

## **Note technique projet**

### **« Participez à la conception d'une voiture autonome »**

## **Introduction:**

Le projet « Participez à la conception d'une voiture autonome » se rapporte à la conception des systèmes embarqués de vision par ordinateur pour les véhicules autonomes. Il consiste principalement en la segmentation des images qui est alimentée par le bloc de traitement des images et qui alimente le système de décision.

Les objectifs retenus sont résumés dans les points suivants :

- Mise en disposition des images en format png provenant du jeu de données « Cityscapes Dataset » et contenant deux archives pour les images et les masques associés. Les annotations distinguent 35 sous catégories qui doivent être fusionnées en 8.
- Préparation d'un data generator pour la manipulation de jeux de données très volumineux.
- Entraînement d'un modèle de Deep Learning sur des images en utilisant Azure Machine Learning et Keras comme framework
- Préparation d'une API simple d'utilisation qui prend en entrée l'identifiant d'une image et renvoie la segmentation de l'image de l'algo et de l'image réelle.
- Déploiement d'une API Flask grâce au service Azure, affichée par Streamlit et qui sera utilisée par le système de décision.

## **Implémentation d'un générateur de données:**

Jusqu'au projet 7, le chargement de données a été relativement simple puisque les étapes se résumaient principalement à la récupération d'un dataframe à convertir en array numpy puis sa lecture.

Dans ce projet c'est plus compliqué. Les fichiers sont contenus dans des répertoires différents qu'il faut aller les récupérer à des endroits un peu partout. L'extraction de données est plus complexe compte tenu de leur taille volumineuse d'où l'intérêt d'un data générateur. Il s'agit d'une classe qui

regroupe un ensemble d'attributs et de méthodes qui sert lors du fit d'un réseau de neurones. Ainsi, les données seront extraites directement par un ensemble de méthodes.

Pour ce projet avant de passer directement au data générateur un pré-traitement des fichiers a été effectué pour faciliter leurs utilisations. Les deux archives `P8_Cityscapes_gtFine_trainvaltest` et `P8_Cityscapes_leftImg8bit_trainvaltest` sont associées aux images en RGB d'entrée et aux masques d'annotation respectivement [1]. Elles comportent la même architecture contenant des données de train, test et validation pour les différentes villes. L'association image/mask est faite grâce à un identifiant image unique « `image_id` ».

Dans le pré-retraitement :

(1) les paires image/mask sont regroupées sous le même répertoire en respectant la séparation du jeu de données en jeu d'entraînement, en jeu de test et en jeu de validation.

(3) réadaptation des labels pour passer de 35 à 8. Par défaut ils utilisent les Id et dans ce projet le catId a été adopté.

**TABLEAU : RÉADAPTATIONS DE CATÉGORIES [1]**

Nouvelles catégories (catIds)	Ancienne catégorie (Ids)
Flat	Road - sidewalk - parking - rail track
Human	Person - Rider
Vehicle	Car - truck - bus - on rails - motorcycle - bicycle - caravan - trailer
Construction	Building - wall - fence - guard rail - bridge - tunnel
Pole	Pole - pole group - traffic sign - traffic light
Nature	Vegetation terrain
Sky	Sky
Background	Ground - dynamic -static

Le script du générateur de données est entièrement automatisé et permet le traitement des images sur plusieurs cœurs de calcul.

Dans le générateur de données:

(1) redimensionnement de la taille de l'image pour améliorer les performances du réseau de neurones.

(2) l'application ou non de l'augmentation des données.

(3) la normalisation des images.

- (4) L'application de l'encodage « one hot encoding » sur les labels du masque.
- (5) l'application ou non du shuffle pour mélanger les images.

## **Segmentation d'images:**

On appelle segmentation d'images l'opération consistant à identifier les structures d'intérêt dans cette image, c'est à dire la délimitation des objets la composant.

L'approche utilisée dans ce projet cherche à identifier des régions de pixels homogènes au sein de l'image suivant un critère d'homogénéité. Le résultat se présente soit sous la forme d'une image binaire (exemple: route ou pas route), soit d'une image étiquetée, où chaque étiquette ou label correspond à une région. Dans notre projet, la segmentation multiclasse délimite nos 8 différentes catégories à savoir les panneaux, les piétons, les trottoirs, les voitures, la végétation, la construction, le ciel ou le background.

*Un problème de classification pixellique où chacun des pixels de l'image doit fournir une catégorie.*

Pour ce faire, il y a recours au deep learning et ses différentes architectures dont les algorithmes sont à la base de CNN, devenus une référence en traitement d'image depuis plusieurs années.

## **Architectures**

### **1) UNET:**

Le U-Net est une architecture à réseau convolutionnel pour une segmentation rapide et précise des images (cf figure 6). Il a la particularité d'avoir une architecture symétrique composée d'une partie de "contraction", qui va permettre de détecter le contexte et les objets, et une partie "expansion" composée de succession d'up-convolution qui fait le chemin inverse qui va permettre de reconstituer l'image composée de macro-features extraites de la partie "contraction" et finalement de localiser précisément le contour des objets détectés.

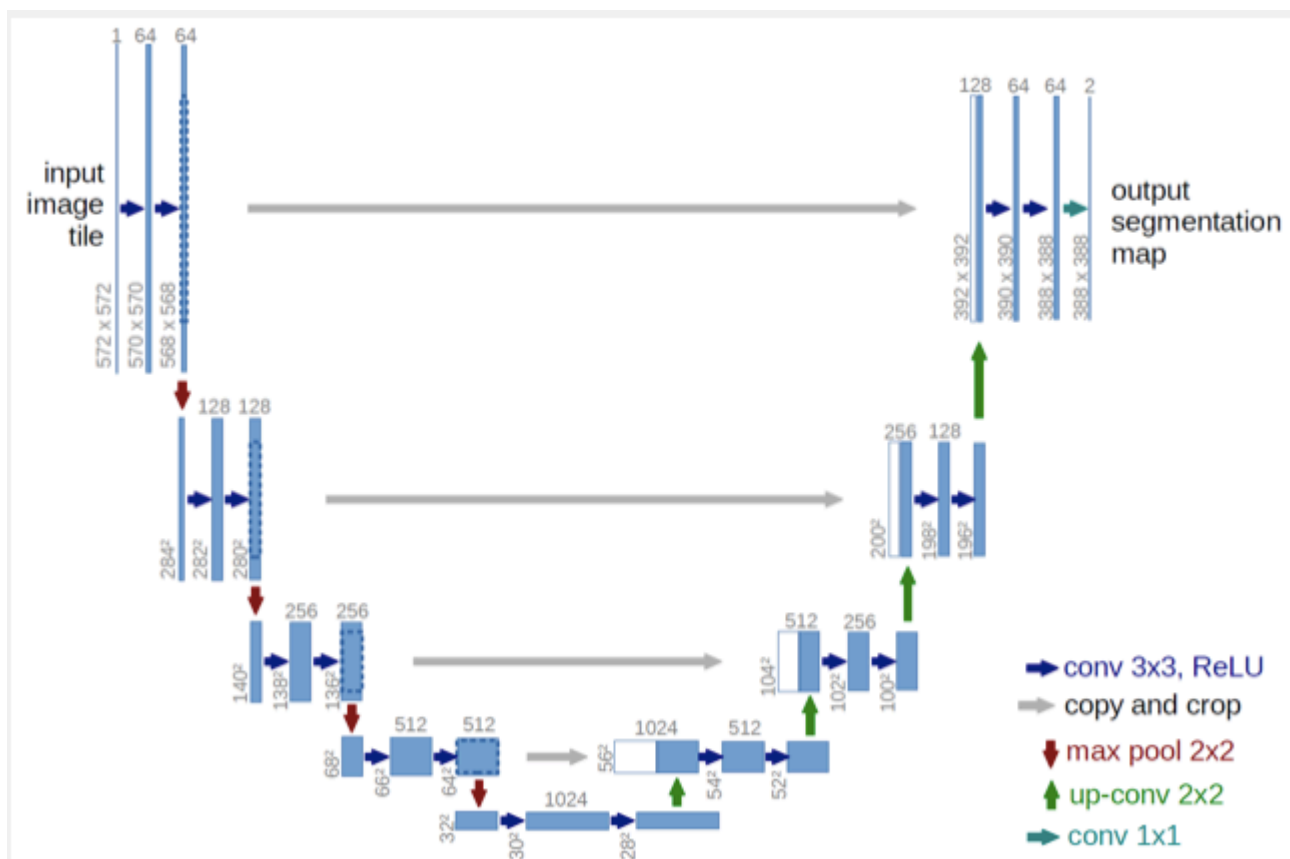


FIGURE: ARCHITECTURE UNET

Comme le montre la figure Architecture Unet, Unet a la forme d'un réseau en U. En entrée on a une image, on applique par la suite des opérations de convolution 3 X3 en bleu puis des max pools pour réduire les dimensions pour arriver jusqu'au moment d'obtention d'un vecteur carrément de profondeur 1024. La partie gauche est la partie encodeur équivalent au réseau de neurones classique. Cette partie gauche on peut la remplacer par la suite par un réseau de neurones qui a été déjà entraîné sur d'autres données.

Par la suite, il y a la partie décodeur qui à l'aide d'opération up-sampling ou de dé-convolution inversée va ré-augmenter la dimension spéciale réinventer l'effet de max pool pour avoir un masque de segmentation composé de valeurs entre 0 et 8.

A la fin L2 la sortie il y a deux canaux. Dans le cas de ce projet on aura besoin de 8. Pour chacun des pixels il va y avoir un vecteur de taille 8. Il va

Zeineb Guizani

indiquer la probabilité que le pixel appartient à telle catégorie. Au lieu de faire une prédiction on fait autant de prédiction qu'il y a de pixels. Donc en sortie, ce sera un masque de taille 8 avec l'opération argmax qui retient la catégorie la plus forte.

Dans ce projet on a adapté une implémentation déjà faite où :

- (i) on a modifié la sortie en remplaçant la valeur nombre de filtre en sortie par 8 et supprimer la conv 9 afin de passer directement de 64 à 8.
- (ii) On a remplacé sigmoid par softmax.

## 2) Unet de segmentation models:

Utilisation d'une architecture un peu plus moderne où les librairies de segmentation models fournissent des Unets dont l'encodeur est un réseau de neurone pré-entraîné afin d'améliorer les performances.

Il est à noter que les librairies segmentation models ont l'avantage de modifier l'architecture en plus de l'encodeur. Ces autres architectures sont Linknet, FPN et PSPNet mais le choix dans ce projet s'est fait sur l'architecture UNet légendaire en segmentation d'images et de vérifier plutôt l'importance des encodeurs.

Le choix du type du backbone dépend des données et ressources de calcul disponibles, de la tâche à effectuer et du type d'utilisation du modèle.

**TABLEAU COMPARATIF BACKBONES**

Utilisation	Architecture du backbone	Avantage
La construction d'un modèle pour une utilisation en routine clinique	Xception, MobileNet et EfficientNet	compromis entre consommation des ressources et niveau de précision.
La classification des images où l'information spatiale est fortement discriminante	Squeeze-and-ExcitationNet et Inception	S'adaptent mieux à la forme discriminante.
La classification de plusieurs structures anatomiques	ResNet ou DenseNet	Architecture robuste et performante

## **Stratégies d'apprentissage**

Afin de mieux converger et en un temps plus court, nous avons adopté la stratégie du transfert learning.

Le transfert learning consiste à transférer les connaissances acquises d'un modèle lors de la résolution d'un problème généraliste à un problème différent, plus spécifique mais connexe. Par exemple, les connaissances acquises en apprenant à reconnaître les voitures pourraient s'appliquer lorsqu'on essaie de reconnaître les camions. Cela consiste concrètement à transférer les poids d'un réseau à un autre.

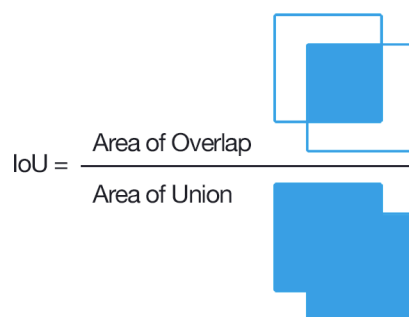
Nous sommes partis des poids du réseaux Resnet34 entraîné sur le dataset ImageNet à classifier 1000 catégories d'objets. Puis le réseau EfficientNetb7 qui a la particularité d'atteindre à la fois une plus grande précision et une meilleure efficacité par rapport aux CNN existants, réduisant la taille des paramètres.

La présentation des résultats de cette partie sera faite après l'augmentation.

## Métriques d'évaluation:

1) La détection des objets dans une image est principalement fondée sur un calcul d'IOU (Intersection over Union), une métrique classique en traitement d'image.

Le ratio entre l'aire d'intersection et l'aire d'union) est la valeur utilisée dans la détection d'objets pour mesurer le chevauchement d'un cadre de sélection prévu par rapport au cadre réel d'un objet. On dispose ainsi d'un indicateur permettant de déterminer si la prédiction recouvre pertinemment la ground truth.



**FIGURE : METRIQUE IOU**

La valeur de l'IOU est comprise entre 0 et 1. 0 est obtenu lorsque les deux ensembles ne se superposent pas (deux ensembles disjoints c'est le plus mauvais). Quand c'est parfaitement aligné on obtient 1.

Cette métrique sera calculée pour toutes les images à prédire et sera suivie au cours de l'apprentissage. Pour ne pas avoir à l'implémenter, utiliser la librairie `segmentation models`.

2) Hormis IOU, un autre indicateur le temps nécessaire pour l'entraînement du modèle a été calculé pour comparer les modèles.

## Fonctions de Coût:

Une fonction de perte, ou Loss function, est une fonction qui évalue l'écart entre les prédictions réalisées par le réseau de neurones et les valeurs réelles des observations utilisées pendant l'apprentissage. Plus le résultat de cette fonction est minimisé, plus le réseau de neurones est performant.

Le tableau suivant montre une sélection de fonctions de coût:

**TABLEAU : FONCTIONS DE COÛT**

Fonction de perte	Utilisation
Binary Cross-Entropy	la classification binaire, ou classification multi-classes ou l'on souhaite plusieurs label en sortie. Utilisable aussi pour de la segmentation sémantique. Fonction basé sur la distribution de Bernoulli. Référence de base à utiliser comme première fonction pour tout nouveau réseau. Non adapté à des datasets biaisé/déséquilibré.
Categorical Cross-Entropy	classification multi-class à simple label de sortie.
Weighted Cross-Entropy	Pour des datasets biaisé/déséquilibré
Balenced Cross-Entropy	Pour des datasets biaisé/déséquilibré.
Focal	Pour des datasets TRES biaisé/déséquilibré.
Pixel-Wise Cross-Entropy	Référence de base à utiliser comme première fonction pour tout nouveau réseau de segmentation d'image. Non adapté à des datasets biaisé/déséquilibré.
Dice	Inspiré des coefficients Dice, métriques utilisé pour évaluer la performance d'un réseau de segmentation sémantique d'image en mesurant le chevauchement entre deux objets.  - problématiques de segmentation d'image
Tversky	Très apprécié pour des problématiques de segmentation d'image.
Focal Tversky	Très apprécié pour des problématiques de segmentation d'image.
BCE-Dice	Très apprécié pour des problématiques de segmentation d'image.
Jaccard / Intersection Over Union (IoU)	Très apprécié pour des problématiques de segmentation d'image.

Fonction de perte	Utilisation
Lovasz Hinge	Conçu comme étant une IoU optimisé à des fin de segmentations d'images à multi-classes.

Pour ce projet le choix s'est effectué sur Cross Entropie et Dice.

## Augmentations de données:

Une technique qui consistent à appliquer plusieurs transformations d'images pour créer artificiellement de nouvelles données.

Le tableau suivant récapitule les différents résultats obtenus suivant les différentes techniques utilisées à savoir: rotation, changement d'échelle, ajout de bruit, ajout brouillage, décalage de couleurs, changement d'éclairage, etc.

Il est à noter que l'architecture du réseau est un Unet simple avec et sans augmentation.

**TABLEAU COMPARATIFS DES AUGMENTATIONS**

Augmentation	IoU	Loss	Temps de Traitement
Aucune	<b>0.620</b>	0.341	2 h 58 m
RandomGamma(p=0.25), Blur(p=0.25, blur_limit=7), GaussNoise(p=0.5)	0.599	0.365	3 h 9 m
HorizontalFlip(p=0.5), RandomGamma(p=0.25), Blur(p=0.25, blur_limit=7)	0.613	0.337	3 h 25 m
HorizontalFlip(p=0.5), RandomGamma(p=0.25),	<b>0.626</b>	0.362	3 h 26 m
RandomGamma(p=0.25), Blur(p=0.25, blur_limit=7),	<b>0.630</b>	0.390	3 h 25 m
ShiftScaleRotate(), RGBShift(), Blur(), GaussNoise()	0.536	0.407	3 h 24
RGBShift(), RandomGamma(p=0.25), Blur(p=0.25, blur_limit=7),	0.588	0.364	3 h 23



Augmentation	IoU	Loss	Temps de Traitement
RandomGamma(p=0.25), GaussNoise(p=0.5), Blur(p=0.25, blur_limit=7),	0.606	0.349	3 h 23 m
ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=45, p=0.2), HorizontalFlip(p=0.5), RandomGamma(p=0.25), GaussNoise(p=0.2), Blur(p=0.25, blur_limit=7)#,	0.577	0.369	3 h 24 m
A.HorizontalFlip(p = 0.5), # apply horizontal flip to 50% of images OneOf( [ # apply one of transforms to 50% of images A.RandomContrast(), # apply random contrast A.RandomGamma(), # apply random gamma A.RandomBrightness(), # apply random brightness ], p = 0.5 ) OneOf( [ # apply one of transforms to 50% images A.ElasticTransform( alpha = 120, sigma = 120 * 0.05, alpha_affine = 120 * 0.03 ) A.GridDistortion(), A.OpticalDistortion( distort_limit = 2, shift_limit = 0.5 ) ], p = 0.5 )	0.591	0.330	3 h 25

Les résultats montrent que les techniques simples d'augmentation de données (visualisées en rouge) sont plus performantes que celles complexes et dépassent les performances du modèle sans augmentation (visualisées en vert).

## Unet pré-entraînés:

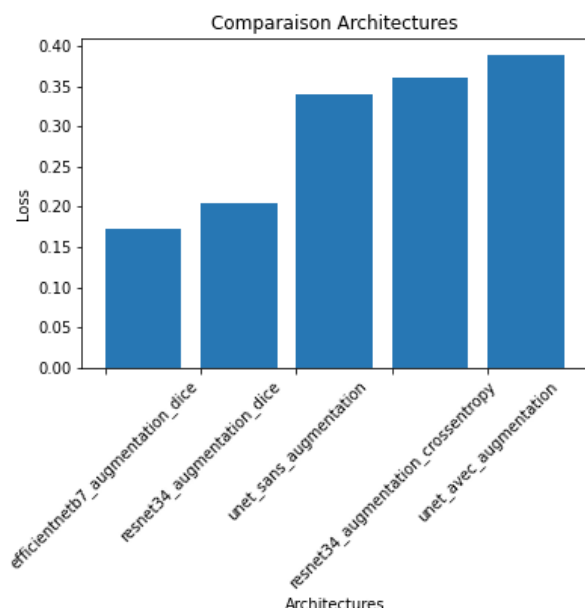
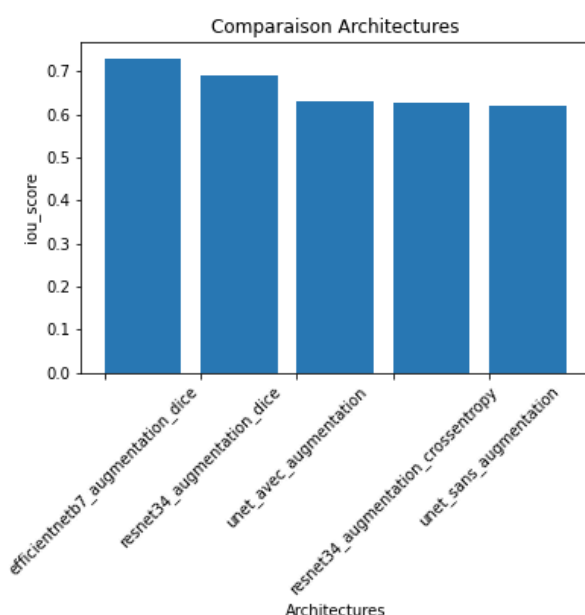
Dans cette partie il y a utilisation des architecture d'UNet de la librairie segmentation models comprenant les meilleures augmentations présentées dans la partie Augmentation de Données.

Au départ, le test est effectué avec la fonction de coût cross entropie (CE) et l'architecture UNet avec le backbone resnet34. Les performances d'IoU obtenues sont moins bonnes que l'Unet simple avec augmentation bien que les pertes se sont améliorées d'où l'idée de l'utilisation de la fonction de pertes Dice. Et là les résultats se sont améliorés en fonction de l'IoU, loss et le temps de traitement.

En poussant les expériences plus loin, une expérience avec les mêmes paramètres mais utilisant le backbone efficientb7 a été lancée et elle a donné les meilleurs résultats. C'est ce modèle qui a été choisi pour le déploiement sur la plateforme Azure Machine Learning.

TABLEAUX DE PERFORMANCE DES ARCHITECTURES

Architecture	IoU	Loss	Temps de Traitement
UNet simple sans augmentation	0.620	0.341	2 h 58 m
UNet simple avec augmentation	0.630	0.390	3 h 25 m
backbone_resnet34_CE	0.626	0.361	1 h 0 m 35
backbone_resnet34_Dice	0.690	0.205	1 h 15 m
backbone_efficientnetb7_Dice	0.731	0.172	4 h 36 m



## Interface graphique Streamlit de l'API Flask déployée grâce au service Azure

### Streamlit Image Segmentation using flask

Welcome to machine learning model APIs!

#### List of images


Select Image Id

frankfurt\_000001\_004327


frankfurt\_000001\_004327

Predict Mask


#### Results



Input image



Ground truth mask



Predicted mask

## Conclusion et perspectives

Pour récapituler, l'objectif de ce projet est de produire un modèle à base de réseaux de neurones qui permet de segmenter les images. Il a nécessité la comparaison de plusieurs types d'architectures de réseaux de neurones pour segmenter, l'utilisation de différentes augmentations de données ainsi que le test de deux fonctions de coût.

Commençant par l'implémentation d'un générateur de données, on a dérivé à la fin du projet un modèle de segmentation déployé sur azure machine Learning utilisant Flask qui est une librairie python qui permet de créer des apis et Streamlit pour la génération d'une interface qui sert à envoyer les images.

Comme perspectives, il est nécessaire d'envisager une étude plus poussée des architectures et de les tester afin de trouver la meilleure. De même, il est important de trouver une solution pour pallier la dégradation de la prédiction suite à la petite résolution d'images. Entre autres, il est indispensable de prendre en considération le temps de prédiction car les applications qui utilisent ce type d'api nécessitent des réponses en temps réel.

## References:

[1] WWW: Web page of Cityscapes Dataset, <https://www.cityscapes-dataset.com/dataset-overview/>

[2] WWW: Web page of the Intersection over Union (IoU) for object detection, <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

[3] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham.

[4] zhixuhao, unet, GitHub repository, <https://github.com/zhixuhao/unet/blob/master/model.py>

[5] Fonctions de perte/coût (loss function), Bastien Maurice, Jan 2021, <https://deeplylearning.fr/cours-theoriques-deep-learning/fonctions-de-perte-cout-loss-function/>