

SEGMENTATION SÉMANTIQUE D'IMAGE

ROMAIN BOYRIE

17 Février 2022

CONTENTS

1	Segmentation Sémantique	2
2	Les Réseaux Convolutifs	2
3	Les Réseaux et ImageNet	4
4	Les Encodeurs-Décodeurs	6
5	Nos Masques R-CNN	6
6	Data Augmentation	7
7	Data Generator	7
8	Comparaison	9
9	Conclusion	9
10	Annexe	10

LIST OF FIGURES

Figure 1	Chronologie des Algorithmes d'Apprentissages	3
Figure 2	Les Réseaux Convolutifs sur ImageNet	4
Figure 3	U-Net	4
Figure 4	DeepLabV3 plus	5
Figure 5	L'architecture encoder-decoder	6
Figure 6	Data Augmentation	7
Figure 7	Factory Pattern et Data Generator	8
Figure 8	Mesure des Performances	10

LIST OF TABLES

Table 1	Comparaison de quelques optimisations	9
Table 2	Tableau d'expérimentations	11

ABSTRACT

Ce document détaille l'étendue des prolégomènes du domaine de la segmentation sémantique d'image sur une architecture basée en réseau neuronal profond. À cet effet, nous avons parcouru, pour les résumer, des papiers scientifiques, en profitant de leur disponibilité, sur le site Arxiv. À ceux-là nous avons ajouté une présentation de nos modèles en exposant notre démarche et nos résultats. Dans un premier temps, nous expliquons pourquoi le champ de la segmentation d'image est lié aux réseaux convolutifs en passant en revue l'état de l'art (**SOTA**) en le contextualisant avec la recherche, de manière synthétique et pour notre cas particulier, i.e. avec des réseaux comme Xception ou ResNet 50. Dans un second temps, nous parlons de la préparation de nos données et de quelques-unes des solutions que nous avons élaborées en apprentissage profond. Enfin, dans un troisième temps, nous présentons les résultats de l'architecture qui se démarque dans nos expériences, **DeepLabV3**, laquelle a réalisé 62.1% **mIoU** (mean of intersection over union) et 71% de coefficient **Sørensen-Dice** sur la base Cityscapes, avec une augmentation de données.

1 SEGMENTATION SÉMANTIQUE

À partir des **R-CNN**, La segmentation sémantique classifie chaque pixel en un nombre fixe de catégories mais sans différentiation du nombre d'instance, ce paramètre n'étant pas pris en compte dans la tâche de segmentation. Cette dernière traite de l'identification ou de la classification d'objets similaires en une seule et même classe au niveau du pixel, bien qu'il puisse se trouver plusieurs classes d'objets dans une image.

Les Masques R-CNN, qui sont la traduction en calque d'une segmentation d'image, ont été construits en utilisant le **Faster R-CNN**. Toutefois, alors que le Faster R-CNN a 2 sorties pour les objets (un label de classe et un cadre délimitant l'objet), les Masques R-CNN sont l'addition d'une troisième branche dont la sortie est le masque d'objet(s).

Ce masque de sortie est distinct de la classe des cadres délimiteurs d'objets. Ils requièrent l'extraction d'une délimitation en plus, de l'objet dans l'espace, par un calque. Les Masques R-CNN sont une extension des Faster R-CNN et fonctionnent en ajoutant une branche de prédiction d'un calque d'objet (Région d'Intérêt) en parallèle d'une branche existante, car cet ajout délimite les limites encadrant l'objet reconnu dans l'image.

2 LES RÉSEAUX CONVOLUTIFS

LeNet est le premier réseau convolutif, lequel fut créé par Bell Labs. La fin des années 1980 et le début des années 1990 sont le moment du développement des réseaux de neurones multicouches.

Cependant, ce n'est qu'en 2012 que cette technologie montre concrètement des résultats impressionnantes. C'est cette année-là que l'équipe de Geoffrey Hinton de l'Université de Toronto devient le SOTA avec 16% d'erreurs sur la base ImageNet (5 catégories, 1,3 million d'images). Le réseau convolutif est, à ce moment-ci, programmé sur une carte GPU.

Le réseau convolutif a des similarités avec la vision du cortex des mammifères. Ces derniers détectent des motifs quelque-soient leurs positions dans l'image (invariance par translation : Pooling), et affectent des millions de neurones à cette tâche et pour toutes les orientations de motifs, lesquels sont les contours des objets

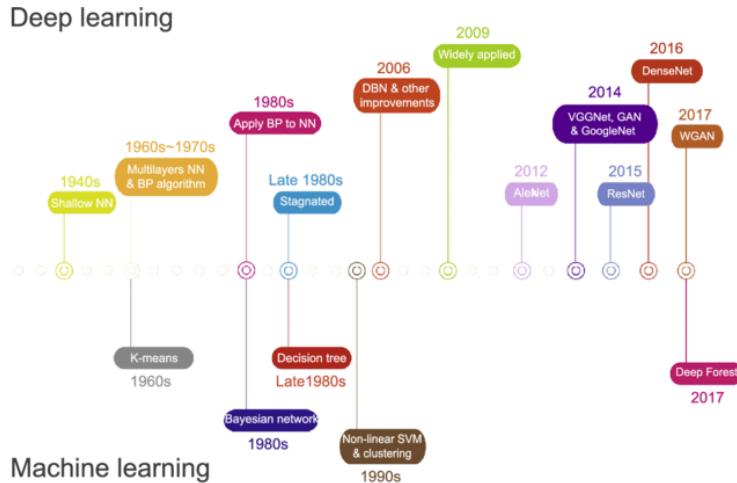


Figure 1: Chronologie des algorithmes d'apprentissages.

de l'image. Grâce au chercheur japonais Kunihiko Fukushima, qui a travaillé en allant plus loin que le modèle d'Hubel et Wiesel¹, un pas est franchi : la progression dans le réseau passe de couche en couche et entraîne la dernière couche mais la rétropropagation n'existe pas encore[1].

Ces travaux, dont on ne comprend pas bien la cause, sont peu acceptés par la communauté scientifique. Pourtant, ils seront une source d'inspiration pour la création des extracteurs de caractéristiques **SIFT** ou **HOG**. Notons que ces algorithmes ne sont pas des réseaux de neurones mais une série de calculs complexes.

Les réseaux convolutifs mélangeant dans leur architecture, des cellules simples et complexes (semblables au cortex visuel), mais l'entraînement du système est muni d'une rétropropagation de gradient, donc chaque couche est entraînée pour la reconnaissance de contours.

Chaque couche de convolution utilise une fenêtre se déplaçant sur l'image, dans laquelle chaque neurone va tenter de détecter un motif qui lui est propre, partout dans l'image. Cela donne une feature map (carte de caractéristique). Chaque feature map de sortie, est la somme de convolutions effectuées sur les "feature maps" d'entrée avec des noyaux différents.

Ci-joint le programme d'une couche complète de convolution [2] :

```

1 def convlayer(X, W, Y):
2     for u in range(len(Y)): # boucle sur les f.maps de Y
3         for v in range(len(X)):
4             conv(X[v], W[u,v], Y[u])

```

L'opération de convolution passe dans un **ReLU** pour mettre en évidence ce qui est détecté, et c'est naturellement que le ReLU est suivi par un Pooling, puisque ce dernier permet de produire des représentations d'invariants, lesquels améliorent la sortie des neurones (malgré les petites variations à l'intérieur d'une fenêtre).

L'architecture typique d'un réseau convolutif est :

Conv -> ReLU -> Pool -> Conv -> ReLU -> Pool -> Conv -> ReLU -> Conv

¹. i.e. Ils prouvaient que les aires du cortex visuel primaire jouent le rôle d'un extracteur de caractéristique.

3 LES RÉSEAUX ET IMAGENET

Il y a plusieurs réseaux existants, lesquels ont représentés le SOTA². Pour notre segmentation d'image, nous avons codé implicitement un apprentissage standard sur une structure U-Net avec Xception³, mais nous avons poussé plus loin, c'est-à-dire que nous avons utilisé la méthode de [Transfer learning](#) avec le réseau ResNet-50 sur une structure de DeepLabV3 plus.

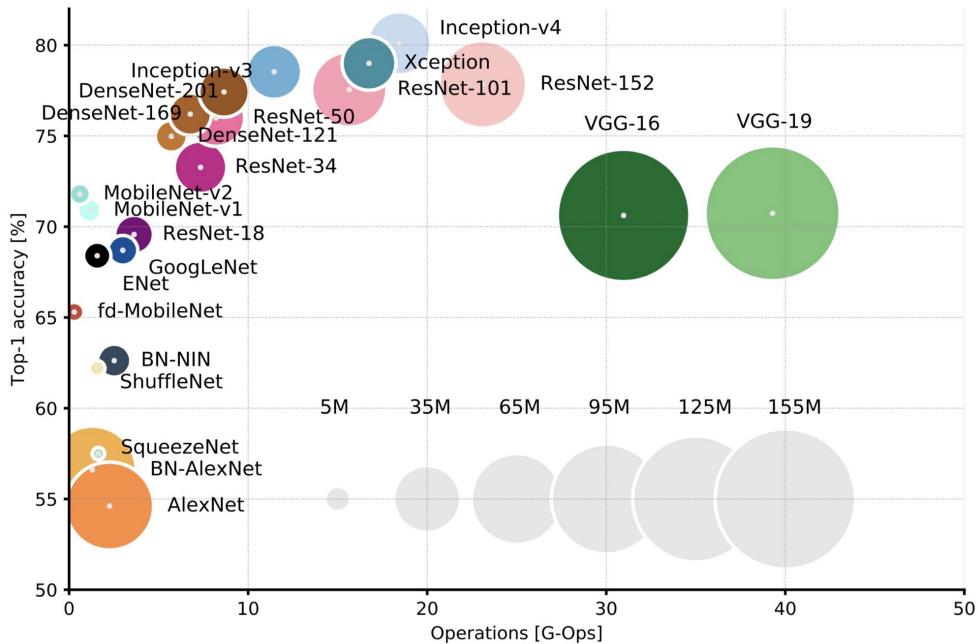


Figure 2: Les réseaux convolutifs d'un type particulier selon accuracy sur ImageNet).

U-NET Il a été proposé par le département d'informatique de l'Université de Fribourg en Allemagne. C'est un réseau convolutif. C'est un réseau qui se compose d'une partie contractante et d'une voie expansive.

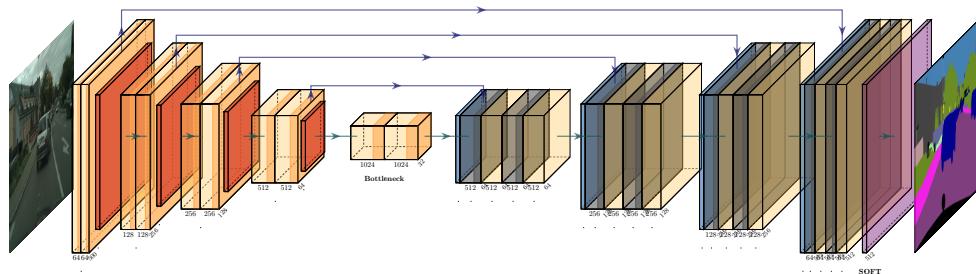


Figure 3: Un réseau U-Net pour Mask R-CNN. Le réseau encode puis décode l'image

XCEPTION Ce modèle représente une interprétation des modules Inception dans les réseaux neuronaux de convolution. Pour être simple, il marque une étape intermédiaire entre la convolution régulière et l'opération de convolution séparable en profondeur^[3] (une convolution profonde suivie par une convolution par point). Inversement, une convolution séparable en profondeur peut être comprise comme un module d'Inception avec un grand nombre de tours maximum.

Cette observation a mené à proposer une nouvelle architecture de réseau neuronal profond de convolution inspirée par Inception, où les modules d'Inception ont été

remplacés avec les convolutions séparables profondes.

Cette architecture, surnommé Xception, dépasse légèrement Inception V3 sur la base ImageNet (pour laquelle le réseau Inception V3 avait été fabriqué), et dépasse significativement Inception V3 sur des bases de données de classification d'image plus grande comprenant 350 millions d'images et 17000 classes.

Comme l'architecture Xception a le même nombre de paramètres qu'Inception V3, les gains de performance sont essentiellement dus à l'augmentation des capacités plutôt qu'à un usage plus efficient des paramètres du modèle, d'où la réalisation du SOTA.

RESNET-50 Il a été proposé par Kaiming He, un chercheur du laboratoire de Microsoft-Research à Beijing. C'est un standard de la reconnaissance d'image. Sa particularité est de comporter des connexions « saute-mouton » que l'on appelle connexions résiduelles[2], et qui court-circuitent des paires de couches.

DEEPLABV3 PLUS Primo, il est composé d'une convolution Atrous[4] qui permet de contrôler explicitement la résolution à laquelle les réponses de features sont calculées à l'intérieur du DCNN (Deep Convolutional Neural Networks). Ceci permet aussi d'élargir significativement le champ de vue des filtres à incorporer pour de plus large contexte sans augmentation du nombre de paramètres ou du nombre de calcul.

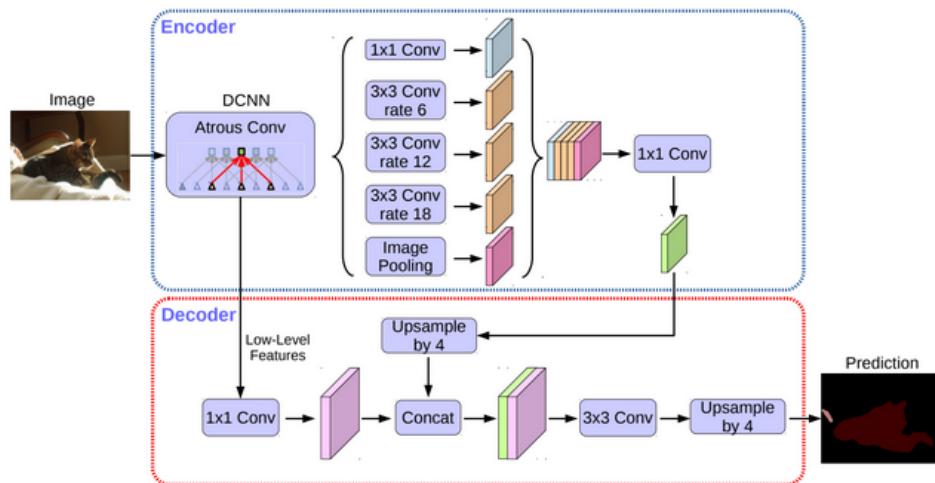


Figure 4: Le réseau neuronal DeepLabV3 plus

Secundo, il propose ASPP pour [Atrous Spatial Pyramid Pooling](#), qui segmente robustement des objets à de multiples échelles. ASPP pourvoit une couche convective de caractéristiques, en entrée, avec des filtres à des taux de multiple échantillonnage et des champs de vue efficaces, bien que capturant les objets aussi bien que le contexte de l'image à de multiples échelles.

Tertio, la localisation des limites d'objets est améliorée en combinant des méthodes des DCNN et des modèles graphiques probabilistes. Le déploiement commun des combinaisons de max-pooling et de downsampling dans le DCNN aboutit à une invariance mais celle-ci impacte l'exactitude sur la localisation.

Le réseau atténue ce problème en combinant les réponses à une couche finale DCNN avec un fully connected Conditional Random Field (CRF), laquelle donne des améliorations qualitatives et quantitatives dans la performance de localisation.

Le système "DeepLabV3 plus" règle le SOTA sur PASCAL VOC-2012 de la tâche de segmentation sémantique d'image, atteignant 79.7% mIoU dans le jeu de test, et avance des résultats sur 3 autres bases : PASCAL-Context, PASCAL-Person-Part, et Cityscapes.

4 LES ENCODEURS-DÉCODEURS

Nos réseaux utilisent l'architecture d'encodeur-décodeur[5]. Le premier prend une entrée de taille variable et la conformise à un format, le second mappe l'état encodé (dans un état contraint) vers une séquence de longueur variable.

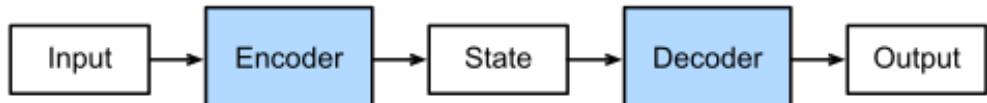


Figure 5: L'architecture encoder-decoder

Dans la bibliothèque Keras, son intégration se fait comme suit :

```

5  class EncoderDecoder(tf.keras.Model):
6      """Classe de base d'une architecture d'encodeur-décodeur"""
7      def __init__(self, encoder, decoder, **kwargs):
8          super(EncoderDecoder, self).__init__(**kwargs)
9          self.encoder = encoder
10         self.decoder = decoder
11
12     def call(self, enc_X, dec_X, *args, **kwargs):
13         enc_outputs = self.encoder(enc_X, *args, **kwargs)
14         dec_state = self.decoder.init_state(enc_outputs, *args)
15         return self.decoder(dec_X, dec_state, **kwargs)
  
```

5 NOS MASQUES R-CNN

Pour réaliser notre travail de segmentation d'image, nous avons réduit le nombre de classes de nos masques à 8, et ceci de façon astucieuse : Plutôt que de "mapper" chaque pixel des masques à 32 dimensions, nous avons utilisé les fichiers de cotation de nos polygones, ces premiers, qui sont au format JSON, nous leur avons "mapper" les catégories dans un tableau de valeur, sérialisé lui-aussi en JSON. Finalement, ce tableau est d'une dimension plus raisonnable que le nombre de pixel d'une image, qui dans Citiscapes, est de 1024x2048 pixels.

Il ne nous restait ensuite plus qu'à utiliser la fonction fillPoly d'openCV pour créer nos nouveaux masques, ceux-ci portant seulement les 8 dimensions nécessaires à notre travail de segmentation d'image. Cette technique nous a permis d'économiser beaucoup de temps de calcul. Nous avons élaboré cette dernière en recherchant comment optimiser le nombre de lecture/écriture sur le disque dur pour limiter le nombre d'accès disque.

```

16 for pic_num in range(len(layer_list)):
17     # pic_num : boucle de chacun de nos fichiers json à 8 catégories
18     dfl = pd.read_json(layer_list[pic_num])
19     h = dfl.at[0, 'imgHeight']
20     w = dfl.at[0, 'imgWidth']
21     img_mask = np.zeros(np.hstack((h, w)), dtype='uint8')
22     for poly_num in range(len(dfl)):
23         label = dfl.at[poly_num, 'objects']['label']
  
```

```

24     poly = np.array([dfl.at[poly_num, 'objects']['polygon']])
25     img_mask = cv2.fillPoly(img_mask, poly, dict_label_clr[label])

```

Grâce à ces préparatifs, nous avons obtenu un jeu de données d'entraînement et de validation. Pour pallier aux 500 masques de tests complètement opaques de la base fournie, nous avons tiré aléatoirement 500 images à chaque entraînement parmi les données d'entraînement. Les images furent isolées ipso facto, car une utilisation post-apprentissage est requise en vue de réaliser les mesures sur données de test, donc des performances (mIoU et Sørensen-Dice de modèle de Deep Learning).

6 DATA AUGMENTATION

Pour réaliser l'augmentation de données nous avons réalisé des retouches d'images en série pour toutes les images de l'entraînement, soit 2475 images. Nous avons testé plusieurs bibliothèques avant de retenir `imgaug`. Nous avons été limité dans notre choix par la différence de dimension entre les images d'entrée et de sortie, elle ne respecte pas la structure (la signature) de la méthode centrale de la classe `ImageDataGenerator` de la bibliothèque TensorFlow. Cependant, l'égalisation des dimensions d'entrée et de sortie semble être une piste pour travailler la data augmentation en temps réel, même si cela nous rapprocherait de la méthode dont on use pour un réseau GAN. Par contre, dans notre situation, nous avons chargé le double du nombre de fichier originaux sur le disque-dur, du moins dans les apprentissages qui ont eu lieu en mode "augmentation de données".

```

26 from imgaug import augmenters as iaa
27 seq = iaa.Sequential([
28     iaa.Affine(rotate=(-25, 25)), # Rotation des images
29     iaa.AdditiveGaussianNoise(scale=(15, 60)), # Bruit
30     iaa.Crop(percent=(0, 0.2)), # Retire des lignes/colonnes
31     iaa.Dropout([0.05, 0.2]), # Enlève des pixels uniformément
32     iaa.Sharpen((0.0, 1.0)), # Augmente les contours
33     iaa.ElasticTransformation(alpha=50, sigma=5) # Bouge les pixels localement
34 ], random_order=True)
35 segmap = SegmentationMapsOnImage(segmap, shape=image.shape)

```



Figure 6: Data Augmentation des Images et des Masks R-CNN

7 DATA GENERATOR

Pour permettre le chargement de nos données en cachant la complexité du code, nous avons créé plusieurs classes et un fichier regroupant les fonctions statiques. Ainsi notre classe `DataGenerator` nous permet-elle d'utiliser une écriture claire avec un switch dans une fonction appelante `__call__` et des méthodes utilisant le pattern d'objet Factory pour masquer la complexité du code dans notre classe, laquelle ne contient que le concept auquel nous faisons référence lors du chargement de nos data, dont le code de l'action plus bas niveau se trouve dans un fichier regroupant les fonctions statiques (tools). Dans notre code, c'est le cadriel DVC qui joue le rôle de commande à travers un pattern "Command" cette fois, mais l'idée reste la même bien que le concept soit de plus haut niveau.

Les deux patterns sont exposés dans le schéma UML suivant :

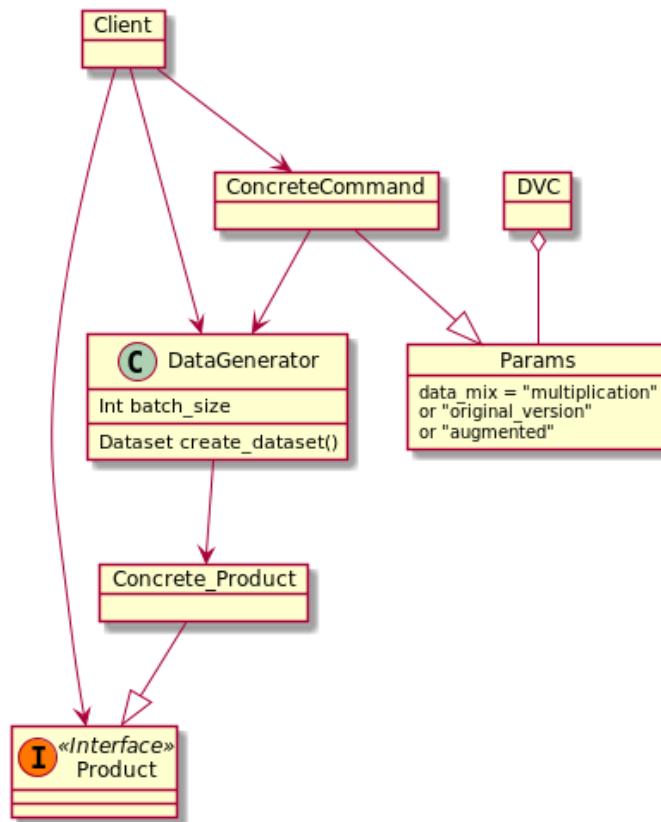


Figure 7: Utilisation de pattern Factory et Command pour le DataGenerator

Ci-dessous un extrait de la classe DataGenerator.

```

36 import sys
37 from .ModelVariables import ModelVariables
38 from tools import data_simple_treatment
39 from tools import data_augmented
40 from tools import BATCH_SIZE
41
42 class DataGenerator(ModelVariables):
43     def __init__(self, batch_size=BATCH_SIZE):
44         super().__init__()
45         print('Eager loading ready')
46         self.size_batch = batch_size
47
48     def __call__(self, path_images, path_masks, data_mix='NA'):
49         if data_mix == "original_version":
50             return self.data_vo()
51         elif data_mix == "augmented":
52             return self.data_aug()
53         elif data_mix == "multiplication":
54             return self.data_multiplication()
55         else:
56             print('paramètre data_mix vaut "augmented", "multiplication" ou "original_version"')
57             sys.exit()
58
59     def data_vo(self):
60         print('data_vo')
61         return data_simple_treatment(self.path_images, self.path_masks, self.size_batch)
62
63     def data_aug(self):
64         print('data_aug')
65         return data_augmented(self.path_images, self.path_masks, self.size_batch)
66
67     def data_multiplication(self):
68         print('data_multiplication')
69         return data_simple_treatment(self.path_images, self.path_masks, self.size_batch)
  
```

Ce genre d'écriture est utilisé en Programmation Orientée Objet et représente un ensemble de bonnes pratiques de codage pour la maintenabilité et notamment le travail en équipe.

8 COMPARAISON

Nous avons travaillé deux sortes d'algorithmes. Le premier, simple et classique, est un réseau R-CNN, sur le modèle de U-Net et Xception. Il utilise la structure du modèle, mais sans le modèle pré-entraîné. Le second est plus complexe car il comporte une méthode d'apprentissage par transfer learning, celui-ci est obtenu par le modèle pré-entraîné ResNet 50, sur une structure globale de plus.

Nous avons effectué beaucoup d'expériences pour régler nos hypers paramètres d'apprentissage profond². Nous avons de plus rangé les expériences selon le type de data en présence, car nous voulions mettre en évidence les améliorations apportées par l'augmentation de data.

name	mIoU	dice	time	data_mix	optim_type	learning_rate
deeplab	0,6	0,7	1586,42	original_version	rmsprop	0,00019
deeplab	0,62	0,71	1589,59	multiplication	rmsprop	0,00019
deeplab	0,61	0,7	1686,6	original_version	nadam	8E-05
deeplab	0,6	0,7	1698,6	multiplication	nadam	8E-05
UNet	0,54	0,63	995,57	original_version	adam	0,00015
UNet	0,48	0,58	996,59	multiplication	adam	0,00015
UNet	0,54	0,63	997,76	original_version	rmsprop	0,00037
UNet	0,5	0,6	991,54	multiplication	rmsprop	0,00037

Table 1: Comparaison de quelques optimisations : Notre meilleure performance est avec augmentation de data et optimiseur rmsprop

Le tableau ?? nous montre que le meilleur score parmi nos modèles est de 71% pour le coefficient de Dice et de 62% pour le mIoU. Nous pouvons nous apercevoir que l'augmentation de data a une réelle valeur ajoutée seulement pour les cas où le taux d'apprentissage est bas ($> 2e^{-4}$) et en utilisant l'optimiseur rmsprop, dont la particularité est de supporter la mise à jour, même pour une valeur nulle du gradient, quand l'optimiseur est dans une implémentation dense.

9 CONCLUSION

Dans cet article, nous avons présenté nos différentes approches, celle de U-Net avec Xception et celle de DeepLab avec ResNet. De plus nous avons parcouru l'histoire des réseaux convolutifs du siècle dernier à nos jours. Nous avons pris le soin d'énumérer les réseaux ayant dépassés les SOTA puis de nous concentrer sur une sélection parmi une poignée d'entre-eux??.

Ensuite nous avons mis l'accent sur la construction d'une base de données augmentées?? et sur l'amélioration notable qu'elle pourvoie à l'apprentissage du modèle¹.

Pour poursuivre cette étude, il serait utile de modifier l'usage du ResNet-50 par son petit-frère le ResNet-152, comportant plus de couches de convolution. De plus, il nous semblerait utile de tester les performances d'autres réseaux pré-entraînés

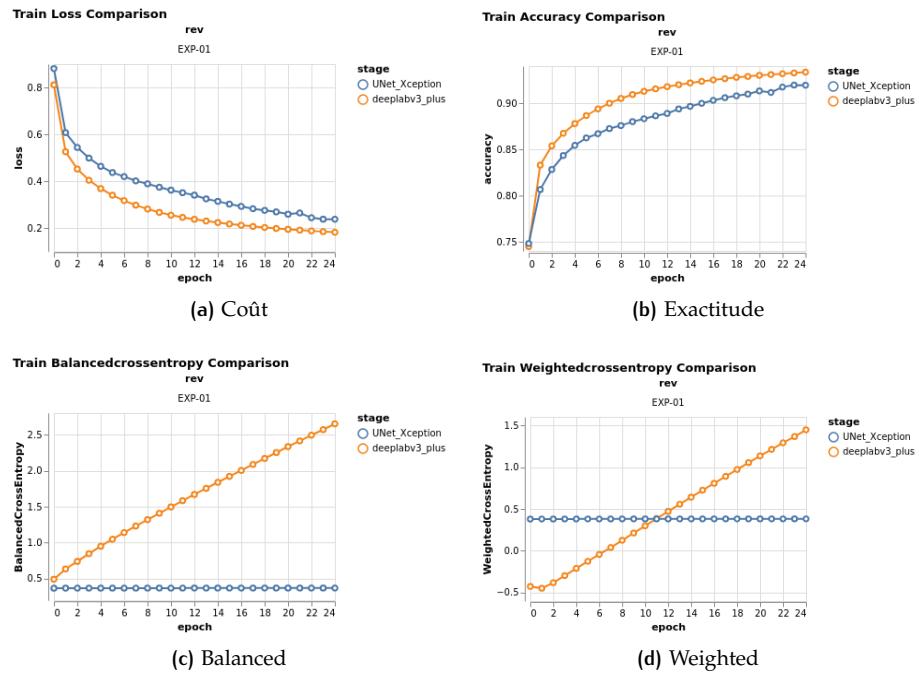


Figure 8: Comparaison de métriques et de coûts des meilleurs modèles pour chaque réseau.

dans le modèle utilisé, ou encore de poursuivre l'entraînement en réduisant encore un peu plus le taux d'apprentissage.

De plus, il nous paraît pertinent d'imaginer l'ajout d'une autre série de transformation au moins, à nos données, pour leurs formuler de plus nombreuses augmentations, puisqu'en l'occurrence, le réseau semble être plus performant quand il a plus de data.

Enfin, pour nous intégrer au mode de lecture en temps réel de notre voiture autonome, nous envisageons de réduire drastiquement la dimension dans les 3 axes du tenseur représentant les images d'entrée, pour assurer une rapidité d'exécution, au moment du fonctionnement du véhicule électrique.

	Experiment	BCE	WCE	accuracy	loss	mIoU	step	time	dice	epochs	data_mix	name	optim_type	learning_rate
0	exp-7bc0f	10,604	8,728	0,934	0,173	0,6	19	1724,62	0,69	20	original_version	deeplab	nadam	0,00085
1	exp-3ca96	10,708	8,916	0,934	0,176	0,59	19	1697,2	0,69	20	multiplication	deeplab	nadam	0,00085
2	exp-doe65	5,960	4,429	0,940	0,153	0,61	19	1533,93	0,7	20	original_version	deeplab	adam	0,00015
3	exp-c9ac1	6,044	4,608	0,940	0,155	0,61	19	1522,94	0,7	20	multiplication	deeplab	adam	0,00015
4	exp-e321a	4,251	2,994	0,943	0,142	0,61	19	1686,6	0,7	20	original_version	deeplab	nadam	8E-05
5	exp-c76d4	3,956	2,605	0,943	0,142	0,6	19	1698,6	0,7	20	multiplication	deeplab	nadam	8E-05
6	exp-c6a43	21,457	20,301	0,940	0,151	0,61	19	1585,62	0,7	20	original_version	deeplab	rmsprop	0,00037
7	exp-e4dff	20,901	19,685	0,941	0,150	0,6	19	1586,27	0,7	20	multiplication	deeplab	rmsprop	0,00037
8	exp-e93a2	1,452	0,536	0,822	0,550	0,45	19	1486,55	0,55	20	original_version	deeplab	sgd	0,00047
9	exp-78dfdf	1,412	0,528	0,823	0,546	0,45	19	1491,36	0,55	20	multiplication	deeplab	sgd	0,00047
10	exp-7ocad	1,318	0,171	0,822	0,558	0,46	19	1535,1	0,55	20	original_version	deeplab	adadelta	0,00082
11	exp-92faf	1,081	-0,241	0,821	0,561	0,46	19	1537,94	0,55	20	multiplication	deeplab	adadelta	0,00082
12	exp-aof8c	17,876	16,164	0,943	0,141	0,6	19	1586,42	0,7	20	original_version	deeplab	rmsprop	0,00019
13	exp-34ef2	17,287	15,537	0,943	0,142	0,62	19	1589,59	0,71	20	multiplication	deeplab	sgd	0,00019
14	exp-db829	1,542	0,205	0,825	0,539	0,46	19	1504,13	0,55	20	original_version	deeplab	rmsprop	0,0005
15	exp-29f17	1,666	0,364	0,824	0,541	0,46	19	1494,5	0,56	20	multiplication	deeplab	sgd	0,0005
16	exp-ba3d8	1,165	0,381	0,911	0,264	0,54	19	991,8	0,63	20	original_version	U-Net	nadam	0,00085
17	exp-53c97	1,164	0,382	0,911	0,267	0,51	19	1008,77	0,61	20	multiplication	U-Net	nadam	0,00085
18	exp-d8c86	0,624	0,382	0,912	0,261	0,54	19	995,57	0,63	20	original_version	U-Net	adam	0,00015
19	exp-7c564	0,624	0,381	0,910	0,268	0,48	19	996,59	0,58	20	multiplication	U-Net	adam	0,00015
20	exp-b4ea4	0,299	0,382	0,912	0,263	0,53	19	990,16	0,62	20	original_version	U-Net	nadam	8E-05
21	exp-8d64e	0,299	0,381	0,913	0,259	0,46	19	992,64	0,56	20	multiplication	U-Net	nadam	8E-05
22	exp-30f96	1,382	0,381	0,910	0,268	0,54	19	997,76	0,63	20	original_version	U-Net	rmsprop	0,00037
23	exp-f13b2	1,381	0,381	0,910	0,267	0,5	19	991,54	0,6	20	multiplication	U-Net	sgd	0,00047
24	exp-oeff32	0,515	0,382	0,913	0,260	0,49	19	992,04	0,59	20	original_version	U-Net	sgd	0,00047
25	exp-42da6	0,516	0,381	0,917	0,245	0,5	19	992,42	0,6	20	multiplication	U-Net	adadelta	0,00082
26	exp-9od63	0,696	0,381	0,913	0,261	0,52	19	990,81	0,62	20	original_version	U-Net	adadelta	0,00082
27	exp-b8fg1	0,696	0,381	0,913	0,259	0,5	19	994,91	0,6	20	multiplication	U-Net	rmsprop	0,00019
28	exp-77c21	1,600	0,381	0,911	0,265	0,52	19	990,24	0,61	20	original_version	U-Net	sgd	0,0005
29	exp-7043e	1,600	0,381	0,913	0,257	0,49	19	989,4	0,59	20	multiplication	U-Net	sgd	0,0005
30	exp-78211	0,840	0,381	0,910	0,268	0,51	19	992,08	0,61	20	original_version	U-Net	sgd	0,0005
31	exp-5ca61	0,841	0,381	0,913	0,260	0,49	19	996,07	0,59	20	multiplication	U-Net	sgd	0,0005

Table 2: Tableau d'expérimentations

INDEX

ASPP, 5
DCNN, 5
Deep Learning, 2, 6
DeepLabV3, 2, 4–7
Faster R-CNN, 2
GAN, 6
HOG, 3
Hubel, 3
mIoU, 2, 6, 8
R-CNN, 2, 7
ReLU, 3
ResNet 50, 7
ResNet-50, 4
SIFT, 3
Sørensen-Dice, 2, 6, 8
TensorFlow, 6
Transfer Learning, 4, 7
U-Net, 7
Wiesel, 3
Xception, 7

GLOSSARY

Atrous Spatial Pyramid Pooling La convolution Atrous permet de contrôler explicitement la résolution à laquelle les réponses de features sont calculées à l'intérieur du Deep Convolutional Neural Networks. Cette convolution Atrous provient d'une technique plus ancienne mais qui reste une stratégie orienté sur le pooling. [5](#)

DCNN Deep Convolutional Neural Networks. Ils sont utilisés dans la classification d'image, la reconnaissance d'objet, et dans la segmentation d'image, notamment combiné à un "algorithme à trous" (issu de la transformé en ondelette) pour réduire l'impact des invariants de DCNN sur la segmentation d'image, car ils augmente l'exactitude de localisation, voir DeepLab.

[5](#)

Deep Learning Apprentissage profond. Ce terme regroupe les méthodes d'apprentissage utilisées dans des graphes ou des réseaux ou encore des modules paramétrés et interconnectés. L'apprentissage a court en mettant à jour les poids dans le réseau par descente de gradient. Plusieurs optimiseurs sont possible, de plus la fonction de coût s'adapte à l'apprentissage du réseau. [6](#)

DeepLabV3 Algorithme de deep learning utilisant le principe d'encodeur et décodeur avec des convolutions séparables Atrous pour la segmentation sémantique d'image. [2](#)

Faster R-CNN C'est un réseau qui extrait les caractéristiques en utilisant le RoIPool (Region of Interest Pooling) de chaque boîte évaluée en réalisant une classification et une régression sur les boîte délimitant le contour de l'objet. Le RoIPool est une opération d'extraction de carte de petites caractéristiques de chaque RoI durant la détection. [2](#)

HOG Les Histogrammes des Gradients Orientés sont des descripteurs saisissant l'apparence et la forme locale d'un objet par la distribution de l'intensité du gradient et de la direction des contours.. [3](#)

mIoU La moyenne d'une « Intersection over Union » (IoU), aussi connue sous le nom de l'index Jaccard, est la métrique d'évaluation la plus populaire pour la tâche de segmentation, de la détection d'objet ainsi que du suivi. [2](#)

R-CNN Region-Based Convolutional Neural Network, c'est un type de réseau convolutif basé sur des régions de l'image. [2](#)

ReLU Rectified Linear Unit activation function. Cette fonction est définie sur $[-\infty, \infty]$, mais prend pour valeur 0 avec les valeurs négatives. De plus elle ne change pas de valeur si cette dernière est positive ou même nulle. [3](#)

SIFT Shift Invariant Feature Transform. [3](#)

SOTA State of the art. Défini les standards les plus exigeants de l'industrie et notamment pour la recherche, dans le domaine de la vision par ordinateur. [2](#)

Sørensen-Dice Indice de Sørensen ou coefficient de Dice, développé par les botanistes Thorvald Sørensen et Lee Raymond Dice dans des articles publiés respectivement en 1948 et 1945. Il exprime, en statistique, la similarité entre 2 échantillons. [2](#)

Transfer learning Apprentissage visant à incorporer les poids d'entraînement d'un premier réseau, sur le dessus d'un second réseau, le premier étant souvent spécialisé dans l'exécution d'une tâche précise. [4](#)

REFERENCES

- [1] Y. Bengio Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.
- [2] Yann Le Cun. *Quand la machine apprend*. Odile Jacob, Paris, 2019.
- [3] François Chollet. Xception: Deep learning with depthwise separable convolutions. Apr 2017.
- [4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018.
- [5] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. <https://d2l.ai>.