Segmentation sémantique sur le jeu de données “Cityscapes” à l’aide de FCN, U-Net et SegNet : une approche comparative

Abdessamad DERRAZ
abdessamad@derraz.fr

Résumé. Le Deep Learning s’est avéré extrêmement efficace lorsqu’il s’agit de travailler avec des images en tant qu’informations et il est actuellement dans une phase où il fonctionne mieux que les humains dans de nombreux cas d’utilisation. Les principaux problèmes que la vision par ordinateur a permis de résoudre sont la classification des images, la segmentation des objets et la détection. Il existe plusieurs modèles préexistants qui obtiennent déjà d’excellentes performances sur le jeu de données “Cityscapes”. En conséquence, il devient très difficile pour quelqu’un qui veut travailler sur la segmentation sémantique, de choisir le bon modèle avec les bons paramètres. Dans cette note technique, nous présentons un retour approfondi sur la segmentation sémantique sur le jeu de données “Cityscapes”. Pour cette analyse, nous utilisons trois modèles préexistants différents : SegNet, U-Net, et Fully Convolutional Networks (FCN) avec VGG-16 comme encodeur. Ensuite, nous réalisons plusieurs expériences sur ces modèles et les comparons entre eux. En outre, nous utilisons également certaines techniques traditionnelles et modernes afin d’améliorer la précision des performances des modèles. Des expériences et des analyses approfondies montrent que le meilleur score est obtenue par le modèle SegNet associé à certains hyperparamètres, obtenant un Pixel Accuracy de **87,49%** et un Mean IoU de **51,67%**. Le code et les résultats sont disponibles sur `https://github.com/Abdess/Future-Vision-Transport`.

1 Introduction

Dans le traitement avancé de l’image et la vision par ordinateur, la segmentation de l’image est la méthode qui consiste à diviser une image informatisée en plusieurs sections. L’objectif de la division est de simplifier et de modifier la représentation d’une image en un contenu plus significatif et plus simple à examiner. La segmentation est couramment utilisée pour localiser des objets et des limites dans des images. C’est le moyen le plus commun d’attribuer une étiquette à chaque pixel d’une image, de telle sorte que les pixels portant un nom similaire partagent des qualités spécifiques. Le résultat de la segmentation d’une image est un ensemble de sections qui couvrent globalement l’ensemble de l’image ou un ensemble de formes extraites de l’image. Chacun des pixels d’une région est comparable en ce qui concerne une caractéristique ou une propriété enregistrée, comme la texture, la couleur ou l’intensité. Les régions adjacentes ont des couleurs sensiblement différentes en ce qui concerne la caractéristique équivalente. Lorsqu’ils sont appliqués à une pile d’images, normalement dans l’imagerie médicale, les contours qui résultent de la segmentation de l’image peuvent être utilisés pour faire des recompositions en 3D.

L’une des méthodologies prédominantes pour cette innovation est le deep learning (apprentissage profond), qui entre dans la catégorie des connaissances artificielles et qui peut imiter ou adopter le processus de développement de la réflexion d’un être humain. La méthodologie actuelle, est défini avec des centaines, voir, un grand nombre d’informations pour rendre la session d’entraînement plus productive et rapide. La principale tâche de la segmentation d’image consiste à regrouper chaque pixel d’une image appartenant à une classe spécifique, ce qui peut être considéré comme un problème de classification pour chaque pixel. En général, il existe deux types principaux de segmentation d’image : la segmentation sémantique et la segmentation d’instance. L’illustration 1 montre la différence entre la reconnaissance d’images, la segmentation sémantique, la détection d’objets et la segmentation d’instances. La segmentation sémantique est la manière la plus courante de classer chaque pixel appartenant à une étiquette particulière. Elle ne fait pas de distinction entre les différents exemples d’un élément similaire. Par exemple, dans le cas où il y a 3 moutons dans une image, la segmentation sémantique

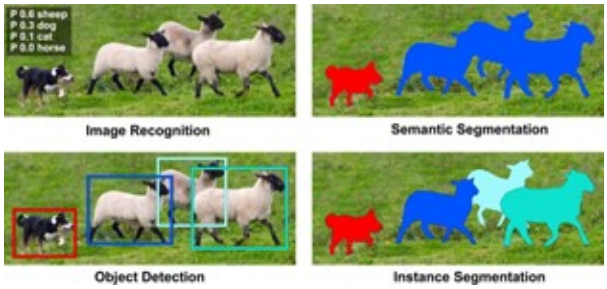

Figure 1: Différence entre la segmentation sémantique et la segmentation par instance. Image tirée de Google Image.

donne la même étiquette à tous les pixels des 3 moutons. La segmentation d'instance est différente de la segmentation sémantique, car elle donne un nouveau nom à chaque occurrence d'un élément spécifique dans l'image. Comme on peut le voir dans l'illustration, chaque mouton se voit attribuer des couleurs différentes, c'est-à-dire des étiquettes différentes. Avec la segmentation sémantique, chacun d'entre eux se serait vu attribuer une couleur similaire. Il existe de nombreuses applications de la segmentation d'images, comme la reconnaissance de l'écriture manuscrite pour extraire le texte des documents manuscrits, le mode portrait de Google Photo pour flouter l'arrière-plan dans une image, Google Meet avec les différents arrière-plans lors de visioconférence, les voitures autonomes, etc.

Dans cet essai, nous utilisons trois différents modèles préexistants tels que SegNet, U-Net et FCN avec VGG-16 comme encodeur pour la tâche de segmentation d'images sur un ensemble de données de paysages urbains. En outre, nous utilisons plusieurs techniques afin d'améliorer la précision du modèle préexistant, comme l'augmentation traditionnelle et moderne des données, différentes fonctions d'activation, différents optimiseurs, tuning et taux d'apprentissage. Les principales interventions de cet essai sont les suivantes :

- Nous réalisons des expériences approfondies avec plusieurs modèles préexistants de segmentation sémantique sur le jeu de données "Cityscapes" et présentons une comparaison globale et précise entre eux.
- Nous utilisons également différentes techniques sur chaque modèle pour améliorer encore les performances.

2 Travaux connexes

De nombreux travaux ont déjà été réalisés sur la segmentation sémantique sur le jeu de données "Cityscapes". Dans cette section, nous allons décrire une brève introduction de tous les modèles préexistants utilisés dans cette analyse.

2.1 SegNet

SegNet [1] se veut une architecture productive pour la division sémantique par pixel. Cette architecture est motivée par des applications de compréhension de scènes de rue qui requièrent la capacité de montrer l'apparence (rue, bâtiment), la forme (voitures, piétons) et de comprendre la relation spatiale (contexte) entre différentes classes comme la rue et le trottoir. Dans les scènes de rue moyennes, la plupart des pixels ont une place avec des classes énormes comme la rue, le bâtiment et donc le réseau doit créer une division sans accrocs.

L'illustration 2 décrit l'architecture globale du modèle SegNet. Il comporte un réseau encodeur et un réseau décodeur associé, suivis d'une dernière couche de classification par pixel. Le réseau encodeur comprend 13 couches convolutives qui correspondent aux 13 couches convolutives initiales du réseau VGG16 destinées à la classification des objets. Elles disposent de couches entièrement connectées pour conserver des cartes de caractéristiques de plus haute résolution à la sortie la plus profonde de l'encodeur. Cela réduit également le volume des données dans le réseau d'encodage SegNet (de 134Mo à 14,7Mo) par rapport à d'autres structures récentes. Chaque couche encodeur a une couche décodeur correspondante et, par conséquent, le réseau décodeur compte 13 couches. La dernière sortie du décodeur est prise en charge par un classificateur soft-max multi-classes pour générer librement des probabilités de classe pour chaque pixel.

2.2 Fully Convolutional Network

L'architecture globale d'un CNN comprend quelques couches convolutionnelles et de pooling suivies de quelques couches complètement connectées vers la fin. Une étude sur les réseaux entièrement convolutifs [2] soutient que la dernière couche "Fully-connected" peut être considérée comme effectuant une convolution 1x1 qui couvre toute la région. Par conséquent, les dernières couches denses peuvent être supplantées par une couche de convolution accomplissant un résultat similaire. Quoi qu'il en soit, l'avantage actuel de cette méthode est que la taille de l'entrée ne doit plus être fixe. En incluant des couches denses, la taille de l'information est obligatoire, et dorénavant, lorsqu'une autre information mesurée est donnée, elle doit être redimensionnée. Mais en supplantant une couche dense par une convolution, cette exigence n'existe pas. De plus, lorsqu'une plus grande taille de l'image est donnée comme information, le résultat fourni sera une carte de caractéristiques et en plus un rendement de classe comme pour une image typique estimée par l'information. De même, le comportement remarqué de la dernière carte de caractéristiques s'adresse à la carte de caractéristiques de la classe nécessaire, c'est-à-dire que la situation de l'article figure dans la carte de caractéristiques. Puisque le résultat de la composante, la carte, est une carte de caractéristiques de l'article nécessaire, il s'agit de données légitimes pour notre exemple d'utilisation de la division.

Étant donné que la carte de caractéristiques obtenue au niveau de la couche de résultat est examinée à la baisse en raison de l'arrangement des convolutions effectuées, nous devrions l'échantillonner à la hausse en utilisant une méthode supplémentaire. L'échantillonnage ascendant bilinéaire fonctionne, mais l'article propose d'utiliser l'échantillonnage ascendant appris avec la déconvolution qui peut même devenir familière avec un échantillonnage ascendant non linéaire. La partie descendante du réseau est appelée encodeur

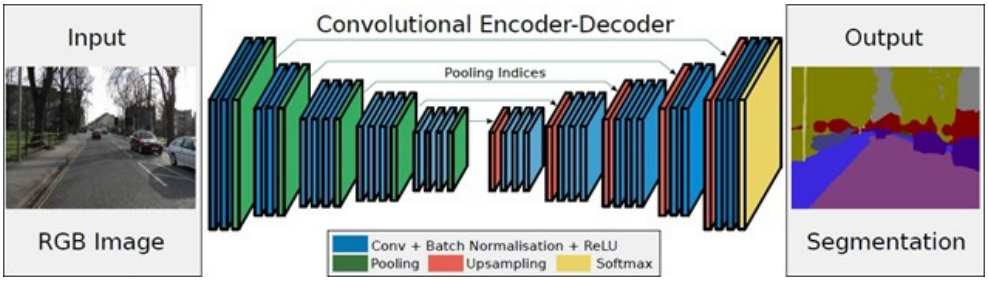

Figure 2: Architecture du modèle SegNet.

et la partie de suréchantillonnage est appelée décodeur. C'est un exemple que nous verrons dans de nombreuses structures, c'est-à-dire la diminution de la taille avec l'encodeur et ensuite l'inspection vers le haut avec le décodeur. Dans un monde idéal, nous ne voudrions pas sous-échantillonner en utilisant le pooling et garder une taille similaire tout au long du processus, mais cela demanderait une quantité énorme de limites et serait infaisable d'un point de vue informatique. Dans notre analyse, nous utilisons comme encodeur le VGG-16 entraîné sur la classification ImageNet [3].

2.3 U-Net

U-Net [4] s'étend sur le réseau entièrement convolutif. Il a été construit à des fins cliniques pour dépister les croissances dans les poumons ou le cerveau. Il comprend également un encodeur qui sous-échantillonne l'image d'entrée en une carte de caractéristiques et un décodeur qui suréchantillonne le guide d'éléments pour inclure la taille de l'image en utilisant des couches de déconvolution apprises. Nous avons vu plus haut dans FCN que puisque nous sous-échantillonnons une image en tant que composant de l'encodeur, nous avons perdu une quantité de données qui ne peuvent pas être récupérées efficacement dans la partie encodeur. FCN tente de résoudre ce problème en prenant les données des couches de mise en commun avant la dernière couche de composants.

U-Net propose une autre façon de s'occuper de ce problème de déficit de données. Il propose d'envoyer des données à chaque couche de suréchantillonnage dans le décodeur en commençant par la couche de sous-échantillonnage correspondante dans l'encodeur. De cette façon, on peut obtenir de meilleures données tout en maintenant le calcul à un niveau bas. Étant donné que les couches situées au début de l'encodeur disposent de plus de données, elles renforcent l'activité de suréchantillonnage du décodeur en fournissant des détails précis sur les images d'entrée, ce qui améliore considérablement les résultats. L'article propose en outre l'utilisation d'un travail astucieux sur les erreurs, que nous allons examiner ci-dessous.

3 Méthode

Dans cette section, nous décrivons toutes ces techniques utilisées avec les modèles préexistants afin d'augmenter leur précision.

Figure 3: Un exemple de technique de recadrage aléatoire. Image tirée de Google Images.

Figure 4: Un exemple de la technique de retournement horizontal. Image tirée du jeu de données "Cityscapes".

3.1 Augmentation des données

C'est une technique utilisée pour augmenter la quantité de données en ajoutant des doublons quelque peu modifiés de données existantes. Il existe plusieurs méthodes pour y parvenir, mais dans notre analyse, nous nous sommes concentrés sur quelques-unes d'entre elles, telles que le recadrage aléatoire, les retournements horizontaux, la normalisation et la teinte. L'augmentation des données n'est pas seulement utilisée pour éviter l'overfitting. En règle générale, le fait de disposer d'un énorme jeu de données est crucial pour les performances des modèles d'apprentissage automatique (ML) et d'apprentissage profond (DL). Nous pouvons nous efforcer d'améliorer les performances du modèle en élargissant les données. Cela implique que l'augmentation des données est en outre utile pour améliorer les performances du modèle et joue un rôle important pour une meilleure précision des modèles. Dans cette étude, nous utilisons toutes les techniques d'augmentation de données mentionnées afin d'améliorer les performances du modèle.

Figure 5: Un exemple de technique de variation des couleurs avec un changement aléatoire de la saturation, de la teinte et du niveau de luminosité de l'image. Image prise de google images.

3.1.1 Recadrage aléatoire

Le recadrage aléatoire est une stratégie d'augmentation des données dans laquelle nous produisons un sous-ensemble aléatoire d'une image originale. Cela permet à notre modèle de mieux se généraliser car les objets d'intérêt dont nous avons besoin pour l'apprentissage de nos modèles ne sont pas toujours entièrement visibles dans l'image ou à une échelle similaire dans nos données d'apprentissage. L'illustration 3 montre un exemple de technique de recadrage aléatoire.

Par exemple, lorsque nous réalisons un modèle de Deep Learning qui reconnait les véhicules dans une rue, nous pourrions avoir une caméra qui transmet des images à un modèle de découverte d'éléments révélant quand un véhicule est apparent. Les véhicules ne sont pas toujours entièrement dans le cadre, et ils ne sont pas non plus toujours à la même distance de la caméra. Un recadrage aléatoire est une décision intéressante comme procédure d'expansion pour cette situation. Pour cette raison, nous utilisons le recadrage aléatoire avec un padding de 4 pour améliorer encore les performances de chaque modèle.

3.1.2 Retournement horizontal

Le retournement implique la rotation d'une image sur un axe horizontal ou vertical. Dans le cas d'un retournement horizontal, le retournement se fait sur l'axe vertical, et dans le cas d'un retournement vertical, le retournement se fait sur l'axe horizontal. Le retournement de lignes entières et de sections de pixels d'une image dans une direction horizontale est appelé augmentation par retournement horizontal. Nous utilisons ensuite cette technique pour ajouter plus de variétés dans l'ensemble de données. L'illustration 4 montre un exemple de technique de retournement horizontal réalisée sur le jeu de données cityscapes.

3.1.3 Normalisation

La normalisation est une méthode de prétraitement des données utilisée pour porter les données mathématiques à une échelle typique sans en déformer la forme. En général, lorsque nous entrons les informations dans un calcul de machine learning ou de deep learning, nous modifions assez souvent les valeurs à une échelle raisonnable. La raison pour laquelle nous normalisons est en quelque sorte de garantir que notre modèle peut se résumer de manière appropriée. L'optimisation est plus rapide car la normalisation ne permet pas aux poids de s'envoler dans tous les sens et les limite à une plage spécifique.

3.1.4 Variation des couleurs

La variation des couleurs est l'une des stratégies d'augmentation des données d'image. Elle consiste à modifier de manière aléatoire la luminosité, la teinte et la saturation de l'image. HSL (Hue, Saturation, Lightness) est une autre représentation du modèle RVB. Il présente une structure de couleur géométrique courbée qui est le début d'un cylindre. La luminosité et la saturation sont des lignes droites et la chromaticité a une forme ronde.

La teinte représente la couleur. 0 degré est rouge, 120 degrés est vert, et 240 degrés est bleu. Elle a une plage de 0 à 360 (degrés). La saturation est une couleur plus distinctive est appelée à être plus saturée, et la couleur est plus probablement un ombrage achromatique, par exemple, le gris ou le blanc, ou le noir est censé être moins saturé. Lorsqu'il s'agit d'une teinte à forte saturation, l'expression "sombre" est régulièrement utilisée. En revanche, lorsqu'il s'agit d'une teinte à faible saturation, on utilise régulièrement l'expression "flou". L'échelle de saturation est comprise entre 0 et 100%. La clarté désigne la luminosité d'une couleur. Elle a une plage comprise entre 0 et 100%. L'illustration 5 montre un exemple de technique de variation de couleur appliquée à une image. L'image de gauche est l'image originale et l'image de droite est le résultat après application de différentes valeurs de teinte, de saturation et de luminosité. De même, nous appliquons de manière aléatoire différentes valeurs de teinte, de saturation et de luminosité à différentes images pour augmenter l'ensemble de données, ce qui accroît progressivement les performances du modèle.

3.2 Fonction d'activation

Une fonction d'activation est une fonction qui est ajoutée à un réseau de neuronne artificiel pour aider le réseau à apprendre des modèles complexes dans les données. Par rapport à un modèle à base de neurones qui se trouve dans le cerveau humain, la

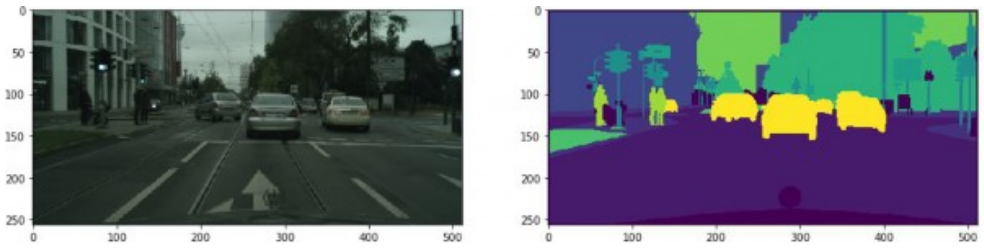

Figure 6: Un exemple du jeu de données “Cityscapes” avec son ground truth.

fonction d’activation conclut vers la fin ce qui doit être envoyé au neurone suivant. Les architectures préexistantes utilisent un certain type de fonction d’activation, comme SegNet qui utilise la fonction d’activation “Relu” et U-Net qui utilise la fonction d’activation “ELU”. Nous essayons également de remplacer ces fonctions d’activation par d’autres pour voir l’effet sur les performances du modèle.

3.3 Optimiseurs

Les optimiseurs intègrent la fonction de perte et les paramètres du modèle en rafraichissant le modèle en réponse au résultat de la fonction de perte. En termes moins techniques, les optimiseurs façonnent et moulent le modèle dans sa structure la plus précise possible en jouant avec les poids. La fonction de perte est le ”GPS” du terrain, indiquant à l’optimiseur quand il va dans la bonne ou la mauvaise direction. Elle est utilisée pour maximiser l’efficacité ou pour minimiser la perte. Différentes architectures utilisent différents optimiseurs, par exemple, certaines utilisent l’optimiseur SGD et d’autres l’optimiseur Adam. Pour voir l’effet sur la performance, nous changeons également l’optimiseur et rapportons les résultats.

3.4 Taux d’apprentissage

Le taux d’apprentissage est un hyperparamètre configurable utilisé dans la préparation des réseaux neuronaux. Il a une petite valeur en positif, quelque part dans la plage de 0,0 et 1,0. Le taux d’apprentissage contrôle la rapidité avec laquelle le modèle s’adapte au problème. Les petits taux d’apprentissage nécessitent des époques d’entraînement supplémentaires étant donné le nombre de petits changements apportés aux charges à chaque mise à jour, tandis que les taux d’apprentissage plus élevés entrainent des changements rapides et nécessitent moins d’époques de préparation.

Un taux d’apprentissage trop élevé peut faire converger le modèle trop rapidement vers une solution sous-optimale, tandis qu’un taux d’apprentissage trop faible peut bloquer le processus. Le défi de tout apprentissage de réseaux neuronaux de deep learning consiste à choisir avec précaution le taux d’apprentissage. Il pourrait s’agir de l’hyperparamètre le plus important pour le modèle. C’est pourquoi nous essayons également différents taux d’apprentissage pour chaque modèle afin d’évaluer les performances.

4 Expériences

Pour valider les performances des différents modèles avant et après l’application de toute modification. Nous menons des expériences approfondies sur le jeu de données Cityscapes. Dans cette section, nous précisons d’abord les détails de la configuration expérimentale qui comprend l’environnement, le jeu de données, les modèles, les hyperparamètres et les métriques utilisés dans cette étude, puis les résultats expérimentaux qui montrent l’analyse quantitative et qualitative de chaque modèle et une comparaison détaillée complète.

4.1 Dispositif d’expérimentation

Nous réalisons nos expériences sur un seul GPU (NVIDIA GTX 1070) ayant une mémoire de 8 GB avec CUDA version 11.7. Notre implémentation est totalement basée sur Tensorflow avec Python 3.9 comme langage de programmation. Certains programmes sont exécutés sur colab et d’autres sur l’environnement PyCharm.

4.1.1 Jeu de données

Nous utilisons le jeu de données cityscapes, qui comprend diverses scènes de rues métropolitaines dans 50 villes différentes à des années différentes, ainsi que des vérités de terrain pour de multiples tâches de vision, y compris la division sémantique, la segmentation au niveau de l’instance et l’inférence de disparité de couple stéréoscopique. L’illustration 6 montre un exemple de l’ensemble de données Cityscapes avec son étiquette réelle. Pour les tâches de segmentation, Cityscapes fournit des annotations denses au niveau des pixels pour 5000 images à une résolution de 1024 * 2048, préalablement divisées en ensembles d’apprentissages (2975), de validation (500) et de tests (1525). Les étiquettes annotées pour les tâches de segmentation couvrent plus de 30 classes telles que personne, voiture, camion, route, etc., généralement rencontrés lors du repérage de scènes de conduite.

Modèle	Activation	Optimiseur	Taux d'apprentissage	Augmentations	Accuracy (%)	IoU (%)	Temps
SegNet	ReLu	Adam	10^{-4}	Sans	86.47	50.65	46m00s
SegNet	ReLu	Adam	10^{-4}	Avec	**87.49**	**51.67**	50m45s
SegNet	ELU	Adam	10^{-4}	Avec	84.73	47.63	52m47s
SegNet	ELU	Adam	10^{-2}	Avec	79.08	44.21	56m21s
SegNet	ELU	SGD	10^{-4}	Avec	65.31	39.54	57m54s
SegNet	ELU	Adam	10^{-5}	Avec	79.64	46.71	55m07s
U-Net	ReLu	Adam	10^{-4}	Sans	84.33	47.04	**13m37s**
U-Net	ReLu	Adam	10^{-4}	Avec	85.01	46.98	15m08s
U-Net	ELU	Adam	10^{-4}	Avec	82.65	43.62	16m32s
U-Net	ELU	Adam	10^{-2}	Avec	33.06	27.36	18m47s
U-Net	ELU	SGD	10^{-4}	Avec	56.32	39.89	20m41s
U-Net	ELU	Adam	10^{-5}	Avec	75.71	38.95	19m28s
FCN-8s	ReLu	Adam	10^{-4}	Sans	83.68	30.13	17m05s
FCN-8s	ReLu	Adam	10^{-4}	Avec	83.92	30.21	20m18s

Table 1: Les chiffres en **gras** indiquent le meilleur score.

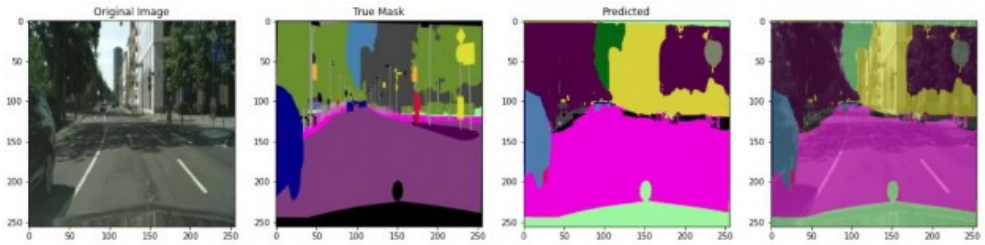

Figure 7: Résultat du modèle SegNet

4.1.2 Modèles

Comme nous l'avons vu dans la section consacrée aux travaux connexes, nous utilisons trois modèles préexistants différents tels que SegNet, U-Net et FCN avec VGG-16 comme bloc encodeur. Nous utilisons ces modèles avec différents hyperparamètres et observons leurs performances. Nous combinons également ces modèles avec certaines techniques de régularisation.

4.1.3 Hyperparamètres

Dans cette étude, nous utilisons différents paramètres pour différents modèles afin de mesurer les résultats de chacun d'entre eux. Pour SegNet et U-Net, nous exécutons nos programmes avec les fonctions d'activation ReLu et ELU, les optimiseurs Adam et SGD, la perte d'entropie croisée catégorielle, avec et sans techniques d'augmentation des données, et différents taux d'apprentissage (10^{-4}, 10^{-2}, 10^{-5}). Nous utilisons un Learning Rate Schedule, si les étapes sont inférieures à 10 000, le taux d'apprentissage est de 10^{-4}, si les étapes sont entre 10 000 et 20 000, le taux d'apprentissage est de 10^{-5}, si le taux d'apprentissage est entre 20 000 et 40 000, le taux d'apprentissage est de 3×10^{-6}, et pour toute autre valeur, le taux d'apprentissage est de 10^{-6}. Nous utilisons des techniques d'augmentation des données et tous les autres paramètres sont tirés de la publication [3]. Nous exécutons les modèles sur 10 epochs. Pour plus de renseignements, nous avons rédiger un notebook Jupyter dans le dépôt github dont le lien est mentionné dans la section "Introduction".

4.1.4 Métriques

Nous utilisons 2 métriques d'évaluation, Pixel Accuracy et Mean IoU. Le pixel accuracy pour évaluer une segmentation sémantique qui rapporte le pourcentage de pixels de l'image qui ont été classés avec précision. Il s'agit de la mesure la plus facile à comprendre sur le plan conceptuel, par rapport aux autres mesures d'évaluation. La précision des pixels est normalement rapportée pour chaque classe indépendamment, tout comme au plan global pour toutes les classes. En considérant la précision des pixels par classe, nous évaluons essentiellement un masque binaire ; un vrai positif correspond à un pixel dont on prévoit avec précision qu'il appartient à la classe donnée, tandis qu'un vrai négatif correspond à un pixel qui est effectivement distingué comme n'appartenant pas à la classe donnée.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Ici, TP est la fréquence à laquelle la valeur réelle et la valeur prédite sont toutes deux vraies, et TN est la fréquence à laquelle la valeur réelle et la valeur prédite sont toutes deux fausses. FP signifie la fréquence à laquelle la valeur réelle est fausse ou la

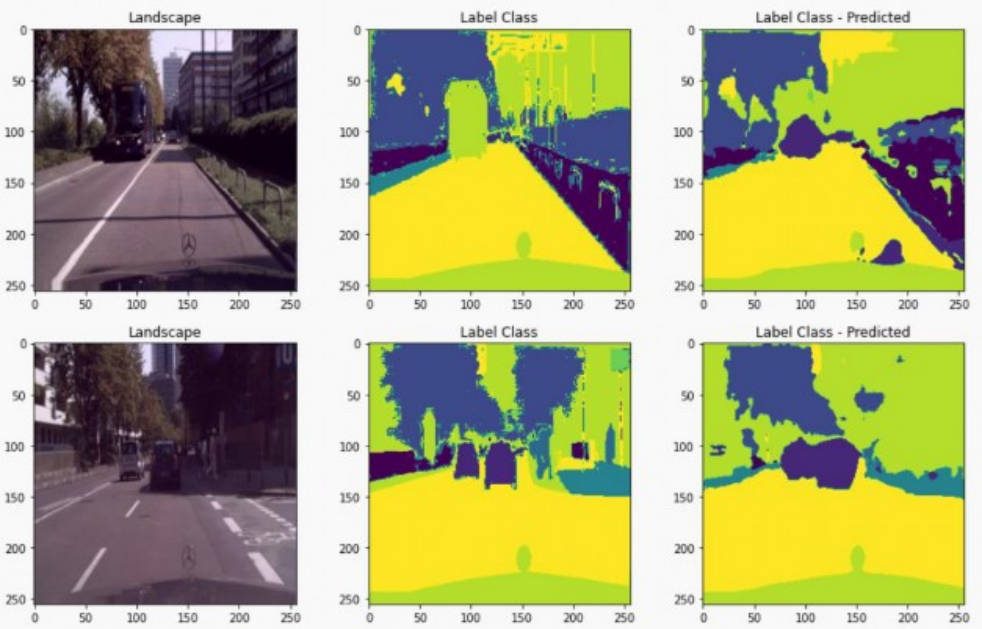

Figure 8: Résultats du modèle U-Net

valeur prédite est vraie, et FN signifie la fréquence à laquelle la valeur réelle est vraie ou la valeur prédite est fausse.

En complément, nous utilisons la métrique Intersection over Union (IoU), également appelée indice de Jaccard, est essentiellement une méthode permettant de quantifier le pourcentage de chevauchement entre le masque cible et notre sortie de prédiction. Cette métrique est étroitement liée au coefficient de Dice qui est souvent utilisé comme fonction de perte pendant un apprentissage.

Pour faire simple, la métrique IoU mesure le nombre de pixels communs entre le masque cible et le masque de prédiction, divisé par le nombre total de pixels présents dans les deux masques.

$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

Le score IoU est calculé pour chaque classe séparément, puis la moyenne est calculée pour toutes les classes afin de fournir un score IoU global et moyen de notre prédiction de segmentation sémantique. C'est probablement la métrique la plus importante pour les modèles de segmentation sémantique,

4.2 Résultat de l'expérience

Nous conduisons nos expériences sur quatre bases pour chaque modèle. Premièrement, nous exécutons le modèle en changeant la fonction d'activation, comme nous utilisons ReLu et ELU pour notre étude. Deuxièmement, nous exécutons le modèle en changeant l'optimiseur. Dans notre cas, nous utilisons Adam et SGD. Troisièmement, nous exécutons le modèle en changeant les taux d'apprentissage à une valeur plus grande et plus petite et quatrièmement en ajoutant ou non des techniques d'augmentation des données. Nous discuterons d'abord de l'analyse quantitative des résultats obtenus avec les différents modèles, puis de l'analyse qualitative.

4.2.1 Analyse quantitative

Le tableau 1 montre le score de chaque modèle avec différents hyperparamètres. Tout d'abord, nous exécutons le modèle SegNet avec toutes les différentes variantes. La meilleure précision a été obtenue par la fonction d'activation ELU avec un taux d'apprentissage de 10^{-4} avec l'optimiseur Adam et les techniques d'augmentation des données, obtenant un score de 87,49%. Cependant, le changement de l'optimiseur en SGD n'a pas été concluant, car ses performances ont chuté de manière drastique à 65,31%. En augmentant ou en diminuant les taux d'apprentissage à 10^{-2} et 10^{-5} respectivement, on obtient 79,08% et 79,64%, ce qui est une perte de performance. L'exécution du modèle SegNet avec la fonction d'activation ReLu, l'optimiseur Adam, avec un taux d'apprentissage de 10^{-4}, et sans techniques d'augmentation des données donne un score de 86,47%. Cette dégradation de la performance montre l'importance de la technique d'augmentation des données pour la segmentation sémantique. Ensuite, pour le modèle U-Net, la meilleure performance a été obtenue avec les mêmes paramètres que ceux obtenus pour le modèle SegNet. Avec la fonction ELU, l'optimiseur Adam, le taux d'apprentissage 10^{-4} et l'augmentation des données, il obtient un score de 84,33%. Cependant, en changeant le taux d'apprentissage à 10^{-2}, on obtient une très mauvaise performance avec une précision de 33,06% seulement. En changeant l'optimiseur en SGD avec la fonction ELU, un taux d'apprentissage de 10^{-4} et

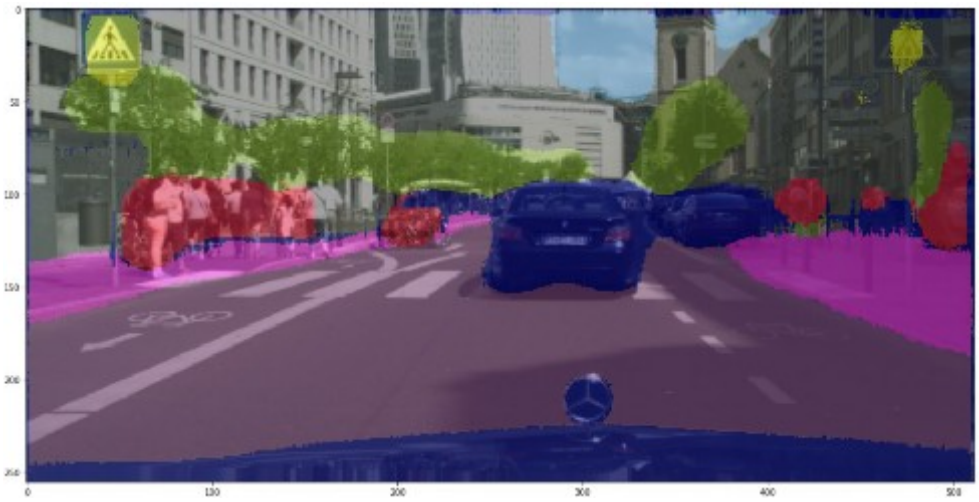

Figure 9: Résultat du modèle FCN

une technique d'augmentation des données, on obtient un score de 56,32%, ce qui n'est pas aussi mauvais qu'avec un taux d'apprentissage de 10^{-2}. De même, lorsque nous essayons d'augmenter le taux d'apprentissage à 10^{-5}, cela donne une précision de 75,71%, ce qui est une précision acceptable par rapport à la quantité de données entraîné. Puis avec la fonction d'activation ReLu, l'optimiseur Adam, un taux d'apprentissage de 10^{-4} et sans technique d'augmentation des données, on obtient un score de 84,33%. Enfin, pour le modèle FCN avec les techniques d'augmentation des données, nous avons un score de 83,92% qui est un excellent score par rapport aux autres modèles.

Après avoir comparé tous les modèles avec différents hyperparamètres, il semble que le modèle avec la fonction d'activation ReLu, l'optimiseur Adam, le taux d'apprentissage 10^{-4} et en utilisant l'augmentation des données, les performances du modèle augmentent.

4.2.2 Analyse qualitative

L'illustration 7 montre la sortie d'une image du modèle SegNet, où la première image est l'image originale, la deuxième est son ground truth, la troisième est la segmentation prédite dans l'image et la quatrième est la superposition de la prédiction avec l'image original. On peut clairement voir que le modèle SegNet est performant. L'illustration 8 montre la sortie de deux images du modèle U-Net, où la première colonne est l'image originale, la deuxième est l'image du ground truth et la troisième est la prédiction par le modèle U-Net. La performance du modèle U-Net légérement comparable au modèle SegNet. L'étiquette de classe prédite est presque égale au ground truth. Enfin, cette illustration 9 montre la sortie du modèle FCN et ce modèle réussit à distinguer les piétons, les panneaux de signalisation et les voitures. Globalement, il montre une bonne performance pour la segmentation sémantique.

5 Conclusion

Dans cette étude, nous avons étudié trois différents modèles préexistants tels que SegNet, U-Net et FCN avec VGG-16 comme encodeur pour la tâche de segmentation d'images sur le jeu de données "Cityscapes". De plus, nous avons essayé différentes techniques afin d'améliorer les performances. Nous avons également présenté une comparaison complète et détaillée entre trois DCNNs différents exécutés avec différents hyperparamètres. Nous avons également utilisé différentes techniques afin d'améliorer les modèles préexistants et comparé les performances des modèles entre eux. Après avoir comparé chaque modèle avec chaque technique, nous avons constaté que le modèle SegNet avec l'optimiseur Adam, le taux d'apprentissage de 10^{-4}, la fonction d'activation ReLu et les techniques d'augmentation des données donne le meilleur résultat par rapport aux autres modèles, et permet d'obtenir le score le plus élevé avecc un Pixel Accuracy de **87,49%** et un mean IoU de **51,67%**.

6 Travaux futurs

À l'avenir, nous chercherons à mener davantage d'expériences avec différentes techniques. En raison des ressources limitées, telles que le nombre de GPU et la mémoire, l'entraînement d'un seul modèle prend davantage de temps et, en raison de la courte période de temps, nous n'avons pas pu tester plus de techniques et de modèles différents.

References

[1] Badrinarayanan, V., Handa, A., and Cipolla, R. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling.* 2015. DOI: 10 . 48550 / ARXIV . 1505 . 07293. URL: https://arxiv.org/abs/1505.07293.

[2] Long, J., Shelhamer, E., and Darrell, T. *Fully Convolutional Networks for Semantic Segmentation*. 2014. DOI: 10.48550/ARXIV.1411.4038. URL: https://arxiv.org/abs/1411.4038.

[3] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[4] Ronneberger, O., Fischer, P., and Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597.