# Conception du système

## Acquisition d’un dataset

Afin de réaliser un modèle de segmentation sémantique d’image il est nécessaire d’avoir un dataset assez particulier.

* Dataset en Machine Learning Classique

Dans d’autre problème classique de **supervised Learning** (apprentissage supervisé) le **dataset** est sous forme de matrice contenant des scalaires. Ces scalaires sont de deux catégories on a en premier lieu les **features** puis la **target.**

**Les features** désignent ici un ensemble de valeurs qui seront pris en paramètres dont dépendent les valeurs de notre **target**

**Les targets** désignent les valeurs en résultantes des features.

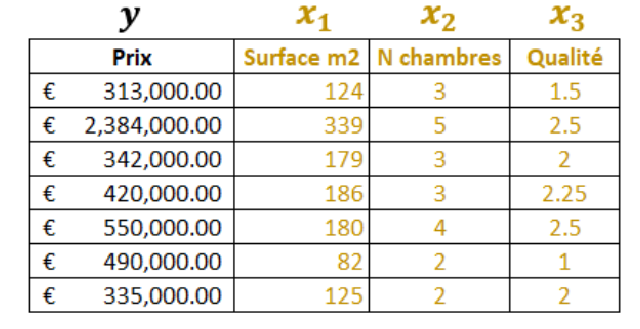


Figure 1: Exemple de dataset Scalaire

La figure ci-dessus nous montre l’exemple d’un dataset. Les valeurs de **X** sont nos **features** et les valeurs de **Y** sont nos **targets**. Le but de ce type de modèle est d’étudier la relation entre les **features** et les **targets** afin de fournir une fonction sous la forme qui permettra de prédire les valeurs de **Y** en sorties

* Dataset en Deep Learning

Le Deep Learning vient comment une sorte d’extension au Machine Learning. Il permet de faire les mêmes opérations que celle faites en Machine Learning mais sur une quantité significativement plus importante de données. Dans le cadre de notre projet notre dataset sera constituer de ce qu’on appelle **les tile images** et **les tile masks** comme illustre la figure ci-dessous



Figure : Image du dataset



Figure 3: Mask d'une image du dataset

- Une **tile image** : celle-ci désigne un morceau d’une image satellitaire de type **matrice** qui sera traiter afin de servir à entrainer notre modèle. Une **tile image** est en réalité un morceau d’une image satellitaire beaucoup plus grande cette technique consistant à découper les images satellitaires est nécessaire afin de faciliter le stockage, le traitement et la diffusion de ce type d’image.

- Une **mask image :** chaque tile image du dataset possède sa **mask** **image**. Cette image est dite « masquer » ou « filtrer » le but de filtre ici est d’éliminer les zones de l’images qui ne sont pas pertinentes afin de ne conserver que nos zones d’intérêts. Chaque zone d’intérêts est représentée par une couleur et en ce qui concerne ce projet nous en avons 06 au total

L’élaboration d’un telle dataset est assez complexe car il est nécessaire de photographier la zone en question avec l’aide d’un satellite ou d’un drone et ensuite il faut pouvoir extraire les zones d’intérêt ce qui est une tâche assez fastidieuse.

Notre dataset a été élaborer à partir de données recueillies dans la ville de **Dubaï au Emirates Arabes Unies** et qui est mis à la disposition de tous sur le site web **Kaggle.com**.

## Traitement des images

### Représentation d’une image en mémoire

Pour l’élaboration de modèle de segmentation sémantique d’image il est important de premièrement comprendre comment une image est stockée en mémoire. Il faut d’abord que l’on distingue que type d’image : Les images en échelle de gris et les images en couleur.

* Dans le cas particulier des images en échelle de gris, chaque **pixel** est stocké sur un **octet** c’est-à-dire une valeur entre 0 et 255. La bibliothèque **opencv** de python nous permet de lire des images sous ce format avec la commande « cv2.imread(‘/path/image.jpg’, 0) » comme l’illustre la figure suivante



* Dimension : (644, 797)
* Taille : 513268 pixels

Figure 4: Image en echelle de gris

Ce type d’image est représentée dans la mémoire de l’ordinateur sous forme d’une **matrice à deux** dimensions.

* Nous avons aussi les images de couleur. La couleur ici est former d’une association de valeur de trois couleur fondamentale le **rouge, le vert et le bleu** (RGB). Chaque pixel ici au lieu d’avoir une simple valeur est constituer d’un vecteur de 03 valeurs donc chaque valeur est représentée sur 1 octet. C’est typiquement ce type d’images que sera utiliser pour l’entrainement de notre modèle



* Dimension : (644, 797, 3)
* Taille : 1539804 pixels

Figure : Image en couleur

### Preprocessing des images

Le preprocessing des images est une étape primordiale pour la conception d’un modèle de segmentation sémantique d’images. Le but essentiel de cette étape est de pouvoir préparer les images à être utiliser pour entrainer notre modèle. Les images qui présentent dans le dataset n’ont pas toujours les mêmes dimensions or pour entrainer un modèle de ce type il est nécessaire de toutes les données en entrée soit de la même dimension appelé **patch size.**

* **Fixation du patch size**

Le **patch size** est la valeur de dimension que nous utiliserons pour notre modèle cette taille est généralement de pixels mais elle dépend grandement de la résolution de l’image.

* Pour des images de hautes résolution le patch size peut aller de pixels
* Pour des images de moyennes résolutions le patch size peut aller de pixels

Il est important de noter que plus la résolution de l’image satellitaire est grande plus la taille des pixels sera réduite et par conséquent une petite zone de l’images contiendra une grande quantité d’information c’est cela qui explique le fait que la taille du patch size est inversement proportionnelle à la résolution de l’image.

De façon général la segmentation sémantique permet d’attribuer une classe à chaque pixel de l’images c’est pour qu’il est nécessaire que le patch size soit aussi réduit afin de faciliter la classification de pixels. De plus il est nécessaire de trouver un certain équilibre en choisissant la valeur du patch size. Une patch size trop petit cela va grandement augmenter la charge de calcul et réduire lé vitesse de traitement et choisir un patch size trop faible causera un manque de précision du modèle

Il est aussi important de noter à la fois les **tile images et les mask images**

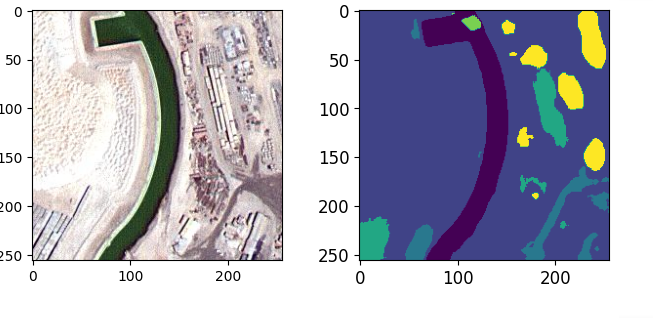


Figure 6: Images et mask au patch size

La figure ci-dessus nous montre le résultat que nous avons obtenu après avoir découper à la fois une **tile image et sa mask image.** Ce tracé a été obtenu avec la bibliothèque **matplotlib** de python et nous montre parfaitement que chaque image est de la taille du patch size c’est-à-dire de pixels.

* **Classification des pixels**

Cette opération les uniquement effectuer sur le **tile mask** de notre dataset. Lors du préprocessing des données il est crucial de pouvoir identifier le **land marks** (zones d’intérêt de notre modèle ou classe d’objet) et associer **chaque pixel de chaque tile image à une classe d’objet** dans le cadre de ce projet nous identifierons 06 classe d’objet avec chacun ayant sa propre couleur en hexadécimale

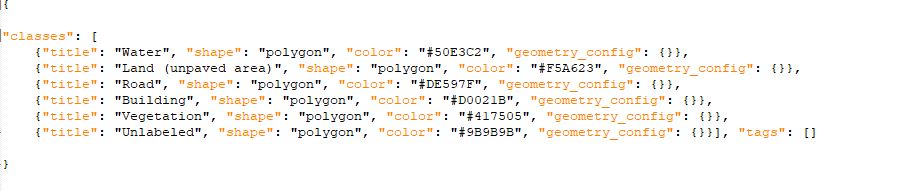


Figure 7: Classes identifiables

Un fois les classe identifier nous allons créer une nouvelle matrice qui contient **la valeur de l’indice de la classe de chaque pixel dans le fichier ci-dessus** pour bien comprendre cette étape on va prendre l’exemple d’une image M

Notre image M est une image en couleur ce qui signifie qu’elle peur être assimiler à un **tenseur d’ordre 3 (**vecteur à 3 dimensions) notre image M est alors constituée de N pixel. Chaque pixel P est caractérisé par

*l = indice sur ligne*

*c = indice colonne*

*v = vecteur rgb (r, g, b)*

Si notre pixel fait par exemple partie du la classe « Water » une autre matrice M’ sera générer et **aura l’indice de la classe de ce pixel P à cette position**

Ainsi à la fin de cette étape si on avait N images au départ, on aura également N matrices chacune caractérisant chaque image avec l’indice à la place de chaque pixel.

* **Encodage OneHot Division du dataset et**

A cette étape d’avancement de réalisation notre dataset se divise en trois parties grandes parties

* La partie A former de nos **tile image**
* La partie B former de nos **tile mask**
* Enfin la partie C constituer de nos matrices contenant la classe de chaque pixel pour chaque image de la partie B du dataset
* Encodage OneHot

L’encodage OneHot est une méthode utiliser lors de la segmentation sémantique d’image afin de pouvoir plus facilement stocker la partie C du dataset en mémoire. En effet pour que la valeur de l’indice du pixel n’est pas d’incidence significatif lors de l’entrainement du modèle (par exemple la classe « water » à pour valeur « 0 » et la classe « vegetation » à pour valeur « 4 ») il est nécessaire de trouver une méthode qui permettrai non seulement d’éviter le problème énoncer précédemment mais en plus rendrai la tâche de traitement de l’ordinateur plus simple à réaliser c’est donc là l’importance de cette méthode d’encodage son principe de fonctionnement est le suivant :

Premièrement nous allons identifier le nombre total de classe dans notre dataset 06 dans notre cas particulier) puis nous allons créer un vecteur contenant 06 valeurs soit à « 0 » ou à « 1 » **la seul valeur à « 1 » de ce vecteur sera celle qui correspond à l’indice de la classe qui lui est associer dans notre dataset**.

Par exemple si nous avons un pixel P qui appartient à la classe « vegetation » par exemple (indice 4) ce pixel sera représenté par un vecteur V sous la forme V [0, 0, 0, 0, 1, 0]. Cet encodage est rendu possible grâce à la fonction **to\_categorical** de la bibliothèque **tensorflow.keras.utils** illustrer comme suit



Figure 8: Encodage OneHot

Cette fonction prend 02 paramètre : le premier est le nombre total de classe présent de notre dataset (num\_class) et la second est le dataset précèdent contenant les valeurs de l’indice de chaque classe (labels). A la fin cette étape chaque image sera sous la forme suivante :

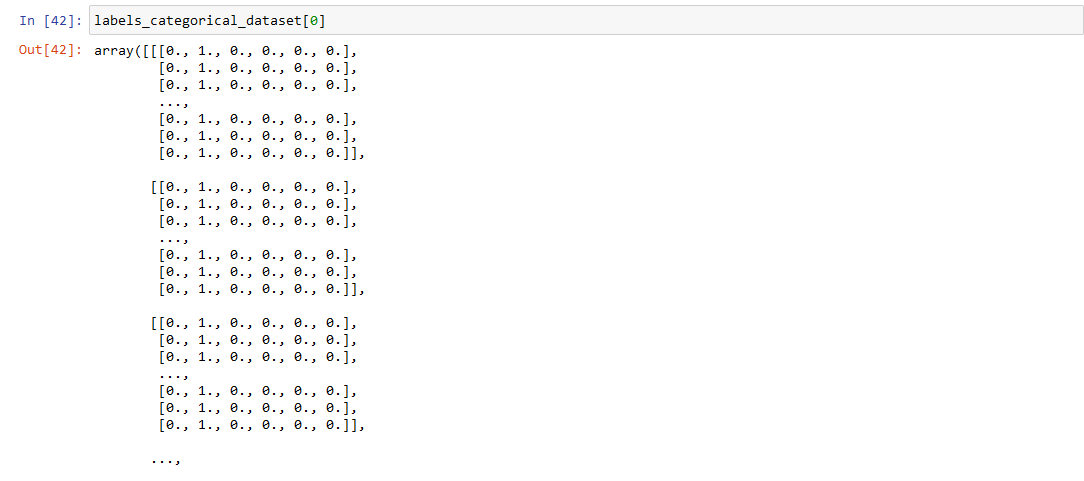


Figure : Image après encodage OneHot

* Division du dataset

Dans le Machine Learning en général il est toujours nécessaire de séparer le dataset de départ en 04 parties comme suit :

* X\_train : c’est la partie des **tile image du dataset** qui va être utiliser pour entrainer notre modèle
* Y\_train : c’est la partie des **tile mask** qui sera utiliser pour entrainer notre modèle
* X\_test : il s’agit de la partie des **tile image** sur lesquelles nous allons tester notre modèle
* Y\_test : qui représente ici nos données de validation c’est à partir de ces **tile mask** que nous allons voir à quel point notre modèle épouse la réalité et est précis

Il existe aussi une fonction qui nous permet de réaliser cette séparation du dataset du non de **train\_test\_split** de la bibliothèque **sklearn.model\_selection.** Cette opération est réalisée comme suit :

La fonction **train\_test\_split** prend ici 04 paramètres

* : représente notre dataset contenant les **tile image**
* : représente notre dataset contenant les **tile mask**
* : représente la proportion entre les images contenues dans la X\_test et dans la X\_train
* : indique que les images doivent être prise de façon aléatoire dans le dataset

Une fois notre dataset diviser nous somme désormais prêt à entamer l’entrainement du modèle.

## Entrainement du modèle