KGIC : Intégration de graphe de connaissances pour la classification d'images

Franck Anaël Mbiaya[*,**], Christel Vrain[*], Frédéric Ros[**], Thi-Bich-Hanh Dao[*]
Yves Lucas[**]

[*] Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, 45067, Orléans, France
[**] Univ. Orléans, PRISME EA 4229, 45067, Orléans, France
prenom.nom@univ-orleans.fr

Résumé. Nous présentons une méthode d'apprentissage profond pour la classification supervisée d'images, intégrant des connaissances sous forme de graphe. A ces fins, nous introduisons une fonction de coût combinant à la fois une mesure traditionnellement utilisée en apprentissage profond (entropie croisée) et une mesure originale qui prend en compte la représentation des nœuds après plongement du graphe de connaissances. Les connaissances ne sont utilisées que pendant la phase d'apprentissage et ne sont pas nécessaires pour l'évaluation d'un exemple en mode test. Les expérimentations sur plusieurs bases d'images démontrent l'amélioration en performances de notre méthode par rapport à l'état de l'art : d'une part en comparaison avec des algorithmes classiques d'apprentissage profond et d'autre part avec un algorithme très récent basé aussi sur la connaissance issue d'un graphe.

1 Introduction

Ces dernières années, la classification d'images par apprentissage profond a connu un immense succès. Pour améliorer les performances, de nombreux travaux (Khan et al., 2020) ont porté sur le développement de nouvelles architectures. Cependant, ces architectures ont montré leurs limites sur des problèmes plus complexes, parmi lesquels la classification à grains fins (Wei et al., 2021) qui requiert une analyse plus fine des images. Un exemple des jeux de données les plus utilisés pour la classification à grains fins est Caltech-UCSD-2011 (Wah et al., 2011), base de données d'images d'oiseaux regroupés en 200 catégories.

Pour faire face à ces difficultés, de nombreuses solutions ont été proposées, dont l'utilisation de la connaissance a priori pour améliorer le pouvoir de généralisation des architectures profondes (von R. et al., 2021). L'intégration de cette connaissance dans des architectures profondes dépend de la façon dont celle-ci est formalisée (graphe de connaissances, équations, règles logiques ...). Une manière classique consiste à la considérer comme une nouvelle modalité de données (Yang et al., 2019). Dans les architectures profondes, l'intégration se fait généralement à travers la définition de l'architecture neuronale (Lu et al., 2017). Elle peut aussi être utilisée dans la fonction de perte (Xu et al., 2018) ou pendant la phase de test (Glavaš et Vulić, 2018).

Nous proposons une nouvelle méthode KGIC (Knowledge Graph for Image Classification) pour la classification d'images qui se fonde non seulement sur les images mais aussi sur des connaissances complémentaires formalisées sous forme de graphe pondéré. Ce graphe contient un nœud pour chaque image et chaque classe du jeu de données, mais aussi des nœuds représentant des propriétés ; les arêtes relient les images et les classes à leurs propriétés. Un plongement de ce graphe de connaissance permet d'apprendre une nouvelle représentation des images, et nous définissons une fonction de coût sur cette nouvelle représentation. La connaissance est utilisée uniquement pendant la phase d'apprentissage : la fonction de coût permet de guider l'apprentissage des poids du réseau profond. Ainsi que nous le verrons dans les expérimentations, notons que cette fonction de coût peut être intégrée dans diverses architectures.

Nos contributions sont les suivantes :

— intégration de connaissances sous forme de graphe en phase d'apprentissage
— définition d'une nouvelle fonction de perte se fondant sur un plongement du graphe de connaissance
— expérimentations sur 2 jeux de données (Caltech et SUN Attributes)

Cet article est organisé comme suit. La section 2 présente diverses approches d'intégration de la connaissance dans des architectures profondes. La section 3 est dédiée à la présentation de notre méthode tandis que la section 4 présente nos résultats expérimentaux.

2 État de l'art

L'apprentissage profond (Alzubaidi et al., 2021) domine les méthodes d'apprentissage en raison de leur très haute performance dans les tâches de prédiction et de classification. Cependant les modèles actuels restent encore des "boîtes noires", nécessitent beaucoup de données étiquetées, et leur efficience est réduite sur des problèmes plus fins de classification. L'intégration de la connaissance dans les architectures profondes est une voie émergente et prometteuse (von R. et al., 2021). Elle peut se faire à travers des données d'entraînement, au niveau de l'ensemble d'hypothèses, durant l'apprentissage ou encore pendant la phase d'inférence. Son intégration dépend de la façon dont celle-ci est représentée.

Concernant l'intégration de la connaissance dans les données d'entraînement, la connaissance est vue ici comme une source d'informations complémentaire et distincte des données d'apprentissage. La représentation peut se faire à l'aide d'équations algébriques (Yang et al., 2019), en créant des nouvelles données à partir de simulations (Pfrommer et al., 2018). L'utilisation de la connaissance en mode test cependant constitue une limite pour cette stratégie.

La connaissance peut être utilisée pour construire l'ensemble d'hypothèses : à travers la définition de l'architecture neuronale, via les hyperparamètres du modèle (Lu et al., 2017) ou encore en utilisant des neurones symboliques (Bach et al., 2017), qui utilisent des règles comme base de la structure du modèle.

La connaissance est classiquement intégrée par des techniques de régularisation, comme par exemple l'intégration de règles logiques dans une fonction de perte sémantique (Xu et al., 2018; Diligenti et al., 2017) ou un terme de régularisation basé sur la matrice laplacienne d'un graphe (Ma et Zhang, 2018). Enfin, d'autres travaux visent à vérifier la cohérence de la prédiction du modèle avec des contraintes issues de la connaissance, comme par exemple Fang et al. (2017) ou Glavaš et Vulić (2018).

Si les travaux de l'état de l'art sont prometteurs, ils restent encore souvent ad'hoc. Notre proposition basée sur les graphes de connaissances et des techniques de régularisation vise à être plus générique et ne requiert pas la connaissance en mode test.

Nous appliquons notre méthode à la classification d'images à grains fins (Wei et al., 2021). Il s'agit d'un problème complexe dont l'objectif est de discriminer les classes d'images ayant de grandes variations intra-classes et de petites variations inter-classes. Les méthodes développées peuvent être regroupées en trois familles : l'extraction de caractéristiques fines pendant l'apprentissage (Sun et al., 2020), la localisation de l'objet dans l'image (Liu et al., 2020), et l'utilisation de la connaissance. Le seul travail utilisant un graphe de connaissances est Chen et al. (2018) et notre proposition diffère dans la construction du graphe de connaissances, dans son utilisation dans la fonction de perte, et dans son utilisation seulement dans la phase d'apprentissage.

3 Méthode proposée

Pendant la phase d'apprentissage d'un réseau neuronal convolutif, les couches convolutives sont censées capter les informations visuelles dans les images pour discriminer les différentes classes. Cependant, la similarité entre certaines classes est souvent très élevée, et il est alors difficile pour le modèle de les différencier. Cette difficulté est d'autant plus importante dans les problèmes de classification à grains fins. La méthode KGIC (Figure 1) que nous proposons utilise un graphe de connaissances pour améliorer les performances d'un réseau neuronal convolutif pour la classification d'images. Tel que décrit dans l'algorithme 1, notre méthode nécessite (1) de calculer un plongement des différents nœuds du graphe de connaissances associé au jeu de données d'entraînement, (2) de calculer une fonction de perte, prenant en compte la similarité entre la représentation (par plongement) d'une image et des classes. Notre méthode a deux avantages principaux : d'une part, elle peut être utilisée dans toute architecture profonde, et d'autre part, elle ne nécessite pas de disposer de connaissances sur les images test, puisque le plongement n'est utilisé que pendant la phase d'apprentissage.

3.1 Graphe de connaissances

Pour être applicable, notre méthode nécessite de disposer d'un graphe contenant un nœud par instance d'image, un nœud par classe et des nœuds représentant des propriétés binaires. Les arcs peuvent être pondérés par un facteur de certitude.

Pour les jeux de données utilisées dans les expérimentations, de tels graphes n'étaient pas disponibles et ils ont été engendrés à partir de descriptions attribut-valeur des images (décrivant des concepts visuels, représentation sémantique du contenu des images) : un nœud par instance d'image, un nœud par classe et un nœud par couple attribut-valeur.

Formellement, dans la suite $G = (V, E)$ où $v_i \in V$ est un nœud (représentant une image, une classe ou une propriété) et $e_{i,j} \in E$ est une arête reliant v_i et v_j. Chaque arête $e_{i,j}$ est associée à un poids $s_{i,j}$ qui correspond à la certitude avec laquelle l'image (ou les images de la classe) i possède l'attribut correspondant à j. Pour une classe, la pondération est calculée en moyennant la pondération de cette classe. Si $s_{i,j} = 0$, les nœuds i et j ne sont pas connectés par une arête. Le graphe peut être enrichi avec des connaissances, comme une hiérarchie sur les classes.

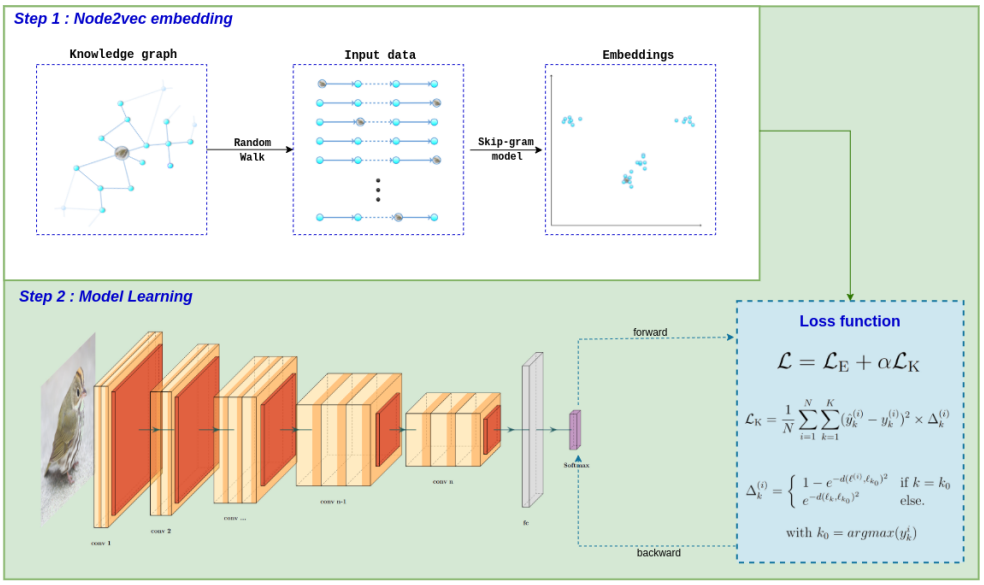


FIG. 1 – Vue générale de KGIC : un plongement (Node2Vec) prend en entrée le graphe de connaissances et calcule une nouvelle représentation des nœuds en fonction de leurs voisinages. Cette représentation est utilisée dans la fonction de perte lors de l'apprentissage.

La figure 2 donne un exemple de graphe de connaissances pour Caltech (Wah et al., 2011). Le graphe a été enrichi en regroupant les différentes classes (oiseaux) en genre, famille et ordre.

3.2 Plongement du graphe de connaissances

Le plongement du graphe de connaissances consiste à calculer une représentation de ses nœuds dans un espace de dimension $d \ll |V|$. Nous utilisons Node2vec (Grover et Leskovec, 2016) qui se décompose en deux étapes : une marche aléatoire pour générer le voisinage de chaque nœud et un algorithme de type Skip-gram pour apprendre les nouvelles représentations.

La marche aléatoire intègre un facteur de biais α pour réévaluer les poids en fonction de l'état du nœud précédent. Formellement, étant donné un nœud source u, et en considérant c_i comme le $i^{\text{ième}}$ nœud lors d'une marche composée de l étapes ($i \leq l$), la probabilité d'aller à un nœud est :

$$\mathcal{P}(c_i = v | c_{i-1} = t) = \begin{cases} \alpha_{pq} \times w_{tv} & \text{si } (t,v) \in E \\ 0 & \text{sinon} \end{cases} \quad (1)$$

Ici w_{tv} est le poids de l'arête $\{t, v\}$, et α_{pq} est le biais de recherche, dépendant de 2 paramètres p et q, qui permet de restreindre la transition entre les nœuds selon que l'on privilégie une proximité de premier ou de second ordre.

Algorithm 1 : KGIC

Input : X : images, Y : labels, G : Graphe de connaissances, d p q : paramètres pour le plongement, $f(\bullet)$: Réseau neuronal profond, λ : taux d'apprentissage, α : Paramètre du modèle

Output : Θ

\# Plongement du graphe de connaissances
$D \leftarrow Marche\ aleatoire(G, p, q)$ \# Eq. 1
$L \leftarrow Skip - gram(D, d)$

\# Apprentissage du réseau neuronal
$\Theta \leftarrow$ Itialisation des poids de $f(\bullet)$
for $epoque \in [1, num\_epoques]$ **do**
  for $batch \in [1, num\_batches]$ **do**
    $\hat{Y} = f(X, \Theta)$
    $\mathcal{L}_{CE} = Entropie\ croisee(Y, \hat{Y})$ \# Eq. 5
    $\mathcal{L}_K = Perte\ connaissance(Y, \hat{Y}, L)$ \# Eq. 2
    $\mathcal{L}_{batch} = \mathcal{L}_{CE} + \alpha\mathcal{L}_K$ \# Eq. 4
    $\Theta = \Theta - \lambda\partial\mathcal{L}_{batch}$ \# Apprentissage des paramètres
  end for
end for

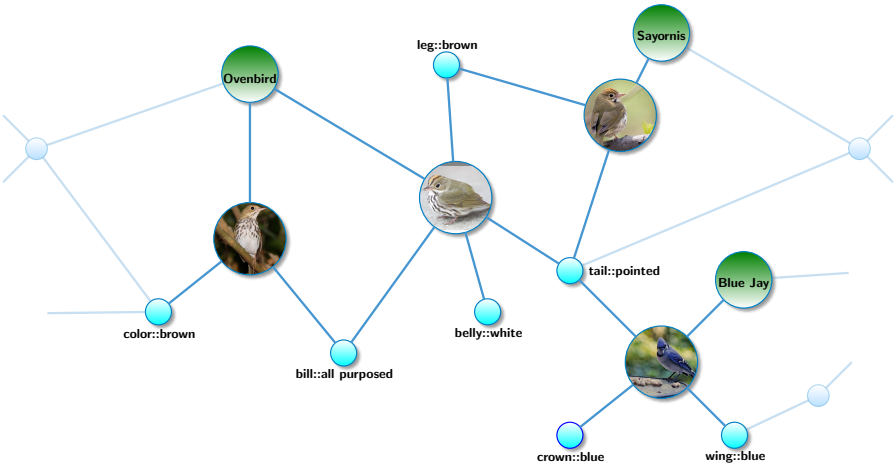
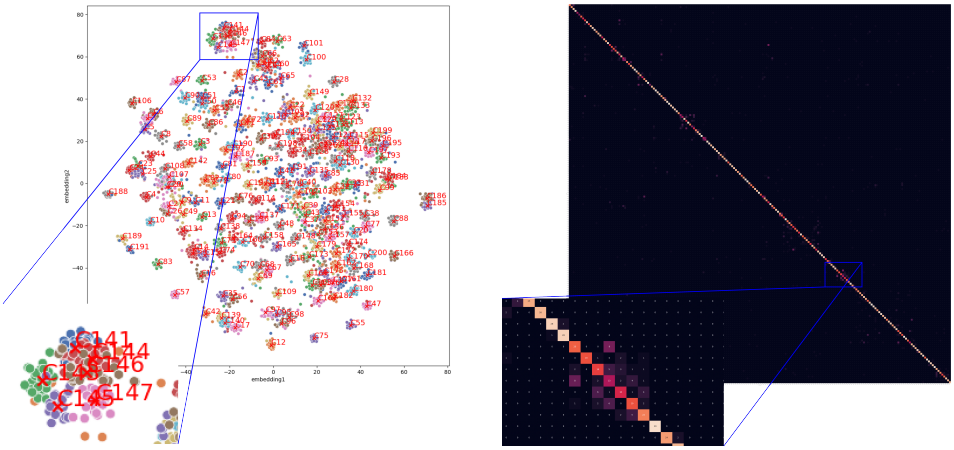


FIG. 2 – Exemple de graphe de connaissances créé à partir du jeu de données Caltech.



(a) (b)

FIG. 3 – (a) Projection avec t-sne des images d'entraînement et des classes obtenues après plongement du jeu de données Caltech. (b) Matrice de confusion de Caltech après un apprentissage sur ResNet-50. Comme on peut le voir, les classes avec des concepts proches dans le graphe de connaissances sont généralement confondues par le modèle.

Une fois la marche aléatoire terminée, l'algorithme Skip-gram formule le plongement comme un problème d'optimisation du maximum de vraisemblance (voir (Grover et Leskovec, 2016)).

Dans notre méthode, l'idée est de conserver les informations sur la structure locale de chaque nœud (en choisissant $p = 0.1$ et $q = 1$) de manière à ce que les nœuds d'image et de classe qui ont des propriétés similaires aient des représentations très proches dans le nouvel espace. La figure 3a montre la projection des images d'entraînement et des classes après plongement de l'ensemble de données Caltech-UCSD-2011. Comme on peut le voir, les images appartenant à une même classe ont généralement des représentations similaires. Il en est de même de la représentation du nœud de leur classe. La figure 3b montre la matrice de confusion obtenue sur ce jeu de données après apprentissage avec un réseau ResNet-50 (sans utilisation de la connaissance). La partie zoomée représente la matrice pour les classes de 141 à 147. Nous pouvons remarquer que les classes qui sont confondues par le modèle appris ont aussi des représentations proches dans l'espace de plongement.

3.3 Apprentissage du réseau

Après avoir obtenu les nouvelles représentations des images et des classes par plongement, notre jeu de données devient alors $\mathcal{D} = \{(x^{(1)}, \ell^{(1)}, y^{(1)}), ..., (x^{(N)}, \ell^{(N)}, y^{(N)})\}$ où $x^{(i)}$, $\ell^{(i)}$, $y^{(i)}$ représentent respectivement la $i^{\text{ième}}$ image, son plongement dans l'espace des connaissances et son label, représenté par un vecteur ("one-hot representation") : nous utilisons la notation $y_k^{(i)}$ avec $y_k^{(i)} = 1$ si $x^{(i)}$ est de classe k, 0 sinon.

Dans une architecture neuronale convolutive, les couches de convolution capturent les informations spatiales pour discriminer les différentes classes. Cependant, le modèle peine à

trouver la frontière entre des classes visuellement similaires. Un exemple est présenté dans la figure 3 où les classes 141 à 147 sont visuellement similaires (Fig 3a).

Perte liée à la connaissance. Les connaissances complémentaires permettent d'apporter des informations au modèle pendant le processus d'apprentissage sur la proximité entre une image et les classes d'une part et la proximité entre les classes d'autre part. Le modèle va alors modifier ses poids, en tenant compte de la complexité à distinguer certaines classes des autres. Plus précisément, nous introduisons une nouvelle fonction de perte, formulée par :

$$\mathcal{L}_K = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}(\hat{y}_k^{(i)} - y_k^{(i)})^2 \times \Delta_k^{(i)} \tag{2}$$

où $\hat{y}_k^{(i)} \in [0,1]$ est la prédiction du modèle et $\Delta_k^{(i)}$ est une pondération d'autant plus grande que la discrimination est difficile dans l'espace de plongement. Nous utilisons l'erreur quadratique moyenne $(\hat{y}_k^{(i)} - y_k^{(i)})^2$ pour assurer la dérivabilité de $\mathcal{L}_K$. Si k est une classe, $\Delta_k^{(i)}$ est formulée par :

$$\Delta_k^{(i)} = \begin{cases} 1 - e^{-d(\ell^{(i)}, \ell_{y^{(i)}})^2} & \text{si } k \text{ classe attendue de } x^{(i)} \\ e^{-d(\ell_k, \ell_{y^{(i)}})^2} & \text{sinon.} \end{cases} \tag{3}$$

où $\ell_{y^{(i)}}$, $\ell^{(i)}$ et ℓ_k sont respectivement les représentations en dimension d de la classe attendue, de la $i^{\text{ième}}$ image, et de la classe k dans le plongement du graphe de connaissances. $d(\bullet)$ représente la distance euclidienne. L'utilisation de l'exponentiel permet de normaliser la distance dans l'intervalle $[0,1]$. Lorsque la $k^{\text{ième}}$ classe correspond à la classe attendue par le modèle ($k = \arg\max(y^{(i)})$), on mesure la dissimilarité entre l'image et la classe attendue : $\Delta_k^{(i)}$ est d'autant plus grand que le point est éloigné du représentant du nœud de sa classe ($1 - e^{-d(\ell^{(i)}, \ell_{y^{(i)}})^2}$ croît lorsque $d(\ell^{(i)}, \ell_{y^{(i)}})$ croît). Ceci permet de mettre plus de poids dans les cas où l'image est éloignée de sa classe dans l'espace de plongement. Par contre, lorsque la $k^{\text{ième}}$ classe ne correspond pas à la classe attendue ($k \neq \arg\max(y^{(i)})$), on mesure la similarité entre la classe k et la classe attendue : $\Delta_k^{(i)}$ est d'autant plus petit que la représentation de la classe k est éloignée de la représentation de la classe attendue ($e^{-d(\ell_k, \ell_{y^{(i)}})^2}$ décroît lorsque $d(\ell_k, \ell_{y^{(i)}})$ croît). Ceci permet de mettre plus de poids lorsque les classes k et $y^{(i)}$ sont difficiles à discriminer dans l'espace de plongement.

Perte finale. En résumé, notre modèle est entraîné avec la somme pondérée de deux pertes : l'entropie croisée $\mathcal{L}_{CE}(y, \hat{y})$ entre la classe prédite $\hat{y}$ et la classe attendue y et la perte liée à la connaissance définie plus haut :

$$\mathcal{L} = \mathcal{L}_{CE}(y, \hat{y}) + \alpha\, \mathcal{L}_K(y, \hat{y}, \ell) \tag{4}$$

où α est un paramètre du modèle. L'entropie croisée est définie par :

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}(y_k^{(i)}\, log(\hat{y}_k^{(i)}) + (1 - y_k^{(i)})\, log(1 - \hat{y}_k^{(i)})) \tag{5}$$

4 Expérimentations

Notre objectif est de montrer expérimentalement que l'intégration de connaissances pour la classification d'images permet d'améliorer les résultats, indépendamment des architectures profondes choisies. Nous montrons aussi que notre méthode est comparable ou supérieure à celles de l'état de l'art, notamment à Chen et al. (2018) qui intègre aussi des connaissances.

4.1 Cadre expérimental

Jeux de données. Notre méthode est évaluée sur deux jeux de données de classification à grains fins Caltech-UCSD-2011 (Wah et al., 2011) et SUN Attribute (P. et H., 2012).

Caltech-UCSD-2011 (Caltech) se compose de 200 catégories d'oiseaux, avec 5994 images pour l'entraînement et 5794 images pour les tests. Chaque image est également annotée avec 312 attributs visuels, associés à une valeur de certitude entre 1 et 4, représentant un niveau de confiance des annotateurs.

SUN Attribute dataset est un jeu de données de scène contenant 707 classes et 14340 images. Chaque classe contient environ 20 images. Chaque image est décrite par 102 attributs décrivant le contenu de la scène. Un attribut représente la présence ou non d'un objet dans la scène et est associé à une valeur de certitude comprise entre 0 et 1, donnée par l'annotateur. Nous avons créé trois sous-ensembles de données : *SUN-In* (scènes d'intérieur, 78 classes), *SUN-Out* (scènes extérieures, 93 classes) et *SUN-InOut* (scènes mélangées, 171 classes).

Cadres expérimentaux Pour la comparaison avec l'état de l'art, nous avons redimensionné les images d'entrées à 448×448. Pour l'évaluation de la contribution de la connaissance, les images d'entrée ont été redimensionnées à 299×299. Toutes les architectures utilisées ont été pré-entraînées sur ImagetNet (Deng et al., 2009) avec 1000 classes. Pour chaque modèle pré-entraîné, la dernière couche du réseau a été modifiée pour qu'elle corresponde au nombre de classes apprises. Pour les images, des retournements horizontaux aléatoires ont été appliqués pour augmenter le nombre d'images. Concernant le plongement de graphe de connaissance (Node2vec) les paramètres ont été fixés à $p = 0,1$ et $q = 1$ pour la marche aléatoire et les nœuds ont été projetés dans un espace de dimension $d = 128$.

Nous avons utilisé l'optimiseur SGD avec un taux d'apprentissage initialisé à 10^{-4} pour les poids pré-entraînés et à 10^{-3} pour les poids initialisés de manière aléatoire (dernière couche du réseau). Le taux d'apprentissage est diminué d'un facteur de 10 toutes les 20 époques. Nous avons utilisé 80 époques pour tous les modèles et une taille de lots de 16. Le paramètre α dans Eq 4 est fixé à 5. Concernant TransFG (He et al., 2022), le modèle a été paramétré tel que décrit par les auteurs.

Pour Caltech, nous avons utilisé les images d'entraînement et de test d'origine. Pour les autres ensembles de données, nous avons réalisé une validation croisée à 5 plis. Le graphe de connaissances est créé uniquement à partir des données d'apprentissage.

Toutes les expériences ont été effectuées sur un GPU NVIDIA Quadro RTX 6000 avec 20 Go de mémoire en utilisant pycharm.

Méthode	Backbone	Taux succès(%)
KERL* (Chen et al., 2018)	VGG-16	86.3
NTS-Net (Yang et al., 2018)	ResNet-50	86.8
DCL (Chen et al., 2019)	ResNet-50	86.8
PMG (Du et al., 2020)	ResNet-50	89.6
ViT-B_16		89.8
TransFG (He et al., 2022)	ViT-B_16	91.3
KGIC-ViT-B	ViT-B_16	90.6
KGIC-TransFG	TransFG	**91.7**

TAB. 1 – Evaluation de KGIC sur le jeu de données Caltech (* résultats publiés).

4.2 Résultats et analyses

Comparaison avec l'état de l'art. Le Tableau 1 présente les performances de notre méthode sur Caltech. Nous n'avons pu exécuter l'algorithme KERL (Chen et al., 2018) (code source non disponible) alors qu'il est le seul à utiliser la connaissance de Caltech, et nous reportons les résultats publiés (Chen et al., 2018). Notre méthode avec une architecture ViT-B16 améliore de manière significative les résultats obtenus avec des architectures de l'état de l'art, hors TransFG (He et al., 2022). Couplée avec TransFG les résultats sont légèrement améliorés (+0.4%).

Méthode	taux de succès (%)		
	SUN-In	SUN-Out	SUN-InOut
TransFG (He et al., 2022)	77.8	73.7	72.3
KGIC-TransFG	**79.2**	**74.7**	**73.2**

TAB. 2 – Evaluation de KGIC sur les jeux de données SUN.

Le Tableau 2 présente les performances de notre méthode et de TransFG (He et al., 2022) sur les jeux de données SUN. Notre méthode permet d'améliorer légèrement les performances. Ce faible gain peut peut-être s'expliquer par le fait que les connaissances seules (sans les images) sont peu discriminantes, comme montré dans le Tableau 3 (taux de succès inférieur à 40% avec un SVM uniquement sur les attributs).

Données	Caltech	SUN-In	SUN-Out	SUN-InOut
taux succès - SVM (%)	46.9	38.3	29.4	32.2
taux succès - AD (%)	23.6	21.7	15.7	14.8

TAB. 3 – Taux de succès avec un modèle SVM et un arbre de décision (AD) sur la description attribut-valeur sans utiliser les images

Contribution de la connaissance. Comme on peut le voir dans le Tableau 4, l'intégration de connaissances dans des architectures classiques permet d'améliorer les performances des modèles de 1% à 4%. Les meilleurs gains sont obtenus sur le jeu Caltech, car la connaissance associée est plus discriminante (voir Tableau 3). Le meilleur gain de performance obtenu est de 4.61% sur Resnet-50. Le gain de performance est plus faible sur SUN-InOut, soit 1.37% avec le modèle Alexnet. On peut également remarquer que les gains de performance sont influencés par le choix de l'architecture.

Données	Modèle	Taux succès (%)	
		$\mathcal{L}_E$	$\mathcal{L}_E + \alpha\mathcal{L}_K$
Caltech	Alexnet	69.0	**72.4**
	Resnet-50	74.4	**79.0**
	Densenet	74.7	**78.7**
	VGG19	69.1	**72.1**
SUN-In	Alexnet	53.3	**57.2**
	Resnet-50	61.0	**64.7**
	Densenet	60.5	**64.2**
	VGG19	53.1	**56.4**
SUN-Out	Alexnet	46.9	**48.9**
	Resnet-50	56.7	**59.3**
	Densenet	53.7	**55.5**
	VGG19	46.6	**48.9**
SUN-InOut	Alexnet	48.5	**49.9**
	Resnet-50	57.1	**59.6**
	Densenet	54.0	**56.6**
	VGG19	46.6	**49.1**

TAB. 4 – Apport de la connaissance sur des modèles classiques (images de taille 229×229).

Influence de α. Le taux de succès sans connaissance sur le jeu de données SUN-In avec l'architecture ResNet-50 est 74.7%. Comme le montre le Tableau 5, quelque soit α, nous obtenons sur ce jeu de données au moins le taux de réussite obtnue sans connaissance (meilleur taux $\alpha = 5$). Le fait que la valeur optimale de α soit supérieure à 1 peut s'expliquer par la valeur très faible de la perte liée à la connaissance.

α	0.2	0.5	0.8	1.0	3.0	5.0	7.0	10.0
taux succès (%)	74.7	75.3	77.2	77.4	78.5	**79.0**	78.8	76.7

TAB. 5 – Taux de succès avec différentes valeurs de α sur SUN-In (ResNet-50)

5 Conclusion

Dans ce travail, nous proposons une méthode d'intégration de la connaissance pour la classification d'images (KGIC). La connaissance est formalisée sous forme d'un graphe qui contient des nœuds représentant les images et les classes. Utilisant un plongement de ce graphe, nous définissons une nouvelle fonction de perte pour aider une architecture profonde à trouver une frontière entre les classes les plus difficiles à discriminer. Notre méthode est générique et ne nécessite pas la connaissance en mode test. Les comparaisons expérimentales avec l'état de l'art montrent une amélioration de performance sur les jeux de données Caltech et SUN Attributes. Pour les travaux futurs, nous envisageons d'élargir le type de connaissances, notamment sous forme de règles.

Remerciement. Ces travaux ont été partiellement financés par le projet ANR-20-THIA-0017-01 : AI.iO Artificial Intelligence in Orléans : Apprentissage à partir de données hétérogènes et de connaissances expert. Application aux sciences géologiques et environnementales.

Références

Alzubaidi, L., J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, et L. Farhan (2021). Review of deep learning : concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data 8*(1), 53.

Bach, S. H., M. Broecheler, B. Huang, et L. Getoor (2017). Hinge-loss markov random fields and probabilistic soft logic. *J. Mach. Learn. Res. 18*(1), 3846–3912.

Chen, T., L. Lin, R. Chen, Y. Wu, et X. Luo (2018). Knowledge-embedded representation learning for fine-grained image recognition. In *27th IJCAI*, IJCAI'18, pp. 627–634. AAAI.

Chen, Y., Y. Bai, W. Zhang, et T. Mei (2019). Destruction and construction learning for fine-grained image recognition. In *CVPR*.

Deng, J., W. Dong, R. Socher, L. Li, K. Li, et L. Fei-Fei (2009). Imagenet : A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee.

Diligenti, M., S. Roychowdhury, et M. Gori (2017). Integrating prior knowledge into deep learning. In *ICMLA*, pp. 920–923.

Du, R., D. Chang, A. K. Bhunia, J. Xie, Y. Song, Z. Ma, et J. Guo (2020). Fine-grained visual classification via progressive multi-granularity training of jigsaw patches. In *ECCV*.

Fang, Y., K. Kuan, J. Lin, C. Tan, et V. Chandrasekhar (2017). Object detection meets knowledge graphs. In *IJCAI-17*, pp. 1661–1667.

Glavaš, G. et I. Vulić (2018). Explicit retrofitting of distributional word vectors. In *56th Annual ACL*, Melbourne, Australia, pp. 34–45. Association for Computational Linguistics.

Grover, A. et J. Leskovec (2016). node2vec : Scalable feature learning for networks. In *ACM SIGKDD*, pp. 855–864.

He, J., J. C., S. L., A. K., C. Y., Y. B., C. W., et A. L. Y. (2022). Transfg : A transformer architecture for fine-grained recognition. In *AAAI*.

Khan, A., A. Sohail, U. Zahoora, et A. S. Qureshi (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 5455–5516.

Liu, C., H. Xie, Z. Zha, L. Ma, L. Yu, et Y. Zhang (2020). Filtration and distillation : Enhancing region attention for fine-grained visual categorization. *AAAI Conference*, 11555–11562.

Lu, Y., M. Rajora, P. Zou, et S. Y. Liang (2017). Physics-embedded machine learning : Case study with electrochemical micro-machining. *Machines 5*(1).

Ma, T. et A. Zhang (2018). Multi-view factorization autoencoder with network constraints for multi-omic integrative analysis. In *BIBM*, pp. 702–707.

P., G. et J. H. (2012). Sun attribute database : Discovering, annotating, and recognizing scene attributes. In *CVPR*.

Pfrommer, J., C. Zimmerling, J. Liu, L. Kärger, F. Henning, et J. Beyerer (2018). Optimisation of manufacturing process parameters using deep neural networks as surrogate models. *Procedia CIRP 72*, 426–431. 51st CIRP Conference on Manufacturing Systems.

Sun, G., H. Cholakkal, S. Khan, F. Khan, et L. Shao (2020). Fine-grained recognition : Accounting for subtle differences between similar classes. *AAAI Conference on Artificial Intelligence 34*(07), 12047–12054.

von R., L., S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy, J. Garcke, C. Bauckhage, et J. Schuecker (2021). Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 1–1.

Wah, C., S. Branson, P. Welinder, P. Perona, et S. Belongie (2011). The caltech-ucsd birds-200-2011 dataset.

Wei, X., Y. Song, O. Mac Aodha, J. Wu, Y. Peng, J. Tang, J. Yang, et S. Belongie (2021). Fine-grained image analysis with deep learning : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

Xu, J., Z. Z., T. F., Y. L., et G. V. den Broeck (2018). A semantic loss function for deep learning with symbolic knowledge.

Yang, Z., R. Al-Bahrani, A. C. E. Reid, S. Papanikolaou, S. R. Kalidindi, W. Liao, A. Choudhary, et A. Agrawal (2019). Deep learning based domain knowledge integration for small datasets : Illustrative applications in materials informatics. In *2019 IJCNN*, pp. 1–8.

Yang, Z., T. Luo, D. Wang, Z. Hu, J. Gao, et L. Wang (2018). Learning to navigate for fine-grained classification. In *ECCV*.

Summary

We present a deep learning method for supervised image classification, integrating knowledge formalized as a graph. We introduce a cost function combining the classical cross-entropy used in deep learning and an original function based on the representation of nodes after an embedding of the knowledge graph. The knowledge is only used during the learning phase and is not necessary to evaluate an example. Experiments on several image databases show an improvement of the performance compared to state-of-the-art: in comparison with classical deep learning algorithms, and with recent algorithms also integrating knowledge represented by a graph.