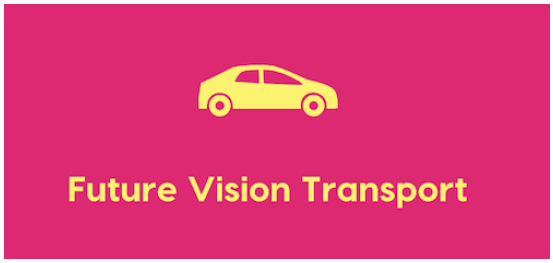

P8 - Note technique - Future vision transport

Segmentation sémantiques d’images

Hourdin Charlène
Parcours Ingénieur IA
Novembre 2022

Table des matières

1 Introduction

1.1 Présentation du projet

Le but de cette note technique est de présenter les travaux réalisés et les résultats obtenus dans ce cadre du **projet 8 - Participez à la conception d'une voiture autonome.**

Future Vision Transport est une entreprise qui conçoit des systèmes de vision par ordinateur embarqués pour les voitures autonomes.

Mon rôle au sein de l'équipe R&D *(composée d'ingénieurs aux profils variés)* de cette entreprise est de concevoir **la partie 3** du premier modèle de segmentation d'images qui devra s'intégrer facilement dans la chaîne complète du système embarqué.

Le système se compose de 4 parties :

— **Partie 1** : acquisition des images en temps réel.

— **Partie 2** : traitement des images.

— **Partie 3** : segmentation des images.

— **Partie 2** : système de décision.

Par conséquent, l'objectif de ce projet est de concevoir un modèle de segmentation d'image. Ce modèle sera fourni par le bloc de traitement d'image, qui sera lui-même fourni au système de décision.

A la suite d'une première phase de cadrage, j'ai récolté les avis des ingénieurs qui travaillent sur les parties avant mon intervention ***(traitement des images)*** et après ***(système de décision)*** :

Le jeu de données utilisé est **Cityscape** *(il contient les images segmentées et annotées de caméras embarquées)*. pour mon modèle, j'ai besoin des 8 catégories principales (et non pas des 32 sous-catégories)

Les résultats de la segmentation doivent être présentés dans une API facile à utiliser.

— **L'API** prend en entrée l'identifiant d'une image et renvoie le masque de l'image réelle *(vérité terrain)* ainsi que le masque prédit par le modèle.

— **Le déploiement** :l'API sera déployé sur le serveur Heroku visible à cette adresse : **https ://fvt-app-flask.herokuapp.com/**

2 État de l'art de la segmentation sémantique d'image

La segmentation d'images consiste à diviser une image en différentes parties ou régions qui représentent des objets ou des parties d'objets différents. Cette tâche est souvent utilisée en traitement d'image et en reconnaissance d'objets, car elle permet de cibler et de traiter de manière plus précise des parties spécifiques de l'image.

Il existe deux grandes catégories de segmentation d'images :

— **La segmentation sémantique** : consiste à regrouper des pixels en fonction de leur signification sémantiques ou de leur appartenance à un certain type d'objets. Par exemple, dans une image de rue, la segmentation sémantique pourrait regrouper les pixels appartenant aux maisons, aux arbres, aux voitures, etc.

— **La segmentation d'instance** : vise à identifier et à séparer chaque instance individuelle d'un objet dans l'image. Par exemple, dans une image de rue contenant plusieurs voitures, la segmentation d'instance consisterait à séparer chaque voiture en une région distincte.

Il existe plusieurs techniques d'apprentissage automatique qui peuvent être utilisées pour la segmentation sémantique d'images.

— **Les réseaux de neurones convolutionnels (CNN)** : les CNN sont un type de réseau de neurones particulièrement adapté aux tâches de traitement d'images. Ils peuvent être entraînés sur des images étiquetées pour apprendre à segmenter les images de manière automatique.

— **Les réseaux de neurones à attention** : les réseaux de neurones à attention permettent de concentrer l'attention du modèle sur certaines parties de l'image, ce qui peut être utile pour la segmentation d'images.

— **Les réseaux de neurones récurrents (RNN)** : les RNN sont un type de réseau de neurones qui peuvent traiter des séquences de données, comme des séries temporelles ou des phrases. Ils peuvent être utilisés pour la segmentation d'images en utilisant des séquences d'images au lieu de simples images.

— **Les réseaux de neurones de Markov cachés (HMM)** : les HMM sont un type de modèle stochastique qui peut être utilisé pour la segmentation d'images en modélisant la probabilité de transitions entre différentes étiquettes sémantiques au sein de l'image.

Chacune de ces techniques a ses propres avantages et inconvénients en fonction des caractéristiques de l'image à segmenter, il peut être nécessaire de combiner plusieurs de ces techniques pour obtenir les meilleurs résultats.

Voici une architecture générale qui peut être utilisée pour la segmentation d'images avec des modèles de réseaux de neurones :

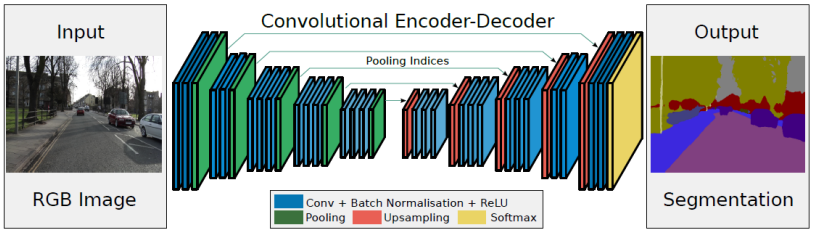

Figure 1 – Architecture d'une segmentation sémantique d'images

2.1 U-net

U-Net est une architecture de réseau de neurones qui a été développé pour la segmentation d'images biomédicales. Elle est conçue pour capturer les caractéristiques les plus importants d'une image tout en conservant une haute résolution pour une meilleure précision dans la segmentation.

U-Net utilise deux parties principales, un encodeur et un décodeur. L'encodeur réduit la taille de l'image en capturant les caractéristiques les plus importantes, tandis que le décodeur agrandit l'image pour prédire la segmentation.

U-Net se distingue des autres réseaux de neurones similaires en utilisant des connections directes entre les couches de l'encodeur et du décodeur, ce qui aide à conserver les détails importants de l'image. Cela permet à U-Net de fournir une segmentation plus précise des images.

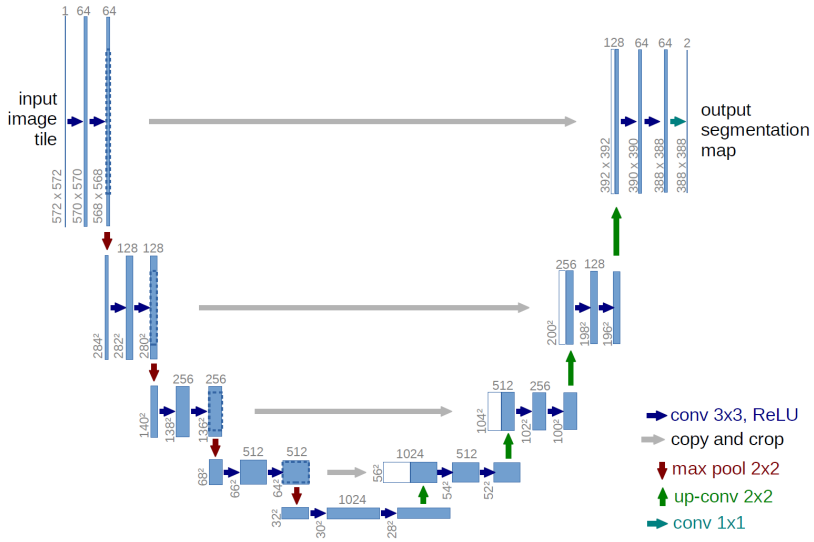

Figure 2 – Architecture U-net

2.2 LinkNet

LinkNet est une architecture de réseau de neurones conçue pour la segmentation d'images. Cela signifie qu'elle peut aider à séparer les différentes parties d'une image.

Elle utilise une structure qui ressemble à U-Net, avec des couches pour réduire la taille de l'image et extraire des informations importantes, et des couches pour remettre ces informations ensemble et prédire la segmentation.

LinkNet est différent de U-Net car il utilise des connexions plus courtes pour réduire le nombre de paramètres et faire l'analyse plus rapidement.

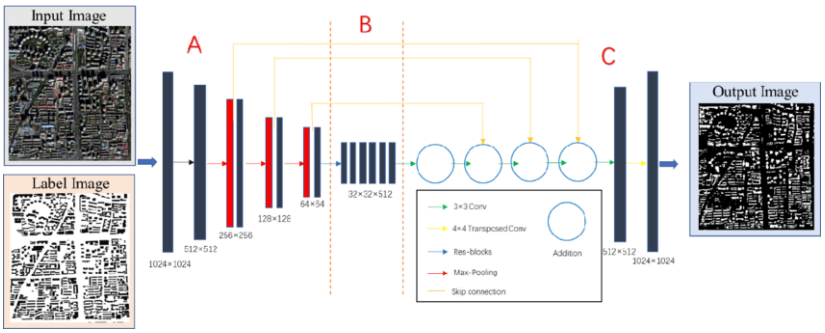

Figure 3 – Architecture LinkNet

2.3 PSPnet

PSPNet *(Pyramid Scene Parsing Network)* est un réseau de neurones qui a été créé pour séparer les différentes parties d'une image, comme les objets et le fond. Il utilise une structure qui ressemble à un puzzle, avec une partie qui réduit la taille de l'image et capture des détails importants, et une autre partie qui agrandit à nouveau l'image pour prédire la séparation.

Ce qui le différencie des autres réseaux de neurones similaires, c'est qu'il utilise une sorte de pyramide pour capturer des informations à différentes tailles. Cela aide PSPNet à mieux comprendre les différentes tailles d'objets dans l'image, ce qui rend la séparation plus précise.

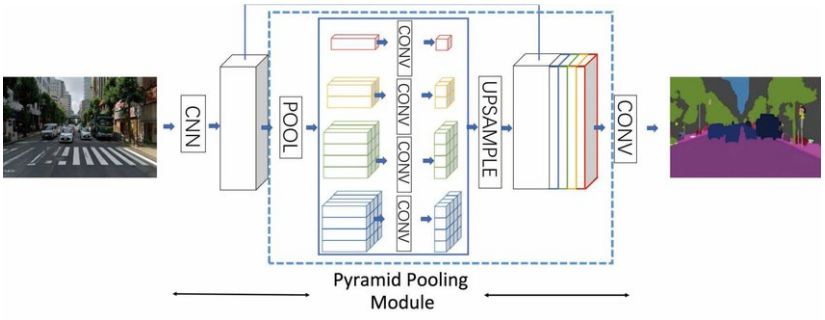

Figure 4 – Architecture PsPnet

2.4 FPN

FPN (*Feature Pyramid Network)* est un type d'architecture de réseau de neurones qui a été créé pour détecter des objets dans les images, mais qui peut aussi être utilisé pour la segmentation d'images. Cette architecture utilise une méthode en deux parties appelée encodeur-decodeur, où les couches de convolution et de pooling dans l'encodeur aident à réduire la taille de l'image et à extraire des informations importantes, tandis que les couches de déconvolution dans le décodeur aident à restaurer la taille originale de l'image et à prédire la segmentation.

FPN se démarque en utilisant une pyramide de caractéristiques pour extraire des informations contextuelles à différentes échelles. Cela aide à mieux gérer les objets de tailles différentes dans l'image et à améliorer la précision de la segmentation.

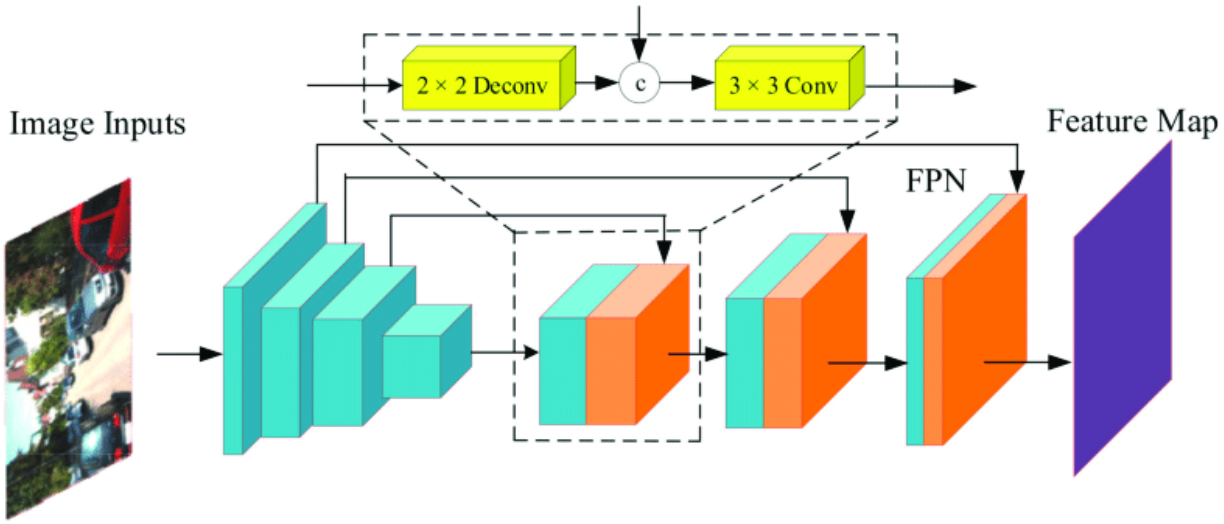

Figure 5 – Architecture FPN

2.5 Métrique et fonction Loss

Il existe différentes métriques d'évaluation et fonctions de perte qui peuvent être utilisées pour évaluer les performances d'un modèle de segmentation sémantique. La métrique choisie est cruciale pour évaluer la qualité du modèle, car elle détermine à quel point la sortie du modèle correspond à la valeur cible.

Les métriques d'évaluation couramment utilisées pour la segmentation sémantique :

- **Pixel Accuracy** : cette métrique mesure le nombre de pixels correctement classés par rapport au nombre total de pixels dans l'image. Elle est simple à calculer, mais peut être biaisée si certaines classes sont beaucoup plus fréquentes que d'autres dans l'image.
- **Dice Coefficient *(F1 Score)*** : cette métrique connu sous le nom d'indice Sørensen – Dice mesure la similitude entre les résultats de la segmentation et la référence en comparant le nombre de pixels correctement classés par rapport au nombre total de pixels dans l'image.
- **Intersection-Over-Union *(IoU, Jaccard Index)*** : cette métrique mesure la similarité entre les prédictions et les étiquettes réelles en comparant la zone d'intersection entre les deux et la zone union. La valeur IoU varie entre 0 et 1, avec 1 signifiant une correspondance parfaite entre les prédictions et les étiquettes réelles.

Les fonctions de perte couramment utilisées pour la segmentation sémantique :

- **Categorical Cross-Entropy *(CCE)*** : cette fonction de perte est utilisée pour les modèles de classification multi-classes, Cette métrique mesure la distance entre les prédictions du modèle et les étiquettes réelles. Plus cette distance est grande, plus le modèle fait de mauvaises prédictions et plus la categorical cross-entropy sera élevée.
- **Dice Loss** : cette fonction de perte mesure la similitude entre les prédictions du modèle et les données réelles en comparant le nombre de pixels correctement classés. Plus la perte de Dice est proche de 1, meilleure est la performance du modèle.
- **Jaccard Loss** : cette fonction de perte également connue sous le nom de perte d'intersection sur l'union (IoU), mesure la similarité entre les masques de segmentation prédits et de vérité terrain en comparant les pixels correctement classés à l'ensemble total de pixels. Une valeur de 0 signifie que les images sont totalement différentes, tandis qu'une valeur de 1 signifie qu'elles sont identiques.
- **Categorical Focal Loss** : cette fonction de perte est une variante de l'entropie croisée catégorielle qui accorde une plus grande importance aux exemples qui sont difficiles à prédire.
- **Balanced Loss** : cette fonction de perte est une technique utilisée pour traiter les déséquilibres de classe dans les données d'entraînement en attribuant une importance plus élevée aux erreurs pour les classes sous-représentées.

2.6 Backbones

Un **backbone** d'un réseau de neurones est la partie principale de l'architecture qui s'occupe de la transformation des données d'entrée en un ensemble de caractéristiques abstraites. Dans le cas de la segmentation d'images, le backbone prend en entrée des images et extrait des caractéristiques pertinentes pour la tâche de segmentation.

Il existe de nombreux backbones différents qui peuvent être utilisés dans les réseaux de neurones de segmentation d'images. Certains exemples populaires incluent :

- **ResNet *(Residual Network)*** : c'est un réseau de neurones à plusieurs couches qui utilise des raccourcis de convolution (appelés "résidus") pour faciliter la propagation des gradients à travers les couches du réseau. ResNet est connu pour ses performances élevées et sa capacité à réduire l'apparition de gradients morts lors de l'entraînement.

- **DenseNet *(Densely Connected Convolutional Network)*** : c'est un réseau de neurones à convolution profondément connecté qui utilise des connections directes entre toutes les couches du réseau. Cette architecture permet de partager efficacement les informations entre les couches il est connue pour ses performances élevées et sa capacité à réduire la surcharge de données lors de l'apprentissage.

- **VGG *(Visual Geometry Group)*** : c'est un réseau de neurones à convolution qui utilise des couches de convolution de taille 3x3, il est connu pour ses performances élevées sur de nombreuses tâches de reconnaissance d'images.

- **Inception** : c'est un réseau de neurones à convolution qui utilise des couches de convolution de différentes tailles, il est connu pour ses performances élevées sur de nombreuses tâches de reconnaissance d'images.

En fonction de la taille et de la complexité des données d'entrée et des ressources de calcul disponibles, il est important de choisir un backbone appropriée pour la tâche de segmentation d'image.

3 Préparation des données

3.1 Présentation du jeu de données

Pour ce projet, nous utilisons les données du jeu de données **Cityscape**. Il s'agit d'un ensemble de données contenant des images prises à partir de voitures dans différentes villes. Le jeu de données contient également des annotations (appelées masques) pour chaque image. Le masque est aussi une image dont les pixels correspondent aux classes de l'image d'origine (piéton, nature, véhicule...)

Voici un exemple du jeu de données **Cityscape**

Figure 6 – Annotations de pixels denses de haute qualité

Le dataset est divisée en 3 parties :

— **Train** (2975 images et masques)
(permet d'entraîner les modèles)

— **Validation** (500 images et masques)
(évalue les modèles durant l'entraînement)

— **Test** (500 images)
(évalue les performances du modèle retenu)

Les masques pour le jeu de Test ne sont pas disponibles, pour remédier à ce problème, nous avons procédé à la séparation du jeu de données de la façon suivante :

— Séparation du jeu de données Train original en un jeu de données Train et Validation d'après un ratio spécifié.

— Création d'un jeu de données Test à partir du jeu de données de Validation original.

Le jeu de données Test original n'est pas utilisé.

3.2 Générateur de données

Au départ, le masque était composé de 32 sous-catégories. Comme nous avons été chargés de le réduire à 8 catégories principales pour ce projet, nous avons créé une fonction qui utilise la méthodologie de Marius Cordts, l'un des créateurs du jeu de données, pour transposer ces 32 sous-catégories en 8 catégories principales : **void, flat, construction, object, nature, sky, human et vehicle**

Pour traiter un ensemble de données volumineux qui ne peut pas être chargé en mémoire vive en une seule fois, nous avons créé une classe qui instancie un objet de type générateur. Ce générateur permet de diviser le jeu de données en batchs (lots), qui peuvent être chargés et traités séparément. Nous avons également intégré des opérations de pré-traitement automatique, comme l'augmentation de données et le redimensionnement des images, dans ce générateur. Cela nous permet de manipuler de grandes quantités de données de manière efficace et sans surcharger la mémoire vive.

3.3 Augmentation des données

L'augmentation des données consiste à générer de nouvelles données à partir des données d'entraînement existantes

Dans le cas de la segmentation sémantique d'images, l'augmentation des données peut être utilisée pour créer de nouvelles images de segmentation à partir des images d'entraînement existantes en utilisant des transformations aléatoires. Cela permet de fournir au modèle un plus grand nombre d'exemples d'apprentissage et de lui permettre de mieux généraliser aux données de test.

L'augmentation des données doit être choisie judicieusement en fonction de la tâche de reconnaissance d'image à accomplir afin de fournir au modèle des exemples qui reflètent de manière réaliste ce qu'il pourrait rencontrer dans la vie réelle.

Dans le cas de la reconnaissance d'objets sur des images, l'inversion horizontale peut être une transformation utile pour ajouter de la variabilité aux données d'entraînement. Cependant, dans le cas de la conduite autonome, l'inversion horizontale ne serait pas réaliste, car les voitures et la route ne seront pas inversées de cette manière dans la vie réelle. Il serait donc préférable d'utiliser d'autres types de transformations, comme la rotation verticale ou le zoom, qui reflètent de manière plus réaliste ce que le modèle peut rencontrer dans la vie réelle.

Afin de fournir un jeu de données d'entraînement plus étendu et améliorer les performances de généralisation du modèle, nous avons implémenté plusieurs techniques d'augmentation de données qui peuvent être pertinentes pour les images utilisées dans les systèmes de conduite autonome, afin de s'assurer que le modèle puisse gérer certaines variations et conditions :

— **Rotation :** les images de caméras embarquées sur les véhicules autonomes peuvent être capturées sous des angles différents.

— **Translation :** les images de caméras embarquées sur les véhicules autonomes peuvent également être capturées avec des décalages horizontaux et verticaux.

— **Zoom :** les images de caméras embarquées sur les véhicules autonomes peuvent être capturées avec des distances de prise de vue différentes.

— **Flou :** Les images de caméras embarquées sur les véhicules autonomes peuvent être capturées avec des niveaux de flou différents en raison de la météo, de la vitesse, de la distance de prise de vue et de la qualité de l'objectif.

— **Bruit :** Les images de caméras embarquées sur les véhicules autonomes peuvent être capturées avec des niveaux de bruit différents en raison de la météo, de la vitesse, de la distance de prise de vue et de la qualité de l'objectif.

— **Modification de couleur :** Les images de caméras embarquées sur les véhicules autonomes peuvent être capturées avec des variations de luminosité et de couleur en raison de la météo et de la luminosité ambiante.

— **Simulation de conditions météorologiques :** Permet de simuler les conditions météorologiques telles que la pluie, la neige, la brume et les reflets.

— **Simulation de la nuit :** Permet de simuler les images capturées la nuit.

Voici un exemple d'augmentation sur une image du dataset **Cityscape**

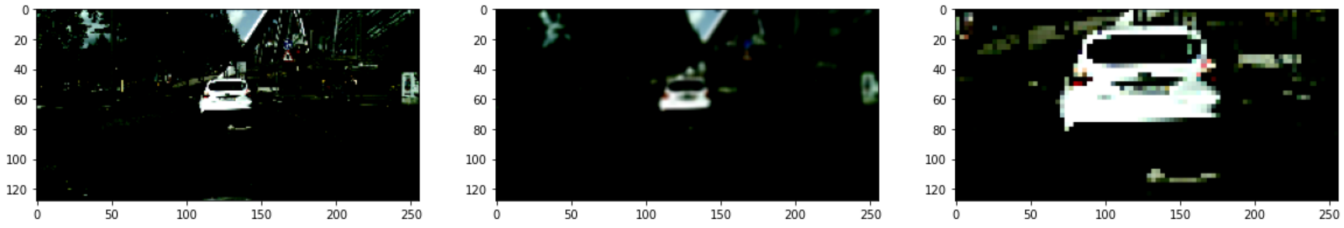

Figure 7 – Augmentation des données sur une image du dataset

A noter : l'augmentation de données peut entraîner une augmentation de la perte **(loss)** lors de l'entraînement du modèle, Cela est dû au fait que les augmentations de données modifient les données d'entraînement de manière aléatoire, ce qui peut rendre le modèle moins capable de généraliser aux données réelles. Pour y remédier, il faut tester différentes combinaisons d'augmentations de données et de paramètres d'entraînement pour trouver la meilleure configuration.

Dans notre cas, nous avons dû réduire les augmentations et utiliser seulement les augmentations suivantes :

- **Flou** *(Ajout d'un flou gaussien avec un sigma allant de 0 à 3.)*
- **Rotation vertical** *(Retourne l'image verticalement avec une probabilité de 0.5)*
- **Zoom** *(Zoom de l'image entre 50% et 150%.)*
- **Dropout2d** *(applique l'abandon avec la probabilité donnée à l'image, il définira aléatoirement une zone rectangulaire de l'image sur zéro.)*

4 Modèle retenu

Pour évaluer et comparées les performances des différents modèles de segmentation d'images, plusieurs tests ont été effectuer :

- **modèles** *(Unet mini et Unet)*
- **backbones** *(ResNet50 et VGG16)*
- **fonctions de perte** *(Categorical crossentropy et Balanced loss)*

Les résultats de ces tests sont résumés dans le tableau récapitulatif ci-dessous :

	Model	Loss Function	Loss	Accuracy	F1-score	Iou score	Time
1	Unet_mini	categorical_crossentropy	0.520794	0.841938	0.597809	0.486344	4863.944373
2	u_net_mini_aug	categorical_crossentropy	0.971151	0.635562	0.315875	0.208688	5085.685824
3	Unet	categorical_crossentropy	0.635241	0.796512	0.543981	0.417973	4247.406577
4	Unet_resnet_CC	categorical_crossentropy	0.543803	0.847009	0.679934	0.560435	3708.879752
5	Unet_vgg16_CC	categorical_crossentropy	0.520273	0.838212	0.642198	0.512760	3675.355234
6	Unet_mini_BL	balanced_loss	0.478510	0.385911	0.069537	0.048239	4467.612423
7	Unet_BL	balanced_loss	0.448479	0.822848	0.619009	0.534574	3858.366940
8	Unet_resnet_BL	balanced_loss	0.444933	0.875154	0.723645	0.632965	3832.857036
9	Unet_VGG16_BL	balanced_loss	0.446515	0.851046	0.700407	0.600885	3747.001836

Figure 8 – Tableau comparatif des résultats

Le modèle ayant obtenus les meilleurs résultats est le modèle **U-NET** avec le Backbones **ResNet50** et la fonction de perte/coût **Balanced loss**

5 Synthèse des résultats obtenus

Fonction de coût : La fonction de coût **Balanced Loss** améliore les résultats par rapport à la fonction **Categorical Cross Entropy**. Cela confirme que la fonction de coût **Balanced Loss** est bien adaptée au problème de Segmentation.

Backbone : le backbone qui donne les meilleurs résultats est un réseau ResNet50 avec les poids pré entraîné sur le jeu de données Imagenet. L'utilisation de poids pré entraîné améliore les résultats, le score IoU augmente tout en diminuant le temps d'exécution.

Architecture : l'architecture qui donne les meilleurs résultats est U-NET. Cette architecture améliore nettement les résultats : le score IoU augmente et le temps d'exécution diminue.

Le meilleur modèle est donc le suivant :

- **Fonction de coût :** Balanced Loss
- **Backbone :** ResNet50 avec poids pré entraînés sur Imagenet
- **Architecture :** U-NET

Grâce à ce modèle, nous obtenons des résultats qui améliorent nettement le modèle de référence (U-NET mini) :

- **La Loss** diminue de 0.52 à **0.44**
- **Le score IoU** augmente nettement de 0. 49 à **0.63**
- **Le temps d'exécution** baisse de 1h35mns et **1h06mns**

6 Conclusion et pistes d'amélioration

Nous obtenons un modèle de qui donne de bons résultats tout en ayant respecté les contraintes imposées.

L'augmentation de données a été testée sur le modèle de base **Unet mini** et n'a pas permis d'améliorer les performances du modèle.

Voici plusieurs **pistes d'améliorations possibles** :

- optimiser d'autres **hyper-paramètres** comme l'optimiser ***(SGD, RMSprop, Adagrad...)***
- tester d'autres versions d'augmentation des images
- tester d'autres fonctions de perte, *(cf chapitre 2.5 Métrique et fonction Loss)*
- tester une approche de type **Transformer**. On pourrait pour cela utiliser le modèle **AutoModelForSemanticSegmentation** de la librairie **Hugging Face** qui est une référence dans ce domaine.