Université de Montpellier
Faculté Des Sciences
Département Informatique

HMIN339M - Méthodes avancées de la science des données

Classification de l'occupation du sol en Dordogne

Github :
`https://github.com/StefanStefanovski/Projet-HMIN339/blob/master/Projet_HMIN339.ipynb`

Stefan Stefanovski
Riadh Guemache
Minyoung Yang

Décembre 2020

Table des matières

1 Introduction

1.1 Contexte

Ce projet a été effectué au sein de la faculté des sciences, Université de Montpellier, durant le premier semestre de Master 2, pour L'UE HMIN339M - Méthodes avancées de la science des données.

1.2 Introduction à la problématique

L'occupation de sol désigne le type d'usage/non-usage fait des terres par l'Homme (eau,forêt ..etc). L'utilisation d'un sol de qualité est nécessaire dans la société actuelle. Que ce soit d'un point de vue écologique, économique ou sociétale, l'occupation de sol prend un aspect décisionnel important : elle explique divers phénomènes qui impactent le territoire (déforestation,artificialisation des terres ...etc).

1.3 Objectif

Dans le cadre de ce projet, il est demandé de proposer une approche automatisée de définition de l'occupation du sol dans le département de Dordogne (région Nouvelle-Aquitaine). Il faut proposer des modèles d'apprentissage profond afin de prédire diverses classes d'occupation du sol parmi lesquelles se trouvent des zones urbaines, de l'eau, de la forêt, des landes et des cultures.

Notre objectif est de réaliser une classification en utilisant un jeu de données représenté par une série temporelle d'images satellites optiques Sentinel-2 et d'une vérité terrain (information de classe et d'identifiant).

1.4 Outils de travail

Pour ce faire nous avons utilisé Google Colab la version cloud de Jupyter Notebook qui permet de créer des modèles d'apprentissage sans avoir besoin d'installer quoi que ce soit sur notre machine.

Nous avons utilisé des outils de Scikit-learn, une bibliothèque libre en Python destinée à l'apprentissage automatique ainsi que des outils de Keras et TensorFlow, des bibliothèques libres en Python destinées à l'apprentissage profond.

2 Les données

Les données brutes sont représentées par une série temporelle de huit images satellites optiques Sentinel-2, la vérité terrain a été échantillonnée sur cinq classes (zones urbaines, de l'eau, de la forêt, des landes et des cultures).

Nous disposons d'un jeu d'entraînement déséquilibré composé de 1863 objets vus comme des pixels ayant le même identifiant ainsi que d'un jeu de test composé de 800 objets dont on doit fournir leur classe d'occupation de sol.

3 Pré-traitement

Nous avons commencé par une lecture d'images pour créer des séries temporelles de bandes de rouge, vert, bleu et du proche infrarouge. Ensuite, pour ne pas bousculer la dynamique de certaines classes d'occupation du sol comme la végétation qui ont un profil temporel bien spécifique, nous avons normalisé les bandes sur les séries temporelles pour avoir des valeurs comprises entre 0 et 1. Après, nous avons récupéré les positions des pixels du jeu d'entraînement et de test tel que : pour le jeu d'entraînement il faut prendre les pixels dont on connaît l'identifiant et la classe, pour ce qui est du test on prend le pixels dont on connaît l'identifiant mais pas la classe. Puis, à partir du jeu d'entraînement nous avons créé un jeu de validation en s'assurant que si deux pixels ont le même identifiant alors ils appartiennent au même lot.

Enfin, en fonction du modèle de prédiction, nous avons sauvegardé des données (les échantillons d'entraînement, de validation et de test) reformées en différentes dimensions, afin de ne pas avoir à refaire la même tâche lorsqu'on change de modèle et pour optimiser la gestion de la mémoire.

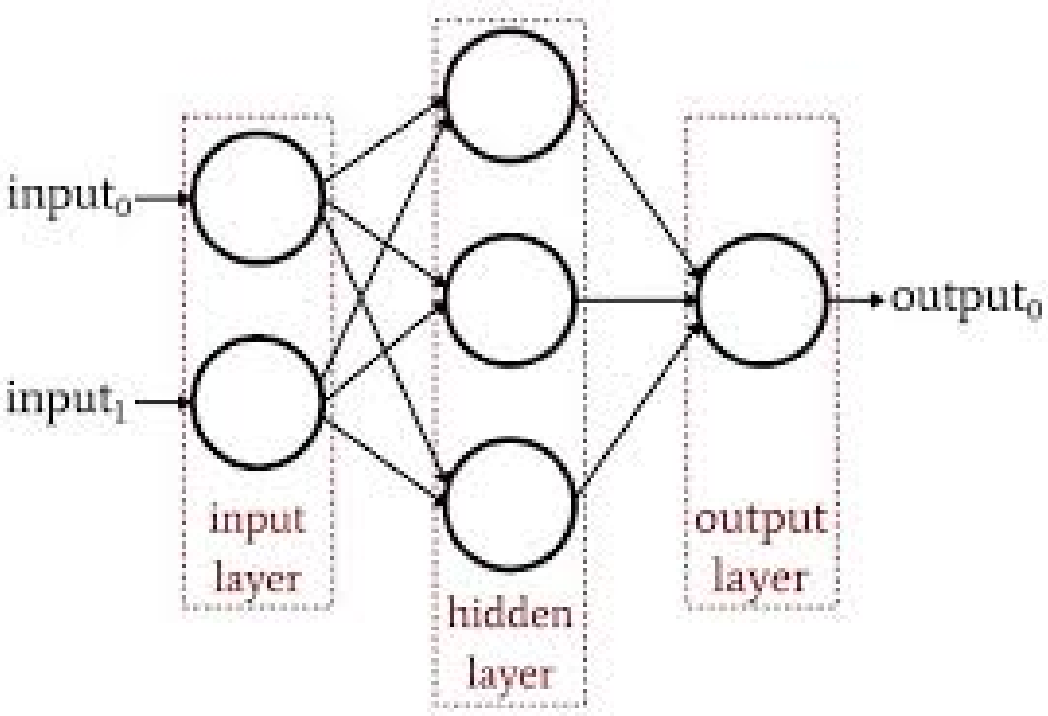
4 Définition des modèles

Nous avons utilisé trois modèles : MLP, CNN1D et CNN2D ainsi que deux combinaisons MLP avec CNN1D et MLP avec CNN2D.

Un réseau de neurones est un système dont la conception est inspirée du fonctionnement des neurones biologiques.

4.1 MLP

Le perceptron multicouche (multilayer perceptron MLP) est un type de réseau de neurones artificiel au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Comme son nom l'indique, il est composé de plusieurs couches - couche d'entrée, couches cachées et couche de sortie. Dans le contexte du traitement d'images, le MPL utilise un perceptron pour chaque pixel tel que chaque nœud soit connecté à tous les autres nœuds de la couche suivante et précédente formant ainsi un réseau très dense.

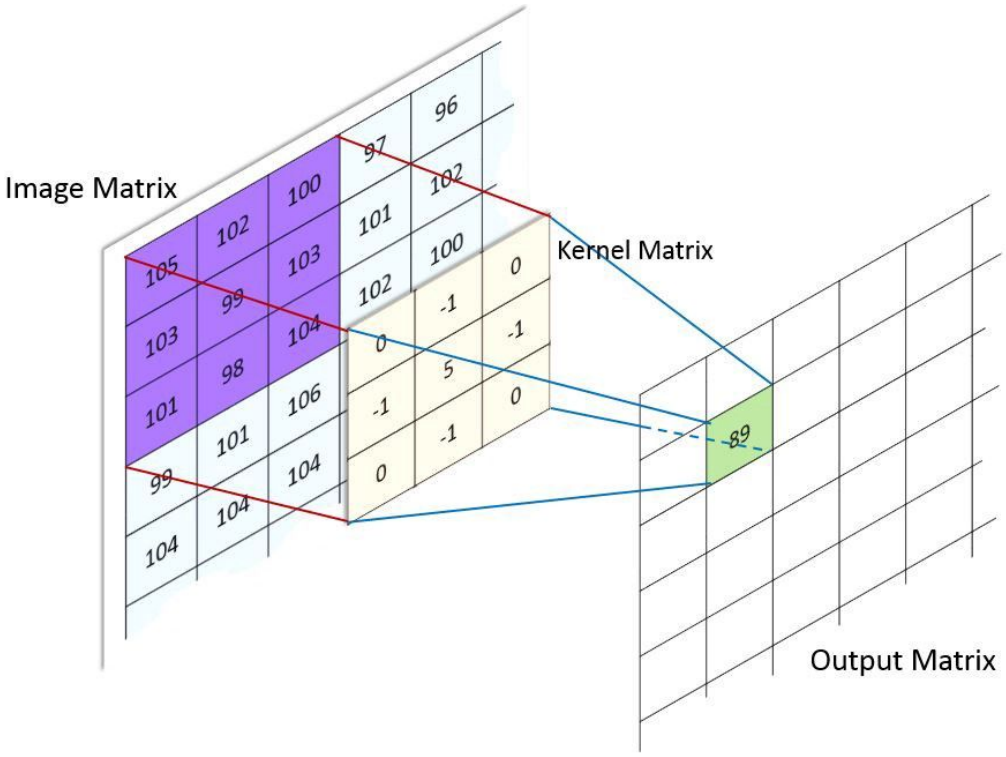


Exemple : MLP

4.2 CNN

C'est un réseau de neurones convolutifs, c'est-à-dire un type de réseau de neurones artificiel dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux.

Des modèles CNN (Convolutional Neural Network) ont été développés pour la classification d'images. Les couches sont peu ou partiellement connectées ou plutôt que complètement. Chaque nœud ne se connecte pas à tous les autres nœuds. CNN tire parti du fait que les pixels proches sont plus étroitement liés que les pixels distants.



Exemple : CNN-2D

4.2.1 CNN-1D

Il s'agit d'un modèle CNN à une dimension car le noyau glisse le long d'une seule dimension sur les données. Ce modèle extrait les entités de données de séquences et mappe les caractéristiques internes de la séquence.

4.2.2 CNN-2D

C'est le CNN standard, il est généralement utilisé pour le traitement d'images. Le modèle accepte une entrée bidimensionnelle représentant les pixels et les canaux de couleur d'une image, il est appelé CNN à 2 dimensions car le noyau glisse le long de 2 dimensions sur les données.

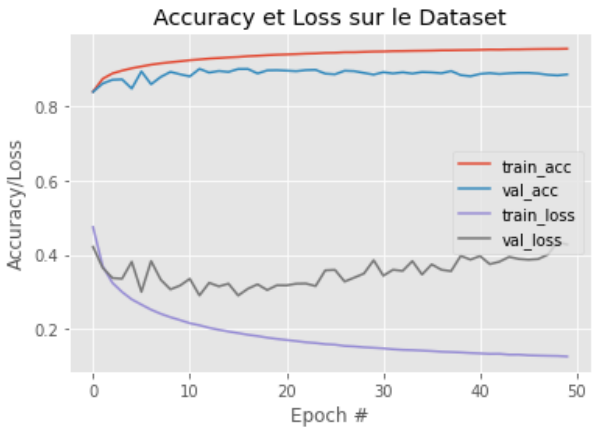
5 Notre expérience avec les modèles

5.1 Les modèles

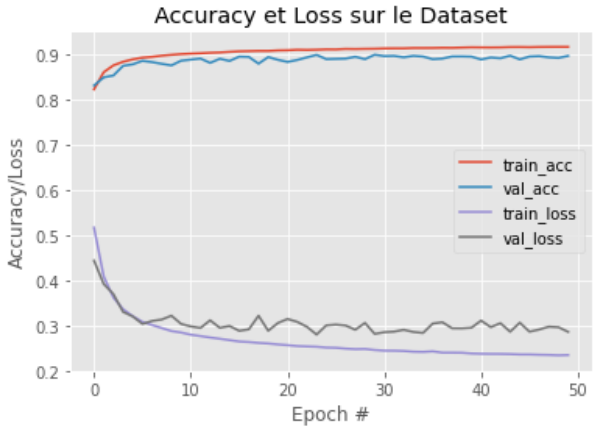
5.1.1 CNN-1D

Nous commençons avec les couches de convolution d'une dimension (Conv1D) en augmentant le nombre de neurones de 32 à 128. Ces dernières prennent les données en entrée et les segmentent afin d'apprendre les caractéristiques locales tout en détectant la présence d'un ensemble de features. En sortie, nous avons un feature map. D'ailleurs nous nous sommes arrêtés à 128 neurones car la couche avec 256 neurones n'améliore pas le résultat.

Ensuite nous avons les couches de Dropout. Sans ces dernières, le modèle peut sur-apprendre. Par conséquent, nous avons obtenu un score assez haut mais la validation loss (l'erreur) a aussi augmenté au fil des époques comme sur le schéma suivant :



Les couches de Dropout sont alors appliquées après chaque couche de convolution afin d'éviter l'overfitting. A chaque itération, ces couches désactivent aléatoirement, en mettant le poids à 0, un pourcentage des neurones d'une couche (30% dans notre cas) . Cette technique permet d'empêcher au modèle l'apprentissage par cœur. Le graphe suivant illustre que nous avons moins d'erreur que précédemment et que la validation loss reste constante ou diminue :



Par la suite nous avons ajouté deux couches de pooling : Maxpooling1D et GlobalAveragePooling1D :

La première prend en entrée la sortie de la couche précédente et réduit sa dimension tout en permettant de garder les informations les plus importantes. En effet, elle ne prend que la valeur maximale sur un intervalle de la taille donnée (pool_size). Dans notre modèle, elle prend donc la matrice de sortie de la couche de convolution (le feature map) et réduit sa dimension de moitié (car Maxpooling1D avec 2 en paramètre).

GlobalAveragePooling1D utilise le même principe que le Maxpooling, mais il calcule la moyenne des valeurs sur chaque feature map venant de la couche précédente.

Enfin nous avons la dernière couche Dense qui permet de réduire le nombre de neurones à 5, ce qui correspond au nombre d'échantillons de notre vérité terrain.

5.1.2 CNN-2D

Comme dans le modèle CNN-1D, nous commençons avec les couches de convolution (Conv2D) en augmentant le nombre de neurones et en diminuant la taille de Kernel.

Nous avons commencé avec des photos de taille 5x5 comme input, car nous ne disposons pas de beaucoup de mémoire pour entraîner notre modèle sur des photos plus grandes. Pour cela il n'est pas possible de définir plus de deux couches de convolution suivies de MaxPooling et Dropout.

Donc pour la première couche de Conv2D, nous avons défini le kernel size à 3x3 et la dimension d'input à 5x5x32 (32 correspond aux couleurs normalisées), elle est suivie d'une couche Dropout.

Comme dans CNN1D, avec Dropout nous avons pénalisé notre modèle mais cette fois-ci de 10% après la première couche Conv2D et de 20% d'après la deuxième couche Conv2D. Les différents et plus petits dro-

pouts (comparé à CNN 1D) s'expliquent par le nombre de neurones que nous avons défini dans les couches de CNN2D. Nous constatons également que le loss et val_loss n'augmentent pas, donc il ne fait pas overfitting même après avoir diminué le dropout et donne le meilleur résultat.

Pour la deuxième couche de convolution, le kernel size est de 1,1 car le output du premier est de 1,1, suivi d'une couche Dropout de 20% et de MaxPooling.

On termine avec un GlobalAveragePooling pour mettre les données en une dimension et d'une couche Dense de 5 afin d'avoir le nombre des classes souhaité.

5.1.3 MLP

Notre modèle de MLP est constitué de 5 couches. Le nombre de neurones augmente de 32 à 256, à finit par 5 grâce aux couches Dense, et les couches de Dropout sont utilisées afin d'éviter le surapprentissage.

5.1.4 Concaténation de MLP et CNN-1D

Nous avons également construit des modèles en concaténant deux modèles. Le premier est la concaténation de MLP et CNN-1D.

Pour cela, nous avons défini séparément les deux modèles, puis nous concaténons les dernières couches de chacun en utilisant la fonction layers.Concatenate(), ensuite nous avons ajouté une couche de Dropout et deux couches de Dense : une avec 256 et l'autre avec 5 pour avoir le nombre de classes correspondant.

Il est important de changer de dimension de données car ces deux modèles n'ont pas la même dimension d'entrée, nous avons donc réduit la dimension pour CNN-1D en utilisant .reshape pour qu'elle soit compatible avec le modèle MLP.

Dans le modèle de CNN-1D, nous avons appliqué la stratégie de Dropout comme dans CNN2D :
Le premier dropout est de 10%, puis de 20% et au final de 30% après chaque couche de convolution. Nous remarquons que cette méthode donne le meilleur score.

5.1.5 Concaténation de MLP et CNN-2D

Ce modèle est le résultat de la concaténation de MLP et CNN-2D, la méthode appliquée est la même que celle du précédent.

Nous avons défini les deux modèles comme ils sont définis précédemment et leur concaténation donne un résultat satisfaisant (f1 score $>$ 0.9). Néanmoins, la validation loss augmente au fil des époques malgré les couches de dropout dans chaque modèle et après la concaténation (figure 1). Nous avons donc augmenté le pourcentage dans chaque couche de Dropout à 0.5, en résultat, le f1 score diminue mais il n'y a plus d'overfitting (figure 2).

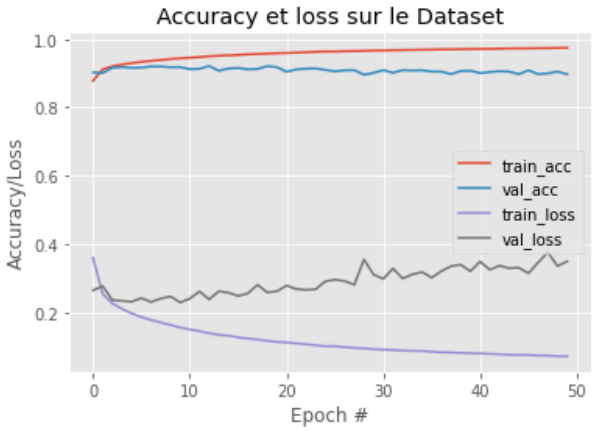


FIGURE 1 – Dropout avec <0.5

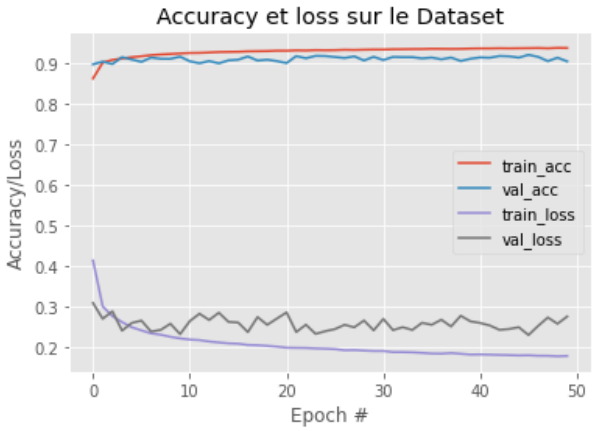


FIGURE 2 – Dropout avec =0.5

5.2 Analyse des résultats

Chaque modèle est ensuite compilé et entraîné. Nous avons défini le nombre d'époques à 50 pour l'entraînement. Le tableau suivant nous permet de comparer les performances de chaque modèle :

	acc	val_acc	f1_score	f1_score sur le jeu de test
CNN-1D	0.90	0.90	0.86	0.84
CNN-2D	0.92	0.92	0.89	0.87
MLP	0.92	0.90	0.87	0.85
CNN-1D + MLP	0.90	0.90	0.88	0.85
CNN-2D + MLP	0.93	0.92	0.85	0.83

Nous remarquons que le CNN-2D donne le meilleur f1 score (0.89). Ceci peut s'expliquer par le fait que le CNN-2D est un modèle spécialisé pour les images.

La concaténation de CNN-1D et MLP donne également un score satisfaisant, car le MLP est un algorithme classique de machine learning qui se comporte bien avec les types numérique et textes.

6 Paramétrage du modèle

Afin de mieux observer le comportement des modèles, nous avons décidé de prendre le modèle donnant le résultat le plus satisfaisant et de l'étudier en changeant les paramètres. D'après l'analyse des résultats faite précédemment, nous allons prendre le CNN-2D.

Pendant l'entraînement des modèles, nous avons mis le nombre d'époques à 50 et la taille de batch à 256. L'optimizer choisi est Adam. Dans la partie suivante, nous allons changer ces paramètres et observer l'évolution de f1 score.

6.1 Le nombre d'époques

Le tableau suivant montre l'évolution du score du modèle en fonction de différents nombres d'époques (avec batch_size=256 et optimizer=adam).

CNN-2D	Epoch# =10	50	100
F1-Score	0.86	0.89	0.88

6.2 La taille de batch

Cette fois-ci, on étudie l'évolution du modèle en fonction de la taille de batch (avec le nombre d'époques =50 et optimizer=adam).

CNN-2D	batchSize=32	128	256
F1-Score	0.89	0.89	0.89

6.3 L'optimizer

Tout en gardant le même nombre d'époques et la taille de batch, nous avons changé l'optimizer.

CNN-2D	adam	SGD	RMSprop
F1-Score	0.89	0.83	0.86

D'après ces résultats, nous pouvons conclure que le nombre d'époques influe sur la qualité de résultat jusqu'à un certain seuil. Mais quand le nombre est trop grand, le score reste constant ou diminue.

Par contre la taille de batch ne fait pas augmenter le score du modèle, et plus le batch size est petit, plus le temps d'entraînement augmente.

Quant à l'optimizer, le SGD n'améliore pas le résultat mais le point intéréssant est que les valeurs loss commencent très haut par rapport à l'optimizer adam et diminuent fortement (figure 3). Changer la valeur de learning rate pourrait changer le résultat. Pour le RMSprop, on constate que le score est plus bas par rapport au deux autres mais l'accuraccy et loss restent inchangés (figure 4).

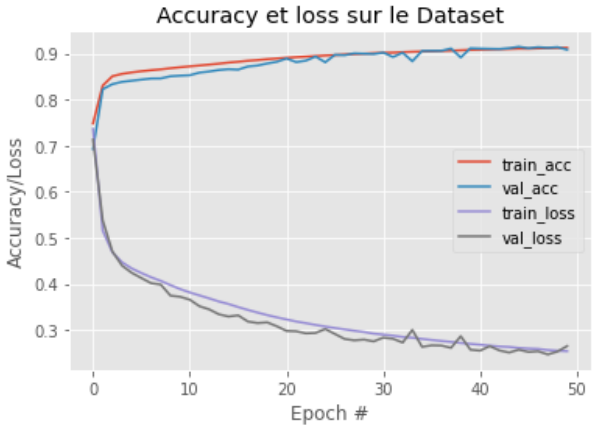


FIGURE 3 – SGD Optimizer

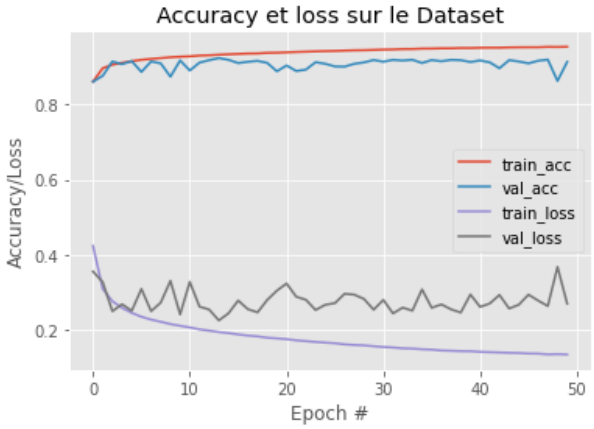


FIGURE 4 – RMSprop Optimizer

7 Conclusion

L'objectif de ce projet était de réaliser une classification de l'occupation du sol en Dordogne, donc de proposer des modéles de deep learning pour prédire des classes d'occupation du sol.

A son terme, nous avons compris et appris ce qu'est réellement la classification d'occupation de sol, ainsi que le traitement d'images en générale.

Malgré nos connaissances limitées dans ce vaste domaine, nous avons pu analyser les jeux de données et les classifier, en utilisant divers modèles d'apprentissage profond (CNN-1D, CNN-2D, MLP et quelques combinaisons entre eux). Toutes les informations et remarques ont été effectuées en détail dans la partie d'analyse.

Le cahier des charges a donc bien été rempli malgré les nombreuses difficultés rencontrées. L'une des principales est la pandémie exceptionnelle que connaît le monde actuellement, et donc le confinement obligatoire : le projet devant être effectué en groupe, l'échange n'a pas été évident. Ce projet fut une occasion de mettre à l'épreuve nos capacité de gestion. Nous nous sommes aussi familiarisés avec les différents outils que nous avons utilisés. Ce qui nous a finalement fourni une approche concrète de notre métier de demain.