0.237092	0.098838	0.179894	0.206745	Transfert_learning_UNET+augmentation	categorical_focal_loss
3	0.812566	0.896592	0.270383	0.450551	0.621213	0.246004	Transfert_learning_UNET+augmentation	categorical_focal_jaccard_loss
4	0.850707	0.919332	0.272344	0.587909	0.740481	0.256467	Transfert_learning_UNET+augmentation	categorical_focal_dice_loss

Figure 10

V.2 - Modèle LinkNet

V.2.1 -Architecture

L'architecture du modèle Linknet est assez similaire à celle du modèle Unet notamment la structure des couches de convolution, les couches de sous-échantillonage et sur-échantillonage (**Max-pooling** et **up-conv**) . . . L'unique différence se situe sur l'opération de concatenation qui est remplacée par une opération

d'addition entre les images issus de l'over-sampling des images dans de l'décodeur et des convolutions dans l'encodeur.

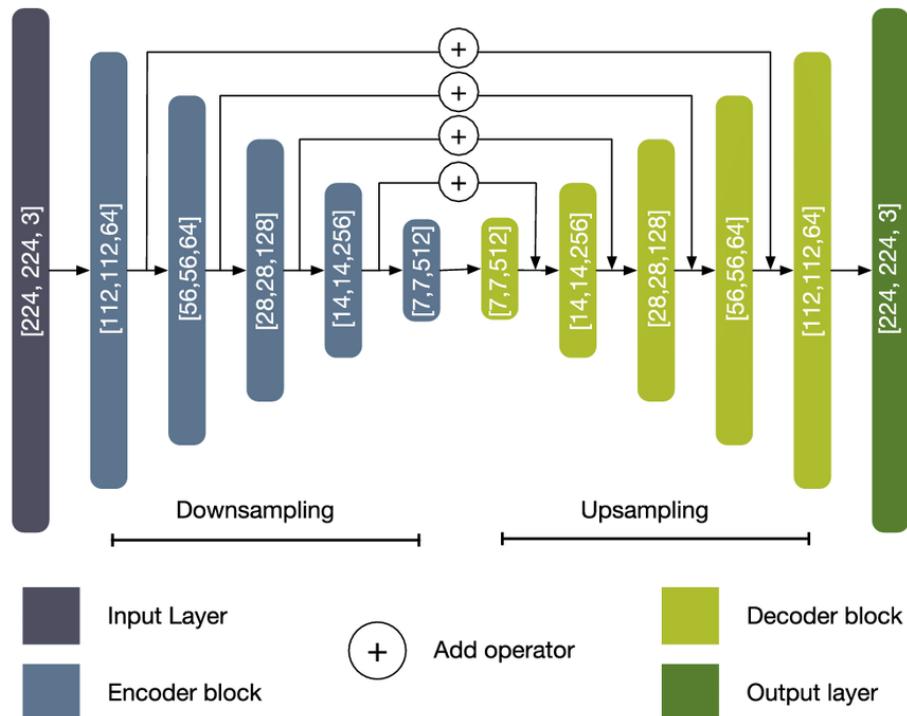


Figure 11

V.2.2 -Résultats et performances

Suite à un entraînement du modèle sur plusieurs paires images-masques, les performances du modèles sont regroupées dans le tableau suivant :

	mean_IoU	dice_coeff	Tversky_coef	val_mean_IoU	val_dice_coeff	val_Tversky_coef	modèle	Loss
0	0.813244	0.896992	0.270263	0.619135	0.763617	0.258577	Transfert_learning_LinkNET	jaccard_loss
1	0.561368	0.718963	0.254621	0.488063	0.654458	0.248815	Transfert_learning_LinkNET	dice_loss
2	0.455832	0.626144	0.247509	0.340543	0.507972	0.236986	Transfert_learning_LinkNET	categorical_focal_loss
3	0.851575	0.919810	0.272310	0.579748	0.733853	0.255934	Transfert_learning_LinkNET	categorical_focal_jaccard_loss
4	0.835549	0.910406	0.271481	0.653147	0.789402	0.260910	Transfert_learning_LinkNET	categorical_focal_dice_loss

Figure 12

Le modèle qui présente les meilleures performances de prédiction de masques est celui entrainé avec la fonction perte **categorical focal dice loss** qui est la superposition de 3 fonctions pertes distinctes à savoir la **categorical Cross entropy** sur nos 8 classes, la **focal loss** (encore assimilée à l'**IoU loss**) et la **dice loss**. La première loss fonction assure une convergence des valeurs de chaque classes vers les valeurs attendues tandis que les deux dernières fonctions perte assure un maximum de valeurs aux métriques IoU et dice coefficient.

V.3 - Modèle PspNet

V.3.1 -Architecture

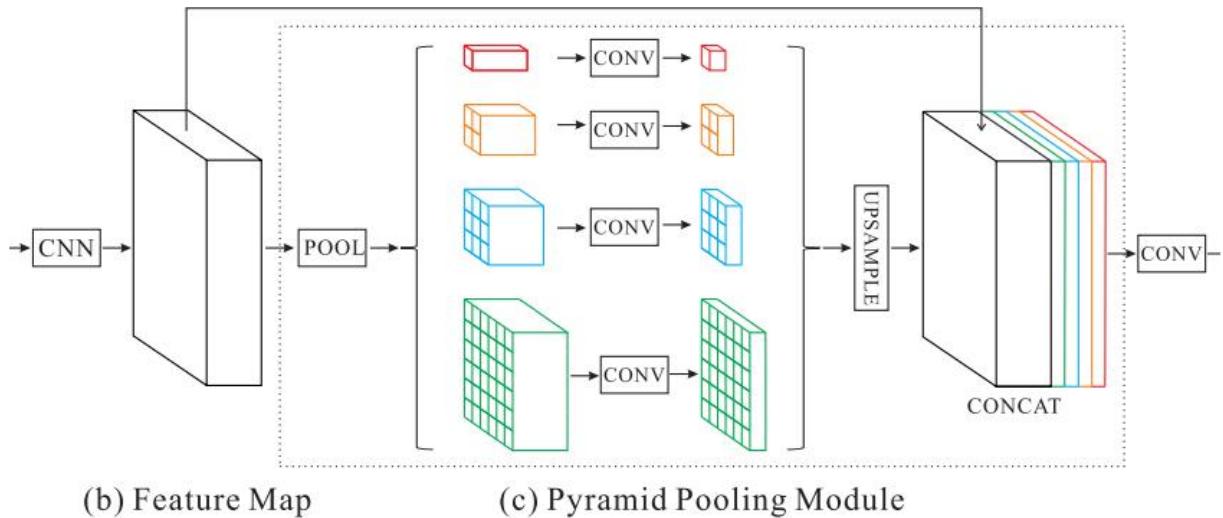


Figure 13

Pour comprendre l'architecture du modèle PspNet, on peut partir de celle du modèle Unet : Si on supprime les différentes couches de concatenation apparaissant dans l'architecture Unet, là où le modèle Unet effectue une succession de série de convolution et d'under-sampling sur les sorties des couches de convolutions précédentes, le modèle PspNet réalise ces même convolutions sur une seule et même sortie: le résultat d'une série de convolution sur l'image de départ.

En effet l'image fournie au modèle subi au départ une série de convolutions afin d'extraire les features importantes de l'images. Le résultat de cette convolution est ensuite sous-échantilloné avec 4 types de filtres dans des canaux parallèles puis convolué dans chaque canaux avant d'être à nouveau sur-échantilloné et au final, les images issues de chaque canaux sont concaténés à la sortie de la première couche de convolution. Une dernière convolution est alors réalisée sur le bloc d'images avant de construire le masque prédit.

Ainsi la différence avec le modèle Unet est les sous-échantillonages et sur-échantillonages ne sont pas réalisés à la chaîne de manière quasi-recurrive mais plutôt en un temps dans des canaux parallèles et complètement indépendants. Aussi la concatenation des images est réalisée une seule fois une fois under-sampling et l'over-sampling effectués.

V.3.2 -Résultats et performances

À la suite de l'entraînement du modèle, les performances ont été reportées dans le tableau ci-dessous :

	mean_IoU	dice_coeff	Tversky_coeff	val_mean_IoU	val_dice_coeff	val_Tversky_coeff	modèle	Loss
0	0.485904	0.653830	0.248807	0.477900	0.645700	0.248111	Transfert_learning_PSPNET	jaccard_loss
1	0.557583	0.715836	0.254248	0.516285	0.680246	0.251151	Transfert_learning_PSPNET	dice_loss
2	0.544090	0.704692	0.254380	0.436823	0.607986	0.245777	Transfert_learning_PSPNET	categorical_focal_loss
3	0.826680	0.905108	0.271071	0.706029	0.827595	0.264290	Transfert_learning_PSPNET	categorical_focal_jaccard_loss
4	0.812116	0.896303	0.270347	0.699375	0.823019	0.263869	Transfert_learning_PSPNET	categorical_focal_dice_loss

Figure 14

Comme on peut le voir, le modèle présentant les meilleures performances toutes métriques confondues est celui construit avec la fonction perte **categorical focal jaccard**

VI - Discussion et Perspectives d'améliorations

Parvenus au terme de notre exploration, 3 modèles de segmentations ont été utilisés avec 5 fonctions de pertes pour l'optimisation et 3 métriques pour l'analyse de convergence des différents modèles. En observant les résultats des différents tableaux récapitulatifs, on observe que le modèle ayant permis d'obtenir les plus grands scores est le modèle Unet ayant successivement 256, 128, 64, 32 et 16 filtres appliqués dans la série de convolutions et construis avec une categorical focal jaccard loss function (voire annexe).

Néanmoins il peut être important de rappeler un certain d'approximations ou de limites ayant été appliqués aux modèles et qui peuvent donner lieu à des pistes d'amélioration :

- Taille des données d'entraînement : Pour des raisons liées au temps nécessaire pour entraîner un modèle, je n'ai utilisé que 50 couples images-masques pour l'entraînement des modèles; ce qui représente un problème dans la représentabilité du jeux de données. Des modèles plus intéressants et prédictifs peuvent donc être obtenus à partir d'un entraînement sur des données plus conséquents.

- Aussi pour des raisons de temps j'ai effectué des tests d'optimisation de paramètres de modèles uniquement sur le modèle Unet. Notamment des tests avec différentes fonctions d'activations et différentes distributions du nombre de filtres sur les couches de convolution (256-128-64-32-16 et 512-256-128-64-32) En principe le modèle devrait fournir de meilleurs résultats quand on augmente le nombre de filtre car on a plus d'informations qui sont extraites des images; une telle observation n'a néanmoins pas été faite avec le modèle Unet peu être à cause de la faible quantité d'images utilisées (50). Il serait alors intéressant d'étendre cette optimisation des paramètres aux autres modèles avec plus d'images;

- Toujours à cause des contraintes de temps l'entraînement a été réalisé avec des images de faible dimension (128x256) au lieu de leur dimensions initiales (1024x2048). Cette réduction de dimension peut aboutir à des images de moindre qualité liée à la méthode d'approximation utilisée pour le redimensionnement; ce qui peut avoir un impact sur les performances des modèles. Il peut donc être intéressant d'entraîner les différents modèles avec des images à leurs dimensions initiales ou intermédiaires;

- Uniquement 3 modèles de base ont été testés mais il reste néanmoins la possibilité de d'utiliser des modèles améliorés dont l'architecture reposera sur les modèles utilisés ici avec quelques modifications par exemple sur le nombre de convolutions utilisées ou en introduisant simultanément les opérateurs de concatenation et d'addition à différents niveau dans leurs architecture sans que celà ne soit utilisé de manière exclusive comme celà est fait dans les modèles Unet et Linknet.

- Une architecture en "W" pourrait aussi être utilisée

VII - Annexe

Tableau récapitulatif des performances de tous les modèles testés :

modèle	Loss	Augmentation	mean_IoU	dice_coeff	Tversky_coeff	val_mean_IoU	val_dice_coeff	val_Tversky_coeff
Transfert_learning_LinkNET	categorical_focal_dice_loss	False	0.835549	0.910406	0.271481	0.653147	0.789402	0.260910
	categorical_focal_jaccard_loss	False	0.851575	0.919810	0.272310	0.579748	0.733853	0.255934
	categorical_focal_loss	False	0.455832	0.626144	0.247509	0.340543	0.507972	0.236986
	dice_loss	False	0.561368	0.718963	0.254621	0.488063	0.654458	0.248815
	jaccard_loss	False	0.813244	0.896992	0.270263	0.619135	0.763617	0.258577
Transfert_learning_PSPNET	categorical_focal_dice_loss	False	0.812116	0.896303	0.270347	0.699375	0.823019	0.263869
	categorical_focal_jaccard_loss	False	0.826680	0.905108	0.271071	0.706029	0.827595	0.264290
	categorical_focal_loss	False	0.544090	0.704692	0.254380	0.436823	0.607986	0.245777
	dice_loss	False	0.557583	0.715836	0.254248	0.516285	0.680246	0.251151
	jaccard_loss	False	0.485904	0.653830	0.248807	0.477900	0.645700	0.248111
Transfert_learning_UNET	categorical_focal_dice_loss	False	0.835610	0.910435	0.271543	0.639473	0.779913	0.259874
	categorical_focal_jaccard_loss	False	0.844026	0.915395	0.271937	0.707761	0.828775	0.264345
	categorical_focal_loss	False	0.519803	0.684039	0.252582	0.369610	0.538498	0.239779
	dice_loss	False	0.464027	0.633378	0.246937	0.383669	0.548988	0.239680
	jaccard_loss	False	0.803224	0.890870	0.269754	0.546726	0.706025	0.253375
Transfert_learning_UNET+512-32	categorical_focal_dice_loss	False	0.865076	0.927636	0.273050	0.640489	0.779926	0.259966
	categorical_focal_jaccard_loss	False	0.880238	0.936301	0.273713	0.686438	0.812918	0.262808
	categorical_focal_loss	False	0.516508	0.681144	0.252429	0.304611	0.465737	0.233218
	dice_loss	False	0.568702	0.724629	0.255021	0.459231	0.627754	0.246580
	jaccard_loss	False	0.842273	0.914383	0.271828	0.699934	0.823430	0.263800
Transfert_learning_UNET+512-32+sigmoid	categorical_focal_dice_loss	False	0.508664	0.674242	0.264489	0.446941	0.617678	0.258399
	categorical_focal_jaccard_loss	False	0.473964	0.643112	0.263179	0.391755	0.562504	0.253398
	categorical_focal_loss	False	0.124952	0.222146	0.213357	0.124953	0.222148	0.213427
	dice_loss	False	0.692255	0.818143	0.257284	0.584506	0.737478	0.251513
	jaccard_loss	False	0.873171	0.932282	0.272423	0.671580	0.803408	0.260746
Transfert_learning_UNET+augmentation	categorical_focal_dice_loss	True	0.850707	0.919332	0.272344	0.587909	0.740481	0.256467
	categorical_focal_jaccard_loss	True	0.812566	0.896592	0.270383	0.450551	0.621213	0.246004
	categorical_focal_loss	True	0.338915	0.506253	0.237092	0.098838	0.179894	0.206745
	dice_loss	True	0.466391	0.636108	0.246872	0.354483	0.523422	0.237347
	jaccard_loss	True	0.773399	0.872222	0.268073	0.462624	0.632595	0.246863
UNET	categorical_crossentropy	False	0.136196	0.239728	0.213035	0.134605	0.237239	0.212787
	categorical_crossentropy	True	0.129495	0.229286	0.212070	0.131148	0.231885	0.212268
	mixte	False	0.223620	0.365356	0.223606	0.224911	0.367083	0.223767
	mixte	True	0.216393	0.355600	0.222735	0.213659	0.351485	0.222398

Tableau 1