Projet
Segmentation de pages de journaux Anciens par Deep Learning

Réalisé par : Nasreddine MENACER

Résumé

Ce Projet consiste à implémenter un réseau de neurone à convolution en langage python, pour déterminer directement à partir de l'image, la structure en articles de pages de journaux numérisés, ce réseau est destiné pour remplacer le système utilisé actuellement dans la plateforme PIVAJ (Plateforme d'Indexation et Visualisation d'Archives de Journaux) développé au laboratoire LITIS, qui s'effectue en deux temps : à travers un CRF pour l'étiquetage bas-niveau des pixels de l'image, et ensuite en appliquant un jeu de règles de structuration de plus haut niveau.

Mots Clés: Segmentation sémantique, FCN, CNN, Réseau de neurones.

Abstract

This project consists in implementing a neural network with python language, to determine directly from the image the structure of a newspaper article. This network is intended to replace the current system used in the platform PIVAJ (Platform for Indexing and Visualization of News Archives) which is developed in the LITIS laboratory, and is carried out in two stages: through a CRF for the low-level labeling of pixels in the image, and then applying a higher-level set of rules structuration.

Key Words: Semantic Segmentation, FCN, CNN, Neural network.

Table des matières

2. Réseaux de Neurones ……..7

2.1. Réseaux de Neurones Entièrement connectés (FC)…….7

2.2. Réseaux de Neurones à Convolutions (CNN)…….....7

2.2.1. Composition des CNNs……..8

Couche d'entrée ……9

La couche de convolution.....9

Nombre de filtres dans une couche de convolution.....10

Les poids des filtres 11

Partage de poids 11

La Couche ReLu.....12

Couche Pooling.....13

2.3. Les Réseaux entièrement Convolutifs (FCN).....14

Métrique 17

PARTIE EXPERIMENTALE.....19

1. Base de Données 19

2. Visualisation des données 19

3. Chargement et prétraitement.....19

4. Création du réseau de neurone 22

VGG16.....22

Entrainement 25

Performance du Modèle 26

TRAVAIL REALISE ET DIFFICULTE RENCONTRE.....27

CONCLUSION ET PERSPECTIVES 28

BIBLIOGRAPHIE……..29

Tableau des Figures

FIGURE 1: DIAGRAMME DES BESOINS ……6
FIGURE 2 : RESEAU DE NEURONE ENTIEREMENT CONN……...7
FIGURE 3 : RESEAU DE NEURONE A CONVOLUTION ……..8
FIGURE 4 : LES COUCHES D'ENTREE D'UN FC, ET D'UN CN……...9
FIGURE 5 : CONVOLUTION ……..10
FIGURE 6 : FONCTION RELU ……... 12
FIGURE 7 : APPLICATION RELU SUR UNE IMAGE ……..12
FIGURE 8 : MAX-POOLING ……..…… .13
FIGURE 9 : FULLY CONVOLUTIONAL NETWORK ……..15
FIGURE 10 : LES VARIANTES D'UN FCN ……... .15
FIGURE 11 : LES CHEMINS DES VARIANTES D'UN FCN ……..…… .16
FIGURE 12 : IMAGES SEGMENTES PAR LES VARIANTES DU FCN ……..…… .17
FIGURE 13 : IMAGE ET MASQUE ……...19
FIGURE 14 : IMAGE ET MASQUE IMAGES REDIMENSIONNEES ……...21
FIGURE 15 : IMAGE ET MASQUE ……...23
FIGURE 16 : PERFORMANCES DES FCN ……....25
FIGURE 17 : Résultats de la segmentation ……25

INTRODUCTION

Le système PIVAJ (Plateforme d'Indexation et Visualisation d'Archives de Journaux) développé au laboratoire LITIS, permet de déterminer et d'extraire la structure en articles à partir de numérisation de pages de journaux, afin d'en faciliter l'indexation et la visualisation. Dans le système actuel l'extraction des articles est effectuée en deux temps : tout d'abord un modèle CRF (champs aléatoires conditionnel) effectue un étiquetage bas-niveau des pixels de l'image afin de déterminer la structure physique de la page. Dans un deuxième temps l'étiquetage logique en articles est déterminé à partir de ces entités de la structure physique, en appliquant un jeu de règles de structuration de plus haut niveau.

Bien que le système actuel donne de bons résultats, la détermination des règles de structuration reste difficile, ce qui rend compliquée l'adaptation du système à différents types de structures.

Compte tenu des avancées spectaculaires réalisées ces dernières années dans tous les domaines de l'analyse d'image grâce à l'apprentissage profond (Deep Learning), y compris en analyse d'images de documents, on se propose d'utiliser des réseaux profonds convolutifs (CNN), pour déterminer directement à partir de l'image, la structure en articles de pages de journauxnumérisés.

OBJECTIF DU PROJET ET L'ETAT DE L'ART

1. Objectif et Cahier de charge du projet

Le but de ce projet est de programmer en langage python, et en utilisant ces librairies tel que KERAS un réseau de neurone capable de segmenter une page de journal, en trois régions, respectivement (région texte, région image, et région fond), pour cela on aura besoin d'une base de données, d'images de journaux déjà segmentées, et d'un fichier de poids d'un réseau prés-entrainé, pour faire entrainer le réseau, après création de notre modèle, il doit être capable de nous fournir un résultat qui nous informe sur les régions existantes dans une page de journal.

De ce fait notre projet peut se diviser en deux parties :

- Réaliser une étude bibliographique complète sur le principe de fonctionnement des réseaux de neurones à convolution.
- Créer une modèle de réseau, l'entrainer, et tester ces performances.

Le diagramme en bêtes à cornes ci-dessous nous permet d'identifier les besoins de notre système.

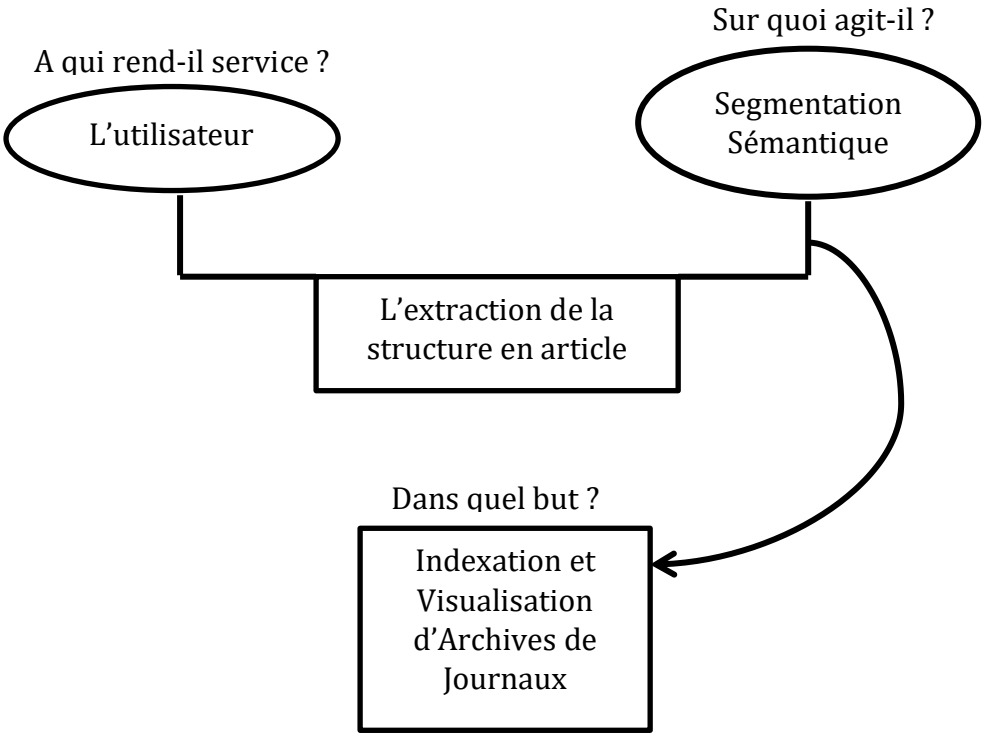


Figure 1: diagramme des besoins

2. Réseaux de Neurones

L'apprentissage en profondeur est un sujet difficile à gérer. Cependant les réseaux convolutifs ont marquaient une avancées segnificative en matière de reconnaissance de forme et de la segmentation et ont surpassaient les technologies les plus avancées.

2.1. Réseaux de Neurones Entièrement connectés (FC)

Les réseaux de neurones entièrement connectés sont des réseaux où chaque neurone est connecté à chaque neurone des couches adjacentes. Ce sont les architectures de réseau de neurones standard et typiques.

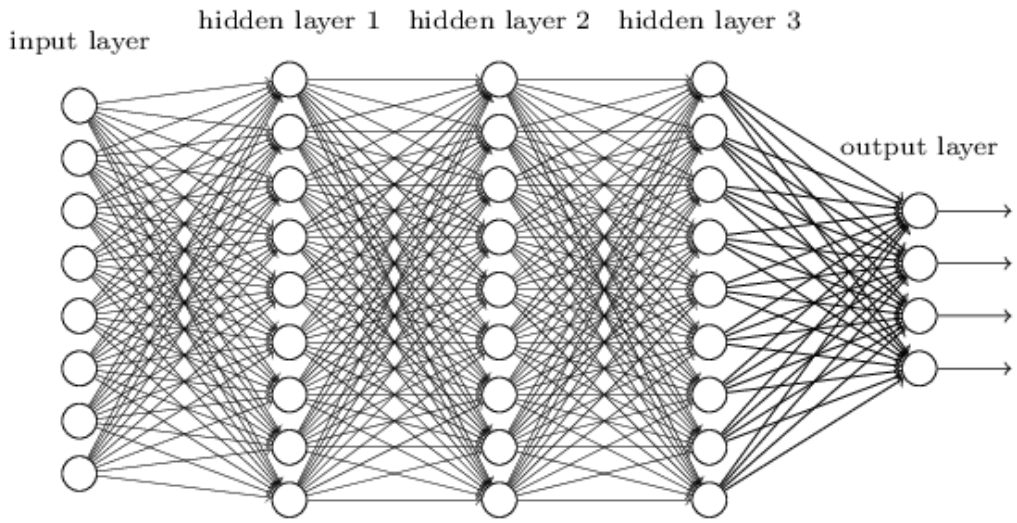


Figure 2: Réseau de neurone entièrement connectés

2.2. Réseaux de Neurones à Convolutions (CNN)

Les réseaux de neurones à convolution (CNN) sont un type particulier d'architectures neuronales spécialement conçues pour gérer les données sous forme d'images. Depuis leur introduction par Monsieur Yann Le Cun, au début des années 1990, les CNN ont démontré d'excellentes performances dans des tâches telles que la classification des chiffres manuscrits et la détection des visages. Au cours des dernières années, plusieurs documents ont montré qu'ils pouvaient également fournir des résultats exceptionnels dans des tâches de classification visuelle plus difficiles. Plus particulièrement (Krizhevsky et al., 2012) affichent des performances record par rapport au référentiel de la classification ImageNet 2012, leur modèle (AlexNet) atteignant un taux d'erreur de 16,4% .

Plusieurs facteurs sont responsables de l'intérêt que les gens ont suscité en eux, par exemple :

- La disponibilité de très grands ensembles de données de formation avec des millions d'exemples étiquetés, tel que (ImageNet) qui est une des bases de données des plus populaires.

- Puissantes implémentations de GPU, rendant pratique la formation de très grands modèles.

- Stratégies de régularisation de modèle améliorées telles que le décrochage.

2.2.1. Composition des CNNs

Les CNN ont leurs propres structure et propriétés. Ils ont un aspect différent des réseaux FC standard, mais ils partagent les mêmes mécanismes. Dans les deux cas, on y trouve les termes (couches cachées, poids, biais, neurones cachés, fonctions de perte, rétropropagation et descente de gradient stochastique).

Les cinq composantes principales d'un CNN sont :

- Couche d'entrée.
- Couches de convolutions.
- Couches ReLu.
- Couches de Pooling.
- Couches entièrement connectées.

Comme le montre la figure ci-dessus, l'image est traitée sur l'ensemble du réseau, sur plusieurs couches, et les neurones de sortie conservent les prédictions de chaque classe.

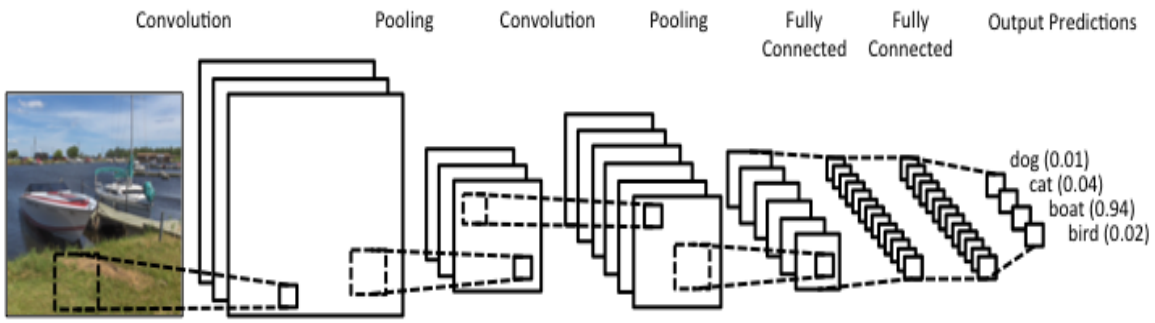


Figure 3 : Réseau de neurone à Convolution

- **Couche d'entrée**

Dans les réseaux entièrement connectés, les entrées sont représentées par des lignes verticales de neurones, essentiellement des vecteurs. Par contre dans les réseaux à convolutions, lorsqu'il s'agit d'images, ils sont considérés comme des carrés ou des neurones, chaque neurone représentant une valeur de pixel. Fondamentalement, les CNN conservent les images telles quelles et n'essayent pas de les compresser dans un vecteur. Le diagramme ci-dessous montre la différence entre une entrée dans un réseau entièrement connectés, et un réseau à convolutions :

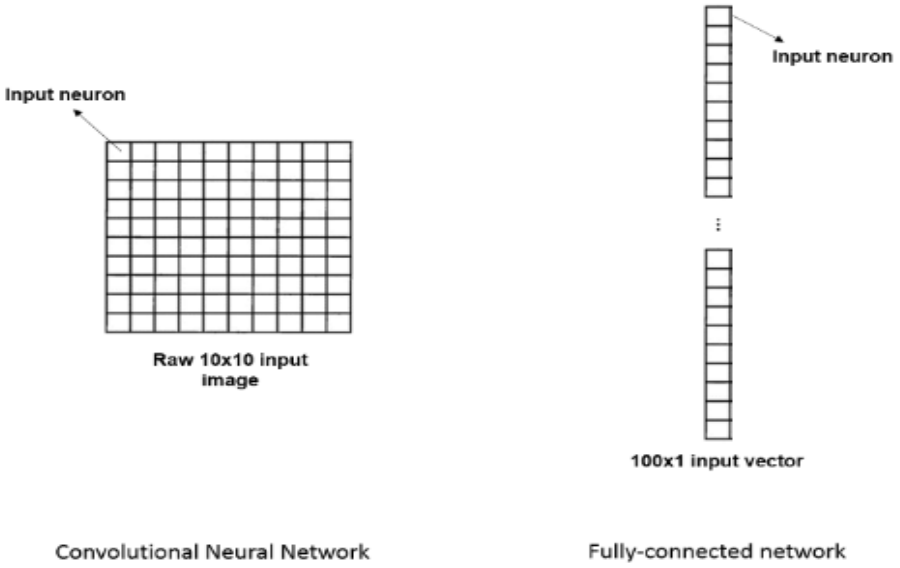


Figure 4: Les couches d'entrée d'un FC, et d'un CNN

- **La couche de convolution**

Cette couche est le composant principal d'un réseau à convolution. Un réseaux entièrement connectés signifie que chaque neurone d'une couche donnée est connecté à tous les neurones des couches adjacentes. Lorsqu'un point de données multidimensionnel circule d'une couche à une autre, l'activation de chaque neurone d'une couche donnée est déterminée par les poids de tous les neurones des couches précédentes. Pour les CNN c'est différent car ils ne sont pas complètement connectés. Cela signifie que chaque neurone d'une couche cachée n'est pas connecté à tous les neurones de la couche précédente. Il est plutôt connecté à un patch (généralement une petite région carrée) de neurones dans la couche précédente.

Comme le montre la figure ci-dessus, le premier neurone de la première couche cachée, est connecté à un patch de 3x3 pixels en entrée. Ce neurone caché ne dépend que de cette petite région et finira par en saisir les caractéristiques tout au long de l'apprentissage. La valeur de ce premier neurone caché est le résultat d'une convolution entre une matrice de pondération appelée noyau (le petit carré gris) et une petite région de même taille dans l'image, appelée champ réceptif.

Il s'agit d'une multiplication élémentaire de deux matrices: la région d'image 3x3 et le noyau de même taille. Les multiplications sont ensuite résumées dans une valeur de sortie. Ce neurone apprend fondamentalement un motif visuel hors du champ réceptif. Sa valeur est considéré comme une intensité qui caractérise la présence, ou non d'une caractéristique dans l'image.

Pour calculer le deuxième neurone caché, le noyau se décale d'un pixel sur l'image d'entrée de gauche à droite et applique la même convolution, toujours avec le même filtre.

Est puisque le noyau glisse sur l'ensemble de l'image en effectuant une convolution à chaque étape et en stockant les sorties dans la Feature Map. On aura le résultat suivant :

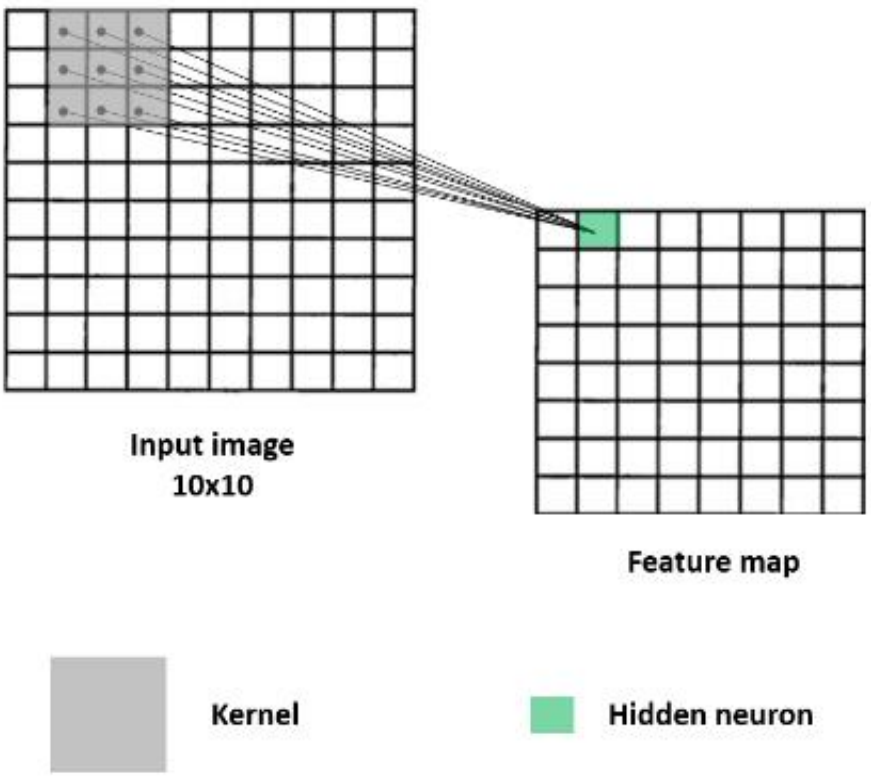


Figure 5: Convolution

- **Nombre de filtres dans une couche de convolution**

Dans une architecture CNN typique, nous n'avons pas un seul filtre par couche de convolution. Parfois, nous aurons 10, 16 ou 32. Parfois plus. Dans ce cas, autant de convolutions par couche que de filtres sont effectuées. L'idée est de générer un ensemble de Features Map, chacune localisant une caractéristique simple et spécifique dans l'image. (Plus nous avons de filtres, plus nous extrayons de détails intrinsèques).

- **Les poids des filtres**

Lors de la formation des CNN, les poids du filtre ne sont pas définis manuellement. Ces valeurs sont apprises par le réseau automatiquement via une rétro-propagation, cela signifie que lorsqu'il ajuste ses pondérations (à partir de valeurs aléatoires) pour réduire les erreurs de classification, le réseau fournit les filtres appropriés qui permettent de caractériser l'objet qui nous intéresse.

- **Partage de poids**

Une carte de caractéristiques est produite par un et un seul filtre, c'est-à-dire que tous les neurones cachés partagent les mêmes poids, car le même filtre les produit tous. C'est une propriété qui permet aux CNN de s'entraîner plus rapidement en réduisant considérablement le nombre de paramètres appris. En plus d'accélérer l'entraînement, le concept de partage de poids est motivé par le fait qu'un motif visuel peut apparaître plusieurs fois dans différentes régions d'une image et qu'il est donc logique de disposer d'un seul filtre qui le détecte sur toute l'image.

- **La Couche ReLu**

Une fois les Features Map extraites de la couche de convolution, l'étape suivante consiste à les déplacer vers une couche ReLu. Les couches ReLu sont généralement associées à des couches de convolution. Ils travaillent généralement ensemble.

Une couche ReLu applique une fonction ReLu sur la Feature Map. Cela règle tous les pixels négatifs à 0. La sortie de cette opération s'appelle une carte de caractéristiques rectifiée.

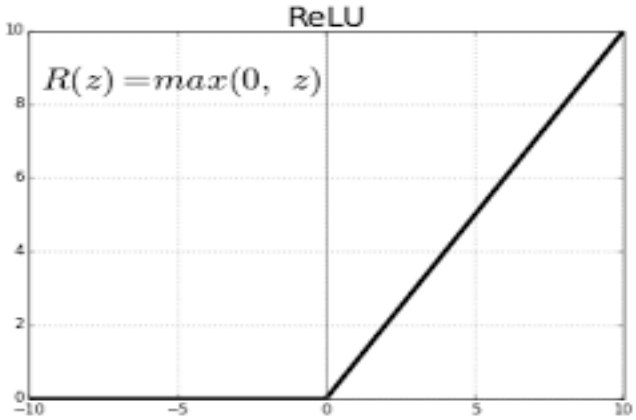


Figure 6: Fonction ReLu

Les couches ReLU présentent deux avantages principaux:

- Ils introduisent une non-linéarité dans le réseau, car les opérations précédentes (convolutions, multiplication et sommation) sont linéaires. Si nous n'avons pas une non-linéarité, nous aurons un modèle linéaire qui échouera dans la tâche de classification.
- Ils accélèrent le processus d'entraînement en évitant le problème de la disparition progressive du gradient.

La figure suivante montre ce qu'une couche ReLU fait sur une d'image:

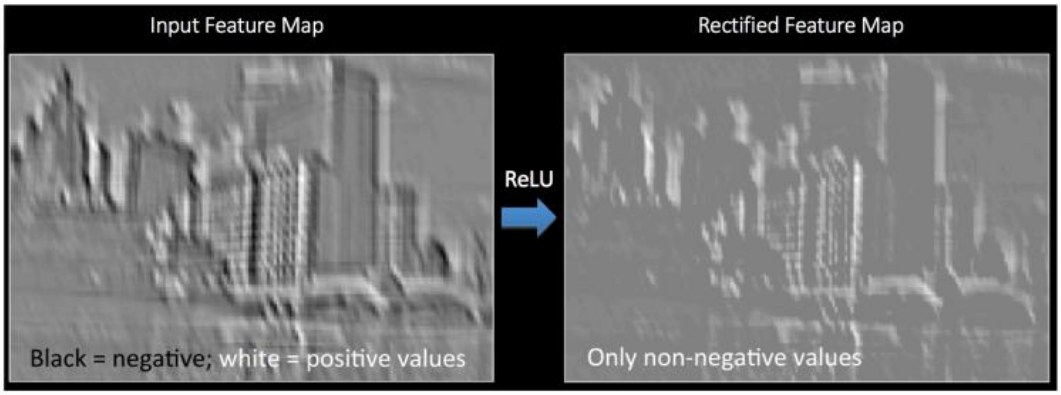


Figure 7: Application ReLu sur une image

- **Couche Pooling**

Les cartes de caractéristiques (Features Map) rectifiées traversent maintenant une couche de regroupement. Le regroupement est une opération de sous-échantillonnage qui réduit la dimensionnalité de la carte de caractéristiques. La variante la plus courante est le (max-pooling). Il s'agit d'une petite fenêtre de taille 2x2 qui glisse sur la carte de caractéristiques rectifiée et prend l'élément le plus grand à chaque étape. Comme le montre la figure suivante :

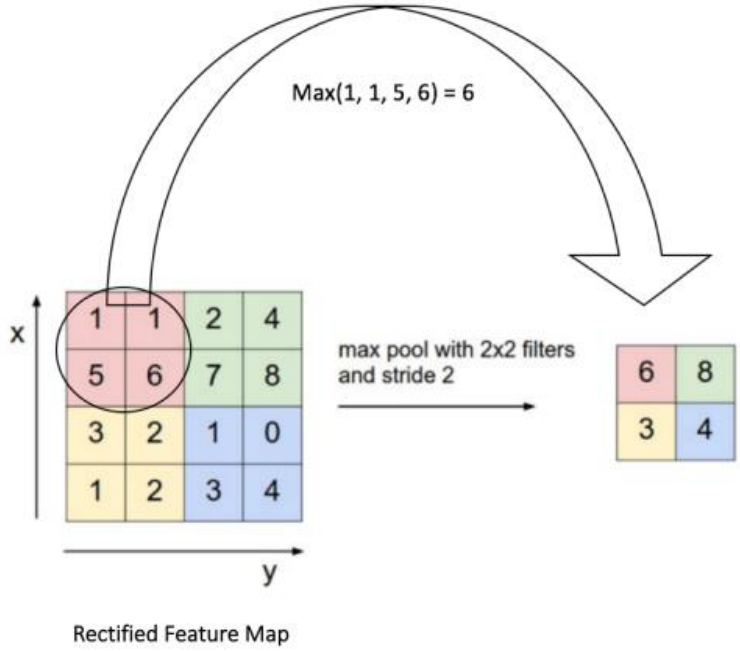


Figure 8: Max-Pooling

Le Pooling présente de nombreux avantages:

- Il réduit la taille des cartes de caractéristiques rectifiées et le nombre de paramètres pouvant être entraînés, contrôlant ainsi le sur-ajustement.
- Il condense les cartes de fonctionnalités en conservant les fonctionnalités les plus importantes
- Cela rend le réseau invariant par rapport aux petites transformations, distorsions et traductions dans l'image d'entrée (une petite distorsion en entrée ne modifiera pas la sortie du Pooling, si on prend la valeur maximale dans un voisinage local.

L'intuition derrière le max-pooling n'est pas simple. Si une caractéristique est détectée, sur une petite partie d'une image telle que le carré rouge dans la figure précédente, nous

ne nous soucions pas de savoir quels pixels exacts la font apparaître. Au lieu de cela, nous choisissons dans cette partie celle qui a la plus grande valeur et supposons que c'est ce pixel qui résume la caractéristique visuelle. Cette approche semble agressive puisqu'elle perd les informations spatiales. Mais c'est vraiment efficace et fonctionne très bien dans la pratique. Et on remarque que les informations spatiales principales restent, et les détails non importants disparaissent. C'est pourquoi la mise en commun maximale évite les sura-justements. Cela permet au réseau de se concentrer sur les informations les plus pertinentes de l'image.

2.3. Les Réseaux entièrement Convolutifs (FCN)

La tâche de segmentation est différente de la tâche de classification car elle nécessite de prévoir une classe pour chaque pixel de l'image en entrée, au lieu d'une seule classe pour l'ensemble de l'entrée.

Les réseaux entièrement convolutifs (FCN) doivent leur nom à leur architecture, qui est construite uniquement à partir de couches connectées localement, aucune couche dense n'est utilisée dans ce type d'architecture. Cela réduit le nombre de paramètres et le temps de calcul. En outre, le réseau peut fonctionner quelle que soit la taille de l'image d'origine, sans nécessiter un nombre fixe d'unités à tout moment, étant donné que toutes les connexions sont locales. Pour obtenir une carte de segmentation, les réseaux de segmentation comportent généralement 2 parties.

- Partie de sous-échantillonnage: capture des informations sémantiques / contextuelles
- Partie de l'échantillonnage: récupérer des informations spatiales

La partie de sous-échantillonnage est utilisé pour extraire et interpréter le contexte, tandis que la partie de ré-échantillonnage est utilisé pour permettre une localisation précise.
Une connexion de saut est une connexion qui contourne au moins une couche. Ici, il est souvent utilisé pour transférer des informations locales en concaténant ou en sommant des cartes de caractéristiques de la partie du sous-échantillonnage avec des cartes de caractéristiques de la partie du ré-échantillonnage. La fusion de caractéristiques de différents niveaux de résolution permet de combiner des informations de contexte avec des informations spatiales.

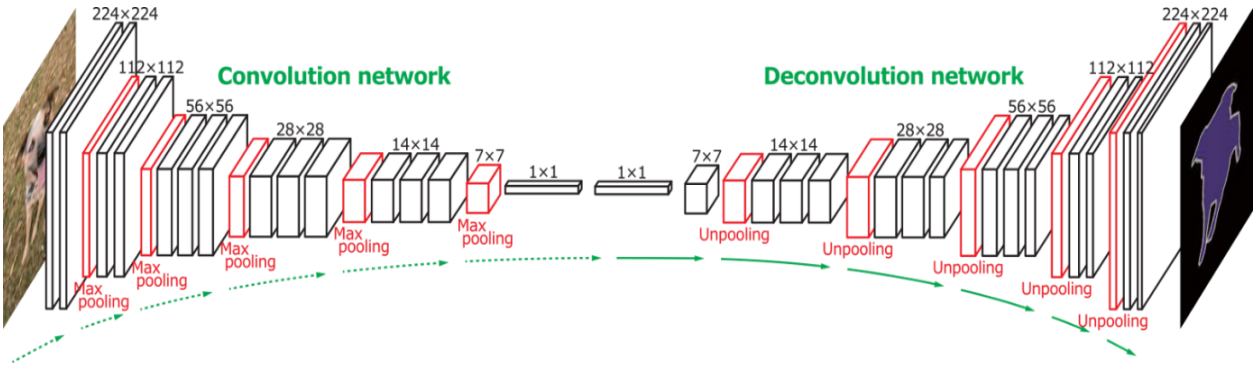


Figure 9: Fully Convolutional Network

Il existe des variantes de l'architecture FCN, qui diffèrent principalement par la précision Spatiale de leur sortie. Par exemple (FCN-32, FCN-16 et FCN-8). Sur la figure ci-dessous, les couches de convolution sont représentées par des lignes verticales entre les couches de regroupement, qui montrent explicitement la taille relative des cartes de caractéristiques

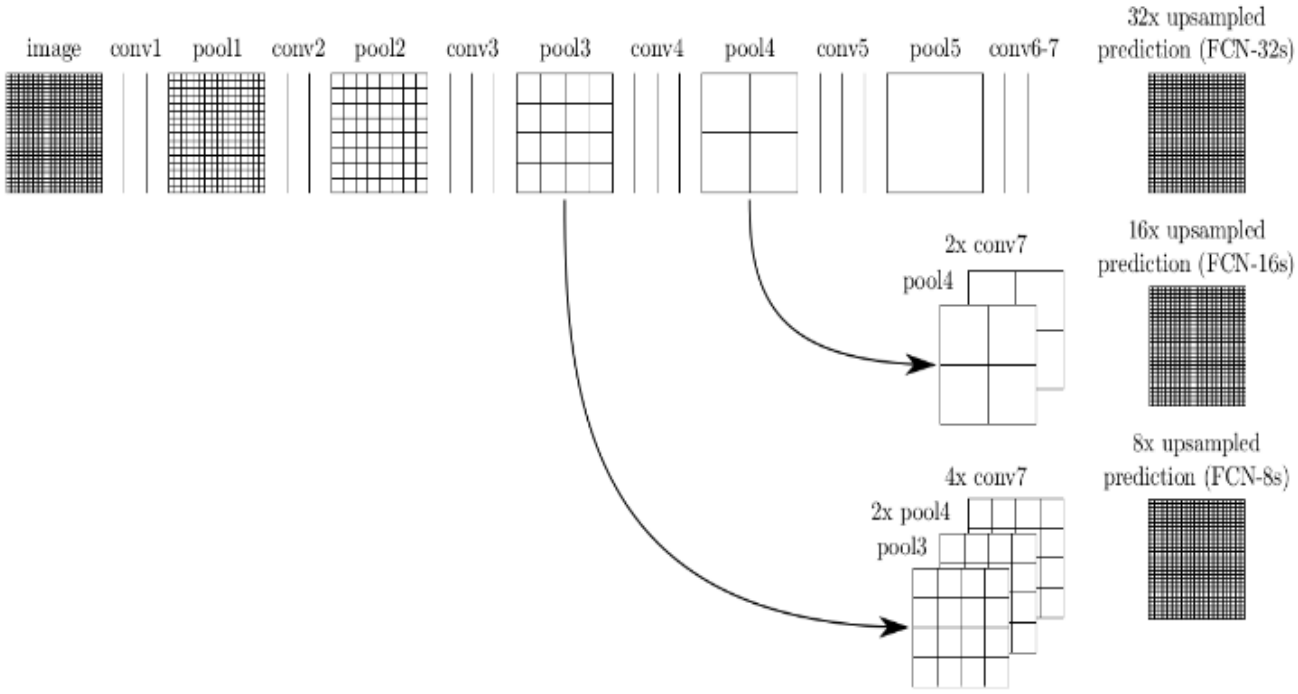


Figure 10: Les variantes d'un FCN

Les trois architectures diffèrent par le pas de la dernière convolution et les connexions ignorées utilisées pour obtenir les cartes de segmentation en sortie. Nous utiliserons le terme chemin de sous-échantillonnage pour désigner le réseau jusqu'à la couche de convolution numéro 7 et nous utiliserons le terme chemin d'échantillonnage pour désigner le réseau composé de toutes les couches après la couche de convolution numéro 7. Les 3 architectures FCN partagent le même chemin de sous-échantillonnage, mais diffèrent par leurs chemins de ré-échantillonnage respectifs.

- FCN-32: Produit directement la carte de segmentation à partir de la couche de convolution numéro 7, en utilisant une couche de convolution transposée avec un pas de 32.
- FCN-16: Additionne la prévision sur-échantillonnée la couche de convolution numéro 7 (en utilisant une convolution transposée avec un pas de 2) avec pool4, puis produit la carte de segmentation en utilisant une couche de convolution transposée avec un pas de 16.
- FCN-8: Additionne la c la couche de convolution numéro 7 sur-échantillonnée (avec une un pas de 2) avec pool4, les échantillonne avec une convolution transposée et les additionne avec pool3, et applique une couche de convolution transposée avec un pas de 8 sur les cartes de caractéristiques pour obtenir la carte de segmentation.

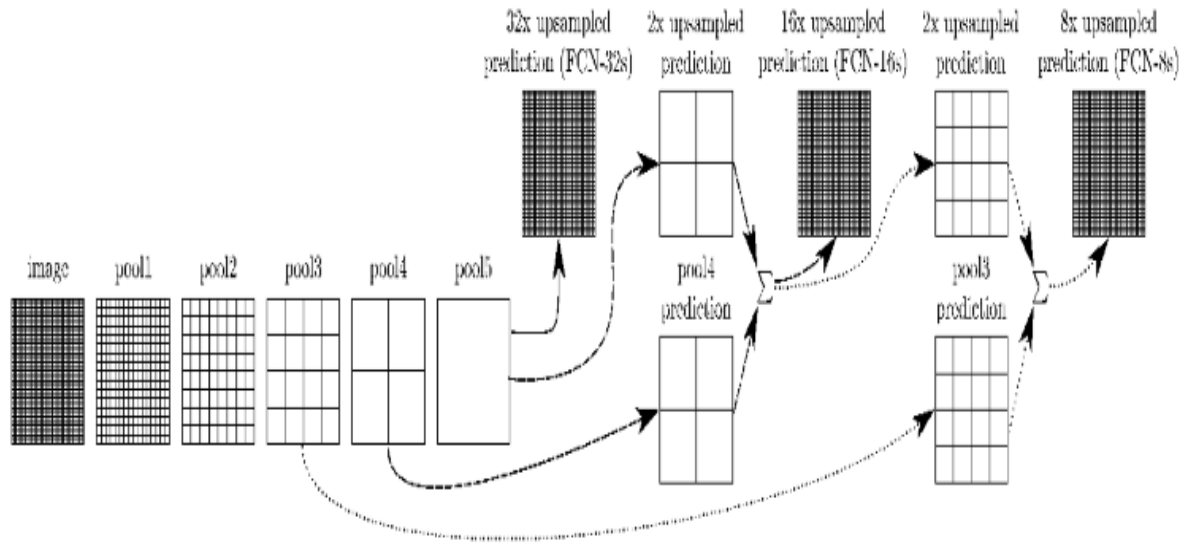


Figure 11: Les chemins des variantes d'un FCN

Comme expliqué ci-dessus, les chemins de ré-échantillonnage des variantes FCN sont différents, car ils utilisent des couches de connexion et des sauts différents pour la dernière convolution, générant différentes segmentations, comme illustré à la figure suivante. La combinaison de couches de précision différente facilite la récupération de données spatiales, ainsi que des informations contextuelles.

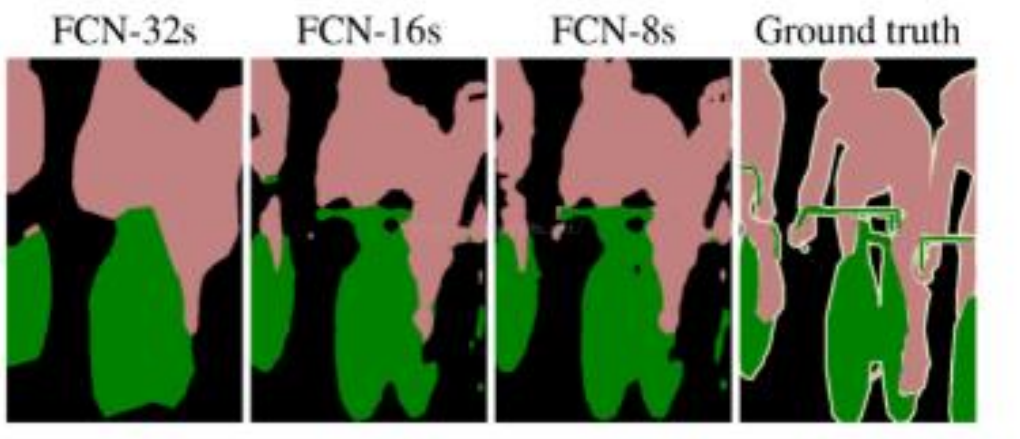


Figure 12 : Images segmentés par les variantes du FCN

- **Métrique**

- **Précision par pixel**

Cette métrique est explicite, car elle génère la précision de prédiction de classe par pixel, dont voici la formule :

$$acc(P, GT) = \frac{|\text{pixels correctly predicted}|}{|\text{total nb of pixels}|}$$

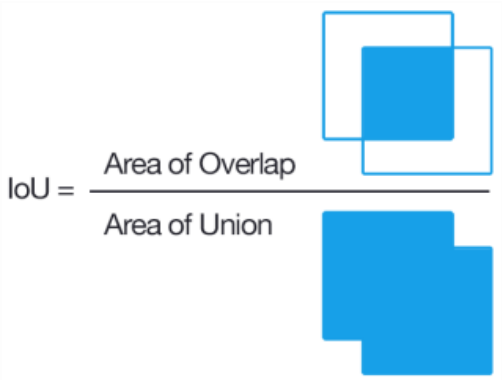
- **Jaccard (intersection sur l'union)**

Cette métrique d'évaluation est souvent utilisée pour la segmentation d'images, car elle est plus structurée. Jaccard est une métrique d'évaluation par classe, qui calcule le nombre de pixels de l'intersection entre les cartes de segmentation de la vérité terrain pour une classe donnée, divisé par le nombre de pixels de l'union entre ces deux cartes de segmentation.

$$jacc(P(class), GT(class)) = \frac{|P(class) \cap GT(class)|}{|P(class) \cup GT(class)|}$$

où P est la carte de segmentation prévue et GT est la carte de segmentation de la vérité terrain. P (classe) est alors le masque binaire indiquant si chaque pixel est prédit comme classe ou non.

Plus la valeur de cette métrique est proche de zéro, plus le modèle est performant, la figure suivant donne une illustration graphique du calcul de l'intersection sur l'union.



PARTIE EXPERIMENTALE

1. Base de Données

Le jeu de données que j'ai utilisé a été collecté pour la formation et la validation d'algorithme d'apprentissage automatique pour les régions de classification des documents sur les zones de texte, d'images et de fond. Il contient 101 images numérisées de divers journaux et magazines en russe. La base est intitulée « Newspaper and magazine images segmentation dataset », crée par « Vil'kin » et « Ilia Safonov » à NRNU MEPhI, Moscow, Russia en 2012.

La plupart des images ont une résolution de 300 dpi et une taille A4, environ 2400x3500 pixels. Des masques à base de pixels ont été créés manuellement pour toutes les images. Les masques de la vérité terrain sont nommés comme des images originales avec le suffixe «_m». Il existe trois classes: zone de texte, zone d'image et arrière-plan. Les pixels du masque de couleur 255, 0, 0 (couleur rouge) correspondent à la zone d'image, les pixels de couleur 0, 0, 255 (couleur bleue) correspondent à la zone de texte, tous les autres pixels correspondent à l'arrière-plan. Les images et les masques sont présentés au format PNG avec compression sans perte.

2. Visualisation des données

Voici un exemple d'image de la base de données, et de son masque, la taille approximative de ces images est = (3329, 2542, 3)

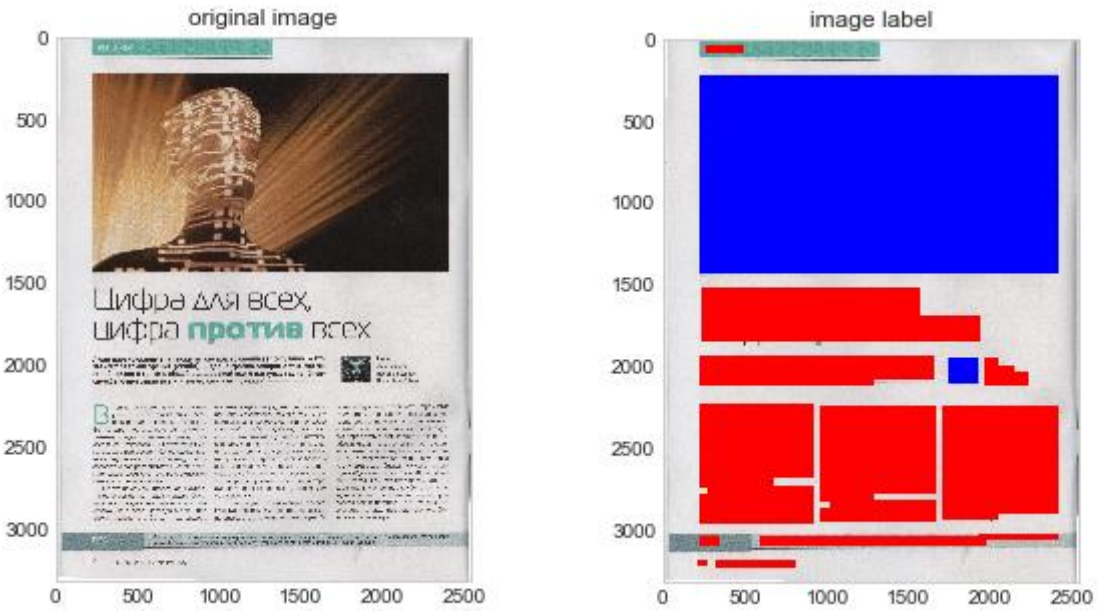


Figure 13: Image et masque

3. Chargement et prétraitement

Pour faire l'entrainement, et la validation du réseau de neurone, j'ai devisé manuellement, les données avant de les charger, comme le nombre d'images et relativement petit, j'ai créé donc quatre répertoire, où on trouve dans les répertoires :

- " images_prepped_train " : les images d'entrainement.
- " annotations_prepped_train " : les masques des images d'entrainement.
- " images_prepped_test " : les images de validations
- " annotations_prepped_test " : les masques des images de validation.

J'ai gardé 80 images, pour l'entrainement, et 21 images pour la validation.

```
print('le nombre dimage dentrainement =',len(ldseg))
print('le nombre dimage label dentrainement =',len(ldseg2))
print('le nombre dimage de test =',len(ldseg3))
print('le nombre dimage label de test =',len(ldseg4))
```

```
le nombre dimage dentrainement = 80
le nombre dimage label dentrainement = 80
le nombre dimage de test = 21
le nombre dimage label de test = 21
```

J'ai chargé ces images dans des listes, pour faciliter leurs manipulations.

```
X_train=[]
Y_train=[]
X_test=[]
Y_test=[]

for i in range (len(ldseg2)):
    X_train.append(cv2.imread(dir_img+ldseg2[i]))
    Y_train.append(cv2.imread(dir_seg+ldseg[i]))

for j in range (len(ldseg4)):
    X_test.append(cv2.imread(dir_img_test+ldseg4[j]))
    Y_test.append(cv2.imread(dir_seg_test+ldseg3[j]))
```

Le seul prétraitement que j'ai réalisé sur les images est, le redimensionnement, pour qu'elles soient toutes de la même taille à l'entrée du réseau, parce que l'architecture VGG, exige une certaine taille d'images (explication dans la partie suivante)

```
X_train_resized=[]
Y_train_resized=[]
X_test_resized=[]
Y_test_resized=[]

for k in range (len(ldseg2)):

    X_train_resized.append(cv2.resize(X_train[k],(224, 224)))
    Y_train_resized.append(cv2.resize(Y_train[k],(224, 224)))

for l in range (len(ldseg4):
    X_test_resized.append(cv2.resize(X_test[l],(224, 224)))
    Y_test_resized.append(cv2.resize(Y_test[l],(224, 224)))
```

Après le redimensionnement des images à une taille égale à : (224, 224,3) voici à quoi elles ressemblent :

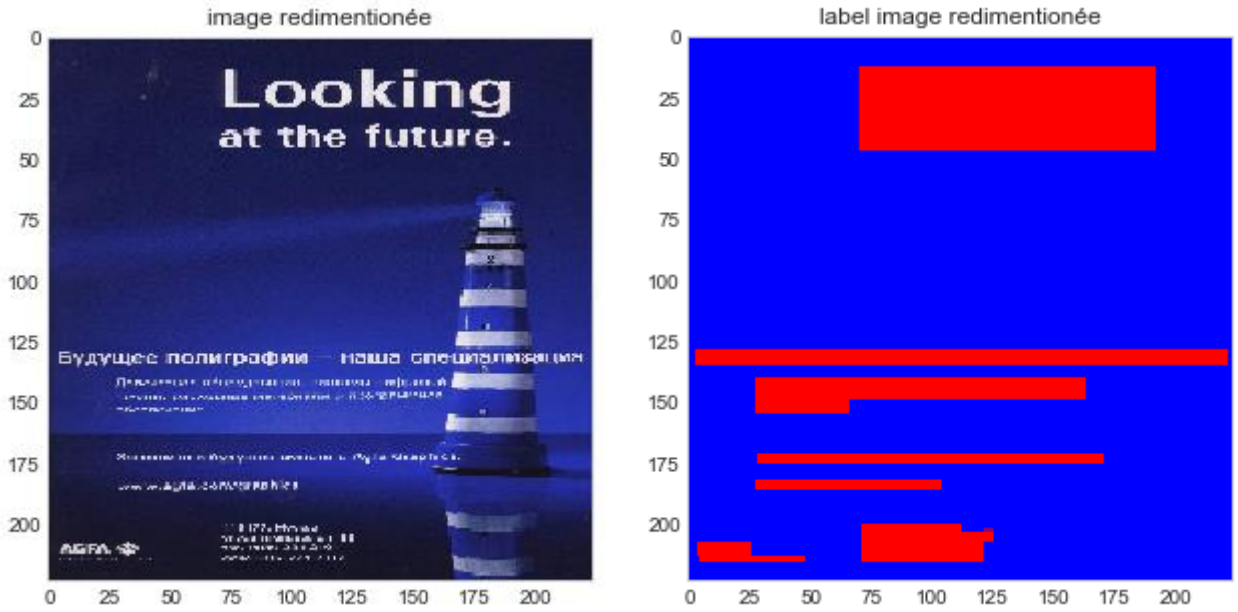
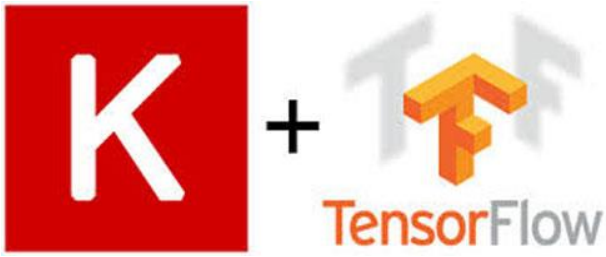


Figure 14: Images redimensionnées

4. Création du réseau de neurone

L'idée est de concevoir un réseau de neurone à convolution qui empile des couches de convolution avec des couches de sous-échantillonnage, telles que le Pooling maximal, et d'éliminer les couche entièrement connectées, pour arriver à réaliser la tache de la segmentation, j'ai remplacé donc ces couches par des couches de convolution, et en stockant les informations spatiales pour permettre une classification par pixel. Les librairies KERAS, et TENSORFLOW de python permettent de créer un tel réseau de neurone.



Le modèle que j'ai créé est d'architecture FCN, en faisant appel au poids du réseau VGG16 pour la segmentation sémantique. Et en éliminant la dernière couche de la classification, et en convertissant toutes les couches entièrement connectées en couches de convolutions. J'ai ajoutent une convolution 1 x 1 avec une dimension de canal identique au nombre de classes de segmentation (dans notre cas, 3) afin de prédire les scores à chacun des emplacements de sortie, suivis par des couches de dé-convolution pour avoir une image de sortie de la même taille que l'image d'entrée (224, 224,3).

- **VGG16**

VGG est une architecture de réseau à convolution populaire, publié par Simonyan et Zisserman en 2014. Il est entrainé sur la base d'images ImageNet. Il existe à l'origine deux versions de VGG : une avec 16 filtres et un autre avec 19 filtres, et les deux réseaux montrent de très bonnes performances comme le montre la figure suivante :

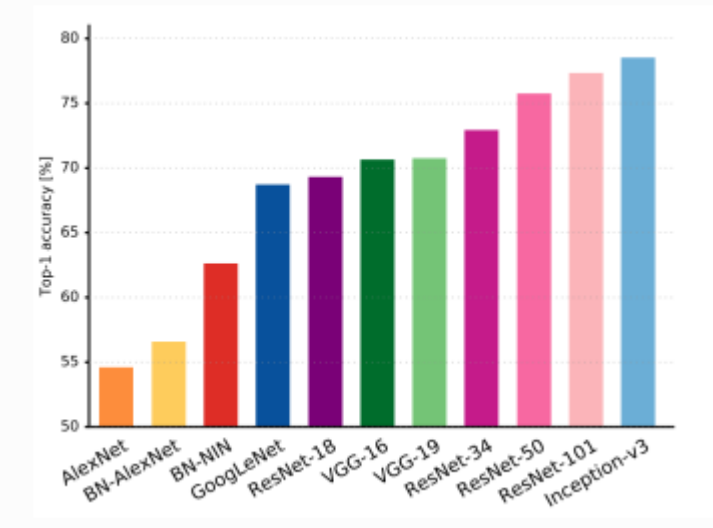


Figure 15: Performances des FCN

Le principal défaut de réseau VGG est sa taille imposante du réseau (environ 500 Mo) comparaît à d'autres réseaux tels que resNet, ce qui rend la vitesse de calcule lente pour une époque. Toutefois, c'est une architecture robuste et relativement lisible pour un réseau profond. Et c'est une bonne référence pour une première estimation des performances possibles pour notre projet.

Voici à quoi ressemble notre réseau

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 112, 112, 128	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 128	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0]
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv1[0][0]
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][0]
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][0]
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0]

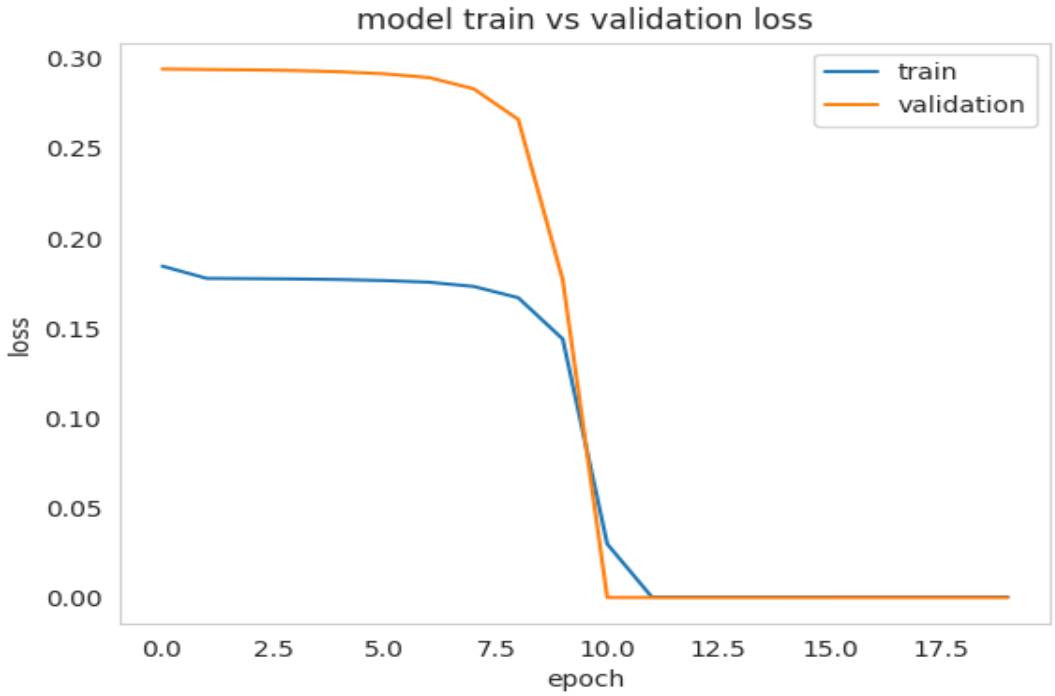
```
____________________________________________________________________________________________________
conv6 (Conv2D)                  (None, 7, 7, 64)     1605696     block5_pool[0][0]
____________________________________________________________________________________________________
pool4_11 (Conv2D)               (None, 14, 14, 3)    1539        block4_pool[0][0]
____________________________________________________________________________________________________
conv7 (Conv2D)                  (None, 7, 7, 64)     4160        conv6[0][0]
____________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 28, 28, 3)    36          pool4_11[0][0]
____________________________________________________________________________________________________
pool3_11 (Conv2D)               (None, 28, 28, 3)    771         block3_pool[0][0]
____________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 28, 28, 3)    3072        conv7[0][0]
____________________________________________________________________________________________________
add (Add)                       (None, 28, 28, 3)    0           conv2d_transpose_2[0][0]
                                                                 pool3_11[0][0]
                                                                 conv2d_transpose_1[0][0]
____________________________________________________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 224, 224, 3)  576         add[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)       (None, 224, 224, 3)  0           conv2d_transpose_3[0][0]
====================================================================================================
Total params: 16,330,538
Trainable params: 16,330,538
Non-trainable params: 0
____________________________________________________________________________________________________
```

Dans la première partie, j'ai utilisé cinq blocks, dans chaque bloque il y'a deux couches de convolutions, qui englobe une fonction d'activation de type ReLu, et un Padding (ajout de zéros sur les bords de l'image pour ne pas réduire sa taille lors de la convolution), et une couche de MaxPolling appliquée sur les images avec un filtre de taille 2X2, tous les filtres utilisées dans les couches de convolutions sont de taille 3X3, avec un nombre allons de 64 sur le premier block, jusqu'à 512 sur le dernier.

Dans le deuxième partie, et lors de la réalisation de la dé-convolution, pour agrandir les images, et avoir une sortie de la même taille que l'entrée, j'ai utilisé des couches de Upsimpling (le contraire de ce que fais le MaxPooling), puis des couches de dé-convolution.

- **Entrainement**

Après la création du modèle, une machine virtuelle avec un GPU assez puissant m'a était affilié pour lancer l'entrainement à distance, une fois l'entrainement terminé, j'ai pu récupérer le modèle sous-forme de fichier (h5), pour faire la prédiction et tester la performance du réseau. Voici la figure du LOSS tracé sur les données d'entrainement et de validation, et l'affichage associer.




```
Epoch 1/20
 - 119s - loss: 0.1846 - acc: 0.3328 - val_loss: 0.2943 - val_acc: 0.3356
Epoch 2/20
 - 116s - loss: 0.1778 - acc: 0.3407 - val_loss: 0.2940 - val_acc: 0.3442
Epoch 3/20
 - 116s - loss: 0.1776 - acc: 0.3454 - val_loss: 0.2938 - val_acc: 0.3492
Epoch 4/20
 - 116s - loss: 0.1774 - acc: 0.3513 - val_loss: 0.2934 - val_acc: 0.3531
Epoch 5/20
 - 116s - loss: 0.1771 - acc: 0.3567 - val_loss: 0.2927 - val_acc: 0.3639
Epoch 6/20
 - 116s - loss: 0.1766 - acc: 0.3696 - val_loss: 0.2916 - val_acc: 0.3777
Epoch 7/20
 - 116s - loss: 0.1756 - acc: 0.3860 - val_loss: 0.2895 - val_acc: 0.3983
Epoch 8/20
 - 116s - loss: 0.1733 - acc: 0.4144 - val_loss: 0.2832 - val_acc: 0.4257
Epoch 9/20
 - 116s - loss: 0.1670 - acc: 0.4441 - val_loss: 0.2662 - val_acc: 0.4796
Epoch 10/20
 - 116s - loss: 0.1440 - acc: 0.5803 - val_loss: 0.1770 - val_acc: 0.7478
Epoch 11/20
 - 116s - loss: 0.0300 - acc: 0.9384 - val_loss: 2.1302e-04 - val_acc: 1.0000
Epoch 12/20
 - 116s - loss: 5.1635e-04 - acc: 1.0000 - val_loss: 1.9151e-04 - val_acc: 1.0000
Epoch 13/20
```

Figure 16: LOSS

- **Performance du Modèle**

Par manque de données, (images, et masque vérité terrain), on ne peut pas vraiment connaitre les performances exacte du modèle, toute fois j'ai utilisé les données de validation pour faire une segmentation sémantique de pages de journaux et voir ce que ça donne.

Je me suis basé sur le calcul de l'intersection sur l'union, pour mesurer la performance de mon modèle, j'ai eu le résultat suivant :

```
class 00: #TP=118198, #FP= 43601, #FN=651328, IoU=0.145
class 01: #TP=   986, #FP=485280, #FN= 1150, IoU=0.002
class 02: #TP=108781, #FP=296850, #FN=173253, IoU=0.188
_________________
Mean IoU: 0.112
```

On remarque que la reconnaissance de la deuxième classe est très mauvaise, vu son intersection sur l'union, et en générale le système ne fournie pas de très bonne résultats

Voici un exemple d'image segmentée avec mon réseau :

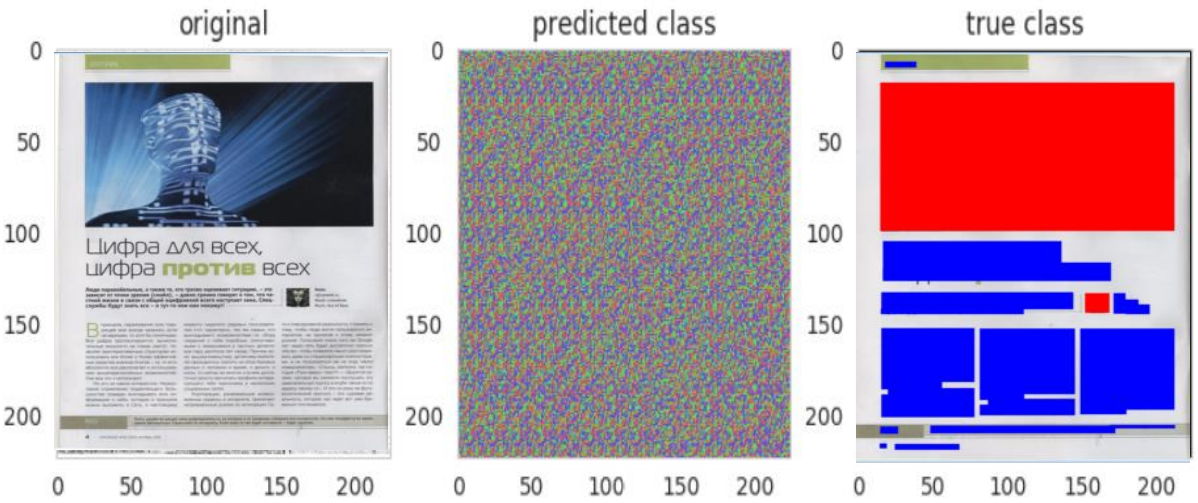


Figure 17: Résultats de la segmentation

TRAVAIL REALISE ET DIFFICULTE RENCONTRE

- **Travail réalisé :**

Durant la période de la réalisation de notre projet j'ai réussi à réaliser :

- Une étude bibliographique sur les réseaux de neurone pour la segmentation sémantique.
- Création d'un réseau de neurone en langage python, l'entrainé et mesurer ses performances.

- **Contrainte technique :**

- Trouver un Data Set suffisamment grand avec des données étiquetées.
- Choix de l'architecture du modèle, et son implémentation.
- Quel prétraitement réaliser sur les images.

CONCLUSION ET PERSPECTIVES

Dans ce travail, nous avons étudié la tâche de segmentation sémantique à l'aide de réseaux de neurones entièrement convolutifs. La segmentation sémantique est souvent utilisée comme première étape pour de nombreuses méthodes de compréhension de scènes et est donc une tâche de grande importance. En dépit des excellents résultats qu'ils permettent d'obtenir, les FCN nécessitent souvent une régularisation explicite afin de pouvoir obtenir une segmentation visuellement régulière, soit en altérant la fonction objectif, soit par post-traitement à base de modèle graphique. Nous avons essayé durant ce projet de créer un modèle de réseau de neurones entièrement convolutif, pour faire de la segmentation des images de journaux, pour en extraire principalement les zones contenant du texte, afin d'utiliser ce texte pour d'autre application, malheureusement le manque de base de données d'images de pages de journaux déjà segmentées manuellement, qui servent de vérité terrain, pour assurer l'entrainement du modèle, a été un problème qui a causé l'obtention de résultats pas très performants. En effet la base de données qui a servie à réaliser l'entrainement, contient seulement 80 images d'entrainement, et 21 images de validations, ce qui est très peu pour un réseau de neurone profond tel que le nôtre.

Le projet de la segmentation par Deep Learning est un travail compliqué où on doit choisir une architecture du réseau adaptée à notre problème et nos besoins, puis avoir assez de données étiquetées pour que le réseau apprend mieux, donc au futur on pourra continuer sur ce travail, avec une quantité d'images plus importante, et en essayant plusieurs architectures possibles, les entrainer, sans avoir recourt au poids d'autres réseaux pré-entrainés, et donc générer ses propres poids, ces perspectives pourraient être des solutions pour un système plus performant.

BIBLIOGRAPHIE

- A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. VillenaMartinez, and J. Garcia-Rodriguez. A Review on Deep Learning Techniques Applied to Semantic Segmentation. arXiv :1704.06857 [cs], Apr. 2017. arXiv : 1704.06857

- X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun. 3d Graph Neural Networks for RGBD Semantic Segmentation. In IEEE International Conference on Computer Vision (ICCV), pages 5199–5208, 2017. 5

- H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520–1528.

- Convolutional Neural Networks for Page Segmentation of Historical Document Images, Kai Chen and Mathias SeuretDIVA (Document, Image and Voice Analysis) research group Department of Informatics, University of Fribourg, Switzerland

- dhSegment: A generic deep-learning approach for document segmentation Sofia Ares, OliveirayBenoit Seguin, Frederic Kaplany yDigital Humaniti Laboratory, EPFL, Lausanne, VD, Switzerland .

- Fully Convolutional Neural Networks for Newspaper Article Segmentation Benjamin Meier, Thilo Stadelmann, Jan Stampfli, Marek Arnold, and Mar Cieliebak Zurich University of Applied Sciences Winterthur, Switzerland