Une approche Deep Learning pour la classification de modèle BIM

T.M

07/01/2021

ABSTRACT

L'utilisation du BIM (building information modeling) à complètement révolutionner la façon d'aborder et de réaliser des projets de construction.
Avec l'avénement du BIM nous sommes passés d'environnement de travail désorganisé et principalement en 2D, à l'utilisation de CDE (common data environment) pour avoir une source unique de d'informations et de données, et des modèles 3D qui sont plus précis et apporte une meilleure compréhension des problèmatique liés à la résolution de problèmes techniques comme par exemple la detection de clashs.
Cela permet également de mieux gérer les composants d'un projets comme par exemple les différents types d'objets BIM que l'on va utiliser pour la modélisation (différentes types de poutres en béton armés par exemple) et de ce fait de mieux en gérer les coûts.
Un objet BIM est composé de sa géométrie en 3D ainsi que d'attributs, qui seront différentes selon la phase du projet ou l'objectif de la maquette (ex: type de béton, ratio de ferraillage).
Mais parce que les modèles 3D sont plus complexes, plus précis, ils sont également plus lourd et la maintenance d'une bibliothèque d'objet BIM est donc beaucoup plus fastidieuse et coûteuse.

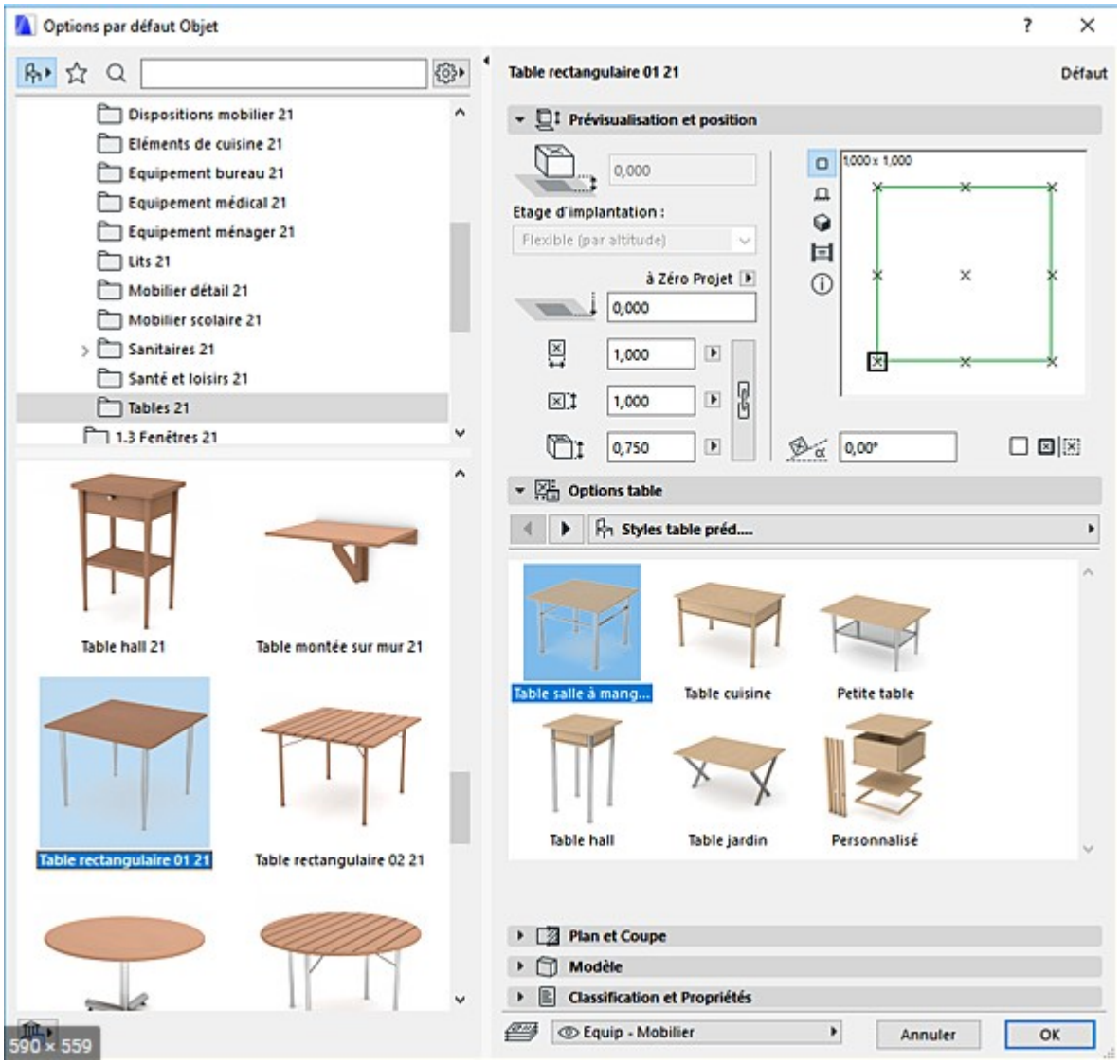

Figure 1: Exemple d'une bibliothèque d'objets BIM

Le processus décrit dans l'article, pour atteindre cet objectif, utilise la méthode suivante : Step 1 Choisir un jeu de données d'entraînement et de test adéquat. Step 2 Traiter les modèles 3D et représenter ses attributs sous forme de matrice. Step 3 Traiter cette donnée comme un auto-encodeur model. Step 4 La sortie d'une couche caché du modèle auto-encodé est utilisé comme entrée de la couche . Step 5 Initialiser les paramètres pour chaque couches de l'étape 4. Step 6 La sortie de la dernière couche cachée est défini comme entrée d'une couche supervisé. Step 7 Ajuster tout les paramètres du réseaux neuronale en fonction des règles de supervision. Ajouter tout les modèles auto-encodé pour former un stacked auto-encoder model.

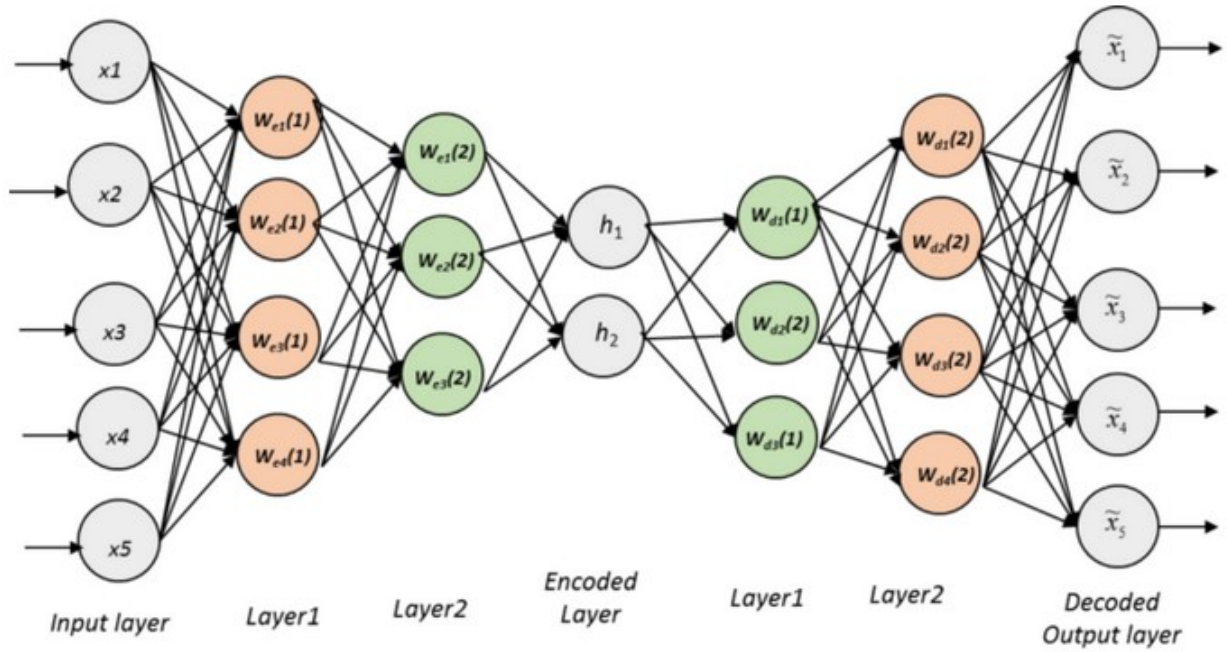

Figure 3: Stacked Autoencoder[3]

Figure 2: Exemple d'un SAE

On va s'intéresser ici dans un premier temps à l'auto-encodeur et son fonctionnement d'un point de vue mathématiques, ensuite on va regarder de plus près un des algorithmes de classification présenté dans le document. Ici le choix à été fait sur SVM (support machine vector ou machine à

vecteurs de support en français).

Partie 1 : Auto-Encodeur

Un auto-encodeur est un réseau de neurones artificiels exploité dans un contexte d'apprentissage non supervisé, il est composé d'un décodeur et d'un encodeur.
L'apprentissage non supervisé c'est quand l'apprentissage par la machine se fait de manière autonome.
L'encodeur va chercher à diminuer les dimensions des données d'entrer afin de les représenter dans un nouvel espace.
Le décodeur va reconstruire les données à partir de l'encodage.
Le but étant de reconstruire une donnée à partir de l'encodage en réduisant au maximum les erreurs de reconstruction.
Dans notre cas ce qui nous intéresse avec cetté méthode c'est que l'auto-encodeur peut servir à réduire l'éffort de labélisation des données.

Mais c'est quoi d'un point de vue mathématiques ?

Dans sa forme la plus simple c'est un réseau de neurones non récurrents avec une couche d'entrée, une couche de sortie ainsi que des couches cachés reliant la couche d'entrée à la couche de sortie. La couche de sortie et la couche d'entrée possède le même nombre de noeuds (l'objectif étant de reconstruire et non de prédire).

Donc on sépare le réseau en 2 parties : l'encodeur et le décodeur :
encodeur :

$$\phi : X \to F$$

décodeur :

$$\psi : F \to X$$

ce qui donne :

$$\phi, \psi = argmin(\phi, \psi)||X - (\phi o \psi)X||^2$$

Si il y a une seule couche cachée, on peut définir l'étape d'encodage tel que

$$x \in \mathbf{R}^d = X$$

Et l'associer à

$$z \in \mathbf{R}^p = F$$

$$z = \sigma(Wx + b)$$

Z représente le code (ou variables latentes),

$$\sigma$$

est une fonction d'activation, W est une matrice de poids, et b un vecteur de biais.

Une variable latente est une variable qui n'est pas directement issue d'une observation, elle est déduite d'autres variables observées. La fonction d'activation c'est une fonction mathématique appliquée à un signal de sortie.

L'étape de décodage est défini comme suit:

$$x' = \sigma'(W'z + b')$$

Attention cependant selon la conception de l'auto-encodeur

$$\sigma', W' etb'$$

peuvent être les mêmes ou non que

$$\sigma, W etb$$

Partie 2 : Algorithme SVM

Dans l'article plusieurs algoritmes ont été essayés pour la classification avec les résultats suivants:

Table 4. The Accuracy of Four Algorithms

Algorithm	Accuracy (%)
KNN	83.5
NN	77.0
SVM	78.0
SAE	88.0

Ici on va regarder d'un peu plus près comment fonctionne le SVM.

Les machines à vecteurs de support sont une technique d'apprentissage supervisé pour résoudre des problèmes de discrimination et de régression. Ici ce qui nous intéresse c'est de résoudre un problème de discrimination (décider à quelle classe appartient notre objet). Dans les 2 cas pour résoudre le problème on utilise une fonction h tel que :

$$y = h(x)$$

Le cas le plus simple, celui d'une fonction discriminante linéaire est tel que :

$$h(x) = \omega^T x + \omega_0$$

Si

$$h(x) > 0$$

alors x est de classe 1, sinon il est de classe -1. La frontière de décision

$$h(x) = 0$$

est apellé un hyperplan séparateur. Le but d'un apprentissage supervisé est d'apprendre la fonction

$$h(x)$$

par le biais d'un ensemble d'apprentissage :

$$(x_1, l_1), (x_2, l_2, ..., (x_k, l_k), ..., (x_p, l_p)) \subset \mathbf{R}^N x - 1, 1$$

Avec

$$l_k$$

qui correspond au labels, p la taille de l'ensemble d'apprentissage, N la dimension des vecteurs d'entrée. Si le problème est linéairement séparable on obtient alors :

$$l_k(\omega^T x_k + \omega_0) \geq 0$$

$$1 \leq k \leq p$$

Conclusion

La conclusion de l'essai est qu'il est possible d'utiliser cette méthode pour classifier des modèles BIM cependant le nombre de couches cachées a un énorme impact sur la précision des résultats de classification.
Le nombres de couches cachés determines si l'objet peut être correctement classifié et représenté.
Si le nombre de couches cachés est trop petit la classe sera principalement de 1 mais à l'opposé si ce nombre est trop grand cela va démultiplier le temps d'entrainement du jeu de données pour une précision pas forcément meilleur.
Il n'y a actuellement pas de méthode qui permet d'obtenir rapidement le nombre de couches cachées idéale, il convient donc d'expérimenter afin de trouver les paramètres optimum selon le cas.