Détection de constructions illégales

Rapport de fin de projet

Projet effectué par Clément Molinier & David Roch

Suivi par Thomas Boehler

Formation DataScientest
Cohorte *Data Scientist* juin 2023

Mars 2024

Sommaire

1 Introduction

Ce document présente la démarche et les résultats d'un projet de détection de constructions illégales, dans le cadre d'un projet de formation au métier de *data scientist*, proposé par la société DataScientest.

L'objectif pédagogique était de réaliser un projet complet, de l'identification des données à l'exploitation des résultats, en mettant en œuvre différentes techniques de *data science*.

Le but du projet était d'identifier des constructions reconnaissables sur des prises de vues aériennes, mais qui ne se retrouvaient pas sur les bases de données topographiques officielles.

Le projet s'est déroulé entre août 2023 et mi-mars 2024.

2 Exploration des données

2.1 Présentation et qualité des données

Nous utilisons deux bases de données en source ouverte, fournies par l'IGN (Institut Géographique National) :

- **BDORTHO** : ensemble d'images aériennes ou satellitaires du territoire français, sous forme de fichiers au format .jp2.
- **BDTOPO** : description vectorielle du territoire français et de ses infrastructures. BDTOPO est un *GeoDataFrame*, à savoir un *DataFrame* portant, pour chaque ligne, une forme géométrique.

La démarche globale consistera à observer BDORTHO à la recherche de bâtiments non décrits dans BDTOPO. En matière de *data science*, nous pouvons considérer que BDORTHO jouera le rôle d'entrée pour nos modèles, tandis que BDTOPO sera la cible. Plus précisément, BDTOPO sera la cible pour entraîner nos modèles à la reconnaissance de bâtiments, et c'est ensuite l'écart entre nos prévisions et BDTOPO qui déterminera les bâtiments suspects d'être illégaux.

Les bases sont bien renseignées :

- BDORTHO couvre l'intégralité du territoire.
- BDTOPO comporte une grande quantité d'informations, dont une bonne partie n'est pas renseignée. Toutefois, les champs qui nous intéressent sont le type de bâti et la colonne **geometry** (qui, dans un *GeoDataFrame*, porte les données géographiques). Or ces deux champs ne comportent aucune valeur absente (NaN).

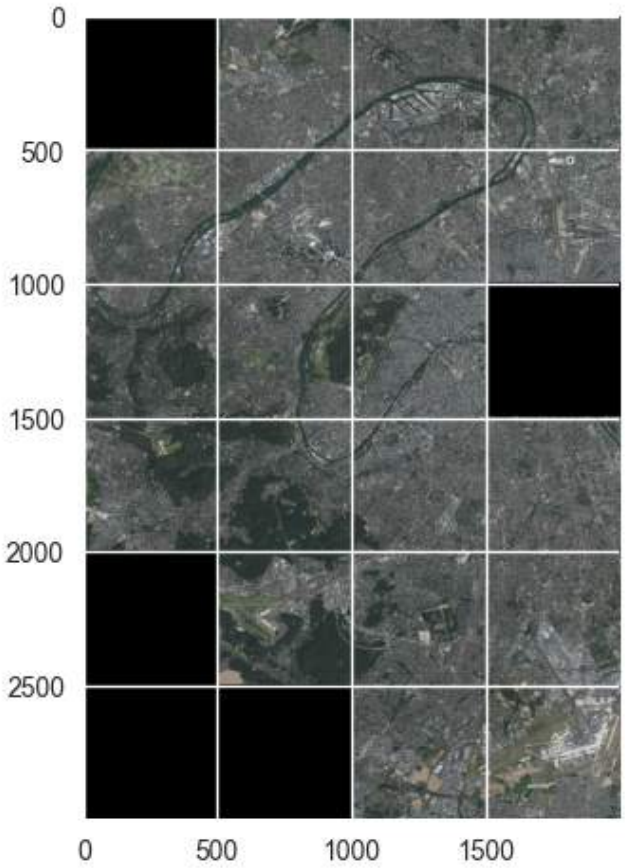
2.2 Exploration des données de BDORTHO

2.2.1 Carte du département

Parmi les différents éléments composant la base de données de cette étude, on retrouve une bibliothèque de 19 images de 25 000 x 25 000 pixels. Les dimensions d'un pixel étant supposées correspondre à 20 cm, ces images couvrent donc une superficie de terrain d'environ 50km2.

Une première étape dans l'étude des images fut de les regrouper et positionner de manière à obtenir une seule image globale.

On peut observer le résultat sur l'image suivante :

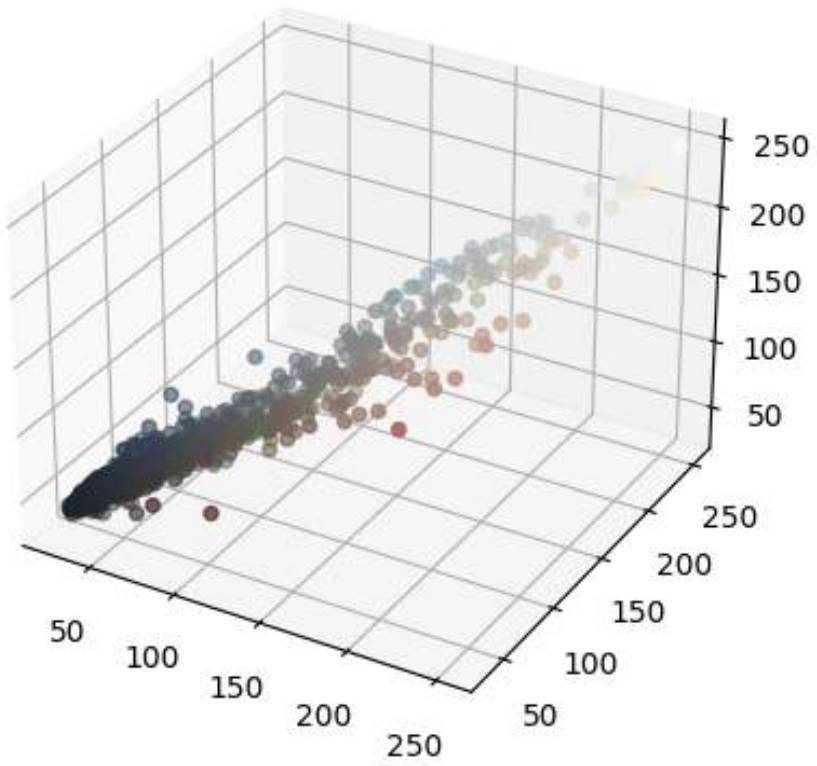


Cette première étape a permis de mieux appréhender le jeu de données et constater la forte densité de bâtiments, comme attendu de ce département. Pour obtenir cette représentation, les images ont été réduites à 500 x 500 pixels et positionnées dans une matrice *numpy* adaptée.

2.2.2 Positionnement des pixels en RGB

Une autre approche pour tenter d'étudier ces images fut de positionner les pixels dans un graphique 3D RGB (chaque axe correspondant à une composante colorimétrique : rouge, vert et bleu). Malheureusement, le nombre élevé de pixels des images ne permettant pas une utilisation fluide et une interprétation simple des résultats, les images ont été réduites.

On peut voir sur la figure suivante l'exemple d'une image réduite à 40 x 40 pixels.

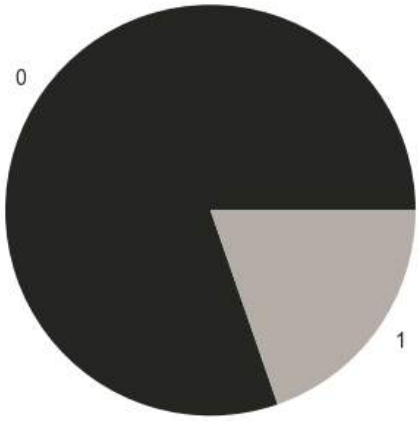


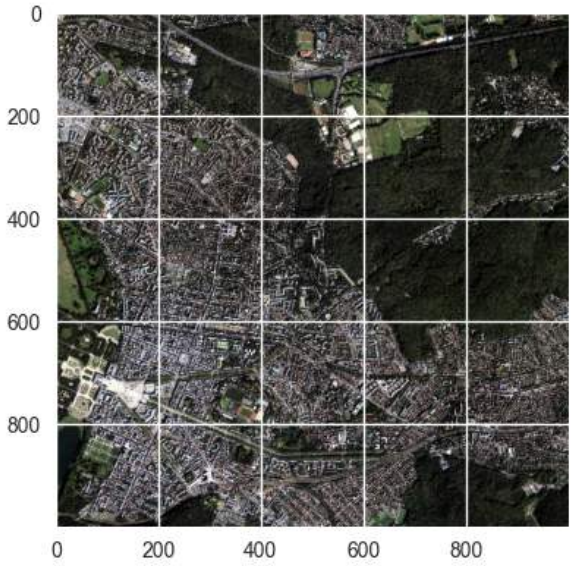
Malgré le fort taux de compression, on peut voir émerger des informations intéressantes. Tout d'abord, la majorité des points est concentrée sur une plage d'intensité faible 0-75. On a donc des couleurs ternes majoritairement et tirant sur le noir. On peut également supposer que la dynamique de l'image est sous-exploitée et qu'un traitement sur la dynamique, avant usage, serait salvateur. Enfin, en étudiant, on peut voir deux clusters, un bleu et un rouge, apparaître avec des valeurs d'intensité plus fortes. Il aurait été intéressant de pouvoir remonter à leur position dans l'image d'origine avant compression pour étudier à quoi ils correspondent.

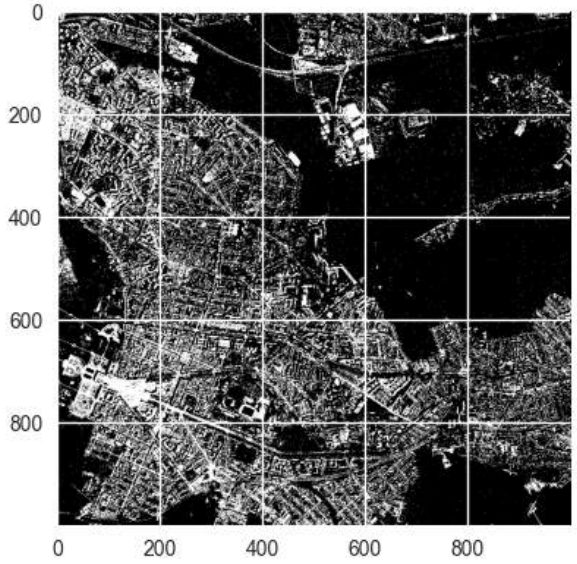
2.2.3 Approche clustering sur les pixels

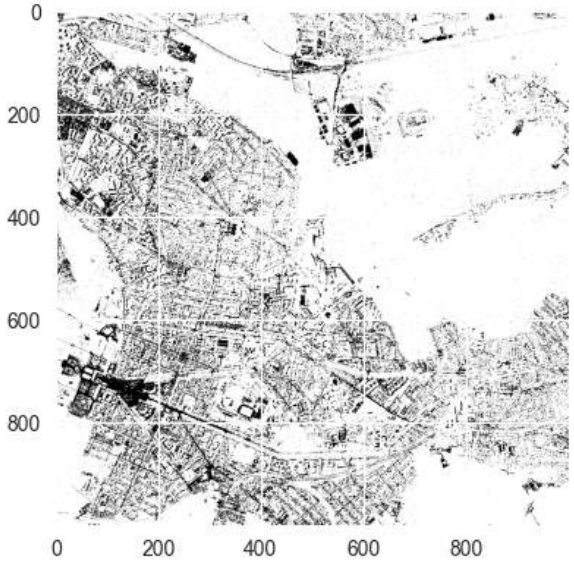
Afin de mettre à profit les algorithmes de Machine Learning étudiés en début de formation, nous avons appliqué un algorithme de clustering sur les images.

On peut voir sur l'image qui suit, le résultat obtenu avec un nombre de clusters de 2 :









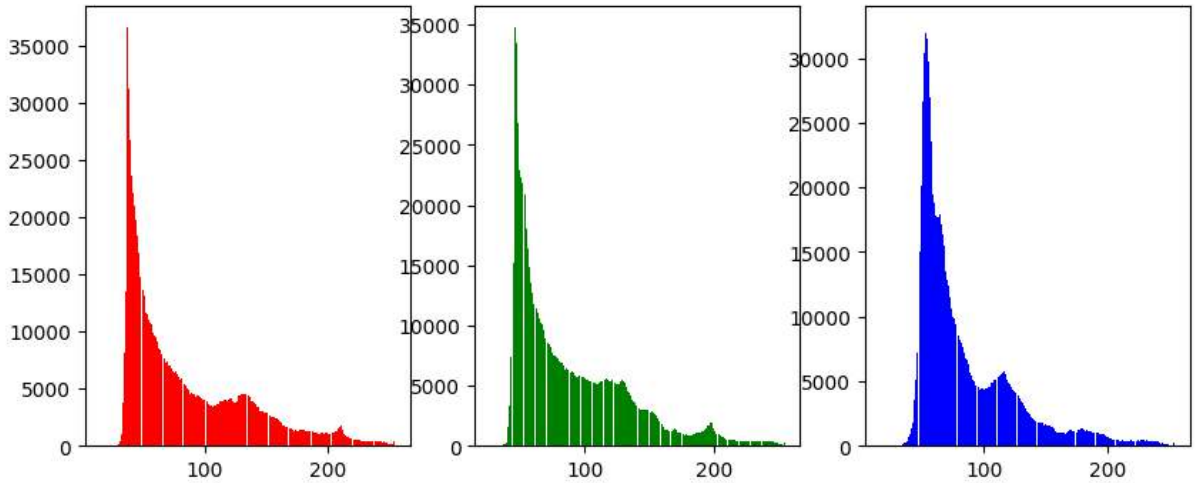
On peut voir une représentation en camembert de la proportion de pixels appartenant aux différents clusters, ainsi que la couleur moyenne du cluster (obtenue en moyennant les valeurs des pixels du cluster). A sa droite, l'image est étudiée. Enfin, les images suivantes correspondent aux masques des différents clusters. On constate que cette approche permet d'obtenir une bonne approximation de la segmentation des bâtiments dans l'image. On peut ainsi avoir une idée du ratio bâtiment / végétation.

2.2.4 Analyse des histogrammes RGB

Enfin, les images ont été préparées pour leur utilisation avec un algorithme de réseaux de neurones. Pour cela, nous avons segmenté la base en images de 1000 x 1000 pixels correspondant, sur le terrain, à un carré de 200 mètres de côté.

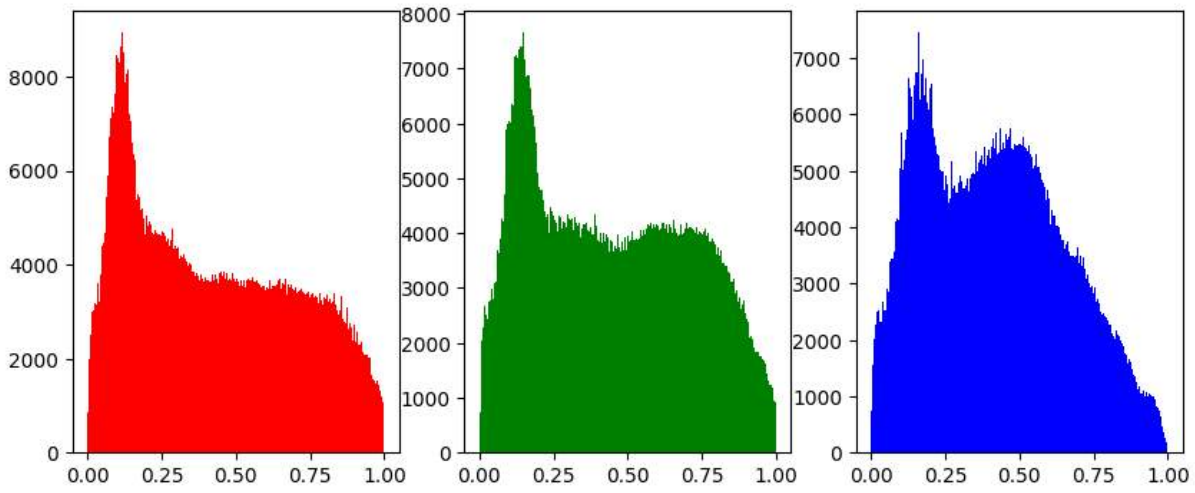
Une fois découpée ainsi, la base contient 11 875 images (contre 19 images initialement).

Cependant, si l'on étudie ces images et leur histogramme…



… On constate que l'hypothèse faite lors de tests précédents sur la sous-exploitation de la dynamique des images est vérifiée.

Pour remédier à ce problème, une fonction d'optimisation d'open cv2 a été appliquée, dont on peut observer les résultats ci-dessous :

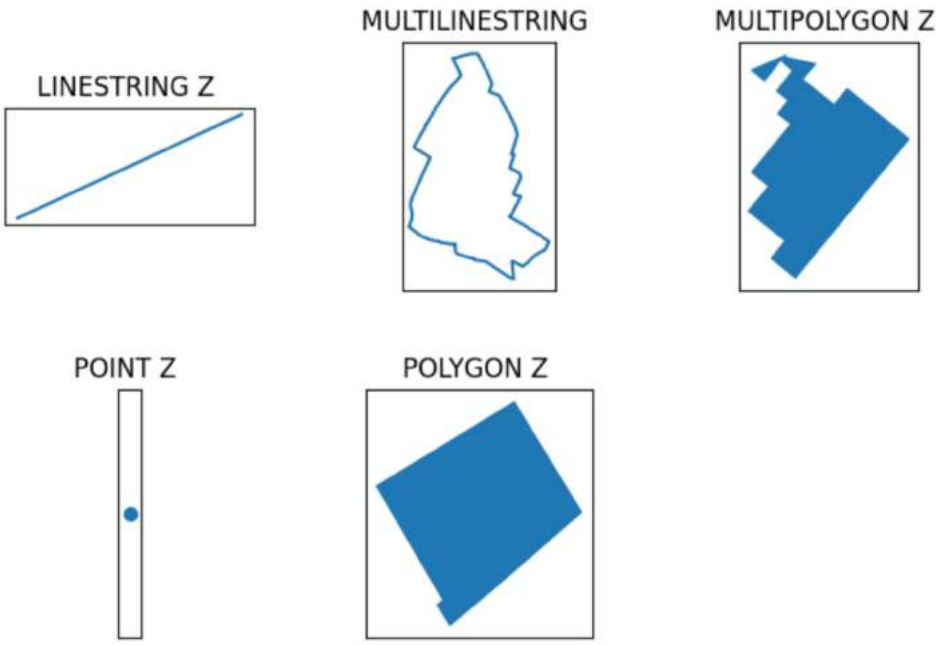


Si l'histogramme confirme le bon fonctionnement de ce pré-traitement, on peut aussi voir à l'œil que nombres de détails dans l'ombre, notamment la façade des bâtiments, sont maintenant visibles.

2.3 Exploration des données de BDTOPO

2.3.1 Sélection des données utiles

Voici un exemple de chacune des figures que l'on trouve dans la base BDTOPO :



Seules les figures ayant une surface nous intéressent. Nous excluons donc les lignes et points, pour nous concentrer sur les figures fermées (**polygones et multi-polygones**).

Une analyse rapide nous montre que les multi-polygones représentent 0,09 % des figures intéressantes, mais 1,3% de leur surface.

Nous choisissons donc, pour l'instant, de conserver les multi-polygones, mêmes s'ils nécessiteront vraisemblablement un traitement différencié.

À ce stade, le dataframe comporte 743 190 polygones et multi-polygones.

Par ailleurs, l'examen des colonnes Category et Type fait apparaître que les constructions qui nous intéressent se trouvent dans la catégorie **BATI**.

Au sein de cette catégorie, une fois les points et lignes supprimés, seuls les types **BATIMENT** et **RESERVOIR** nous intéressent.

Par ailleurs, il pourra être intéressant, selon les résultats des investigations préliminaires, d'exclure certains polygones de l'analyse. En effet, ils représentent de larges surfaces sur lesquelles il n'y aura vraisemblablement pas de construction illégale, et qui risquent de fausser l'apprentissage (présence de formes spécifiques) :

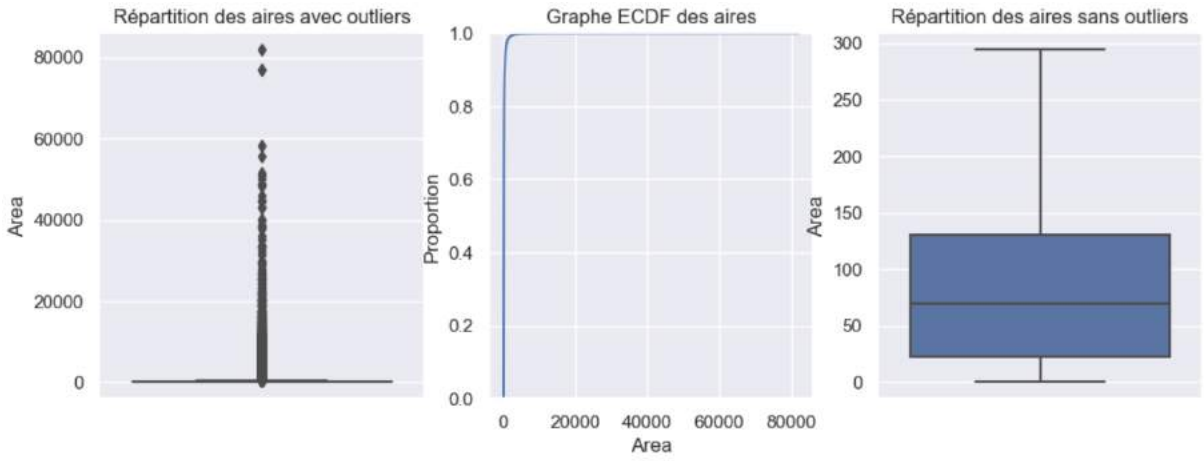
Category	Type
BATI	CIMETIERE
BATI	TERRAIN_DE_SPORT
BATI	CONSTRUCTION_SURFACIQUE (*)
TRANSPORT	AERODROME
TRANSPORT	PISTE_D_AERODROME

(*) barrages, écluses, etc.

Par la suite, nous nous intéressons donc au dataframe issu de BDTOPO qui ne comporte que les objets de type BATIMENT ou RESERVOIR. Il comporte 694 302 lignes (respectivement 693 693 bâtiments et 609 réservoirs).

2.3.2 Aires

Nous calculons ici l'aire de chacun des polygones de la base, pour en visualiser la répartition. Une première analyse montre que les *outliers* sont très représentés :

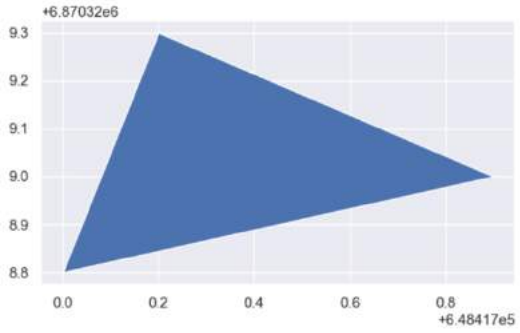


En effet, les outliers représentent la majeure partie de la surface totale de la base :

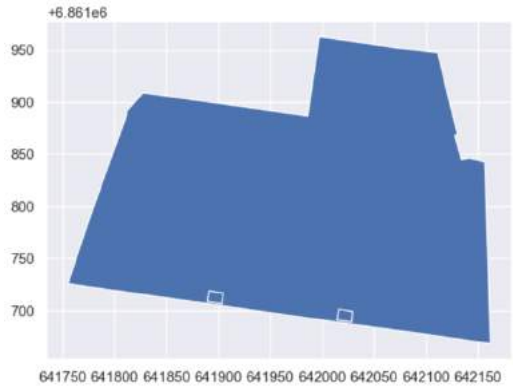
Outliers	*En nombre*	*En pourcentage*
Figures	76 377	11 %
Aire correspondante	69 449 914	61 %

2.3.3 Valeurs extrêmes de l'aire

Le plus petit polygone de la base est un BATIMENT de nature indifférenciée (aire= 0.205) :

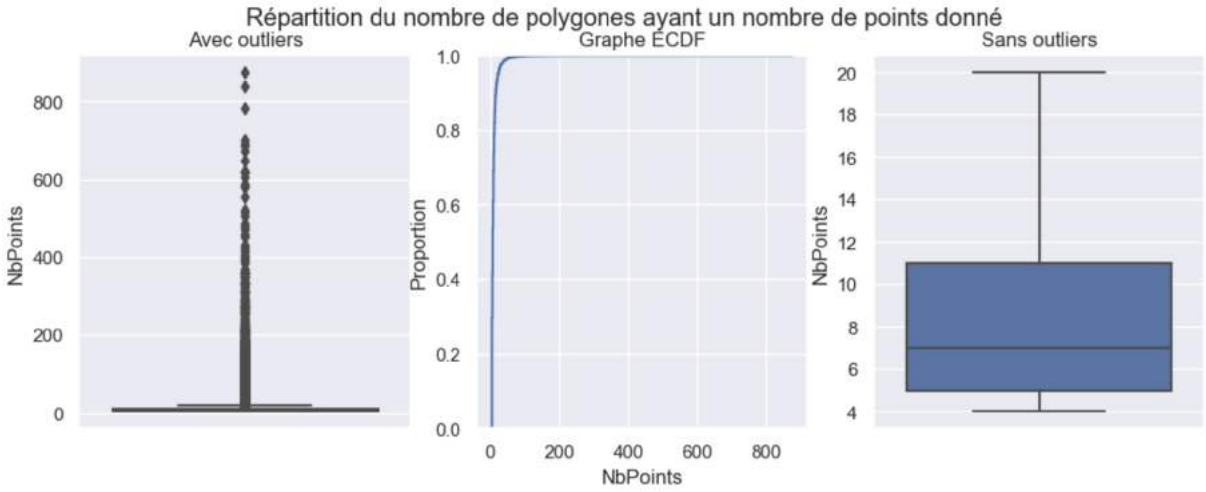


Le plus gros polygone de la base est un multi-polygone RESERVOIR (aire= 81926.91) :



2.3.4 Complexité des polygones

Le nombre de points délimitant les polygones se répartit ainsi (*) :



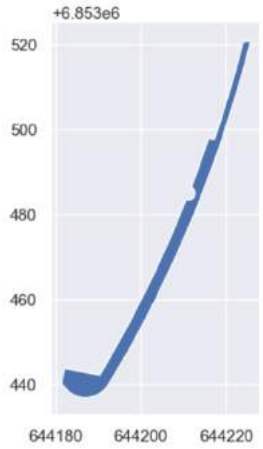
(*) le point fermant le polygone est également compté

Là encore, les outliers sont très significatifs :

Outliers	*En nombre*	*En pourcentage*
Figures	44 787	6 %
Aire correspondante	1 611 376	24 %

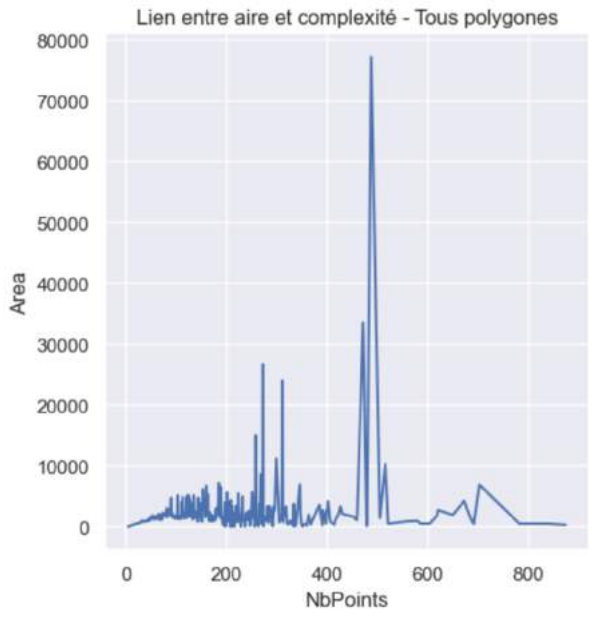
Il s'agira de s'assurer que les polygones très complexes ne pénalisent pas les temps de traitement ; sinon, on pourra envisager de les simplifier pendant le pré-processing.

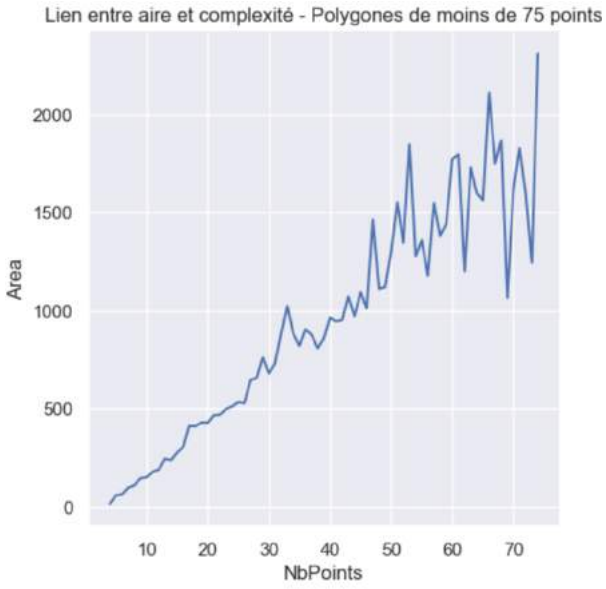
Pour information, la base compte 1 100 polygones de plus de 100 points. Le polygone le plus complexe comporte 875 points, il s'agit d'un BATIMENT de nature indifférenciée :



2.3.5 Lien entre aire et complexité

Les polygones les plus complexes (en nombre de points) sont aussi ceux ayant l'aire la plus , du moins pour les polygones d'une taille inférieure à 50 points environ :

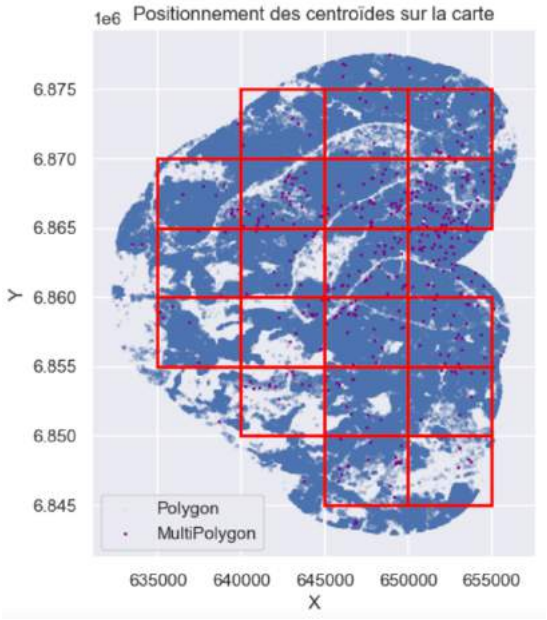




2.3.6 Position des polygones sur la carte BDORTHO

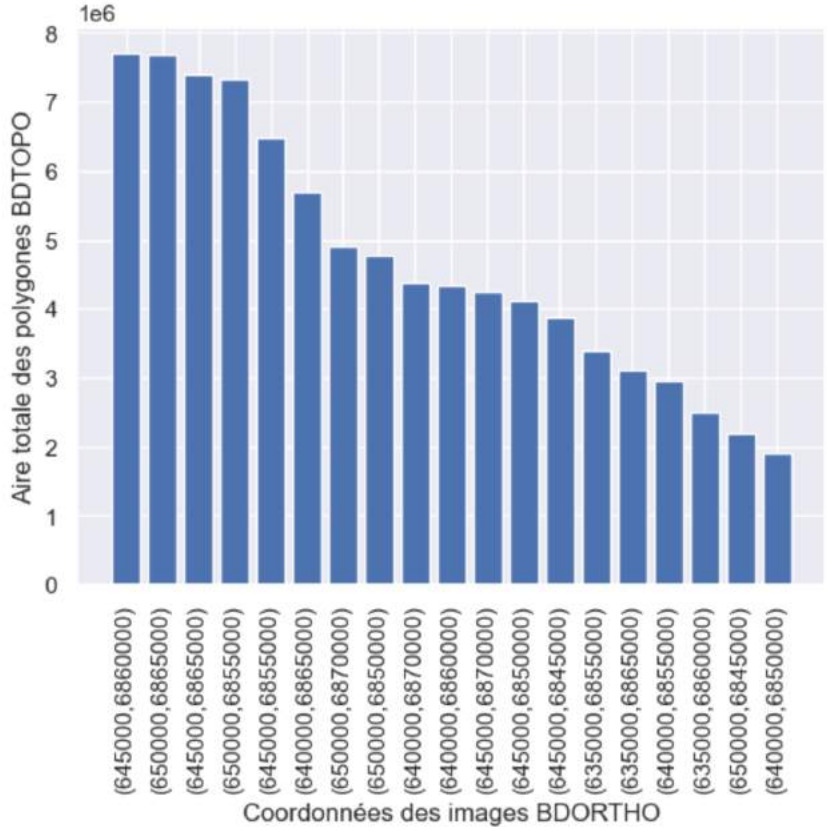
Nous cherchons ici à voir où se trouvent les bâtiments d'un point de vue géographique. Pour cela, nous affichons en bleu les informations sur les polygones, et rappelons en rouge les limites des 19 cartes issues de BDORTHO correspondant au département 92.

Le positionnement des centroïdes (chaque centroïde étant représenté par un point bleu) semble montrer que certaines zones sont plus denses que d'autres :



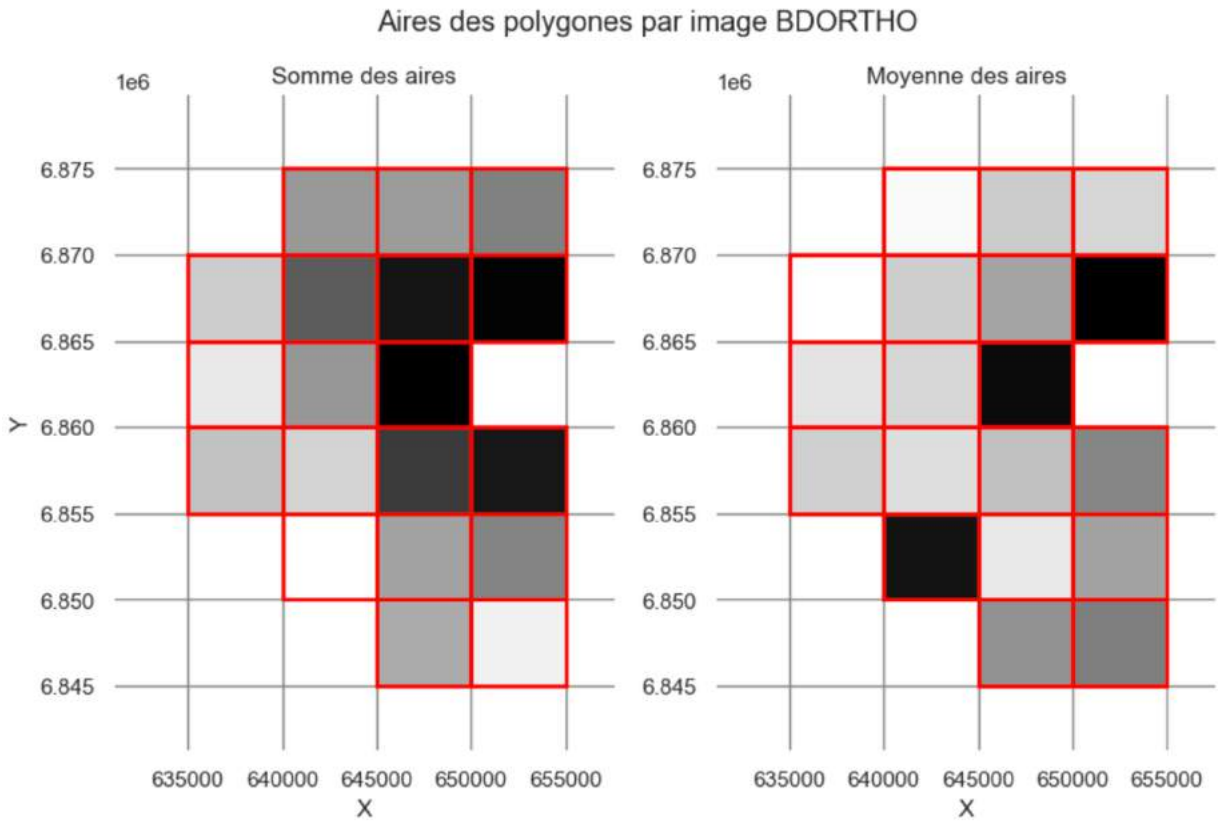
Notons également que certains polygones ont leur centroïde en-dehors des cartes BDORTHO, tout en étant probablement en intersection avec les cartes.

Tentons de préciser la densité de construction en calculant, pour chacune des 19 cartes, les surfaces couvertes par les polygones en intersection avec celles-ci :



Les aires sont très variables d'une carte à l'autre (facteur 3,5 environ).

La carte suivante matérialise les images de BDORTHO (en encadré rouge) et la densité des aires (en noir les zones où l'aire est maximale, en blanc celles où l'aire est minimale, les niveaux de gris présentant les valeurs intermédiaires) :



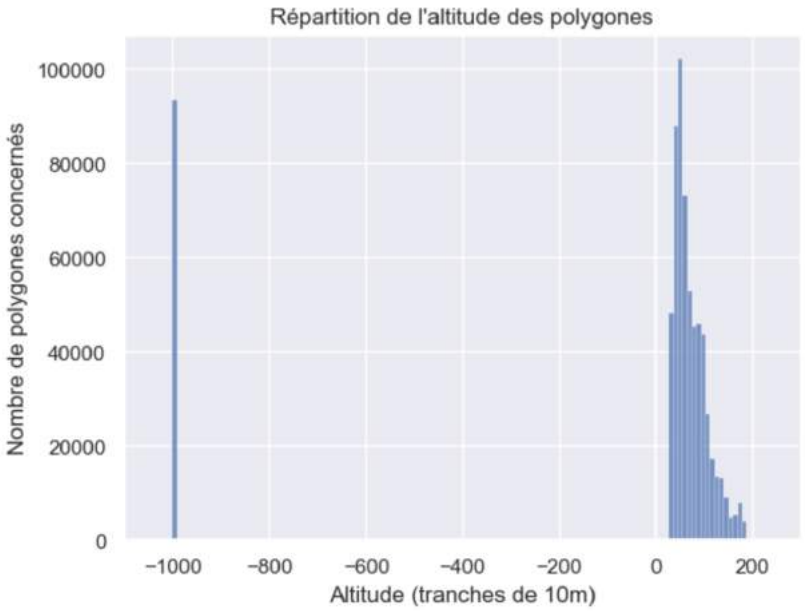
Les zones les plus urbanisées se situent à l'est du département. Pour autant, la carte la plus au sud-ouest présente une aire moyenne importante, avec probablement un nombre de polygones modéré.

NB : le calcul a été fait, pour ce dernier graphe, uniquement sur les polygones (en excluant les multi-polygones).

2.4 Altitude des polygones

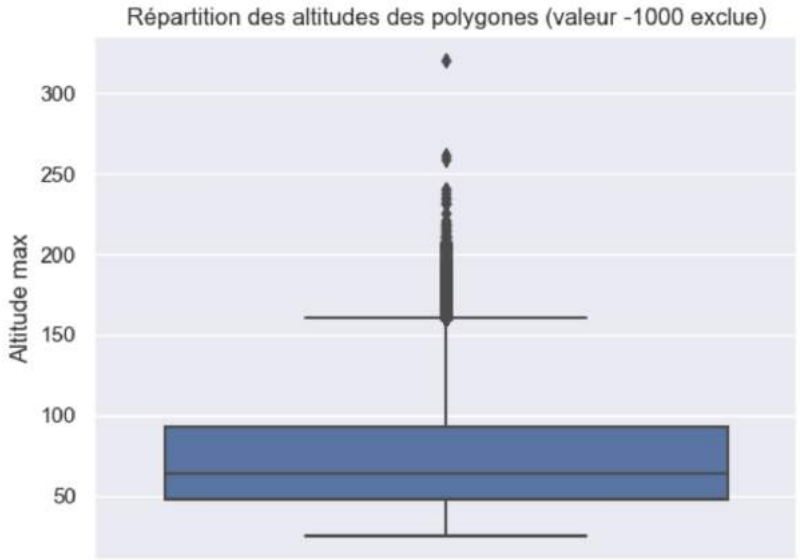
Nous calculons l'altitude de chaque polygone (en prenant le maximum des altitudes des points qui composent celui-ci, les multi-polygones étant exclus).

Les polygones se répartissent ainsi :

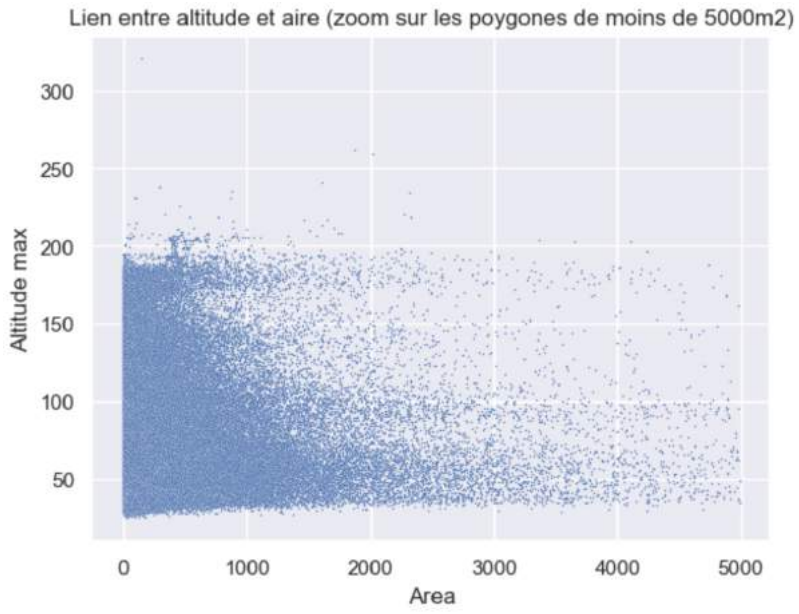


On remarque l'absence d'altitude égale à 0, mais un très grand nombre d'altitudes à -1000m. Il est possible que la valeur -1000 soit utilisée lorsque l'altitude du bâtiment n'est pas connue.

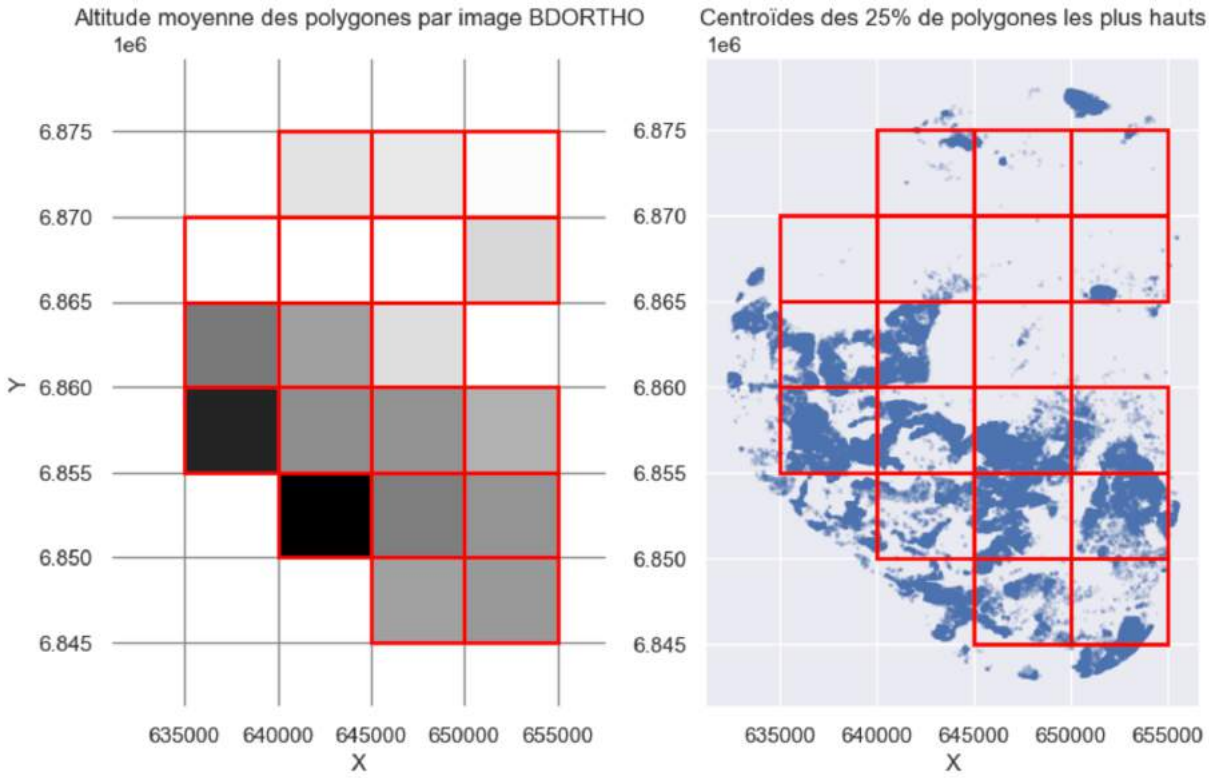
Par ailleurs, la présentation sous forme de boîte à moustaches fait apparaître une quantité non négligeable de bâtiments élevés, puisque le 3ème quartile est égal à 93m :



On n’observe pas de corrélation apparente entre l’altitude des polygones et leur aire, comme l’indique le graphe suivant, où chaque polygone est représenté par un point :



Intéressons-nous enfin à l’altitude par image issue de BDORTHO. La répartition des centroïdes en fonction de l’altitude de leur polygone est la suivante :



On observe que l’altitude moyenne et les polygones les plus hauts sont plutôt situés au sud-ouest de la carte.

3 Modélisation

3.1 Choix des modèles

La problématique étant liée à la reconnaissance de formes sur des images présentant une certaine complexité, l'approche *deep learning* s'est rapidement imposée. En effet, on dispose aujourd'hui de modèles très performants pour résoudre ce type de problème.

Divers réseaux de neurones ont donc été utilisés, soit créés ex nihilo, soit par entraînement complémentaire après *transfer learning* d'un modèle existant.

Nous avons mis en place quatre approches, avec des objectifs légèrement différents :

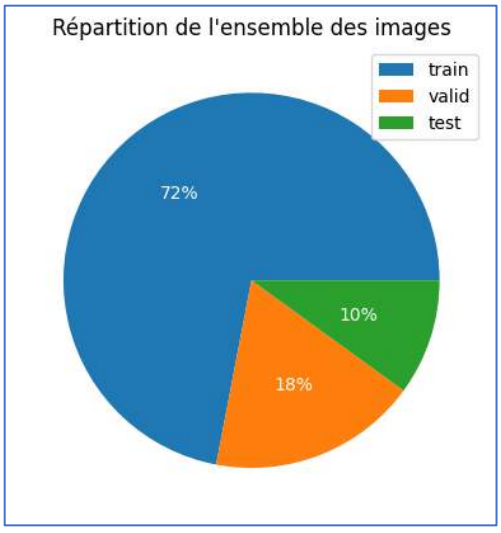
- La régression, afin de compter le nombre de bâtiments par zone géographique.
- Un modèle de régression déjà pré-entraîné (*transfer learning*), afin d'évaluer si nous pouvions aller plus loin dans la précision du comptage.
- La classification sous forme de segmentation sémantique, pour identifier les zones en discordance entre la base d'images aériennes et le cadastre.
- La classification sous forme de segmentation d'instances, de manière à profiter des derniers développements dans les modules de *computer vision* et améliorer ainsi la qualité de la classification.

3.2 Préparation des données

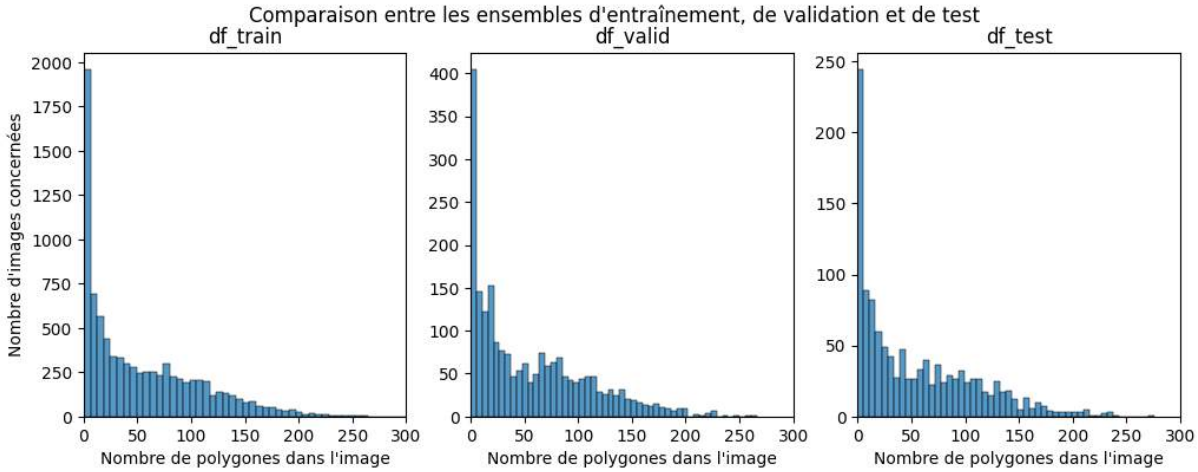
Compte tenu de la volumétrie des données et du choix de modèles consommateurs en ressources, nous avons segmenté la base en images de 1000x1000 pixels ; la résolution étant de 20 cm par pixel, chacune de ces images correspond, sur le terrain, à un carré de 200 mètres de côté.

Une fois découpée ainsi, la base contient 11 875 images (contre 19 images initialement).

Des ensembles d'entraînement (8549 images), validation (2138 images) et test (1188 images) ont été utilisés pour tous les modèles :



Le nombre de polygones est globalement cohérent entre ces ensembles :



Dans la suite du document, les résultats sont présentés sur la base de l'ensemble de test (qui n'a pas servi lors des entraînements).

Par ailleurs, nous appliquons, lorsque les modèles le permettent, un enrichissement des données par translation et symétrie (en excluant la rotation pour des raisons de performances). Cet enrichissement est fait en utilisant les fonctionnalités des générateurs.

3.3 Approche Régression

3.3.1 Objectifs

L'utilisation d'un modèle de régression répondait à deux objectifs :

- Identifier des portions du département sur lesquelles le nombre de bâtiments calculé est très différent du nombre officiellement déclaré. Ce résultat pourrait alors être utilisé comme criblage, pour prioriser les portions du territoire devant faire l'objet d'un examen plus approfondi.
- Valider notre capacité à mettre en œuvre des modèles de deep learning relativement simples avec un résultat satisfaisant.

3.3.2 Modélisation et apprentissage

Nous utilisons un modèle associant des couches de convolution pour capture les caractéristiques des *features*, et des couches denses pour concentrer ces caractéristiques en une variable de régression. Le modèle comporte :

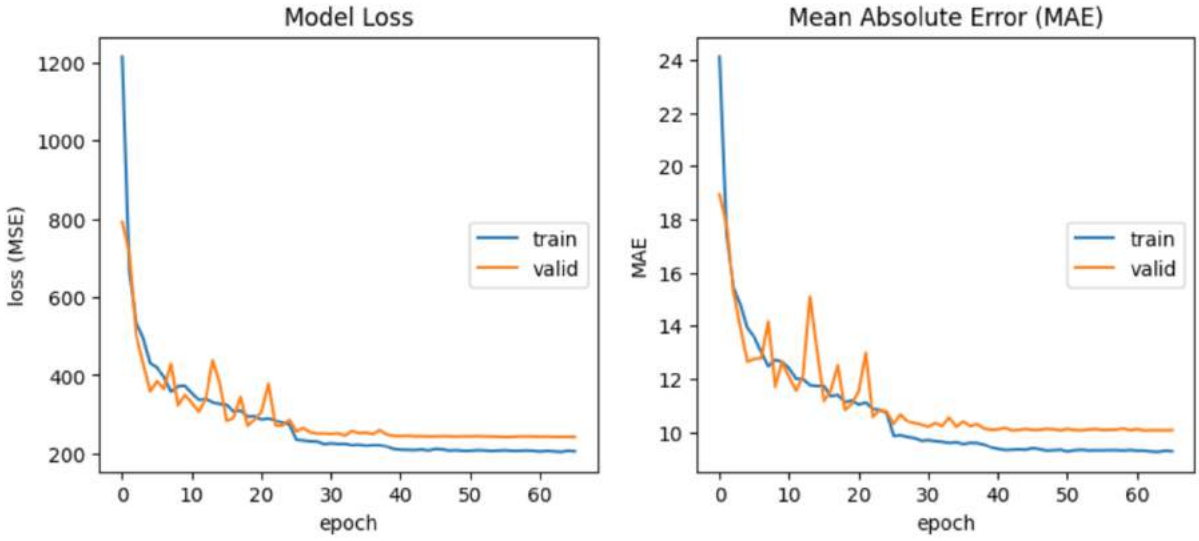
- 5 blocs de convolution associés à un *MaxPooling* et une fonction d'activation *LeakyReLU*, sortie à 512 canaux pour une taille d'image réduite à 14x14.
- 1 couche de *Dropout* à 50%.
- 1 bloc à 2 couches denses, avec une fonction d'activation *Linear*.

Au total, le modèle comporte 1 831 745 paramètres, tous entraînables.

La fonction de perte est la MSE (erreur quadratique moyenne), bien adaptée à la régression, et l'optimiseur *Adam*. Nous ajoutons également des *callbacks* pour enregistrer les améliorations du modèle, adapter le taux d'apprentissage et permettre l'arrêt du modèle.

La taille des images en entrée est fixée à 512x512, afin de rester dans des temps d'entraînement acceptables, et les batchs de 16 images.

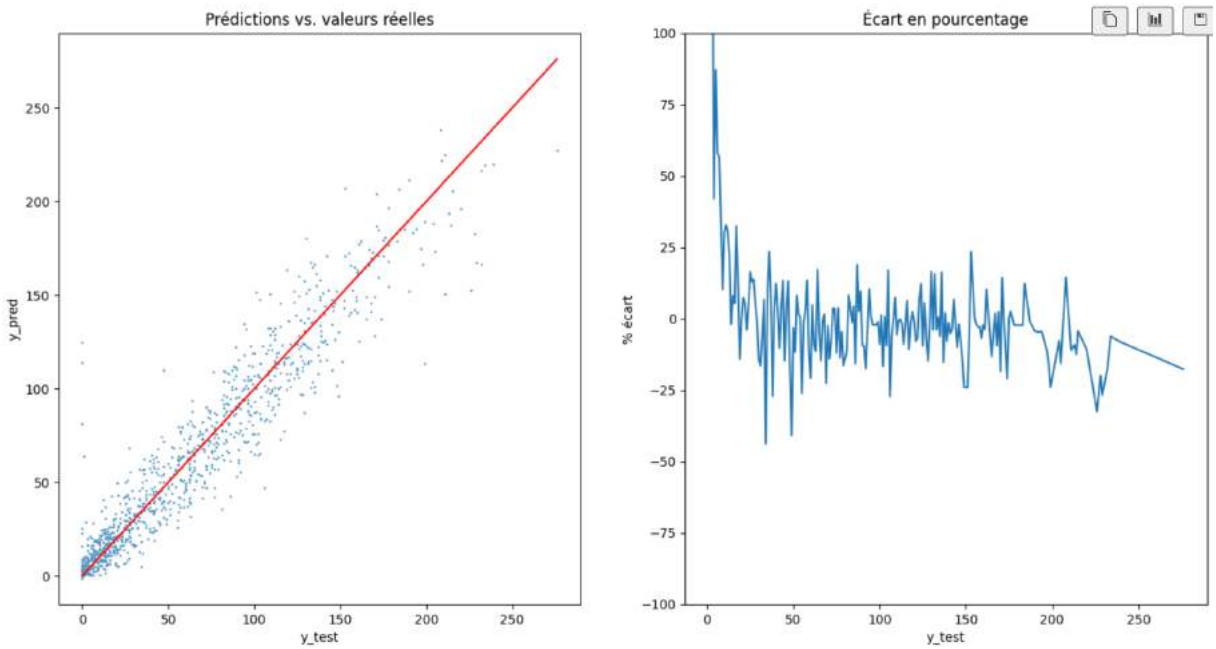
Le modèle se stabilise après environ 45 epochs. Après entraînement, les courbes de perte montrent une convergence vers un minimum, la MAE descendant à 10,05 :



La perte reste un peu supérieure sur l'ensemble de validation, ce qui peut témoigner d'un phénomène d'*overfitting* léger, et donc acceptable.

3.3.3 Résultats

Pour chaque image de l'ensemble de test, le nombre de bâtiments prédits est mis en regard du nombre de bâtiments réels :



On constate que le modèle n'est parvenu à détecter exactement le nombre de bâtiments que dans 13,6% des images.

Toutefois, il a prédit correctement l'ordre de grandeur du nombre de bâtiments :

- Le modèle a détecté le bon nombre de bâtiments à une tolérance de +/- 10 dans 63,6% des cas.
- La moyenne de l'écart entre les données prédites et les données réelles est de -0,6 ; l'écart-type de 15,4.
- Un test statistique a confirmé la corrélation forte entre la prédiction et la réalité :

Test de corrélation de Pearson
H0 : pas de corrélation entre pred et test
H1 : corrélation entre pred et test
Coefficient=0,95 et p-value=0.0
L'hypothèse H0 est donc rejetée

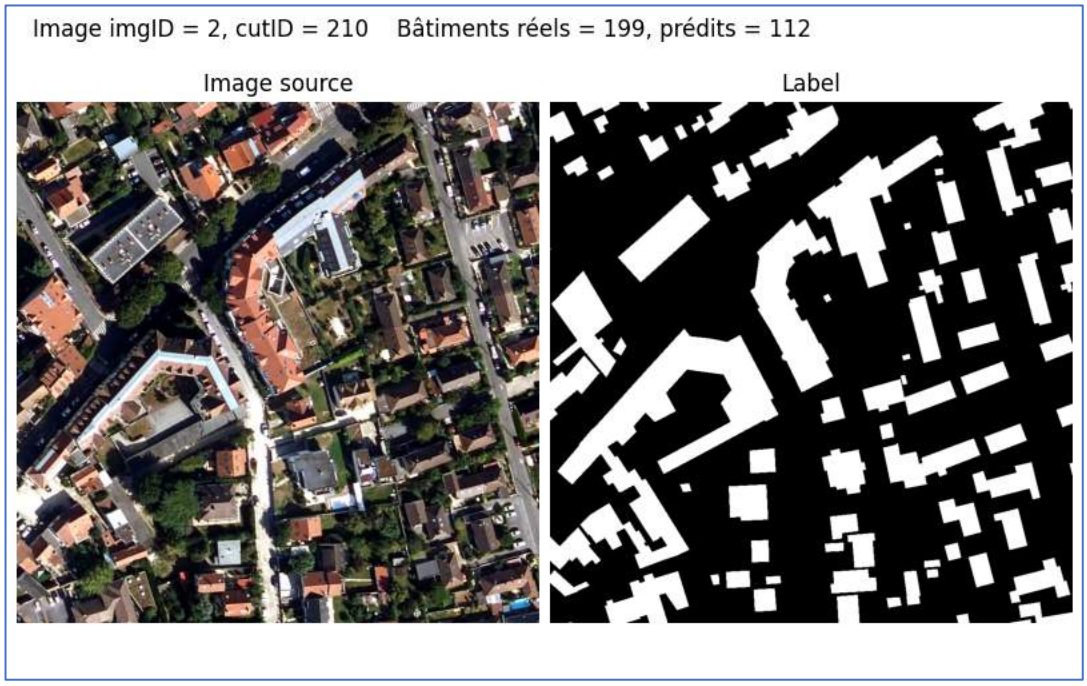
Remarque : nous avons également testé le modèle avec une taille d'image de 256x256 et des batches de 32, vs. (512x512 ; 16) ; les résultats ont été très similaires. Il pourrait être intéressant de tenter une combinaison (512x512 ; 32).

Quelques exemples de résultats sont présentés ci-dessous (l'image, son label et les performances associées).

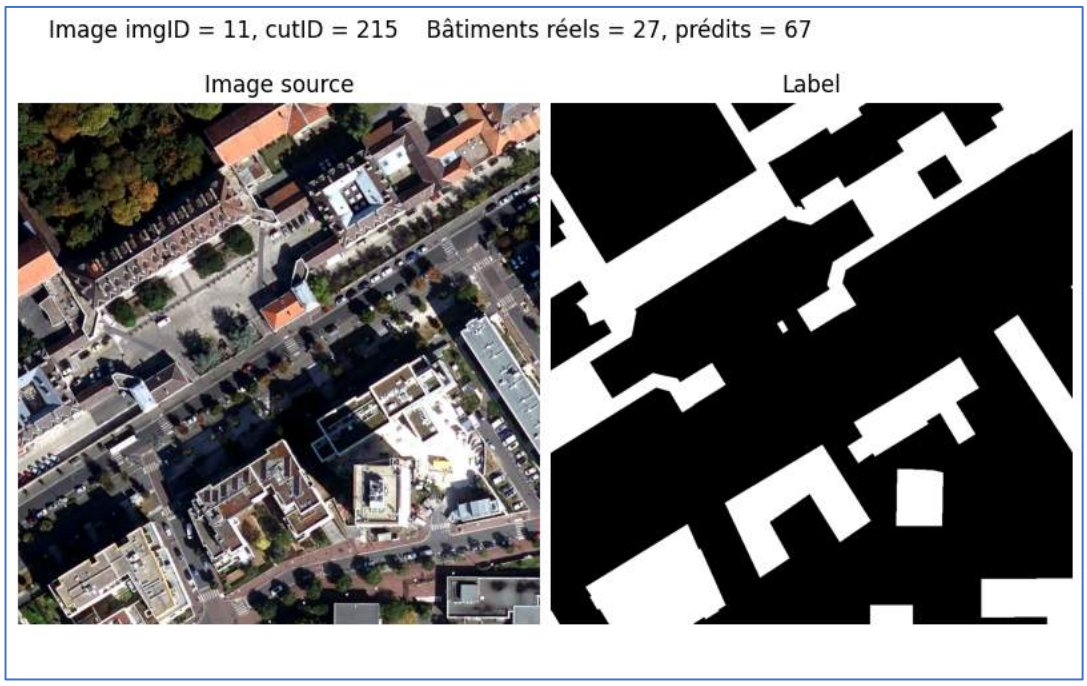
- Une détection correcte avec de nombreux bâtiments :

- Une détection correcte sans bâtiment :

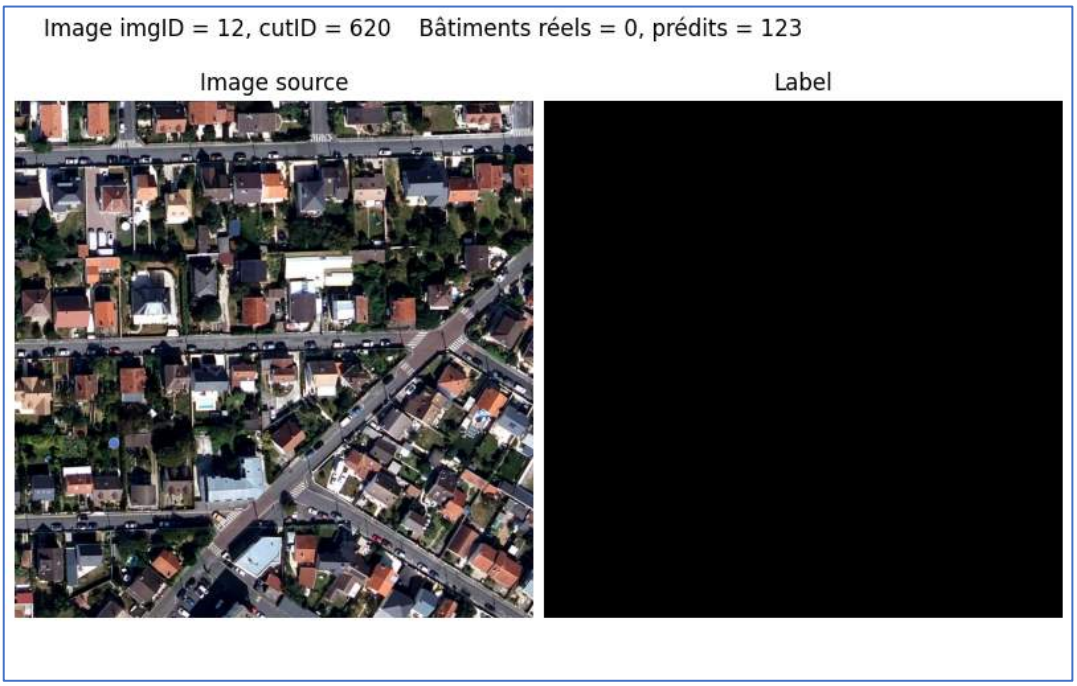
- Une mauvaise détection (pas assez de bâtiments détectés) :



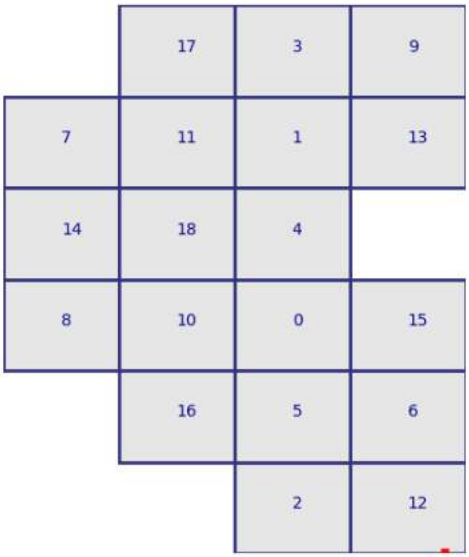
- Une mauvaise détection (trop de bâtiments détectés) :



- Une image dont le masque est vide, alors que l'image regorge de bâtiments :



Cette dernière situation arrive parfois aux abords de la carte, lorsque l'image déborde sur un département voisin ; en l'occurrence, on voit que la position (en rouge) de cet échantillon se situe à l'extrême sud parmi les images couvrant le département :



3.3.4 Base de données infrarouge

À titre de comparaison, nous avons effectué le même travail sur la base de données IRC, i.e. des images comportant les couches Infrarouge, Rouge et Vert au lieu des canaux RGB.

Nous obtenons des résultats similaires à celles de l'ensemble RGB, avec une MAE de 0,95, une moyenne de l'écart entre les données prédites et les données réelles de 0,7 et un écart-type de 15,5.

À titre d'exemple, une image issue de cette base, son label et les performances associées :

Bien que la couleur de la végétation se démarque nettement, on constate que cela ne facilite pas la détection des bâtiments.

3.3.5 Régression : Point d'étape

Le modèle de régression permet d'obtenir un bon ordre de grandeur du nombre de bâtiments dans une image, mais avec une précision limitée. Il semble difficile de l'utiliser pour détecter les constructions illégales, même dans un simple objectif de criblage des images à étudier.

3.4 Approche Régression via Transfer learning

3.4.1 Objectifs

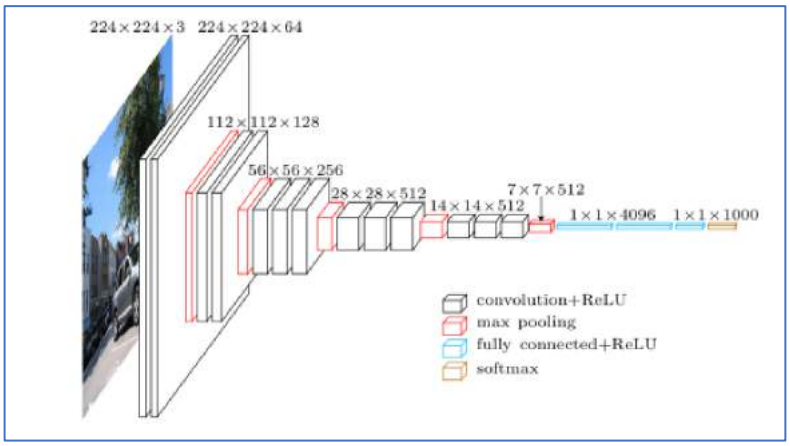
Nous avons également testé une approche par *transfer learning*, consistant à utiliser un modèle pré-entraîné sur un autre ensemble de données, puis à compléter son entraînement.

Il s'agissait de mesurer l'intérêt d'un modèle déjà entraîné par rapport à un modèle créé de toutes pièces, avec une initialisation des poids aléatoire.

Notre choix s'est porté sur un modèle VGG16, réputé pour sa précision sur des images de types très variés (base de données ImageNet).

3.4.2 Modélisation et apprentissage

Pour rappel, l'architecture VGG16 est la suivante :



source Karen Simonyan & Andrew Zisserman, cf. bibliographie

Nous en conservons les couches de convolutions et leurs coefficients, que nous complétons par :

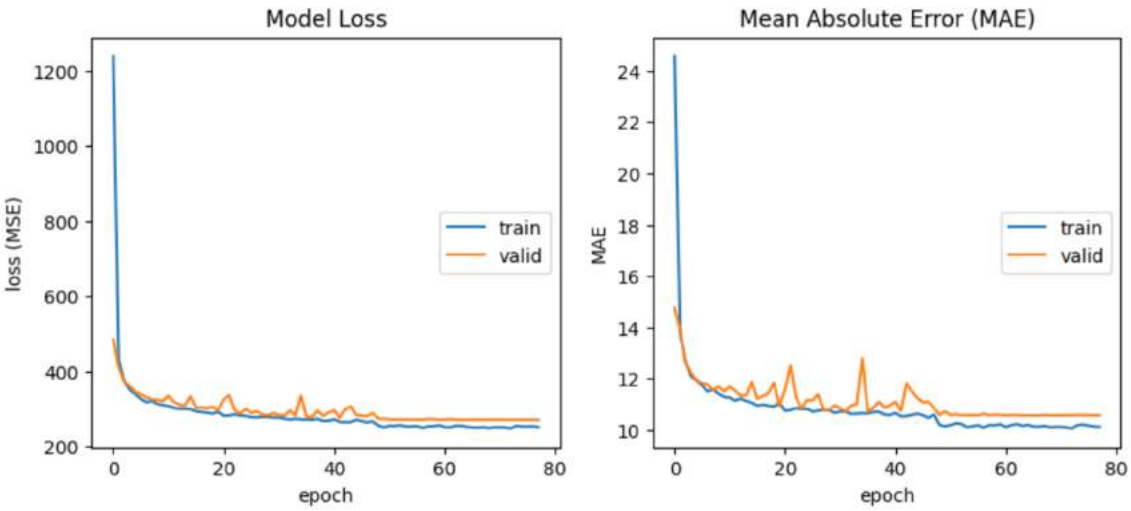
- 1 couche de Dropout à 50%.
- 1 couche de *pooling* moyen.
- 2 couches denses, avec 256 canaux et 1 canal respectivement.

Au total, le modèle comporte 14 714 688 paramètres.

Les autres caractéristiques restent identiques aux modèles de régression présentés ci-dessus : fonctions d'activation linéaires, fonction de perte MSE, optimiseur Adam et *callbacks*.

L'entraînement est effectué en deux temps :

1. Blocage des poids des couches du modèle VGG16, afin de profiter des connaissances de ce dernier pour ajuster les nouvelles couches denses (131 585 paramètres entraînables) :



2. Déblocage des 4 dernières couches du modèle VGG16, pour affiner les paramètres permettant la convergence vers notre *target* (les 14 714 688 paramètres sont désormais entraînables).

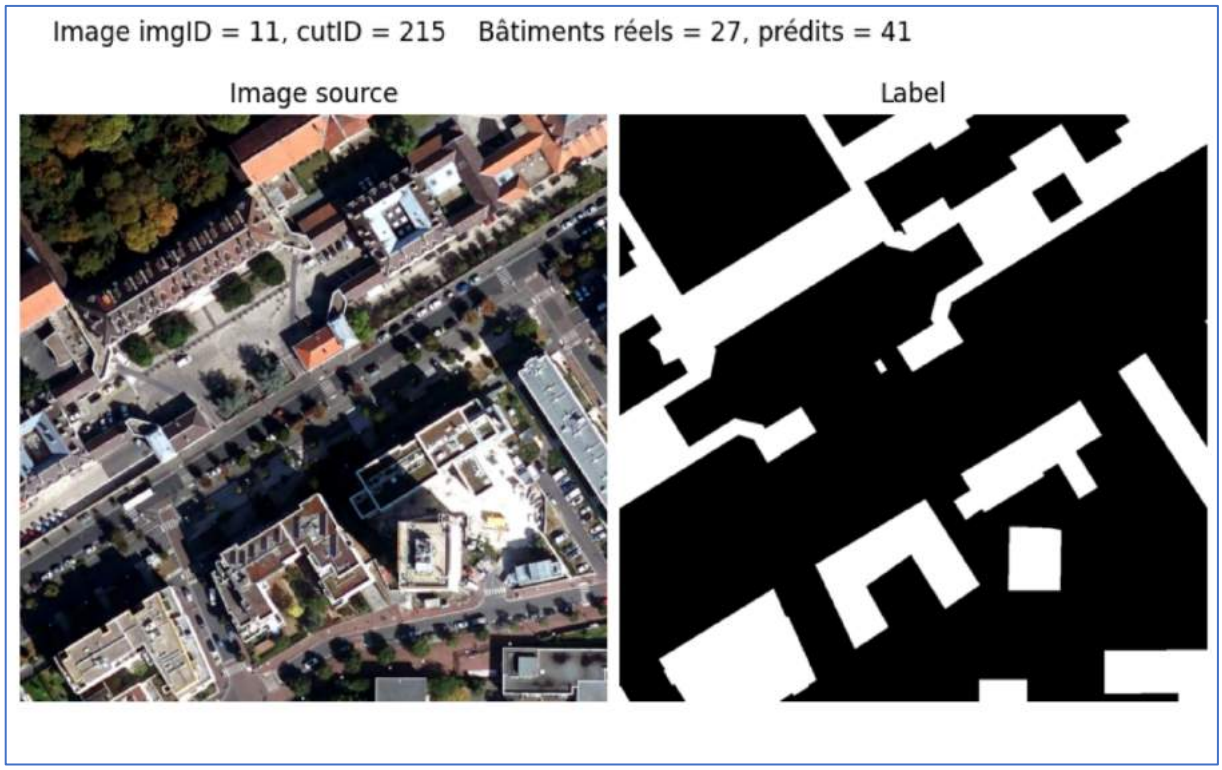
L'entraînement supplémentaire apporte une amélioration significative, dans la mesure où la MAE passe de 10,6 à 8,3.

3.4.3 Résultats

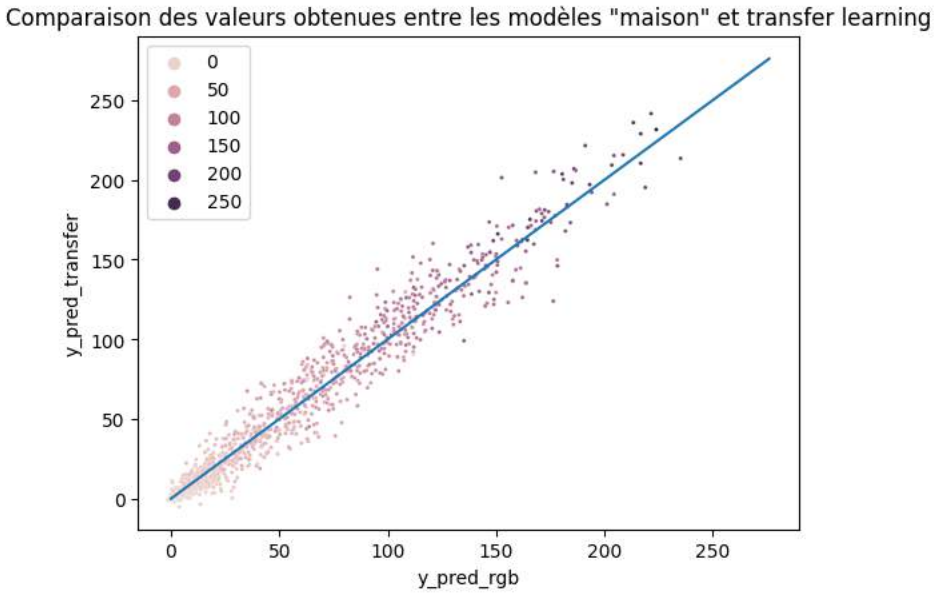
En définitive, les performances du modèle *transfer learning* sont supérieures à celles des modèles « maison », tant RGB qu'IRC :

Indicateur	Transfer learning	Modèle RGB	Modèle IRC
MAE	8,8	10,0	9,9
% nb bâtiments prédits exactement	18,9%	13,6%	13,4%
% nb bâtiments prédits à +/- 10	68,6%	63,6%	65,3%
Moyenne de l'écart (prédit – test)	-0,46	-0,6	0,7
Écart-type de l'écart (prédit – test)	14,6	15,4	15,5

À titre d'exemple, l'image suivante ne présente plus qu'un écart de 14 (vs. 40 avec le modèle « maison ») :



Il est à noter que les deux modèles donnent des résultats différents, comme le montre le faible nombre de points exactement alignés sur la courbe sur ce graphe (qui compare pour chaque image les valeurs prédites par chacun des deux modèles) :



Nous avons tenté d'utiliser la moyenne des deux prédictions pour chaque image. Cela permet d'affiner légèrement les prédictions, l'écart-type des écarts passant à 14.0 (vs. 14,6 et 15,4 pour les deux modèles de base).

3.4.4 Transfer learning : Point d'étape

Cette approche est intéressante dans la mesure où ses performances sont un peu meilleures que notre modèle créé *ex nihilo*. De plus, utiliser la moyenne des prédictions effectuées par chacun des deux modèles permet d'améliorer encore les résultats.

Toutefois, si la régression nous a permis de valider la performance de modèles à base de réseaux de neurones convolutifs, elle ne répond que partiellement au problème posé. C'est pourquoi nous étudions ensuite des modèles de classification portant sur les pixels de l'image.

3.5 Approche Segmentation sémantique

3.5.1 Objectif

L'utilisation d'un modèle de segmentation sémantique répond au besoin de connaître la nature de chaque pixel de l'image. En l'occurrence, c'est une segmentation binaire qui nous intéresse : l'appartenance à une construction ou pas.

Nous utilisons donc un modèle à même de prédire un masque pour chaque image, que nous pourrons comparer au masque réel issu de la base du cadastre.

3.5.2 Modélisation et apprentissage

Nous nous appuyons sur une architecture de type UNET, réputée son efficacité en matière de reconnaissance d'image :

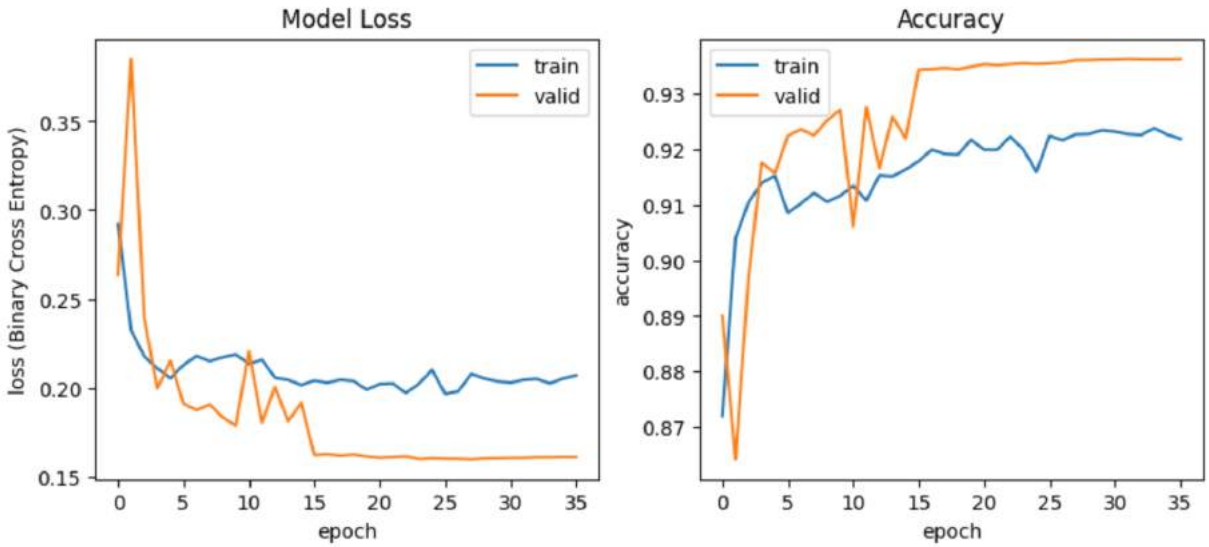
- Encodage : des couches de convolution extraient les caractéristiques de l'image en les concentrant sur une petite matrice composée de nombreux canaux : nous choisissons 6 blocs de convolution avec *MaxPooling* aboutissant à 512 canaux et une taille d'image 8x8.
- Décodage : reconstruction d'une image finale de même taille que l'image d'entrée, grâce à des opérations d'*UpSampling* insérées au sein de couches de convolution, tandis que les matrices sont concaténées avec des matrices antérieures de manière à faire circuler l'information au sein du réseau : nous avons également 6 couches de ce type.

Nous avons choisi de normaliser chaque sortie de couche par *BatchNormalization*, ce qui semble d'autant plus utile que certaines couches sont concaténées entre elles : il est important d'avoir des valeurs du même ordre de grandeur.

La fonction d'activation est *ReLU*, sauf pour la couche de sortie, où la fonction *sigmoïd* est mieux adaptée à une classification binaire, puisque nous avons une seule classe de sortie.

Le modèle comporte 6 738 489, dont 6 735 737 entraînables.

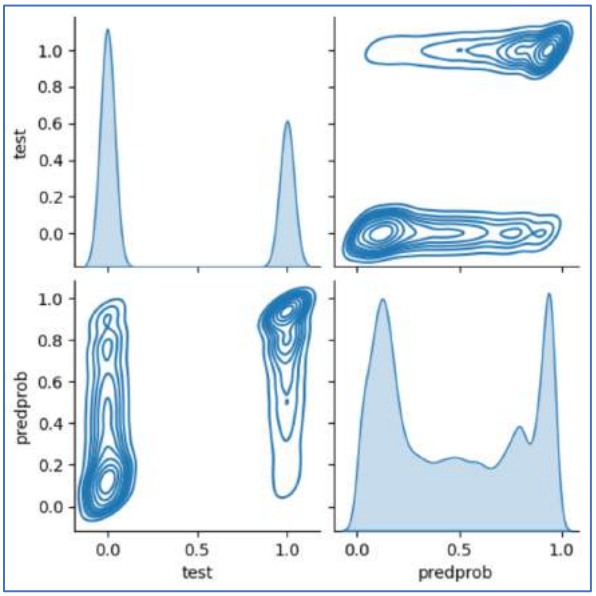
L'entraînement se stabilise très rapidement, après une quinzaine d'epochs. Après entraînement, nous aboutissons à une précision de 93,6% :



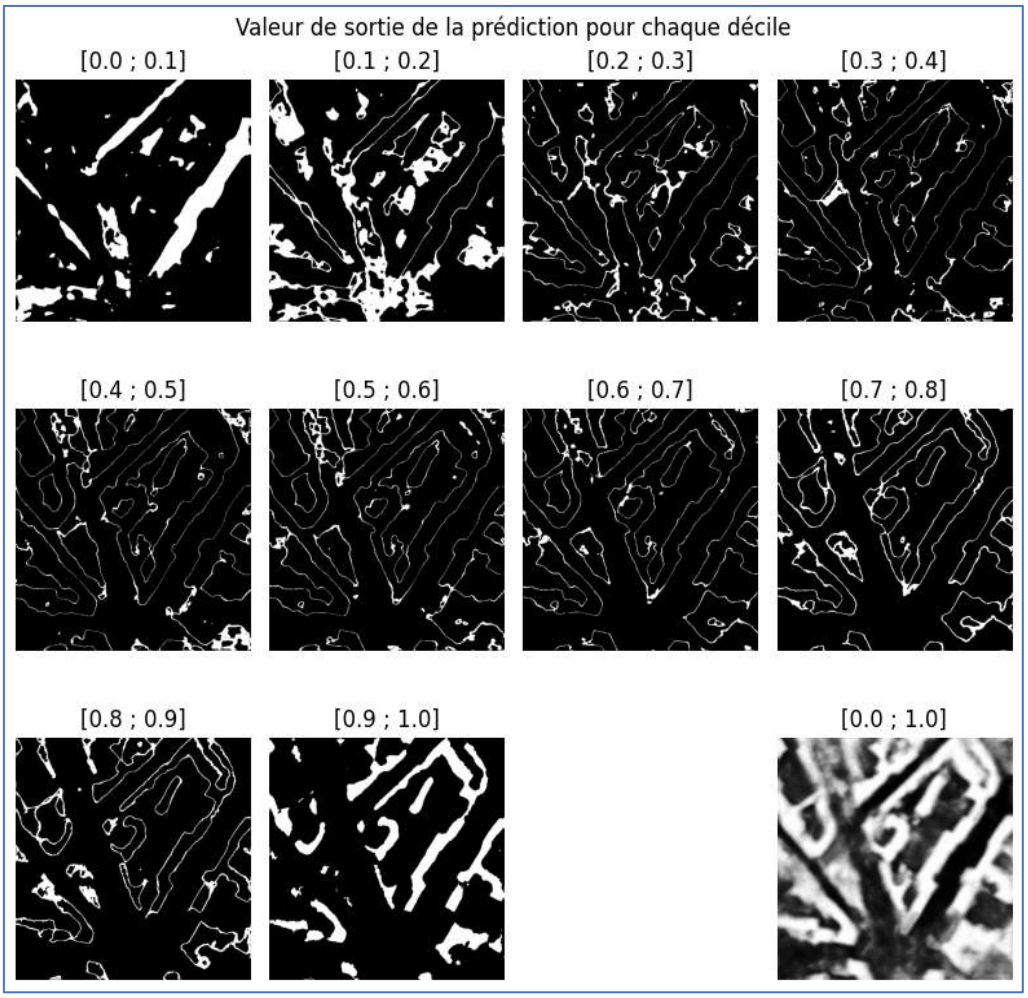
On constate que le comportement est légèrement meilleur sur l'ensemble de validation, ce qui est un signe encourageant quant à la capacité de généralisation du modèle.

3.5.3 Choix du seuil de classification et scores globaux

En mettant en regard, pour chaque pixel d'une image d'exemple, la prédiction et le label, nous constatons que la sortie du réseau est continue (ce qui était attendu avec la fonction sigmoïde) :

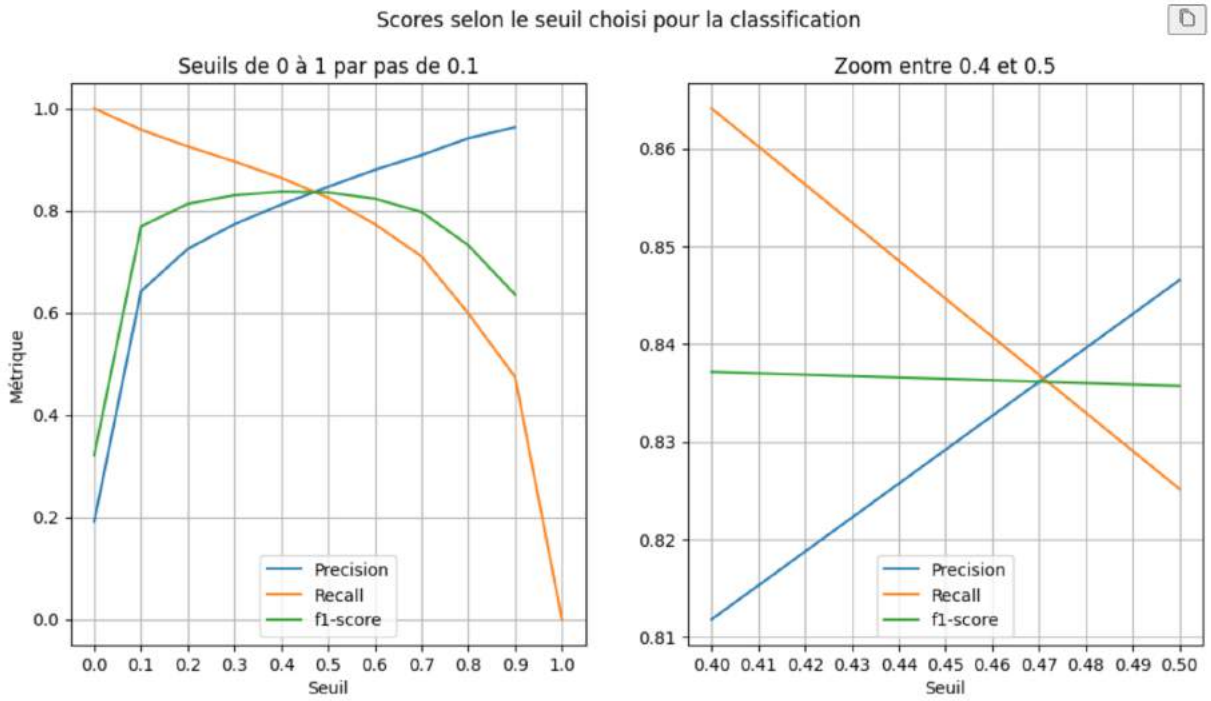


Cela s’illustre en affichant les valeurs appartenant à chaque décile (en blanc sur chaque image) :



Le nombre d’observations en milieu d’intervalle étant assez élevé, la détermination du seuil à utiliser pour déterminer la classe revêt une grande importance.

Pour ce faire, nous établissons, pour chaque décile, 4 cartes des pixels de l'ensemble de test : vrais positifs (TP), vrais négatifs (TN), faux positifs (FP) et faux négatifs (FN). Nous calculons ensuite les scores correspondants :



Nous choisissons le seuil de 0.47 pour séparer les classes, et recalculons ainsi les prédictions pour tout l'ensemble de test :

- valeur de sortie du réseau ≤ 0.47 → prédiction 0 = Non bâtiment
- valeur de sortie du réseau > 0.47 → prédiction 1 = Bâtiment

Ce seuil est une base qui pourra être affinée en fonction des objectifs, selon que nous souhaiterons détecter le maximum de constructions illégales ou, à l'inverse, ne détecter qu'un petit nombre de cas plus à risque.

Avec ce seuil, la matrice de confusion sur l'ensemble des pixels de l'ensemble de test est la suivante :

		Prédiction	
		Positifs	**Négatifs**
Réalité	**Positifs**	16,1 %	3,1 %
	Négatifs	3,1 %	77,7 %

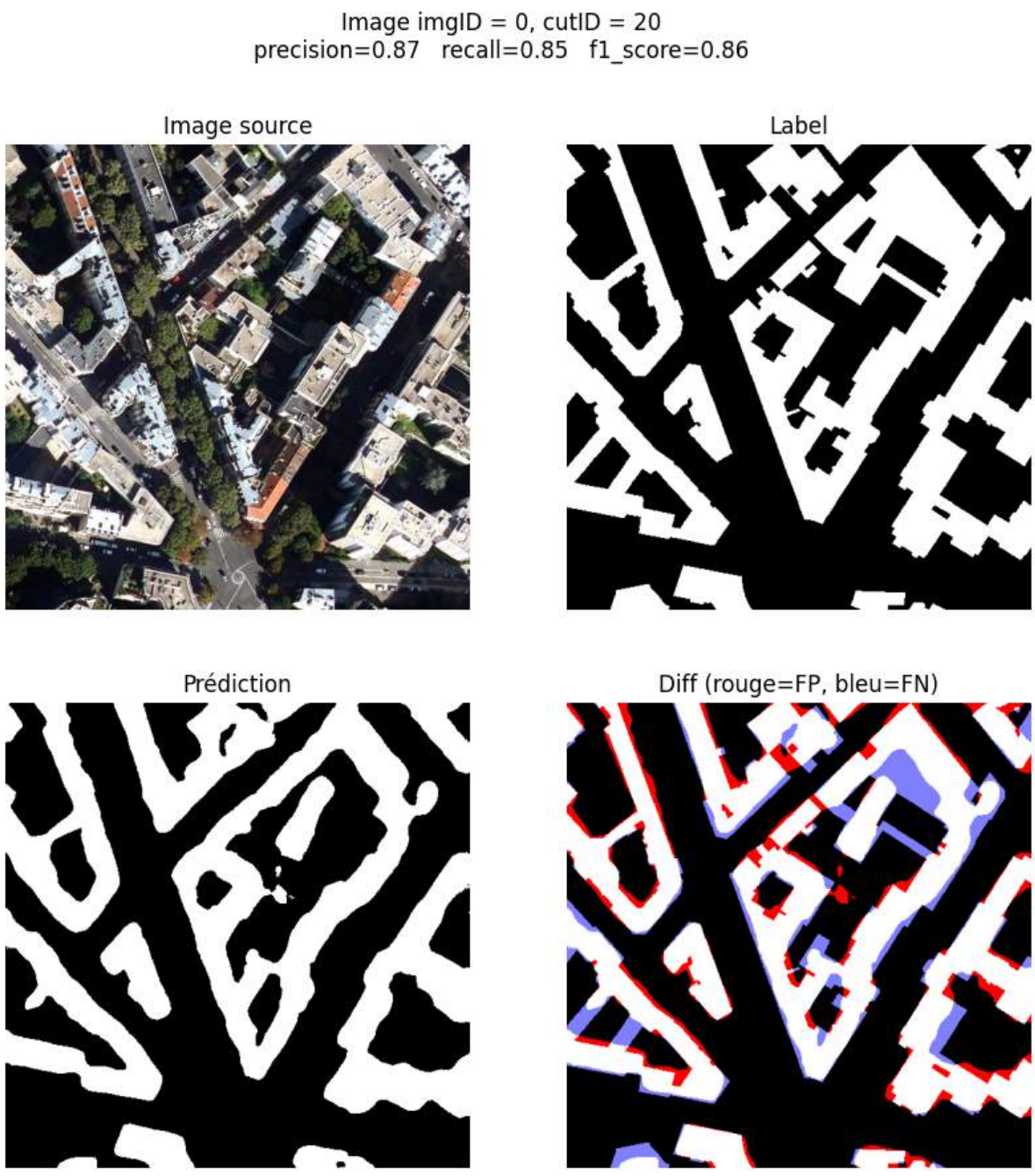
Les scores sont de :

	Classe réelle	
	Positifs	**Négatifs**
Précision	0,84	0,96
Rappel	0,84	0,96
f1-score	0,84	0,96

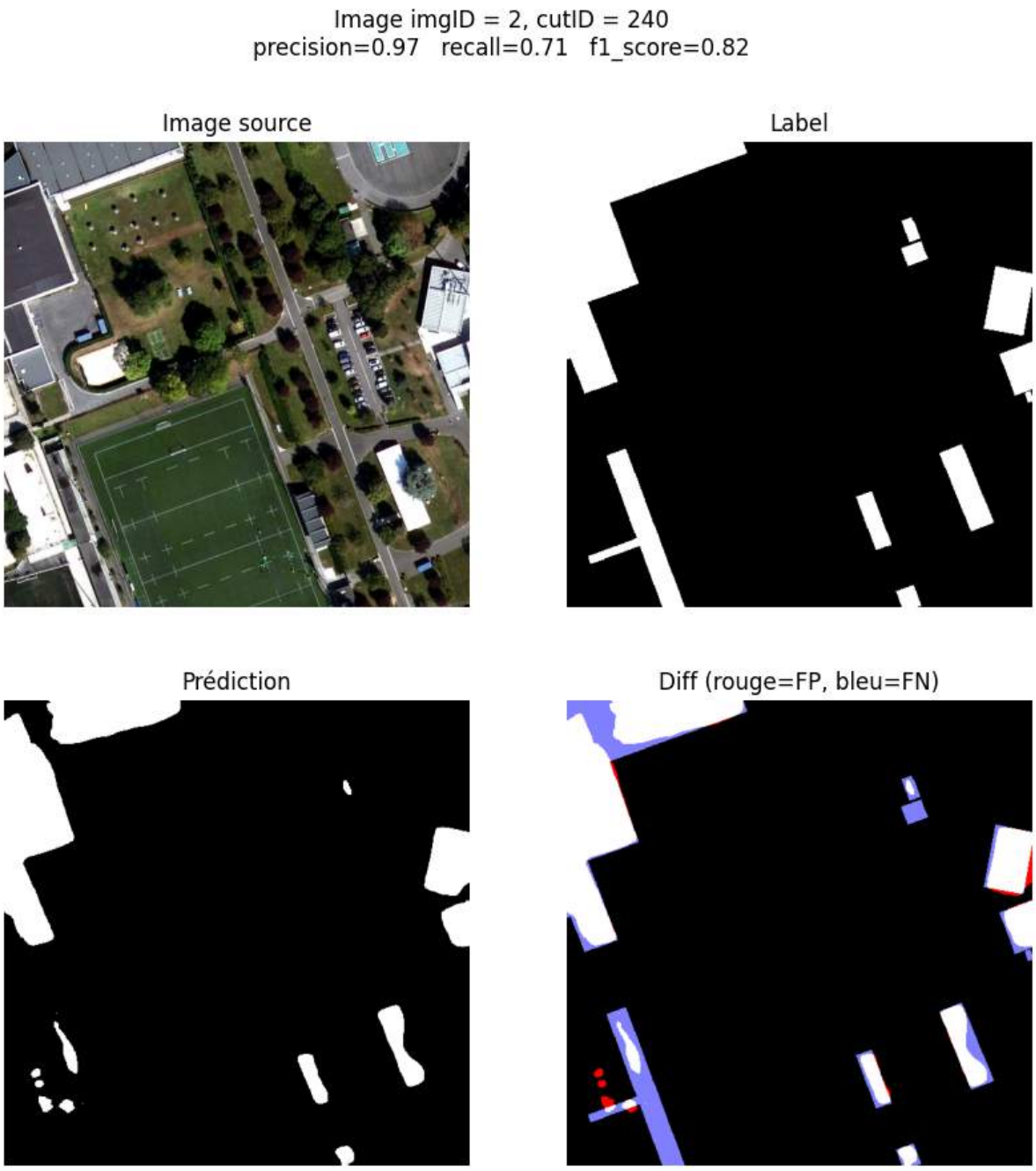
3.5.4 Analyse de quelques images

Nous présentons ici quelques images de l'ensemble de test, avec la photographie source, le masque (label), la prédiction et, enfin, la différence entre la prédiction est le masque, les faux positifs (FP) apparaissant en rouge et les faux négatifs (FN) en bleu.

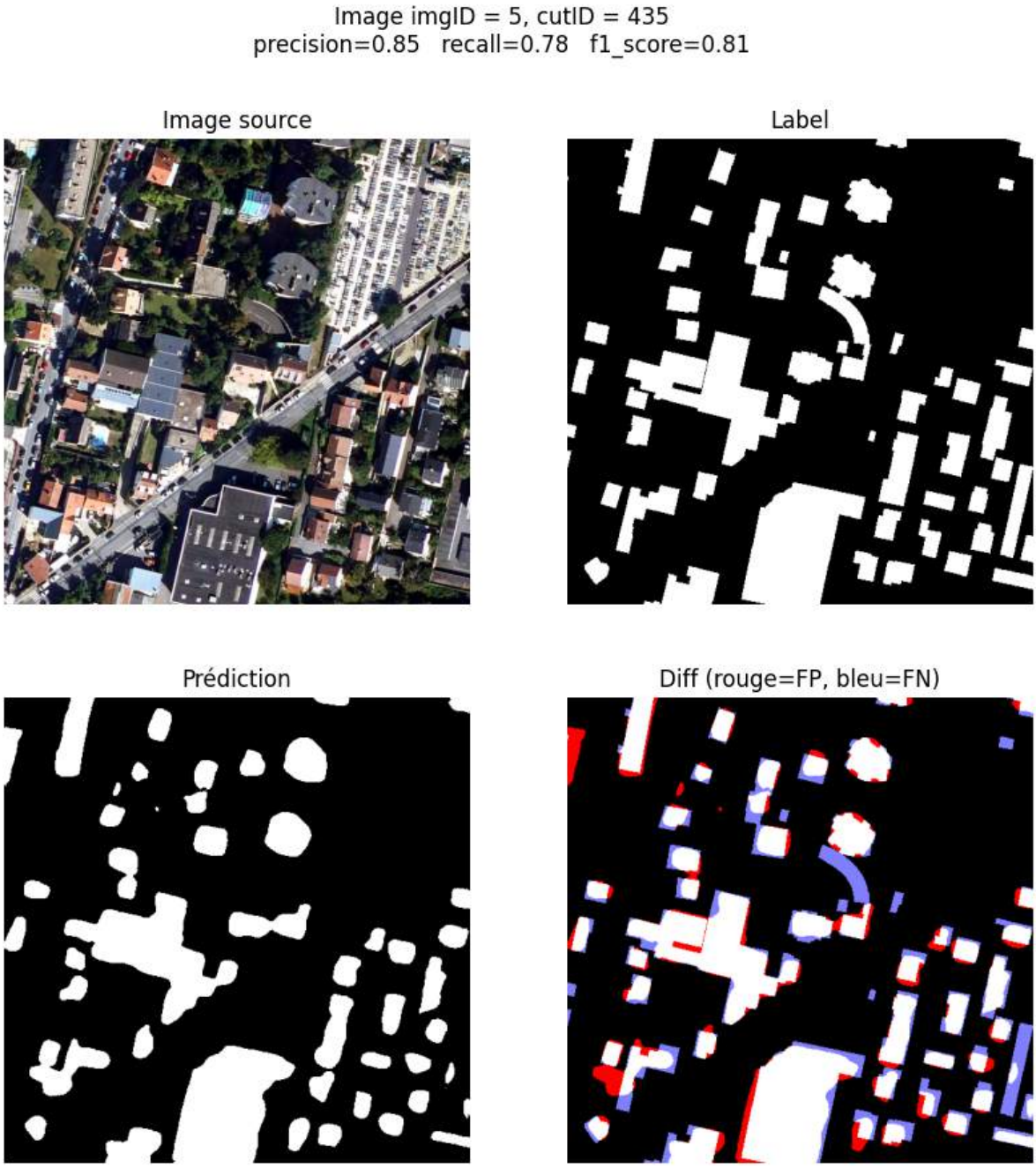
- La plupart des images font apparaître un bon niveau de détection globale, avec des bordures FP et parfois FN, ainsi que certains blocs globalement mal identifiés :



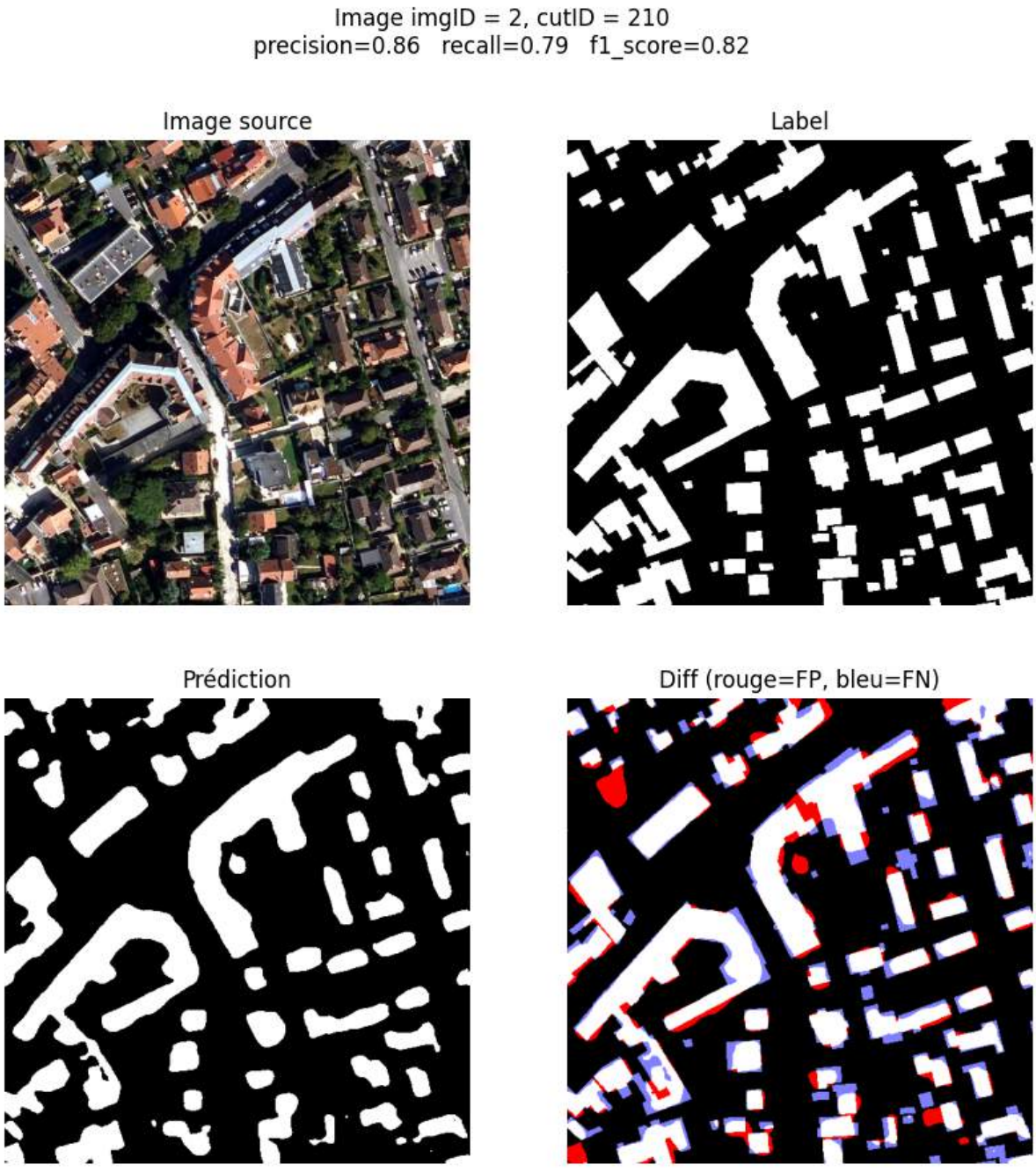
- Certains FN peuvent s'expliquer lorsque les bâtiments ressemblent à d'autres éléments (arbres en partie haute à droite de l'image précédente, ou route en bas à gauche de l'image suivante) :



- De même, certains FP peuvent s'expliquer par des constructions ressemblant à des bâtiments, mais n'en étant pas véritablement, comme cela semble être le cas en bas à gauche de l'image suivante :



- À l’inverse, l’image suivante présente plusieurs zones FP qui pourraient effectivement correspondre à des constructions illégales :



Intéressons-nous à la zone FP en haut à gauche de cette dernière mage. Elle semble effectivement correspondre à un bâtiment non répertorié :

Ce bâtiment fera l'objet d'une étude spécifique en partie 4.3.

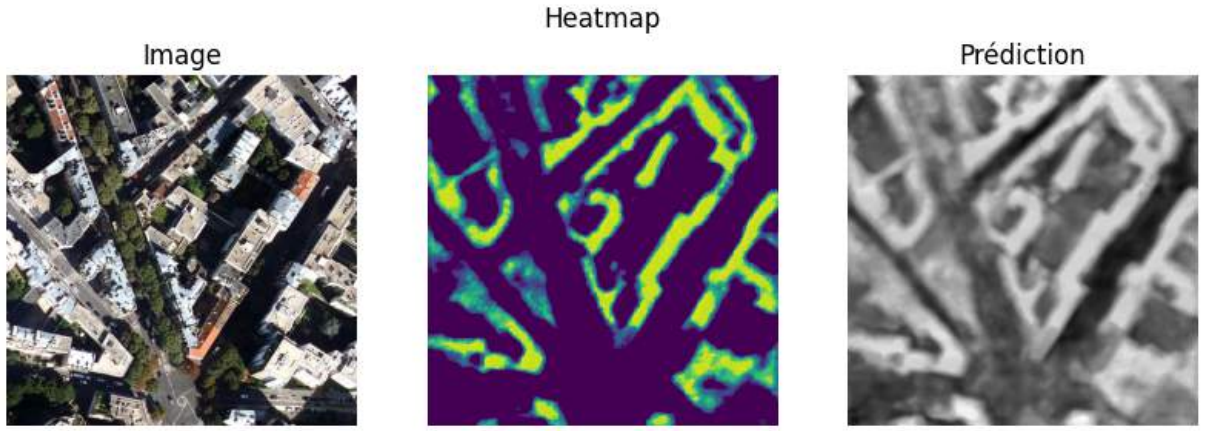
3.5.5 Interprétabilité du modèle

Nous cherchons ici à comprendre la manière dont le réseau de neurones parvient à identifier le nombre de bâtiments.

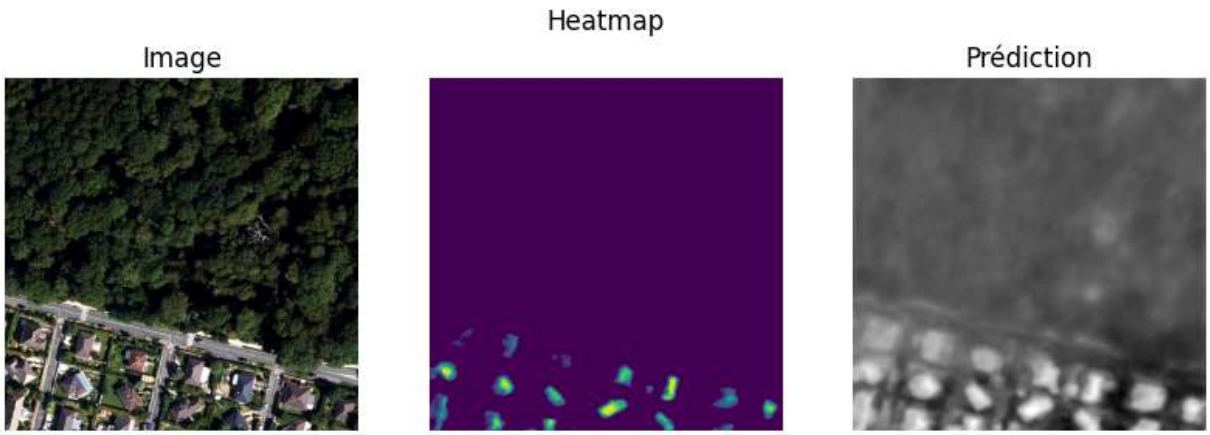
On utilise ici la technique du *Grad-CAM*, qui permet, pour une image donnée de visualiser les zones de l'image ayant la plus grande influence sur la sortie du modèle. Plus précisément, on s'intéresse à la dernière couche de convolution, en calculant :

$$\text{heatmap} = (\text{sortie dernière convolution}) \times \frac{\partial\,(\text{prédiction})}{\partial\,(\text{sortie dernière convolution})}$$

Comme on le voit sur les images suivantes, la *heatmap* préfigure largement les prédictions, ce qui est cohérent puisqu'elle concerne l'avant-dernière couche du modèle (la prédiction est ici affichée sans la dernière couche d'activation et sans seuillage) :



vs



3.5.6 Segmentation sémantique : Point d'étape

Le modèle UNET proposé distingue relativement bien les bâtiments. Pour obtenir ce résultat, nous avons dû augmenter la profondeur du modèle (en ajoutant une couche d'encodage et une couche de décodage) et passer à une taille d'image de 512x512 vs. 256x256 (un premier essai ayant été relativement décevant).

Il a permis d'enrichir la base de données avec, pour chaque image, les indicateurs TP, TN, FP, FN et les scores usuels : cela permet de diminuer considérablement le nombre d'images à analyser par un opérateur humain pour identifier les constructions illégales.

Toutefois, les modèle présente plusieurs inconvénients :

- Il renvoie des masques et non des polygones, ce qui rend difficile d'individualiser, sur une image, chaque faux positif, et donc d'en calculer la surface pour le classer en construction illégale, erreur de précision du modèle ou véritable faux positif.
- Il manque encore de précision aux limites des bâtiments, avec des faux positifs qui conduisent à surestimer l'importance des constructions illégales.
- Enfin, certains faux négatifs apparaissent çà et là sur les images. Cet inconvénient est mineur, car sans influence sur l'objectif de notre étude.

Une approche par segmentation d'instance peut nous permettre de résoudre certains de ces problèmes.

3.6 Approche Segmentation d'instances

3.6.1 Objectif

Les modèles de segmentation d'instances ont pour objectif d'individualiser les objets, contrairement aux modèles présentés précédemment, qui s'intéressent à la classification pixel par pixel. Dans le cadre de notre étude, cette approche devrait permettre de détecter véritablement des bâtiments et de les traiter comme tels, en calculant par exemple leur surface, élément crucial pour mettre en évidence une surface illégale.

3.6.2 Modélisation et apprentissage

Nous utilisons le modèle YOLOv8-seg présenté par Ultralytics. Il s'agit de l'un des modèles actuellement les plus performants pour la segmentation d'instances.

Ce modèle travaille non pas avec des labels constitués de masques (matrices de pixels), mais avec des polygones ; il restitue également des polygones en sortie. C'est pourquoi, préalablement à l'entraînement, une étape de préparation des données est nécessaire, consistant à préparer un fichier texte par image, contenant l'ensemble des polygones normalisés et leur classe associée (en l'occurrence une seule classe, pour les bâtiments).

Le modèle le plus élaboré que nous ayons utilisé possède la structure suivante :

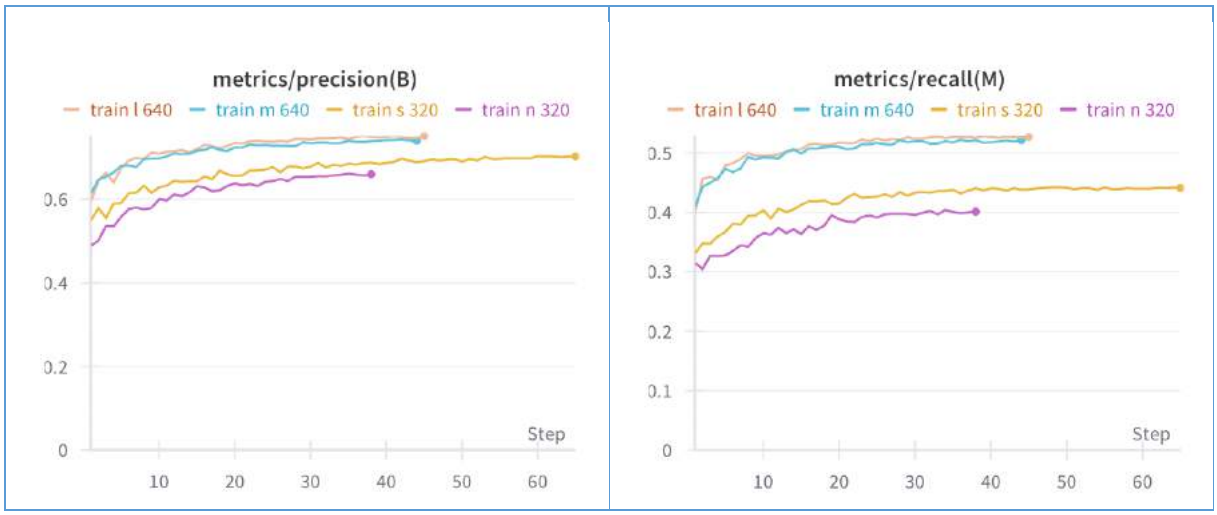
- Un module *backbone* constitué d'un enchaînement de couches de convolution et de couches C2f (*coarse to fine*). Ces dernières remplacent les couches de *max pooling* qui détruisent l'information locale.
- Un module *head* constitué d'un enchaînement de couches d'*upsampling*, concaténation, C2f et convolution. En sortie, une couche *Segment*, qui est la particularité du modèle de segmentation de YOLO (d'autres modèles YOLOv8 existant, par exemple pour la détection).

Précisons qu'il s'agit d'un modèle pré-entraîné ; nous sommes donc dans un contexte de *transfer learning*.

YOLOv8-seg existe en 5 tailles de modèles (n, s, m, l et xl) que nous avons toutes testées, hormis la plus grande. Le modèle « l » comporte 45 937 590 paramètres, dont 45 937 574 entraînables.

Nous avons utilisé les modèles n et s avec des images de 320x320, puis les modèles m et l avec des images de 640x640.

Les résultats sont les suivants :



L'entraînement du modèle « l » a nécessité 11 heures pour 46 *epochs*. À ce stade, il semble qu'il soit stabilisé et possède la meilleure performance : nous le retenons donc pour la résolution de notre problème.

3.6.3 Scores globaux

L'objectif est ici de comparer les résultats obtenus avec ceux du modèle UNET précédent.

Pour cela, nous créons des masques sur l'ensemble de test pour les prédictions et les comparons aux masques réels, de manière à en déterminer la matrice de confusion et les scores :

		Prédiction	
		Positifs	**Négatifs**
Réalité	**Positifs**	18,3 %	3,4 %
	Négatifs	2,5 %	75,8 %

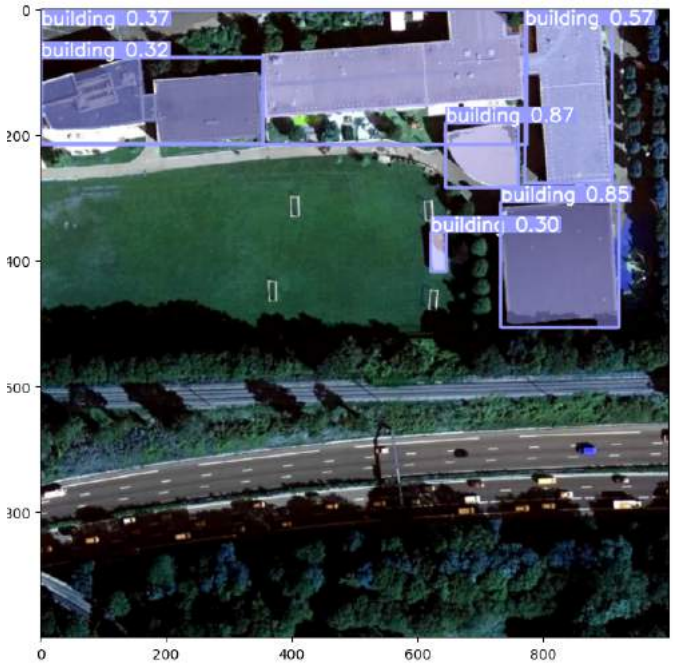
	Classe réelle	
	Positifs	**Négatifs**
Précision	0,88	0,96
Rappel	0,84	0,97
f1-score	0,86	0,96

La précision est légèrement meilleure qu'avec le modèle UNET (0,86 vs. 0,84 sur les bâtiments). Mais surtout, nous disposons de polygones délimités, que nous pourrons traiter individuellement, au lieu d'amas de pixels pouvant couvrir de nombreux bâtiments.

3.6.4 Visualisation des résultats

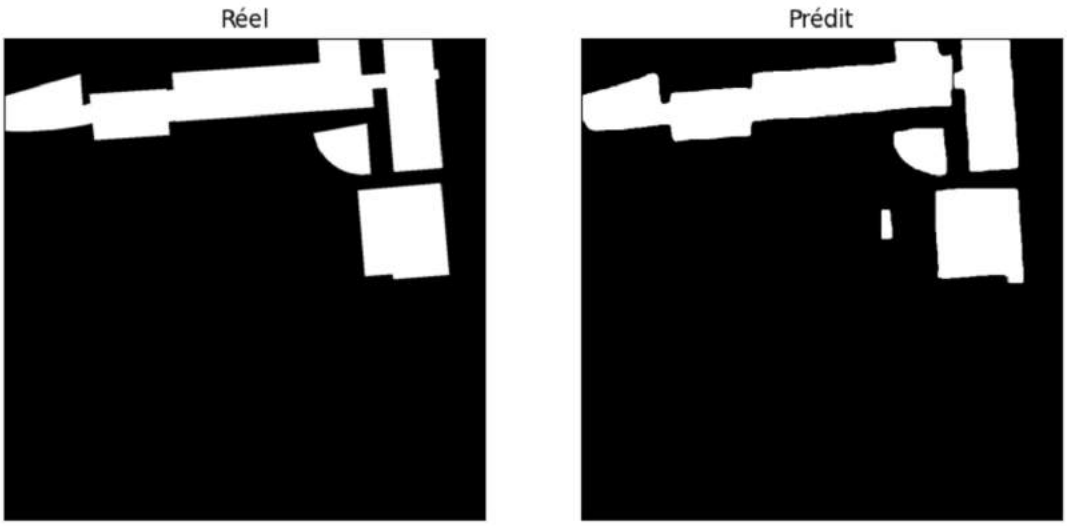
La segmentation d'instances retournant des polygones, des visualisations plus variées sont possibles, en particulier en détourant les polygones, par exemple :

- Visualisation standard de YOLOv8 (avec les probabilités d'appartenance à la classe) :



- Visualisation de l'image source et des polygones (labels en vert pointillé et prédictions en rose) :

- Sous forme de masque, comme pour la segmentation sémantique :



- En mettant en évidence les contours des faux positifs d'une part et leur surface d'autre part :

3.6.5 Segmentation d'instances : Point d'étape

La segmentation d'instances s'est avérée très intéressante. Au-delà de la légère augmentation de précision par rapport au modèle UNET, elle offre beaucoup plus de souplesse grâce aux résultats fournis sous forme de polygones, qui peuvent être mis en rapport avec les labels initiaux grâce aux opérations sur les *GeoSeries*.

Dans la partie suivante, nous utiliserons donc les résultats du modèle YOLOv8 pour détecter les constructions illégales.

4 Interprétation : Détection des constructions illégales

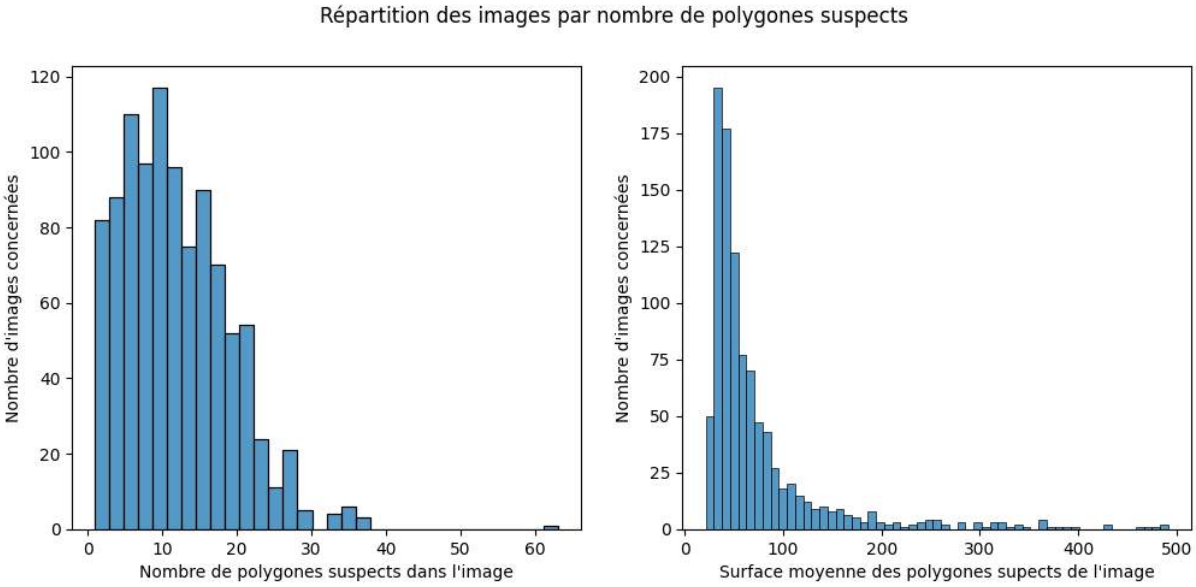
4.1 Criblage en masse

Rappelons que la base complète comporte, pour les Hauts-de-Seine, 11 875 images, chacune représentant un carré de 200 mètres de côté, pour un total d'environ 694 000 polygones correspondant à des bâtiments. Cette volumétrie est incompatible avec un examen visuel exhaustif ; c'est pourquoi, dans un premier temps, nous souhaitons utiliser nos modèles pour cibler les images les plus susceptibles de contenir des bâtiments illégaux.

Nous nous concentrons, pour la démonstration, sur la base de test, qui comporte 1188 images, le même traitement pouvant être répliqué pour l'ensemble de la base.

Les bâtiments peuvent être considérés comme illégaux à partir de 20 m^2 ; nous excluons donc les polygones plus petits. Pour cela, nous calculons les faux positifs grâce aux opérations sur les *GeoSeries* elles-mêmes (union des labels d'une part, des prédictions d'autre part, puis différence entre les deux).

Il reste alors 1006 images réparties ainsi :



Ces distributions montrent qu'un grand nombre d'images possède de nombreux polygones et des surfaces moyennes importantes. Cette répartition ne fait pas apparaître un critère évident pour cibler les images à examiner en priorité.

En fin de compte, cette étape nous a permis d'éliminer 182 images, soit 15% de la base.

4.2 Sélection par examen visuel

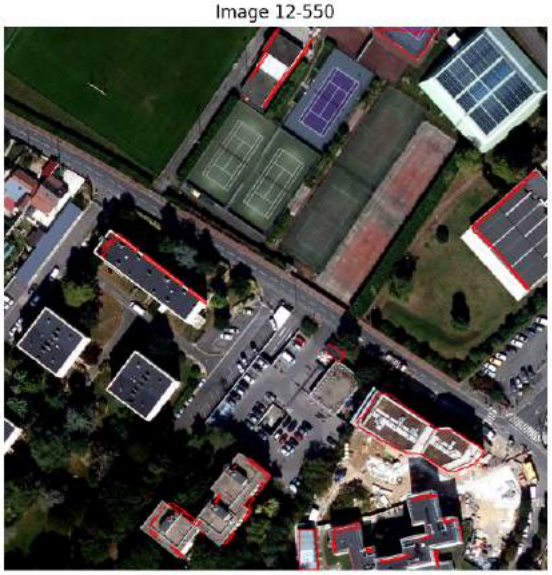
La sélection des images à approfondir devra être faite en les examinant une à une. Compte tenu de la volumétrie restante, la visualisation doit permettre, d'un coup d'œil, de retenir ou rejeter une image. Nous choisissons donc la visualisation qui nous paraît la plus pertinente pour remplir cet objectif.

Quelques exemples :

Image 05-495

Surfaces Faux Positifs

Image 12-550



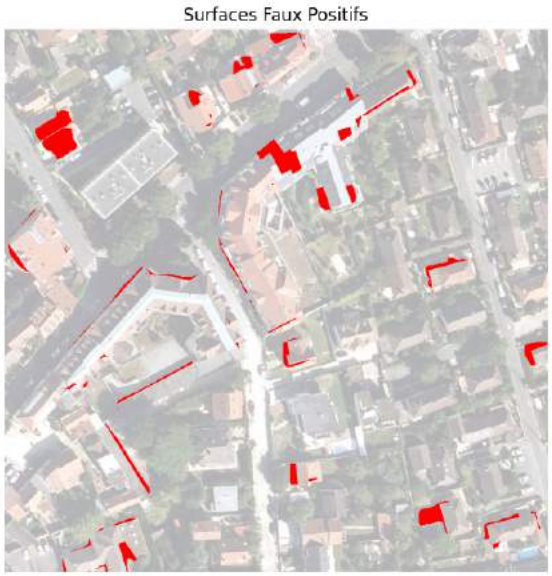
Surfaces Faux Positifs

Image 06-337

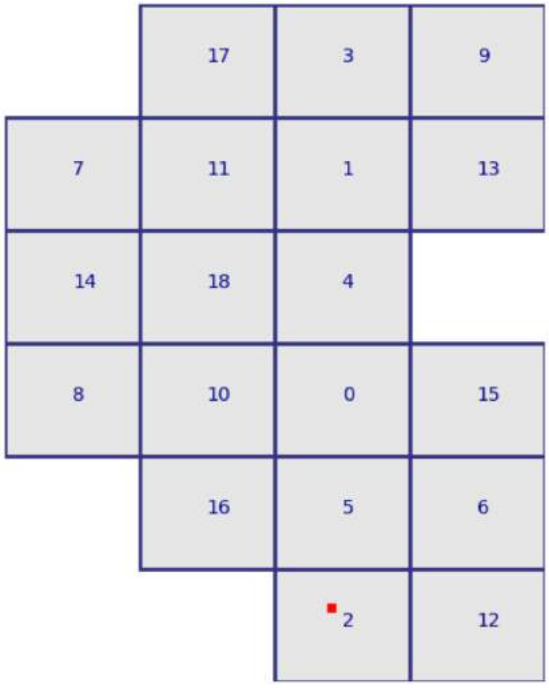
Surfaces Faux Positifs

4.3 Étude d'un cas suspect

Une fois les images à approfondir choisies, chaque polygone suspect devra être étudié. Nous reprenons ici un exemple vu lors de la segmentation sémantique :



Cette image se situe au sud du département :



L'image peut être localisée en transformant les coordonnées Lambert 93 du centroïde (647 100, 68 48 300) en données GPS (2,2807345E 48,7326558N) et en la recherchant sur un outil de cartographie grand public en ligne. Nous sommes dans une zone résidentielle de la ville de Massy.

Une photographie du lieu est accessible :

Source : Apple Plans

Il y a donc bien une construction qui n'est pas présente dans notre *dataframe* des bâtiments.

Plusieurs raisons sont envisageables :

- La discordance entre les bases BDORTHO et BDTOPO est à écarter, car BDTOPO est postérieure (juin 2023) BDORTHO (été 2021). D'ailleurs, la construction est ancienne, comme en témoigne l'historique de Google Maps, qui remonte jusqu'à 2008 :

Source : Google Maps

- Une erreur dans la constitution de la base BDTOPO.
- Une perte de polygone lors des manipulations pour constituer la liste des bâtiments ; par exemple, certains multi-polygones ont dû être simplifiés lors de la préparation des données.
- Une construction effectivement illégale.

Compte tenu de l'ampleur de la construction et de sa localisation dans un endroit dense, nous pensons que cette dernière piste est peu plausible.

Si l'on souhaitait approfondir le cas de cette construction, nous préconiserions de :

- Repartir de la base BDTOPO initiale et faire la jointure spatiale avec la construction, pour s'assurer qu'elle est bien absente.
- Vérifier auprès du cadastre l'existence de cette construction.

5 Bilan et perspectives

5.1 Interprétation des résultats

Notre approche présente une particularité importante par rapport à d'autres projets de *deep learning*. En effet, les labels ne correspondent pas à la caractéristique recherchée, puisque nous visons à détecter des bâtiments manquants, et non les bâtiments qui sont labellisés et font l'objet de l'entraînement du modèle. Cela influe sur la manière d'interpréter les résultats : les faux positifs du modèle peuvent correspondre en réalité à des faux positifs effectifs, mais aussi à des bâtiments illégaux. L'objet de notre recherche est donc inclus dans l'ensemble des faux positifs.

Les modèles en eux-mêmes présentent des performances très honorables, que ce soit UNET comme YOLOv8. Pour y parvenir, nous avons procédé à plusieurs améliorations au cours du projet :

- L'amélioration la plus importante a consisté à passer la taille des images en entrée de 256 à 512 pixels de côté.
- Nous avons également augmenté la profondeur du modèle UNET initial et prolongé les entraînements.
- Pour le modèle YOLOv8, nous avons utilisé un modèle avec plus de paramètres.

In fine, les résultats obtenus permettent d'écarter 15% d'images exemptes de tout risque de construction illégale, et d'effectuer un contrôle visuel très rapide sur le reste des images grâce à la visualisation proposée. En cela, nous rendons possible une opération qui ne l'aurait pas été par un opérateur humain ne disposant que d'une image et de la liste des bâtiments. Nous pouvons donc considérer que l'objectif initial est en grande partie rempli.

Le modèle pourrait aisément s'inscrire dans le processus métier suivant :

- Choix d'un territoire à étudier et chargement des données en source ouverte (BDTOPO et BDORTHO).
- Utilisation du flux de préparation des données proposé (découpage des images, création des fichiers labels).
- Si le territoire concerné est très différent des Hauts-de-Seine d'un point de vue topographique, entraînement complémentaire du modèle sur le nouveau jeu de données.
- Utilisation du modèle en mode prédiction sur l'ensemble du *dataset*.
- En sortie du modèle, filtrage des polygones « faux positifs » de plus de 20 m^2 et des images ne comportant pas de polygone de surface supérieure.
- Examen visuel des images pour ne conserver que celles effectivement suspectes.
- Étude approfondie de chaque image conservée, en lien avec le cadastre et les fonctionnalités de *street view* disponibles sur Internet.

Ce processus métier nécessiterait au préalable d'industrialiser les *notebooks*.

5.2 Difficultés rencontrées

Le projet s'est globalement bien passé, sans difficulté bloquante. Nous pouvons néanmoins noter les points suivants :

- Le projet a nécessité de prendre connaissance des domaines qui n'avaient pas été encore abordés lors de la formation (réseaux de neurones convolutifs, bibliothèque *cv2*) ou n'étaient pas prévus à l'origine dans notre cursus (traitement et analyse d'images, y compris la bibliothèque *rasterio*).
- De même, nous avons dû prendre connaissance des bibliothèques *Geopandas* et *shapely* pour le traitement des polygones.
- Nous avons dû tester plusieurs architectures UNET avant d'obtenir un résultat satisfaisant, et surtout augmenter la taille des images.
- Les temps d'apprentissage ont été assez longs, au moins une dizaine d'heures pour la plupart.
- YOLOv8 a nécessité, quelle que soit la taille du modèle choisi, de passer par une plate-forme en ligne proposant un GPU puissant, en l'occurrence la plate-forme *Kaggle*, dont le maniement s'est avéré peu aisé.
- Les premiers entraînements sur YOLOv8 n'ont pas été concluants, du fait d'un calcul des polygones cible erroné (l'origine se trouvant en haut à gauche de l'image, les coordonnées Y devaient être inversées).
- Il s'est confirmé que le charge de travail était importante tout au long du projet, d'autant que nous n'avons été que deux participants durant l'essentiel de l'année, alors que le projet était prévu pour trois.

L'existence d'un *starter kit* en début de projet a été une aide indispensable. Ensuite, à chaque étape, des indications et des ressources nous été mises à disposition par le coach du projet, ce qui a aidé à surmonter les difficultés.

5.3 Perspectives

Plusieurs pistes d'amélioration sont envisageables :

- Le département des Hauts-de-Seine est extrêmement dense en constructions, hormis quelques cas particuliers : forêts, parcs, aéroport, stades, etc. Il pourrait donc être intéressant de classer les images en fonction du type d'urbanisation : un ensemble d'immeubles, par exemple, présentera souvent peu de séparations entre les bâtiments ; il est également peu susceptible de comporter des bâtiments illégaux. On pourrait alors tenter d'ignorer les faux positifs sur ce type d'images, alors que, dans un contexte de lotissement, ils seraient plus signifiants.
- Lors de la phase d'exploration des données, nous n'avions pas identifié que certaines images en bordure de carte n'avaient pas de labels (car appartenant à un autre département) : cela nous est apparu en examinant les performances du modèle de régression. Ces images gagneraient à être supprimées de la base avant l'entraînement des modèles.

- Avec plus de temps, nous aurions souhaité faire varier les paramètres afin d'en évaluer les effets : taille de batch, nombre de couches, fonction d'activation, optimiseur, etc.
- Sur le modèle de segmentation sémantique, on pourrait tenter d'exclure les bâtiments suspects dont la forme est improbable (par exemple très effilés). En effet, nous avons remarqué que de nombreux artefacts apparaissaient, en particulier des bordures de bâtiments ; sur des constructions volumineuses, ces artefacts dépassent souvent les 20 m^2 : ils deviennent alors de « faux positifs parmi les faux positifs » que sont nos bâtiments suspects !
- Nous disposons de deux modèles avec des performances intéressantes. On pourrait imaginer de créer un méta-modèle prenant, en entrée, les résultats de chacun des modèles, ainsi que des informations annexes : surface du polygone détecté par YOLOv8, position par rapport aux bordures du département, densité en constructions de la zone où est situé le polygone, etc. Ce méta-modèle, qui pourrait être bâti sur un algorithme de *machine learning*, aurait pour objectif d'améliorer la précision de détection.
- Un retravail du code permettrait d'éviter les redondances entre les divers *notebooks*, en factorisant certaines fonctions dans un module.
- Enfin, dans une perspective d'utilisation plus large, un travail d'intégration des *notebooks* dans un programme unique, avec l'ajout d'une interface utilisateur, devrait être mené.

Pour conclure, ce projet s'est avéré passionnant. Nous aurions souhaité aller encore plus loin dans le raffinement des modèles et l'exploitation des résultats et, plus particulièrement, l'automatisation de la détection des constructions illégales sur la base des résultats issus de la segmentation d'instances.

6 Annexes

6.1 Déroulé du projet

	août-23	sept-23	oct-23	nov-23	déc-23	janv-24	févr-24	mars-24	avr-24
Découverte du starter kit									
Exploration des données									
Rapport d'exploration									
Préparation des données									
Modèles de régression									
Modèles UNET									
Modèles YOLOV8									
Interprétation des résultats									
Finalisation du code									
Rédaction du rapport									
Site de présentation Stramlit (à venir)									

6.2 Table des traitements

Notebook
01a - Exploration BDTOPO & BDORTHO.ipynb
01b - Exploration Polygones.ipynb
02a - Création Dataframe BDTOPO.ipynb
02b - Création Dataframe & images BDORTHO.ipynb
03 - Création Masques BDTOPO.ipynb
04a - Régression RGB.ipynb
04b - Régression IRC.ipynb
04c - Régression Transfer learning.ipynb
05 - Segmentation UNET.ipynb
06a - Préparation Yolo.ipynb
06b - Entraînement Yolo.ipynb
06c - Interprétation Yolo.ipynb

6.3 Table des données principales

Données	Type	Contenu
dft	dataframe	Polygones BATIMENT et RESERVOIR de la base BDTOPO
dfi	dataframe	Méta-données des images découpées et de leurs labels sous forme de masque
ii-ccc.jpg	fichiers .jpg	Images RGB ou IRC découpées (ii=numéro d'image d'origine, ccc=numéro de découpe)
Maksk-ii-ccc.png	fichiers .png	Labels sous forme de masque noir & blanc
df_train df_valid	dataframes	Données d'entraînement et de validation pour alimenter les modèles
df_test	Dataframe	Données utilisées pour évaluer le modèle, utilisées comme base des illustrations du présent rapport
df_residus	dataframe	Comparaison entre prévision et nombre de bâtiments pour évaluation des modèles de régression
ii-ccc-txt	fichiers .txt	Liste des polygones de chaque image, en vue d'entraînement du modèle YOLOv8
df_true_polygons	dataframe	Liste de tous les polygones de BDTOPO par image pour évaluation de YOLOv8
df_pred_polygons	dataframe	Liste de tous les polygones prédits par image pour évaluation de YOLOv8
df_suspect_polygons	dataframe	Liste des polygones faux positifs par image pour criblage et analyse des constructions illégales

6.4 Table des modèles

Modèle	Type	Images	Métrique
régression_202402270757	Régression	RGB	MAE=10,05
régression_202402262204	Régression	IRC	MAE=10,66
vgg_202402272245	Régression (transfer learning)	RGB	MAE=12,74[1] MAE=10,06[2]
UNET_202402282311	Segmentation sémantique	RGB	f1=0,84
yolov8l-seg-GPUT100-11h	Segmentation d'instances	RGB	f1=0.86

6.5 Bibliographie

Document	Source
U-NET : le réseau de neurones de Computer Vision	https://datascientest.com/u-net
Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan & Andrew Zisserman	https://arxiv.org/abs/1409.1556v6
Grad-CAM class activation visualization	https://keras.io/examples/vision/grad_cam/
Modèle YOLOv8-seg de segmentation des instances	https://docs.ultralytics.com/tasks/segment/
Dive into YOLOv8: How does this state-of-the-art model work?	https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

[1] Entraînement avec coefficients des couches VGG16 bloqués
[2] Affinement de l'entraînement avec coefficients des 4 dernières couches VGG16 débloqués

Table des matières