Apprentissage profond pour l'approximation d'une distance d'édition entre graphes

Guillaume Renton, Benoît Gaüzère, Pierre Héroux, Sébastien Adam

Apprentissage profond pour l'approximation d'une distance d'édition entre graphes

Guillaume Renton*[1], Benoît Gaüzère[1], Pierre Héroux[1], et Sébastien Adam[1]

[1]Normandie Univ, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS, 76000 Rouen, France

5 mars 2019

Résumé

Dans cet article, nous proposons une méthode d'approximation de la distance d'édition entre graphes basée sur l'utilisation de deux techniques récentes d'apprentissage profond : la convolution dilatée et le Spatial Pyramid Pooling (SPP). L'approche proposée prend en entrée la matrice de taille variable utilisée par l'approximation BP-GED [RNB07]. Après une transformation permettant d'introduire une invariance aux permutations dans la matrice, la combinaison des deux techniques d'apprentissage permet d'obtenir un vecteur de caractéristiques de taille fixe, utilisé en entrée de couches denses modélisant un régresseur. L'approche proposée est évaluée sur deux bases de graphes de référence. Les résultats obtenus sont comparés avec ceux de la seule approche qui exploite de l'apprentissage pour approximer la distance d'édition. Les résultats obtenus, bien qu'à confirmer sur un nombre plus important de bases, montrent la supériorité de l'approche proposée.

Mots-clef : Distance d'édition entre graphe, Approximation de distance, Apprentissage profond.

1 Introduction

La plupart des approches d'apprentissage de la littérature prennent en entrée des données numériques, et s'appuient sur les propriétés mathématiques des espaces euclidiens pour construire des modèles. Toutefois, l'utilisation de vecteurs de caractéristiques ne permet pas toujours de représenter de manière satisfaisante certaines informations structurelles, telles que celles présentes dans les molécules par exemple.

Une solution pour représenter ces informations structurelles repose sur l'utilisation de graphes. Néanmoins, l'espace des graphes ne dispose pas d'autant de bonnes propriétés mathématiques que les espaces euclidiens. Notamment l'absence de produit scalaire dans l'espace des graphes rendant impossible l'utilisation directe des méthodes d'apprentissage à l'état de l'art (SVM, réseaux de neurones).

Pour pallier l'absence de produit scalaire, une mesure de dissimilarité entre graphes est nécessaire. La distance d'édition entre graphes est une approche communément acceptée pour répondre à cette problématique.

La distance d'édition entre graphes (GED) est une métrique entre deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, où V_1 (resp. V_2) désigne l'ensemble des nœuds de G_1 (resp. G_2) et E_1 (resp. E_2) désigne l'ensemble des arêtes de G_1 (resp. G_2). Cette mesure est calculée en évaluant le coût de la transformation de G_1 en G_2. Cette transformation est obtenue par une séquence d'opérations d'édition élémentaire pouvant être de 3 types : substitution, insertion et suppression, à la fois sur les nœuds et sur les arêtes. À chacune de ces opérations est associé un coût d'édition quantifiant la transformation effectuée sur le graphe. Il existe une infinité de séquences d'éditions permettant de transformer un graphe en un autre, chacune de ces séquences définissant un chemin d'édition. La distance d'édition est alors définie comme le coût minimal parmi l'ensemble des chemins d'éditions entre les deux graphes. Le chemin d'édition associé à ce coût minimal est appelé chemin d'édition optimal.

L'utilisation de l'algorithme A^* est une des premières approches proposées pour le calcul de la GED [HNR68]. Cette méthode construit et parcourt l'arbre des so-

*Avec le soutien de la région Normandie, principal financeur.

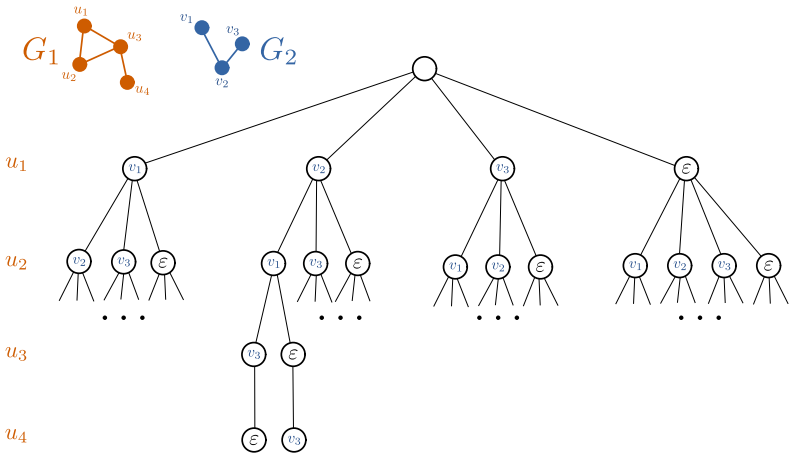


FIGURE 1 – Arbre partiel représentant l'ensemble des chemins d'édition entre G_1 et G_2. L'algorithme A^* permet de parcourir cet arbre afin de trouver un chemin de coût minimal.

lutions afin d'obtenir un chemin d'édition optimal ainsi que le coût correspondant (cf Figure 1). Plus récemment, le calcul de la GED a également été exprimé sous la forme de programme linéaire binaire (BLP), notamment dans [JH06, LAR+15, LAAR+16]. Dans les deux cas, l'obtention d'un chemin optimal étant un problème NP-complet, les méthodes exactes proposées ont une forte complexité. Elle est par exemple de $O(n^m)$ pour A^*, avec n le nombre de nœuds de G_1 et m le nombre de nœuds de G_2, ce qui limite donc le calcul de la GED à de petits graphes [AAGB+17]. Afin de pallier ce problème, des algorithmes de complexité moindre calculant des approximations de la distance d'édition entre graphes ont été proposés. L'une des premières approches est celle proposée dans [RNB07]. Dans cette approche, les auteurs transforment le problème de calcul de la GED en un problème d'affectation de nœuds, résolu par des approches ayant une complexité polynomiales. Il est ainsi possible de traiter des graphes de plus grande taille.

Naturellement, cette amélioration des temps de traitement se fait au détriment de la précision de la valeur obtenue. La problématique devient alors de trouver le compromis entre qualité de l'approximation et temps d'exécution, comme il a été mis en évidence dans [AAGB+17].

Dans cet article, nous proposons une nouvelle méthode d'approximation du calcul de distance d'édition de graphes, utilisant des méthodes récentes d'apprentissage profond afin d'obtenir une précision supérieure tout en maintenant une faible complexité en décision. La section suivante présente les différentes méthodes constituant la base de notre proposition. Ensuite, la section 3 décrit l'approche proposée. Enfin, nous évaluons notre approche dans la section 4 par rapport à l'existant.

2 Approximation de la GED

L'une des premières méthodes à avoir été proposée pour le calcul d'une approximation de la GED est l'appariement de graphe bipartite (BP-GED) [RNB07]. Cette méthode fait le rapprochement entre la distance d'édition entres graphes et un problème de mise en correspondance entre nœuds. La recherche du chemin de coût minimal correspond à un problème d'affectation quadratique, simplifié par [RNB07] en problème d'affectation linéaire afin d'obtenir une approximation.

On cherche à obtenir un appariement optimal, au sens d'un coût minimal d'appariement entre les nœuds. Plus formellement, on cherche la fonction $\varphi : V_1 \to V_2$ qui, au nœud u_k, affecte le nœud $v_{\varphi(k)}$, et qui minimise le coût global de ces affectations. Pour cela, un coût correspondant à l'appariement entre deux nœuds doit être défini. Le coût d'appariement est défini comme étant le coût de transformer un nœud u_i en un nœud v_j. La représentation des suppressions et des insertions de nœuds se fait par l'appariement avec des nœuds vides, appelés ε-nodes. Ainsi, la suppression du nœud u_i correspond à l'appariement entre le nœud u_i et le nœud ε. À l'inverse, l'insertion du nœud v_j correspond à l'appariement entre le nœud ε et le nœud v_j. Il faut donc ajouter m ε-nodes à G_1 et n ε-nodes à G_2. V_1 et V_2 sont ainsi tous deux de taille $n + m$.

Il est important de noter que les nœuds comme les arcs peuvent posséder des attributs, et que le calcul des

coûts doit prendre en compte ces attributs. La coût de substitution de deux nœuds peut ainsi correspondre à la distance euclidienne entre les deux vecteurs de caractéristiques de ces nœuds.

Afin de prendre en compte l'information structurelle présente dans le graphe, le coût minimal d'appariement entre les arêtes incidentes aux deux nœuds est également pris en compte.

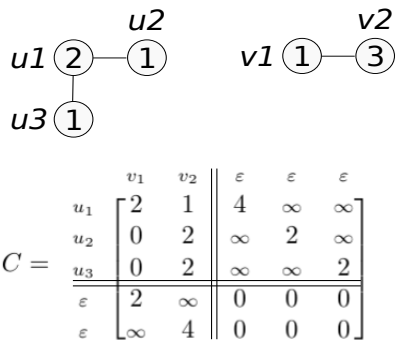
Dans l'équation 1, $u_i, u_k \in V_1$, $v_j, v_{\varphi_k} \in V_2$, a_{ik} correspond à l'arête reliant le nœud u_i au nœud u_k tandis que $b_{j\varphi_k}$ correspond à l'arête reliant le nœud v_j au nœud v_{φ_k}, où φ_k est la fonction d'appariement. Finalement, $\mathscr{S}(n+m)$ correspond à l'ensemble des $(n+m)!$ permutations d'entiers possibles.

$$C(i,j) = c(u_i \to v_j) + \min_{\substack{(\varphi_1,\ldots,\varphi_{n+m}) \\ \in \mathscr{S}(n+m)}} \sum_{k=1}^{n+m} c(a_{ik} \to b_{j\varphi_k}) \tag{1}$$

Finalement, une matrice C est créée, de taille $(n+m) \times (n+m)$ (cf Figure 2), afin de prendre en compte l'ensemble des coûts pour chaque nœud. La taille de la matrice permet d'affecter à chaque nœud u_i un unique nœud v_j, avec u_i et v_j pouvant être des ε-nodes. Afin de s'assurer qu'à chaque nœud ε est associé un unique nœud, le coût entre ce nœud et les autres nœuds ε est fixé à ∞. Finalement, de manière assez naturelle, le coût d'appariement entre deux nœuds ε est fixé à 0, cet appariement n'induisant pas de transformation du graphe. La figure 2 présente un exemple de cette matrice de coût pour deux graphes. On remarque que cette matrice peut être séparée en quatre parties : la partie substitution (en haut à gauche), la partie suppression (en haut à droite), la partie insertion (en bas à gauche) et une zone nulle (en bas à droite).

$$C = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & \varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \varepsilon_4 \end{array} \\ \left[\begin{array}{ccccccc} 1 & 0 & 1 & 3 & \infty & \infty & \infty \\ 1 & 0 & 1 & \infty & 3 & \infty & \infty \\ 2 & 1 & 2 & \infty & \infty & 4 & \infty \\ 0 & 1 & 0 & \infty & \infty & \infty & 2 \\ 2 & \infty & \infty & 0 & 0 & 0 & 0 \\ \infty & 3 & \infty & 0 & 0 & 0 & 0 \\ \infty & \infty & 2 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

Le problème d'affectation des nœuds revient donc à un problème d'appariement linéaire où les coûts d'appariement nœud à nœud sont encodés dans la matrice C



$$C = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \varepsilon \\ \varepsilon \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & \varepsilon & \varepsilon & \varepsilon \end{array} \\ \left[\begin{array}{ccccc} 2 & 1 & 4 & \infty & \infty \\ 0 & 2 & \infty & 2 & \infty \\ 0 & 2 & \infty & \infty & 2 \\ 2 & \infty & 0 & 0 & 0 \\ \infty & 4 & 0 & 0 & 0 \end{array}\right] \end{array}$$

FIGURE 2 – Exemple de matrice de coût pour deux graphes. Les nœuds du graphes sont attribués. Le coût de substituer deux nœuds correspond à la différence entre leurs attributs. Le coût de substituer deux arcs est nul. Les coûts d'insertions/suppressions des nœuds/arcs valent 1.

(Eq. 1). Ce problème peut donc être résolu par l'algorithme Hongrois [Kuh55]. De l'appariement obtenu, il est possible de déduire une séquence d'édition permettant de transformer G_1 en G_2. Étant donnée cette séquence, on peut calculer efficacement le coût associé à cette séquence d'édition. Ce coût correspond à l'approximation faite par BP-GED. Différentes utilisations de l'algorithme Hongrois ont été proposées, que ce soit afin de diminuer les temps de traitements avec des méthodes gloutonnes comme dans [RFB15a], ou avec des méthodes itératives afin d'augmenter la précision, comme dans [BGB16].

Il est important de noter que le chemin obtenu n'est pas forcément optimal, et que le coût n'est ainsi pas forcément minimal. En effet, la construction de la matrice C ne prend en compte qu'une partie limitée de l'information structurelle des graphes, à un voisinage de taille 1. L'appariement et le chemin qui en est déduit ne sont donc optimaux que pour le problème d'appariement linéaire. Le coût associé au chemin obtenu permet de définir une borne supérieure de la distance d'édition.

[RFB14] propose également d'obtenir une borne inférieure, en divisant par deux les coûts des opérations d'éditions sur les arêtes. [RFB15b] utilise cette borne inférieure avec la borne supérieure comme caractéristiques d'entrée d'un SVR afin d'apprendre à prédire une distance d'édition entre graphes plus précise, obtenant des résultats encourageants. Mais cette prédiction est limitée pour deux raisons. Tout d'abord, il est compliqué de prédire une valeur précise à partir de seulement deux caractéristiques. La seconde

limitation vient du fait que l'approximation obtenue par BP-GED est parfois très imprécise.

Bien que ces travaux soient encourageants, il n'y a, à notre connaissance, pas d'autres travaux utilisant l'apprentissage afin d'approximer la GED.

Dans la partie suivante, nous présentons une nouvelle approche permettant d'approximer la distance d'édition via une méthode d'apprentissage profond, en utilisant directement l'information contenue dans la matrice C.

3 Méthode

On peut supposer que l'apprentissage à partir de seulement deux caractéristiques ne permet pas une généralisation satisfaisante. L'objectif de la méthode proposée est de pouvoir extraire directement des caractéristiques à partir de la matrice C. L'utilisation de la matrice C pose deux problèmes lorsque l'on souhaite utiliser des méthodes d'apprentissage sur celle-ci.

Tout d'abord, les graphes sont, par définition, de dimension variable. La matrice C est donc également de taille variable. Cette particularité empêche l'utilisation de la plupart des méthodes d'apprentissage, qui reposent sur l'utilisation de vecteurs de taille fixe.

Le second problème découle du fait qu'il n'existe pas d'ordre d'énumération de l'ensemble des nœuds du graphe. De ce fait, plusieurs matrices C peuvent être générées pour un couple de graphes donné. Chacune de ces matrices correspond à une permutation particulière de l'ordre des nœuds. Dans un tel cas, il faut s'assurer que la prédiction prise soit identique pour un même couple de graphes, et ce selon toutes les permutations possibles.

Pour traiter le premier problème, nous nous inspirons de techniques issues des réseaux convolutifs. Certaines permettent d'extraire des caractéristiques sur des matrices de tailles variables, tandis que d'autres ramènent une matrice de taille quelconque à une matrice de taille fixe, comme le Spatial Pyramid Pooling [HZRS14].

La résolution du second problème est traitée dans la partie suivante.

3.1 Entrée du réseau

L'ordre d'énumération des nœuds d'un graphe n'étant pas fixe, plusieurs ordres peuvent représenter

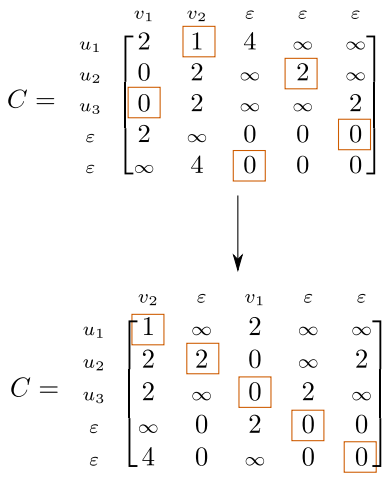


FIGURE 3 – Exemple de permutation de matrice

le même graphe. Ainsi différentes matrices C peuvent être calculée sur une même paire de graphes. Ces matrices C contiennent les mêmes informations, à une permutation près. Afin d'être moins sensible aux permutations, la matrice C est permutée. Cette permutation est réalisée de sorte que les affectations prédites par l'algorithme Hongrois forment la diagonale de la matrice de coûts. La matrice C est permutée selon la matrice de permutation P définie par l'Équation 2. Cette solution ne permet de résoudre que partiellement le problème des permutations, et une méthode plus efficace pourrait être intégré à de prochain travaux.

$$P(i,j) = \begin{cases} 1 & \text{si } \varphi(i) = j \\ 0 & \text{sinon} \end{cases} \tag{2}$$

La Figure 3 présente un exemple de matrice permutée, reprenant l'exemple de la Figure 2. Les matrices ainsi construites et permutées constituent les données utilisées par notre algorithme d'apprentissage utilise pour estimer la GED entre deux graphes. Cette matrice étant de taille variable, des couches de convolution sont utilisées, celles ci n'étant pas restreintes à une taille fixe.

3.2 Réseau convolutif

Les réseaux convolutifs sont basés sur l'utilisation de filtres dont les poids sont appris. Ces filtres sont généralement de taille restreinte (3×3, 5×5) afin de limiter le nombre de paramètres à apprendre, et des

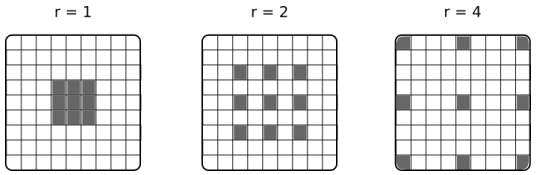


FIGURE 4 – Représentation visuelle des convolutions dilatées.

opérations de *pooling* sont utilisées afin d'augmenter artificiellement le contexte pris en compte par ces filtres. Ce contexte est appelé champ réceptif. Pour cela, dans des fenêtres de petite taille (2×2), le pooling agrège l'ensemble des valeurs de la fenêtre et ne calcule qu'une seule valeur, qui peut être la valeur maximale, minimale, moyenne... La matrice résultante est ainsi une version réduite de la matrice originale, dont le facteur de réduction dépend de la taille des fenêtres utilisées.

La réduction de la taille de la matrice n'est pas la seule solution afin d'augmenter la taille des champs réceptifs. Une autre solution consiste à utiliser les convolutions dilatées, dont l'idée est d'obtenir des filtres creux, plus grand mais gardant le même nombre de paramètres. Un taux de dilatation est utilisé pour cela. Cette méthode est basée sur l'algorithme à trou [HKMMT89] et a été notamment utilisée en segmentation sémantique par [CPK+16]. La Figure 4 illustre le principe des convolutions dilatées.

Par ailleurs, les matrices de coût présentent des tailles très variables, et peuvent atteindre des tailles très faibles. L'utilisation des dilatations semble plus judicieux que le pooling, de façon à ne pas réduire la taille d'une matrice déjà petite.

Finalement, l'architecture utilisée reprend les configuration des premières couches de convolution du réseau VGG-16 [SZ14], cette architecture ayant montré une bonne capacité d'extraction de caractéristiques. Le réseau utilisé possède ainsi 6 couches convolutives, identiques aux 6 première couches de VGG-16, dont la différence vient de l'utilisation de la dilatation plutôt que du pooling. Le réseau possède deux premières couches convolutives de 64 filtres de taille 3×3 et de dilatation 1, suivi de deux couches convolutives de 128 filtres de taille 3×3 et de dilatation 2. Enfin, deux dernières couches sont utilisées, avec 256 filtres de taille 3×3 et de dilatation 4.

L'objectif de nos travaux est dans un premier temps d'évaluer la possibilité d'apprendre des caractéristiques sur la matrice de coût. L'architecture du réseau n'est pas optimisée pour la résolution du problème, ce qui représente toutefois une perspective intéressante.

L'architecture ainsi établie ne modifie pas la taille de la matrice. Si la matrice d'entrée est de taille $(n + m) \times (n + m)$, alors la sortie du réseau convolutif est un tenseur de taille $(n + m) \times (n + m) \times 256$. Afin de réaliser la régression permettant l'approximation de la GED, une représentation matricielle de taille fixe est nécessaire. La solution utilisée ici est le Spatial Pyramid Pooling.

3.3 Spatial Pyramid Pooling

Le principe du Spatial Pyramid Pooling (SPP) est d'appliquer le pooling non pas sur une unique fenêtre de taille fixe mais sur plusieurs fenêtres de tailles adaptables. Pour cela, la matrice d'entrée est divisée en N^2 parts égales. Ainsi, la taille de chaque fenêtre correspond à un ratio de la taille de la matrice. Sur chacune de ces fenêtres, une valeur est sélectionnée. La valeur maximale est généralement sélectionnée, mais l'objectif étant de trouver un coût minimal, la valeur minimale lui est préférée ici.

Cette opération est ensuite réalisée pour plusieurs valeurs de N différentes, afin d'obtenir un certain nombre de caractéristiques par filtre. Les valeurs de N utilisées ici sont 1 et 2, soit 5 caractéristiques par filtre. La dernière couche convolutive ayant 256 filtres, 1280 caractérstiques sont extraites par le Spatial Pyramid Pooling. La Figure 5 présente un exemple d'utilisation du SPP.

La régression est finalement réalisée par des couches dense appliquées aux caractéristiques extraites par le Spatial Pyramid Pooling. Deux couches denses de 21 neurones sont utilisées suivies d'une couche d'un neurone réalisant la régression. Le réseau est ensuite appris en utilisant l'algorithme Adam [KB14]. La figure 6 illustre le processus complet de la méthode ainsi que l'architecture globale du réseau.

Finalement, le réseau complet, incluant convolutions, Spatial Pyramid Pooling et couches denses, comporte 1,172,217 paramètres et a été développé sous Keras [Cho15]. La section suivante présente les différentes expériences réalisées afin d'évaluer le modèle développé.

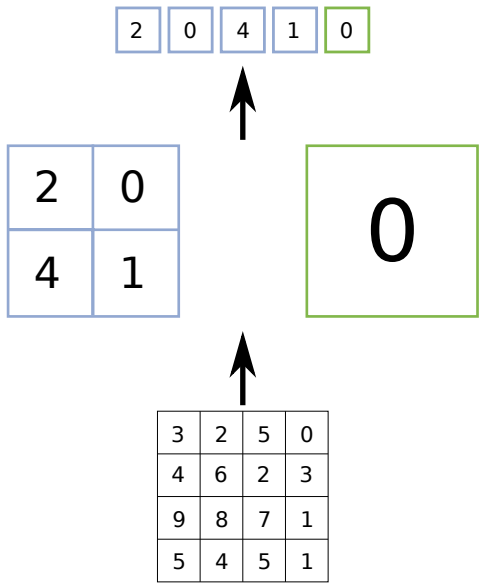


FIGURE 5 – Spatial Pyramid Pooling appliqué à une matrice 4×4 pour des N de valeur 1 et 2.

4 Expériences

Cette section présente les différentes expériences réalisées afin d'évaluer les performances de la méthode proposée. Le protocole expérimental ainsi que les métriques sont tout d'abord présentés. Les résultats obtenus sur deux bases de données de graphes (Letter et Fingerprint [RB08]) sont ensuite présentés.Ces résultats sont comparés à ceux obtenus avec la BP-GED [RNB07] et les SVR [RFB15b], méthode proposant également une approche basée sur l'apprentissage.

4.1 Protocole expérimental et métriques

Le protocole expérimental est le suivant. 1000 graphes sont extraits de la base initiale. Pour chaque paire de graphes, la distance d'édition exacte est calculée, en utilisant l'algorithme A^*. La matrice de coût est également calculée puis permutée, selon la stratégie définie en 3.1. L'ensemble des 5.10^5 paires de graphes est finalement découpé en 3 bases : une base d'apprentissage, comprenant 40% des données, une base de validation avec 10% des données, et les 50% restant en test. Les données de validation sont utilisées afin de s'assurer que notre modèle ne sur-apprend pas lors de l'apprentissage.

Deux métriques de la littérature sont utilisées pour évaluer les résultats : l'erreur moyenne relative (MRE), exprimée en pourcentage dans l'équation 3, ainsi que l'erreur moyenne quadratique (MSE), définie par l'équation 4. Dans les deux équations, d_{a_i} correspond à la distance approximée du couple de graphes i et d_{e_i} à la distance exacte du couple de graphes i.

$$MRE = 100 * \frac{1}{N} \sum_{i=1}^{N} \frac{|d_{a_i} - d_{e_i}|}{d_{e_i}} \tag{3}$$

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (d_{a_i} - d_{e_i})^2 \tag{4}$$

4.2 Letter

Letter [RB08] est une base de graphes représentant des lettres majuscules formées de segments de droite. La figure 7 présente plusieurs exemples de lettres déformées. Chaque nœud représente une extrémité de la lettre, ayant comme attributs les coordonnées x et y de ce point. Les arêtes, quant à elles, représentent l'existence ou non d'un segment entre ces points. Les arêtes ne possèdent pas d'attributs. Différentes distorsions sont appliquées à chacune des lettres.

Le coût de substitution de deux nœuds est calculé selon la distance euclidienne des coordonnées de chaque nœud, tandis que les coûts de substitution entre deux arêtes sont nuls. Les coûts de suppression et d'insertion sont fixés à 0.9 pour les nœuds et à 1.7 pour les arêtes, selon les coûts proposés dans [RB12].

Le tableau 1 présente les résultats obtenus en erreur relative moyenne (MRE) et en erreur quadratique moyenne (MSE) pour les 3 méthodes : BP-GED, SVR ainsi que la méthode proposée. La Figure 8 présente ces résultats graphiquement. Pour une meilleure lisibilité, le log_{10} des résultats est utilisé.

	BP-GED	SVR	Approche proposée
MRE	13.11	7.76	4.2
MSE	3.07	0.86	0.32

TABLE 1 – Comparaison de BP-GED, SVR et notre méthode sur LETTER

Comme on peut le voir, l'approche proposée est nettement meilleure, que ce soit en erreur relative ou en erreur quadratique. L'utilisation de la matrice C apporte ainsi une information supplémentaire, permettant une meilleure précision.

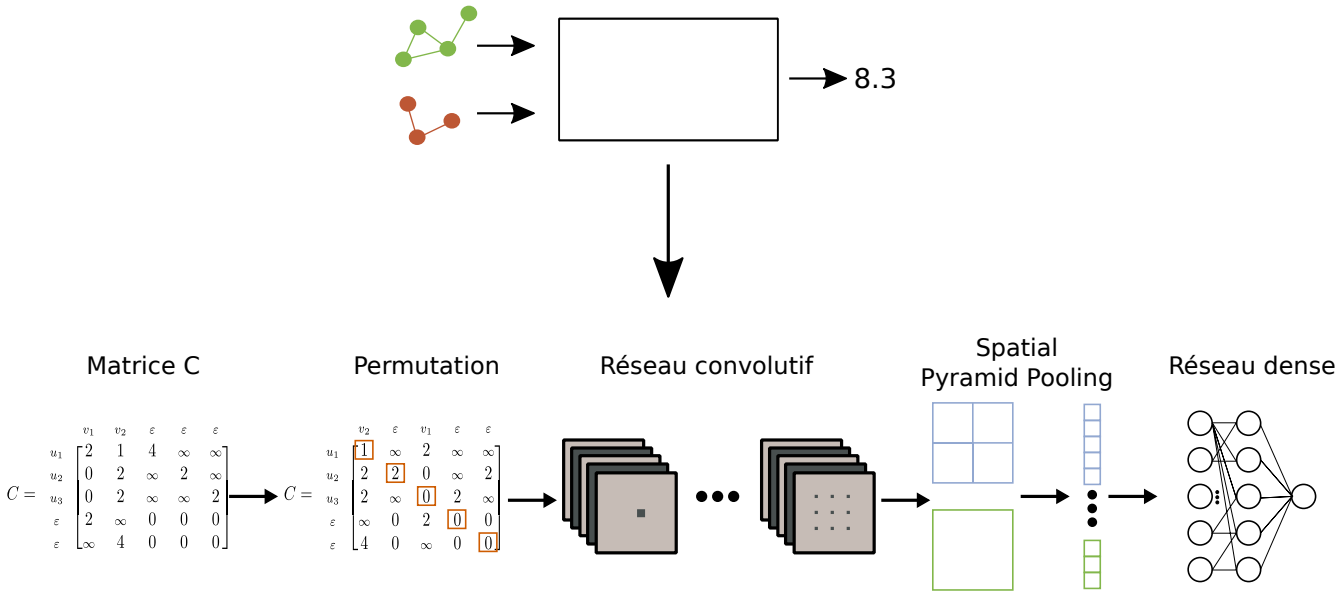


FIGURE 6 – Processus final de la méthode.

FIGURE 7 – Plusieurs déformations de la lettre A.

FIGURE 9 – Empreinte digitale utilisée pour générer la base Fingerprint

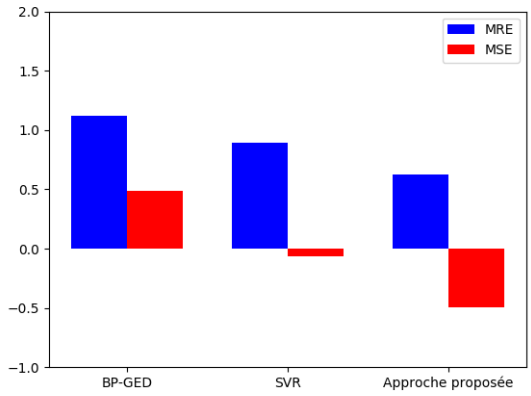


FIGURE 8 – Résultats obtenus par les méthodes BP-GED, SVR et notre méthode sur la base LETTER en MRE et MSE. Le log_{10} des résultats est utilisé pour une meilleure lisibilité.

4.3 Fingerprint

Fingerprint [RB08] est une base de données de graphes représentant des empreintes digitales (cf Figure 9). Elle a été générée en réalisant une squelettisation d'images d'empreintes digitales. Chaque bifurcation et extrémité du squelette obtenu correspond à un nœud, et les arêtes représentent la présence ou non d'un lien entre chaque nœud.

Les nœuds ne possèdent pas d'attribut, et la distance entre deux nœuds dépend donc uniquement de leurs arêtes. Les arêtes sont caractérisées par leur angle. Ainsi, le coût de substitution de deux arêtes dépend de la distance entre leurs angles, à 2π près. Les coûts d'insertion et de substitution sont fixés à 0.525 pour les nœuds et 0.125 pour les arêtes, toujours selon [RB12].

	BP GED	SVR	Approche proposée
MRE	68.39	7.56	1.15
MSE	16.15	0.10	0.006

TABLE 2 – Comparaison de BP-GED, SVR et notre méthode sur FINGERPRINT

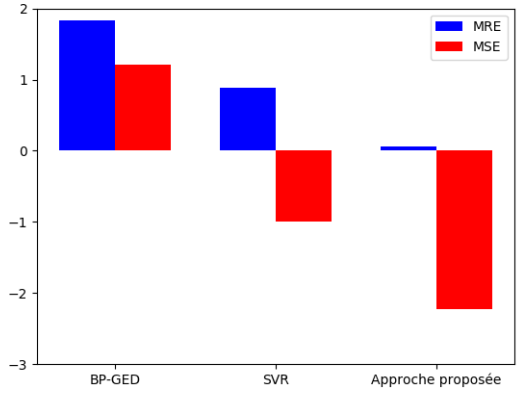


FIGURE 10 – Résultats obtenus par les méthodes BP-GED, SVR et notre méthode sur la base FINGERPRINT en MRE et MSE. Le log_{10} des résultats est utilisé pour une meilleure lisibilité.

Le tableau 2 présente les résultats obtenus en erreur relative moyenne (MRE) et en erreur quadratique moyenne (MSE) pour les 3 méthodes : BP-GED, SVR ainsi que la méthode proposée. La figure 10 illustre graphiquement ces résultats. Pour une meilleure lisibilité, le log_{10} des résultats est utilisé.

On observe que les deux méthodes d'apprentissage permettent une nette amélioration de la précision par rapport à la BP-GED sur cette base. Cela peut s'expliquer par le fait que l'information principale est contenue dans les arêtes, ce que BP-GED prend difficilement en compte. L'approche proposée obtient également des résultats nettement meilleurs que les SVR. On retrouve ainsi l'idée que l'information contenue dans C est plus pertinente que les simples bornes inférieures et supérieures.

5 Conclusion

Dans ce papier, une nouvelle approche d'approximation de la distance d'édition entre graphes est proposée. Celle-ci est basée sur les réseaux convolutifs ainsi que sur le Spatial Pyramid Pooling. La combinaison de ces deux méthodes permet l'extraction de caractéristiques à partir de la matrice de coût, malgré sa taille variable.

L'approche est évaluée sur deux bases de données différentes. Elle présente de bonnes performances sur les deux bases, avec des résultats meilleurs que ceux obtenus par la seule méthode basée sur un apprentissage. Ces résultats pourraient être encore améliorés, notamment en cherchant à optimiser le réseau utilisé. Par ailleurs, la méthode proposée ne permet pas d'obtenir un chemin d'édition qui peut être important selon le domaine d'application. Nos futurs travaux viseront à pallier ce défaut.

Références

[AAGB+17] Zeina Abu-Aisheh, Benoit Gaüzere, Sébastien Bougleux, Jean-Yves Ramel, Luc Brun, Romain Raveaux, Pierre Héroux, and Sébastien Adam. Graph edit distance contest : Results and future challenges. *Pattern Recognition Letters*, 100 :96–103, 2017.

[BGB16] Sébastien Bougleux, Benoit Gaüzère, and Luc Brun. Graph edit distance as a quadratic program. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 1701–1706. IEEE, 2016.

[Cho15] F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[CPK+16] LC. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A.L. Yuille. Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint :1606.00915*, 2016.

[HKMMT89] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1989.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[HZRS14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[JH06] Derek Justice and Alfred Hero. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8) :1200–1214, 2006.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2) :83–97, 1955.

[LAAR+16] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam. Exact graph edit distance computation using a binary linear program. In Antonio Robles-Kelly, Marco Loog, Battista Biggio, Francisco Escolano, and Richard Wilson, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 485–495, Cham, 2016. Springer International Publishing.

[LAR+15] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam. Graph edit distance : a new binary linear programming formulation. *CoRR*, abs/1505.05740, 2015.

[RB08] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In Niels da Vitoria Lobo, Takis Kasparis, Fabio Roli, James T. Kwok, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Marco Loog, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[RB12] Kaspar Riesen and Horst Bunke. Classification and clustering of vector space embedded graphs. In *Emerging Topics in Computer Vision and Its Applications*, pages 49–70. World Scientific, 2012.

[RFB14] Kaspar Riesen, Andreas Fischer, and Horst Bunke. Computing upper and lower bounds of graph edit distance in cubic time. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 129–140. Springer, 2014.

[RFB15a] Kaspar Riesen, Miquel Ferrer, and Horst Bunke. Approximate graph edit distance in quadratic time. *IEEE/ACM transactions on computational biology and bioinformatics*, 2015.

[RFB15b] Kaspar Riesen, Andreas Fischer, and Horst Bunke. Estimating graph edit distance using lower and upper bounds of bipartite approximations. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(02) :1550011, 2015.

[RNB07] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. Bipartite graph matching for computing the edit distance of graphs. In Francisco Escolano and Mario Vento, editors, *Graph-Based Representations in Pattern Recognition*, pages 1–12, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.