IMA206 Rapport
Quid de l'invariance par rotation dans un modèle DL pour des images

Jeanne Salle - Paule Grangette - Théau Blanchard - Julien Bergerot

June 23, 2022

Abstract

Intrinsèquement, les réseaux de neurones à convolution sont invariants aux translations. Cette propriété importante pourrait être étendue aux rotations. Cependant, avec les architectures classiques, celle-ci n'est pas présente. Bien que cela puisse ne pas être recherché sur des images classiques (des images de personnes, de la nature) car l'orientation de l'image donne un réel sens à l'image (une voiture roule forcément sur le sol, sur ses roues), cela est nécessaire dans certaines applications médicales, par exemple lorsque l'on traite des données histopathologiques. Même sur ce type de données, nous avons observé un manque de robustesse à l'orientation et nous avons proposé des solutions comme l'augmentation de données, ou d'autres architectures forçant le réseau à être invariant aux rotations. Les résultats montrent un réel progrès, et nous avons introduit des métriques pour mesurer ces différences.

Introduction

Les réseaux de neurones sont actuellement très utilisés, particulièrement en analyse d'images. Leur structure repose grandement sur des couches de convolution pour extraire différentes caractéristiques des images et travailler dans un espace transposé, une feature map. Cette feature map est souvent associée à quelques couches de régression ou de classification dans la plupart des problèmes. Il existe d'autres cas où l'objectif est de créer une image proche de l'image originale, comme le cas de la segmentation par instance. Dans ce cas, la feature map peut servir de centre du réseau. Par définition de l'opération de convolution, nous voyons que les résultats sont invariants par translation. Il n'y a pas de raison que l'invariance soit aussi présente par rotation mais cela peut être pertinent de la forcer dans certains cas. Comment le faire ?

Dans le problème de segmentation, nous voulons de l'équivariance, signifiant que si l'image d'entrée est tournée, le résultat doit être tourné et identique à ce que l'on aurait sans la rotation (une fois la rotation corrigée). Cependant, cela n'est pas présent avec tous les entraînements.

Pour réaliser cela, nous travaillerons avec des images histopathologiques. Ces images n'ont pas de sens "préféré" car lors d'un examen médical, le docteur peut capturer ces images dans le sens qu'il souhaite, selon comment se présente l'échantillon. Ainsi, on souhaite absolument obtenir les mêmes résultats peu importe l'orientation.

Nous allons tout d'abord identifier et mettre en lumière ce problème. Ensuite, nous allons proposer des méthodes en nous inspirant de la littérature. Finalement, nous allons tester ces méthodes et proposer des métriques pour évaluer cette robustesse aux rotations.

1. Présentation du problème

Dans un premier temps, nous avons décidé de mettre en évidence ce problème. Pour cela, nous avons entraîné des réseaux classiques pour réaliser deux tâches différentes : la segmentation de contours et le comptage de noyaux. Dans un premier temps, nous analysons visuellement les résultats de nos réseaux, ce qui suffit pour voir le problème.

1.1. Présentation des données

Comme discuté dans l'introduction, il ne serait pas le plus logique de travailler avec des bases de données d'images réelles comme ImageNet ou CIFAR10. Nous avons alors choisi de considérer des images histopathologiques, qui n'ont pas d'orientation préférentielle. A ces images-ci sont associés des masques binaires identifiant les contours et aussi une segmentation instancielle, identifiant les noyaux présents sur l'échantillon. Un exemple de trio d'images est disponible Fig. 1. Nous possédons au total de 5154 images, réparties entre 4468 pour le set d'entraînement et 686 pour le set de test.

Afin de préparer une explication aux performances des modèles, nous avons voulu analyser l'orientation des différents noyaux présents sur les images, et voir si l'une était favorisée. A la vue des résultats de la Fig. **??**, nous voyons que les orientations 0, 90 et 180 degrés semblent favorisées, mais cela est peut-être dû aux noyaux ronds, faus-

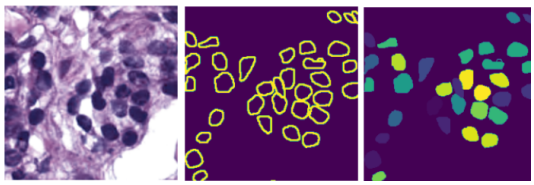


Figure 1. A gauche : L'image originale - Au centre : La segmentation binaire des contours - A droite : La segmentation instancielle des noyaux

sant les résultats.

1.2. Segmentation des contours

Nous nous sommes dans un premier temps penchés sur le problème de la segmentation des contours. Ce problème est un problème de traduction d'images, nous avons alors opté pour un réseau Unet, un réseau introduit en 2015 [2] pour ce problème de segmentation des noyaux dans le domaine médical. Bien que des améliorations de ce réseau aient été proposées, nous avons choisi de rester avec ce simple réseau, dont l'architecture est disponible Fig. 2, car notre but est simplement de montrer que les réseaux de segmentation ne sont pas robuste intrinsèquement aux rotations.

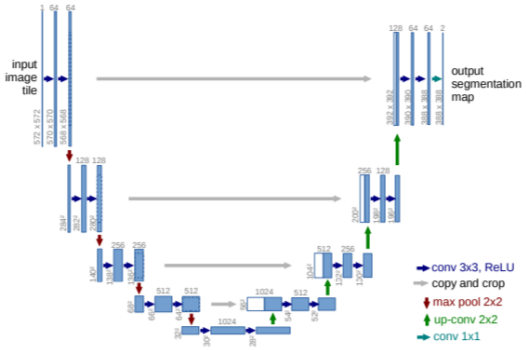


Figure 2. Architecture d'un UNet (example avec un résolution minimale de 32x32 pixels). Chaque boîte bleue correspond à une feature map avec plusieurs canaux, numéro indiqué en haut de la boîte. La dimension spatiale est mentionnée en bas à gauche. Les boîtes blanches sont les copies. Les flèches montrent les différentes opérations. Ici, il y a deux classes prédites en sortie, ce qui est équivalent à notre adaptation.

Afin de réaliser cela, nous avons simplement normalisé les images entre 0 et 1. Le réseau prédit une probabilité pour chaque pixel, qui sera interprétée comme la probabilité d'être un contour.

L'entraînement a été fait durant 50 epochs avec un learning rate de 1e-4. La loss utilisée est la binary cross entropy. Les courbes d'entraînement entre le test et train set sont très proches, ce qui montre une absence d'overfitting.

Pour analyser la robustesse aux rotations, nous avons considéré le set d'entraînement. Pour chacune des images, nous avons fait la prédiction de l'image avec des rotations de 0, 90, 180 et 270 degrés. Afin de pouvoir comparer ces résultats, les rotations inverses ont été appliquées aux prédictions. Un exemple est présenté Fig. 4. Bien que les prédictions semblent proches, l'image de différence montre que le réseau ne prédit pas vraiment exactement la même chose en fonction de la rotation, et n'est pas d'accord avec lui-même. Visuellement, cela met en évidence le problème que l'on souhaite résoudre, qui est présent sur toutes les images de la base. Pour le moment, cela suffit pour vouloir proposer des améliorations que nous ferons dans la seconde partie. De plus, la troisème partie introduira des métriques pour comparer et confirmer notre intuition visuelle.

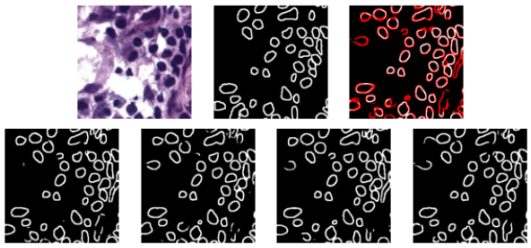


Figure 3. Première ligne, de gauche à droite : L'image originale, la vérité terrain et l'image de différences entre les 4 prédictions du bas. Un pixel est noir si les 4 prédisent un pixel noir, un pixel est blanc si les 4 prédisent un contour, sinon le pixel est rouge. Seconde ligne, de gauche à doirte : La prédiction après un rotation de 0, 90, 180 et 270 degrés.

1.3. Comptage des noyaux

Par la suite nous nous avons cherché à déterminer le nombre de noyau présent sur chaque image de contour donné par le U-Net précedemment décrit. Cela revient alors à effectuer un modèle de regression à partir d'une image binaire. Pour ce faire on va adapter le réseau VGG16 [3] développé en 2014. On récupère les poids de ce réseau entrainé sur ImageNet, on rajoute une couche de convolution sur le haut du réseau pour passer d'une image à un canal à une image avec 3 canaux. Aussi on passe de la sortie du VGG16 à un MaxPooling avec un stride de 2 pour finir sur un couche connectée avec 32 neurones. La sortie est donnée par une dernière couche connecté avec 1 neurone qui correspond au nombre de noyaux visibles sur l'image de contours.

L'entrainement est réalisé en deux temps. Tout d'abord on entraine nos couches personnalisées sur 20 epochs avec un learning rate $l_0 = 0.01$ dégressif en escalier tel que $l_k = l_0 * 0.8^{\lfloor \frac{k}{5} \rfloor}$. Ensuite on va fine-tune le réseau en entrainant toutes les couches sur 20 epochs avec maintenant $l_0 = 10^{-5}$. Dans les deux cas on minimise l'erreur quadratique moyenne et on conserve le modèle qui performe le mieux sur les données de validation (20% des données d'entrainement).

Au final on obtient une erreur absolue moyenne de 2.21.

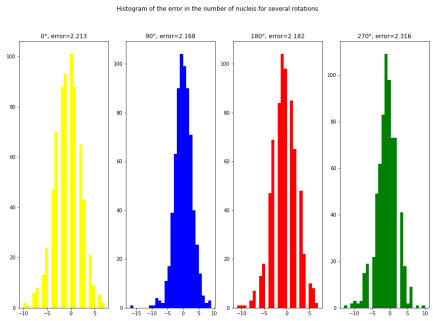


Figure 4. Histogramme de l'erreur absolue moyenne selon les angles de rotation

On constate que peu importe les rotations que l'on donne en entrée, on obtient globalement des résultats similaires. En revanche, si on regarde en détail les résultats pour une image en particulier on se rend compte que le nombre de noyau détectés sur les images tournées est moins précis que sur les images de base.

2. Techniques d'amélioration

Le problème ayant été mis en évidence, nous souhaitons proposer plusieurs approches afin de le résoudre. Une première méthode repose sur l'augmentation de données. Deux autres méthodes ont été proposées dans les papiers [1] et [4], les réseaux SE2CNN et les Harmonic Networks. Il s'agit ici de méthodes dites de "hard-baking" : plutôt que d'être amenée par le biais des données d'entraînement, l'invariance aux rotations est introduite dans le réseau "par construction", en imposant des contraintes sur la nature des filtres.

2.1. Augmentation de données

L'augmentation de données consiste à modifier les images juste avant leur passage dans le réseau lors de l'entraînement pour qu'il ne voit jamais deux fois exactement la même image. Cela évite que le réseau apprenne par coeur sur le set d'entraînement mais aussi peut le forcer à apprendre les caractéristiques importantes des images plus facilement. En général, cette méthode permet d'obtenir un réseau plus puissant et robuste. Ces opérations d'augmentation peuvent consister en l'ajout de bruit gaussien, de flou, modifier le contraste, la luminosité, retourner l'image, la tourner d'un certain angle et bien d'autres. Ici, nous souhaitons simplement le rendre plus robuste aux rotations. De ce fait, afin de pouvoir comparer les performances avec le réseau entraîné précédemment, nous allons simplement appliquer des rotations aux images. Ces rotations peuvent prendre des valeurs uniformément entre 0 et 360 degrés. Nous espérons que cela lui permettra d'ignorer une sorte de distribution qu'il aurait identifiée dans les angles des noyaux, et qu'il prédise la même chose en fonction des rotations que l'on appliquera.

2.1.1 Segmentation des contours

Dans cette partie, le modèle et les paramètres utilisés sont exactement les mêmes que dans la partie associée sans l'augmentation de données. De la même manière, les métriques et erreurs sont très proches et il n'y a pas d'overfitting. Une image très similaire à Fig. 4 est disponible Fig. 5. L'image en input est la même et l'on voit beaucoup moins de rouge sur l'image de différences. De plus, les résultats ont l'air un peu meilleurs, mais cela sera chiffré dans la partie résultat.

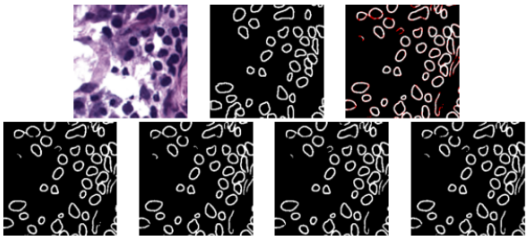


Figure 5. Avec augmentation de données. Première ligne, de gauche à droite : L'image originale, la vérité terrain et l'image de différences entre les 4 prédictions du bas. Un pixel est noir si les 4 prédisent un pixel noir, un pixel est blanc si les 4 prédisent un contour, sinon le pixel est rouge. Seconde ligne, de gauche à droite : La prédiction après un rotation de 0, 90, 180 et 270 degrés.

2.1.2 Comptage de noyaux

Ici le modèle et les paramètres d'entrainement sont identiques à la partie de tout à l'heure. On limite les rotations aux multiples de 90° afin de ne pas changer le nombre de noyaux affichés. On effecture aussi des réflexions verticales et horizontales. On obtient des résutats significativement meilleurs : l'erreur absolue moyenne n'est plus que de 1.63. Aussi si on regarde en détail les images, le fait d'imposer des rotations ne tend pas à dégrader la qualité du résultat.Fig. 6

2.2. SE2CNN

2.2.1 Article : Roto-Translation Equivariant Convolutional Networks: Application to Histopathology Image Analysis [1]

L'augmentation de données apporte, certes, des résultats assez satisfaisants pour obtenir l'équivariance par rotation mais cette méthode pose aussi certains problèmes. En effet, cette approche est très coûteuse en terme de capacité

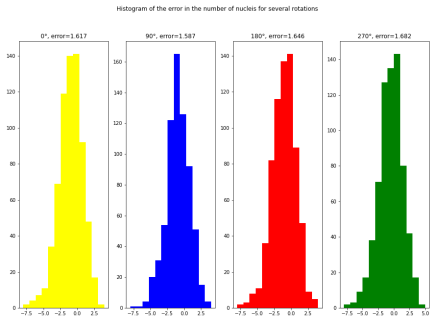


Figure 6. Histogramme de l'erreur absolue moyenne selon les angles de rotation

du réseau (déjà précieuse) pour apprendre le comportement géométrique des noyaux à l'intérieur des images. De plus, l'invariance par rotation n'est pas garantie. Enfin, l'augmentation de données capte l'invariance géométrique de manière globale mais il est impossible de savoir comment elle est obtenue, puisque nous n'avons aucune information sur l'invariance géométrique locale.

Les chercheurs qui ont rédigé ce papier ont proposé une nouvelle méthode afin de palier ces problèmes en s'appuyant sur un cadre mathématique bien plus solide. Ils proposent un nouveau réseau de neurones convolutionnel nommé G-CNN, qu'ils appliquent à trois types de tâches d'imagerie médicale à partir d'images histopathologiques : la détection de mitoses, la classification de tumeurs par méthodes par patch et enfin la segmentation de noyaux multi-organes.

Le réseau G-CNN est composé de trois couches principales : la Lifting Layer, la SE(2) Group Convolution Layer et la Projection Layer (cf Figure 5). Leur construction repose sur l'idée de faire tourner les noyaux de convolutions suivant différents angles et de récupérer les features associées pour apprendre de manière optimale les les features importantes et de pouvoir retourner l'image effectuant la tâche souhaitée (dans notre cas, nous nous sommes majoritairement intéressés à la segmentation).

Les couches sont construites de sorte que chacune préserve l'équivariance par rotation localement. Afin d'assurer cette propriété, les chercheurs ont défini au sein de chaque couche des opérations linéaires qui correspondent à des opérations entre groupes isomorphes au groupe des roto-translations SE(2). Le groupe isomorphe en question est celui des transformations linéaires $R_g : \mathbb{L}^2(X) \longrightarrow \mathbb{L}^2(X)$, pour $g \in G$ et X un ensemble. Les représentations du groupe SE(2) (qui sont des transformations linéaires) vérifient la définition mathématique de l'équivariance :

Définition : *Un opérateur* $\Phi : \mathbb{L}^2(X) \longrightarrow \mathbb{L}^2(Y)$ *est dit* équivariant *par rapport à un groupe* G *avec* X, Y *des ensembles, si pour tout* $f \in \mathbb{L}^2(X)$ *:*

$$\Phi(R_g(f)) = R'_g(\Phi(f))$$

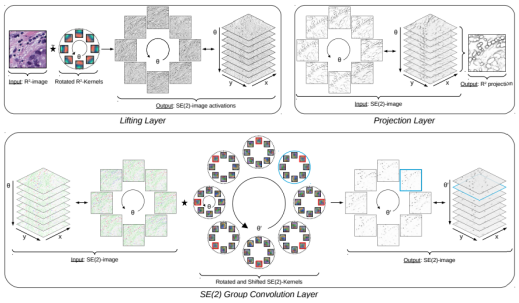


Figure 7. Illustration des trois types de couches utilisées dans le G-CNNs. La *Lifting Layer* utilise des noyaux tournés dans $\mathbb{R}^2$ et renvoie en sortie une fonction d'activation qui est une image sur *SE(2)*. La *SE(2) Group Convolution Layer* applique une convolution de décalage-torsion via un ensemble de noayux tournés et translatés dans *SE(2)*. La *Projection Layer* transforme une *SE(2)*-image sur $\mathbb{R}^2$ via une opération invariante par rotation. Les exemples sont issus d'un modèle de segmentation de noyaux.

Le détail mathématique est développé dans la présentation fournie en annexe. En réalité, on utilise une version discrète du groupe $SE(2)$: $SE(2, N)$ où N est le nombre d'angle de rotations disponibles.

L'article montre finalement que la performance du G-CNNs est globalement meilleure qu'en utilisant l'augmentation des données et que cette méthodes a une meilleure robustesse aux orientations d'entrées. Il est également notable qu'il faut faire un compromis entre la capacité du réseau et l'angle de résolution choisi. On peut voir en figure 7, les résultats obtenus pour divers angles de résolutions et différentes images.

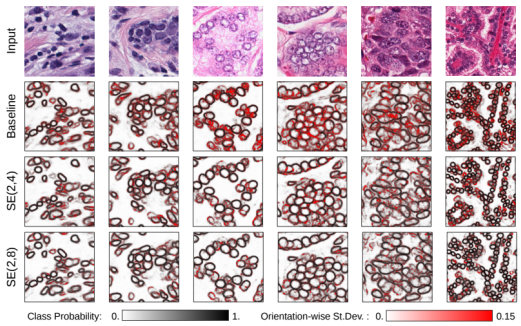


Figure 8. Résultats de la segmentation des images pour la Baseline, $SE(2, 4)$ et $SE(2, 8)$. En rouge, les différences observées entre les images issues des images initiales tournées par rapport à l'image témoin de segmentation.

Finalement, ce qu'on pourrait reprocher à l'article est sa clarté aussi bien dans ses démonstrations mathématiques que dans ses présentations d'implémentations des G-CNNS

pour les trois tâches présentées. En effet, certains passages relatifs aux représentations régulières à gauches du groupe $SE(2)$ ne sont pas toujours bien expliqués et l'intérêt de la *group convolution layer* par rapport à la *lifting layer* n'est pas toujours clair : notamment la dimension en plus dans les noyaux de la *group convolution layer* n'est pas bien définie et on ne sait pas vraiment à quoi correspond cette rotation. Par ailleurs, nous avons aussi rencontrés des problèmes vis-à-vis de la définition du modèle utilisant le G-CNNs sur la segementation de noyaux, dont nous allons parler dans la section suivante.

2.2.2 Test sur le dataset MNIST

Un code pour implémenter en Python les réseaux introduits dans l'article est disponible sur GitHub. Néanmoins il est à noter que le dépôt n'a pas été mis à jour depuis 2018 et souffre de nombreux problèmes de versions. Nous sommes parvenus à le rendre utilisable avec la dernière version de Tensorflow (2.9.1).

Les résultats donnés pour le dataset MNIST étaient plutôt convaincants puisque l'accuracy était de 0.98 et que la matrice de confusion était quasiment diagonale (cf figure 7).

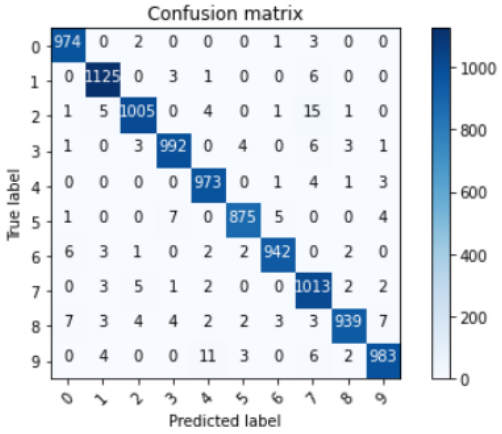


Figure 9. Matrice de confusion pour le réseau G-CNN sur le dataset MNIST.

Nous n'avons pas eu le temps d'implémenter le réseau G-CNN pour la segmentation de noyaux (cf figure 8) car nous avons rencontré de nombreux problèmes lors de l'implémentation. En effet, la création de la pipeline d'entrée était assez difficile étant donné que les images étaient en format .tif et que toutes les versions de tensorflow ne fonctionnaient pas pour ces images. Nous avons ensuite essayé sur le dataset Oxford-IIIT Pet mais n'avons pas eu le temps d'aboutir.

Nous avons tout de même fourni notre code en annexe (le nom du fichier est "SE(2,4)").

Table 2: Architecture and weight counting of the G-CNN models for patch-based tumor classification. The left-most column indicates the operations in each layer. *Concat(HL.x)* indicates the characteristic skip operation of the U-net architecture that consist in concatenating a centered crop of the output activation of the x^{th} layer of the network. *Max. Proj.* indicates the projection operation on $\mathbb{R}^2$, achieved via maximum intensity projection along the orientations.

Layers	*SE(2,N)* Groups N=1 ($\mathbb{R}^2$)	N=4 (p4)	N=8	N=16
Input	60×60×3			
Lifting Layer BN + ReLU MaxPool(2×2)	1×28×28×16 (1040)	4×28×28×10 (650)	8×28×28×8 (520)	16×28×28×6 (390)
Group Conv. BN + ReLU MaxPool(2×2)	1×12×12×16 (5408)	4×12×12×10 (8420)	8×12×12×8 (10768)	16×12×12×6 (12108)
Group Conv. BN + ReLU	1×8×8×16 (5408)	4×8×8×10 (8420)	8×8×8×8 (10768)	16×8×8×6 (12108)
Up-sampling Concat(HL.2) Group Conv. BN + ReLU	1×12×12×16 (10784)	4×12×12×10 (16820)	8×12×12×8 (21520)	16×12×12×6 (24204)
Up-sampling Concat(HL.1) Group Conv. BN + ReLU	1×20×20×64 (43136)	4×20×20×16 (26912)	8×20×20×8 (21520)	16×20×20×4 (16136)
Group Conv. BN + ReLU	1×20×20×16 (1056)	4×20×20×16 (1056)	8×20×20×16 (1056)	16×20×20×16 (1056)
Max. Proj.	20×20×16			
FC Layer Softmax	20×20×3 (54)			
Total Weights	66886	62332	66206	66056

Figure 10. Architecture du réseau G-CNNs pour la segmentation de noyaux

2.3. Harmonic Networks

Comme expliqué précédemment, l'approche traditionnelle pour construire un réseau équivariant aux rotations consiste à avoir recours à l'augmentation de données. Néanmoins, une telle méthode ne garantit pas qu'il y ait invariance à chacune des couches du réseau. Par ailleurs, celles-ci sont généralement assez peu interprétables. L'approche présentée dans [4] propose de surmonter ces deux limitations en remplaçant les filtres classiques par des transformations appartenant à la famille des harmoniques circulaires.

Les harmoniques circulaires sont des filtres dits "orientables" (steerable filters), c'est-à-dire qu'ils sont paramétrés par un terme de phase appris au cours de l'entraînement. Ce terme de phase détermine la sélectivité à l'orientation du filtre. Contrairement à ceux des SE2CNN les filtres des Harmonic Networks ne sont pas "redondants" au sens où on n'y trouve pas de filres similaires à une rotation près. On n'observe pas non plus de phénomène de "copie de filtres" qui est courant dans les CNN entraînés avec augmentation de données.

D'après les auteurs, les H-Nets offrent des performances état-de-l'art sur la classification du jeu de données MNIST avec rotation, et permettraient de réduire le taux d'erreur de plus de 30% par rapport à un CNN classique. Quand à la détection de contours, les H-Nets seraient *a priori*

plus performants que la plupart des réseaux classiques non pré-entraînés, mais tout de même moins que des réseaux spécialisés et pré-entraînés comme DeepContour.

Il est également intéressant de noter que les features maps encodent des structures compliquées (angles, coins, espaces négatifs, etc.) assez clairement visualisables, et ce jusque dans les couches les plus profondes. Il semblerait également que les H-Nets, à accuracy fixée, nécessitent un moindre volume de données d'entraînement que les CNN classiques.

3. Résultats

3.1. Segmentation des contours

Le même algorithme de segmentation a été entraîné sur la même base, avec la seule modification étant l'augmentation de données. Nous avons déjà vu visuellement les différences entre les deux modèles Fig. 4 et Fig. 5 mais nous souhaitons ajouter des métriques. L'image étant principalement composée de noir pour le fond, il faut tenir en compte de cette répartition. Ainsi, le score de dice est utilisé pour comparer une prédiction avec sa vérité terrain. Cela permet de mesurer la précision de nos modèles. Cependant, une seconde chose que l'on souhaite mesurer est l'accord entre les différentes prédictions après rotation. Pour cela nous avons défini un score tel que :

$$S = \frac{2 * score_4 + score_3}{|contours|}$$

où $score_4$ est le nombre de pixel où les 4 prédictions sont d'accord, et $score_3$ est du 3 contre 1. Les pixels considérés sont l'union des contours prédits, ce qui permet d'avoir un score entre 0 et 1.

A titre d'exemple, cela donne un score de 0.63 à Fig. 4 et de 0.91 à Fig. 5, ce qui confirme nos visuels. Sur toute la base de test, ce score vaut **0.60** en moyenne avec un écart-type de **0.05** pour le premier modèle et **0.89** $\pm$ **0.04** pour le second.

Finalement, il est possible que les différences entre les prédictions des rotations soient une force et puisse permettre au modèle de mieux prédire la prédiction initiale. Pour tester cela, nous avons choisi de créer quatre différentes prédictions en se basant sur les 4 segmentations obtenues. La première est simplement la prédiction avec l'image originale. La seconde est un vote majoritaire où 3v1 et 4v0 l'emporte. Le second est un vote majoritaire où 2v2 l'emporte. Pour ces 3 "combats" cela, signifie que si respectivement, 3, 4 ou 2 predictions prédisent un contour, un contour sera le choix final du pixel. Finalement, le dernier est un score aggrégé, c'est-à-dire que tous les pixels considérés à un moment comme un contour seront considérés comme tels. Les résultats sont présentés dans le tableau 1. Ces résultats confirment aussi le fait qu'un modèle avec de l'augmentation de données donne une meilleure segmentation. Une remarque intéressante est que, peu importe la méthode, on a l'impression que les prédictions se complètent, c'est-à-dire que plus l'on concatènent les résultats, plus le score augmente. On peut alors penser que certains contours (ou simplement pixels) sont plus facilement identifiables selon certaines orientations et donc regarder sous plusieurs angles permet de gagner en précision, simplement en rajoutant ce que l'on voit (rajouter les pixels positifs).

Score de dice avec les différentes prédictions		
	Sans augmentation	Avec augmentation
Prédiction 1	0.73 $\pm$0.05	0.81 $\pm$0.05
Prédiction 2	0.62 $\pm$0.04	0.77 $\pm$0.03
Prédiction 3	0.75 $\pm$0.06	0.84 $\pm$0.04
Prédiction 4	0.8 $\pm$0.02	0.89 $\pm$0.04

Table 1. Score de dice moyen (avec l'écart type) sur toute la base de test. Les 4 prédictions sont dans le même ordre que décrit dans la partie associée.

Erreur absolue moyenne selon les stratégies choisies		
	Sans augmentation	Avec augmentation
Moyenne	1.67	1.40
Médiane	1.81	1.47
Minimum	2.09	1.28
Maximum	2.74	2.18

Table 2. Erreur absolue moyenne selon les stratégies choisies

3.2. Comptage des noyaux

On a pu voir que l'augmentation améliore significativement les résulats. On peut donc se demander si on peut combiner les résultats de l'image originale avec ses versions tournées à 90°,180° et 270° pour affiner la prédiction. On va donc chercher à combiner les résultats en faisant leur moyenne, leur médiane, la valeur minimale puis maximale. On constate dans 2 que les stratégies de fusion des résulats améliorent les résultats par rapport à la prédiction de base. Aussi une nouvelle fois on constate que l'augmentation permet d'améliorer les résultats dans toutes les stratégies.

4. Discussion et conclusion

Nous avons montré que l'augmentation de données est un principe qui permet de rendre le modèle plus robuste et plus équivariant. De plus, nous avons montré que regarder une image sous plusieurs angles semble améliorer les résultats (environ 10%) pour le dice score. Bien que certaines prédictions de contours soient uniques, cela n'implique pas qu'elles soient fausses, comme le montre le

score obtenu après aggrégation.

Pour aller plus loin il aurait été pertinent de pouvoir entraîner et tester un SE2CNN et un H-Net et de comparer les performances de ces modèles avec la seule augmentation de données. Nous aurions ainsi pu confirmer ou non les conclusions avancées dans les articles. Leurs auteurs suggèrent par ailleurs plusieurs pistes d'amélioration, ils proposent par exemple d'étendre les filtres des réseaux à d'autres familles de transformations garantissant l'équivariance, voire d'étendre leurs architectures pour des cas à trois dimensions.

References

[1] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis, 2018. 3

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 2

[4] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance, 2016. 3, 5

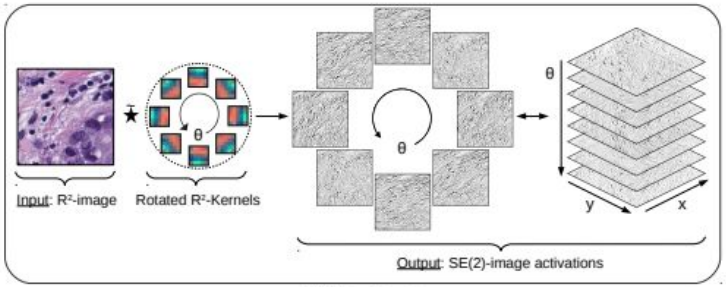
Article : Roto-Translation Equivariant Convolutional Networks : Application to Hispathology Image Analysis

Paule Grangette, le 16/06/2022

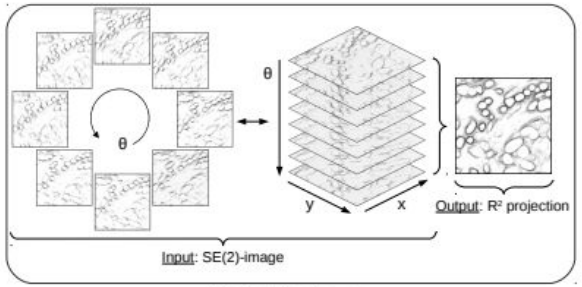
Modèle du G-CNN : basé sur le groupe SE(2)

- Problème d'équivariance : méthode sans data augmentation
- Principe : remplacer les convolutions dans R^2 par des group convolutions utilisant des représentations du groupe SE(2) (des roto-translations) pour encoder explicitement l'orientation des features apprises → assure équivariance
- Pallie les problèmes suivants :
  1) n'a pas besoin d'apprendre les comportements géométriques (pas de perte inutile de capacité du réseau)
  2) invariance par rotation : garantie par construction (pas le cas dans les CNNs avec data augmentation)
  3) equivariance globale : obtenue localement par chaque couche du réseau donc localement (autre méthode : augmentation capture l'invariance géométrique que globalement)

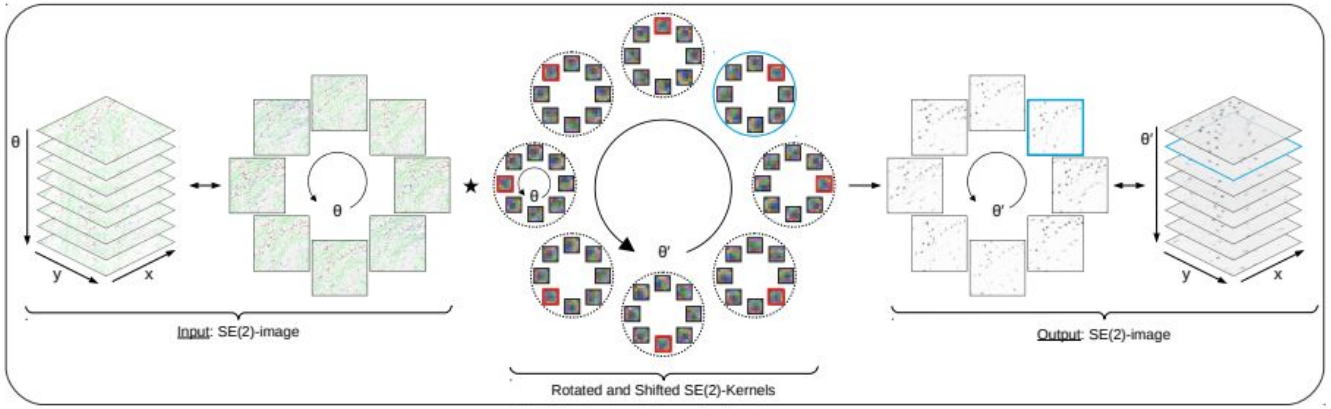
Structure du modèle



Lifting Layer



Projection Layer



SE(2) Group Convolution Layer

Groupe des Roto-Translations SE(2)

Définition groupe (G, .) : loi de composition interne, associativité, élément neutre, symétrique

On considère ici SE(2) = R^2 x SO(2), où la loi est : $g \cdot g' = (\mathbf{x}, \mathbf{R}_\theta) \cdot (\mathbf{x}', \mathbf{R}_{\theta'}) = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \mathbf{R}_{\theta+\theta'})$

Par abus de notation, on définit SE(2) = R^2 x S^1 : $g \cdot (\mathbf{x}', \theta') = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \theta + \theta')$

On peut vérifier qu'il s'agit d'un groupe où l'élément neutre est (0,0) et le symétrique est défini par $g^{-1} = (-\mathbf{R}_\theta^{-1}\mathbf{x}, -\theta)$.

Représentations du groupe

- Utilisation d'isomorphismes pour préserver les propriétés d'un groupe.

- Représentation d'un groupe G par des transformations linéaires

$$\mathcal{R}_g : \mathbb{L}_2(X) \to \mathbb{L}_2(X) \qquad (\mathcal{R}_g \circ \mathcal{R}_h)(f) = \mathcal{R}_{g \cdot h}(f), \qquad \text{with } g, h \in G.$$

- Utilisation de différentes représentation pour $\mathcal{R}_g$: comme la représentation régulière à gauche de SE(2) sur les images 2D : $\mathcal{R} = \mathcal{U}$

$$(\mathcal{U}_g f)(\mathbf{x}') = f(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x})). \qquad g = (\mathbf{x}, \theta) \in SE(2),\ \mathbf{x}' \in \mathbb{R}^2 \qquad f \in \mathbb{L}_2(\mathbb{R}^2)$$

Ou la représentation régulière à gauche définie pour $F \in \mathbb{L}_2(SE(2))$ par $\mathcal{R} = \mathcal{L}$

$$(\mathcal{L}_g F)(g') = F(g^{-1} \cdot g') = F(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x}), \theta' - \theta). \qquad g = (\mathbf{x}, \theta), g' = (\mathbf{x}', \theta') \in SE(2).$$

Equivariance : définition

Un opérateur $\Phi : \mathbb{L}_2(X) \to \mathbb{L}_2(Y)$ est équivariant par rapport au groupe G si :

$$\Phi(\mathcal{R}_g(f)) = \mathcal{R}'_g(\Phi(f)),$$

Afin d'assurer la propriété d'équivariance des opérateurs linéaires, il faut utiliser des représentations du groupe considéré (dans notre cas, SE(2)).

Notations et couches de convolution 2D

On note $(\mathbb{L}_2(X))^N$ l'ensemble des applications sur plusieurs canaux établissant des caractéristiques. Soit f une "feature map" : $\underline{f} = (f_1, \dots, f_N)$.

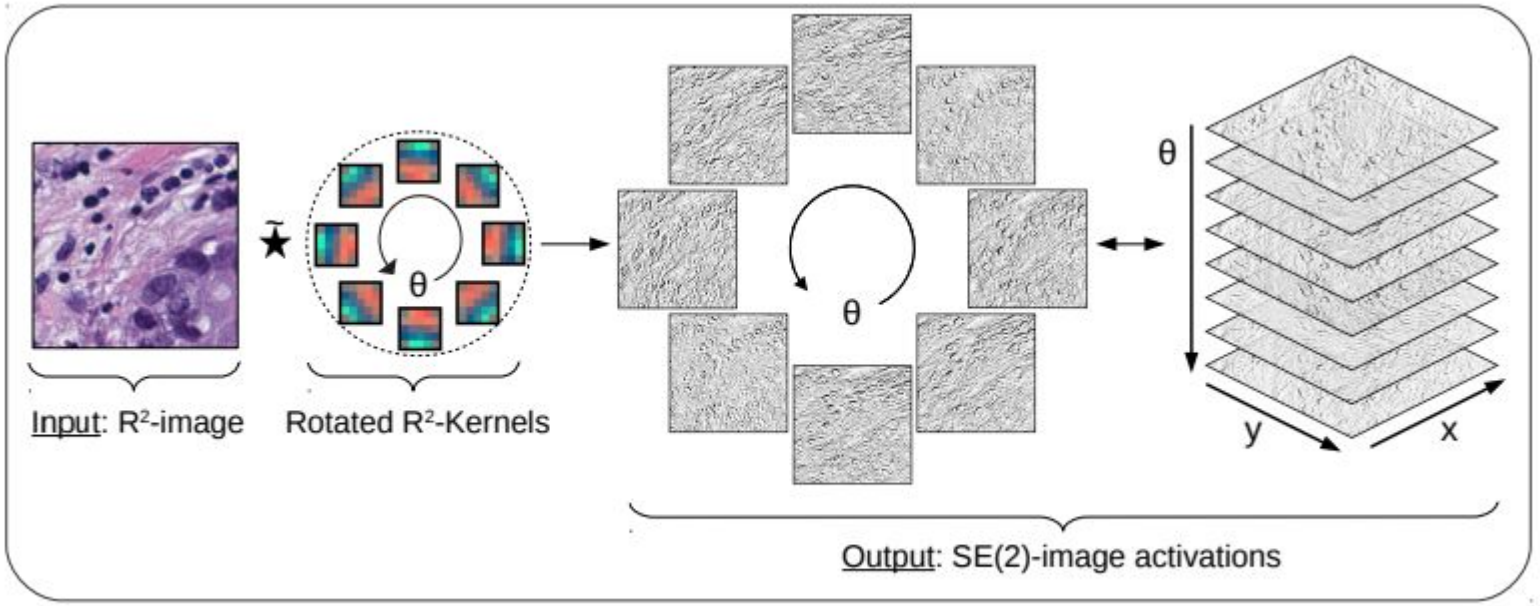
La loi interne de l'endomorphisme est : $(\underline{k}, \underline{f})_{(\mathbb{L}_2(X))^N} := \sum_{c=1}^{N} (k_c, f_c)_{\mathbb{L}_2(X)}$

où : $(k, f)_{\mathbb{L}_2(X)} = \int_X k(\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

On définit ensuite la cross-validation 2D à partir de cette loi interne et de noyau de convolution translaté ($\mathcal{T}_{\mathbf{x}}$ est l'opérateur de translation qui est la représentation régulière à gauche pour le groupe $(R^2,+)$ → équivariance OK)

$$(\underline{k} \star_{\mathbb{R}^2} \underline{f})(\mathbf{x}) := (\mathcal{T}_{\mathbf{x}}\underline{k}, \underline{f})_{(\mathbb{L}_2(\mathbb{R}^2))^N}$$
$$= \sum_{c=1}^{N} \int_{\mathbb{R}^2} k_c(\mathbf{x}' - \mathbf{x}) f_c(\mathbf{x}')\mathrm{d}\mathbf{x}'.$$

Lifting layer



Lifting Layer

Couche qui calcule les features d'origine

Lifting layer

- La ***lifting correlation*** : $(\underline{k}\tilde{\star}\underline{f})(g) := (\mathcal{U}_g\underline{k}, \underline{f})_{(\mathbb{L}_2(\mathbb{R}^2))^N}$ où $\underline{f} \in (\mathbb{L}_2(\mathbb{R}^2))^N$

$$= \sum_{c=1}^{N} \int_{\mathbb{R}^2} k_c(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x})) f_c(\mathbf{x}')\, \mathrm{d}\mathbf{x}'$$

- ***Lifting layer*** : $\tilde{\Phi}^{(l)} : (\mathbb{L}_2(\mathbb{R}^2))^{N_{l-1}} \to (\mathbb{L}_2(SE(2))^{N_l}$ définie par :

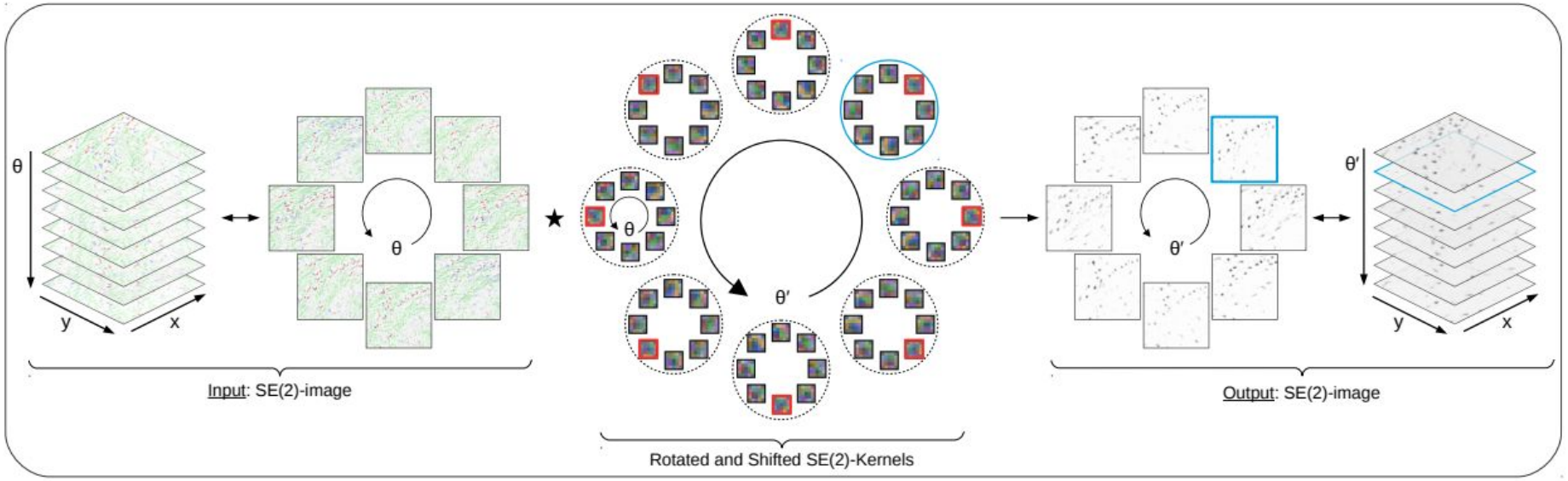
$$\underline{F}^{(l)} = \tilde{\Phi}^{(l)}(\underline{f}^{(l-1)}) := \mathbf{k}^{(l)}\tilde{\star}\underline{f}^{(l-1)} \qquad \underline{F}^{l} \in (\mathbb{L}_2(SE(2))^{N_l} \qquad \mathbf{k}^{(l)} := (\underline{k}_1^{(l)}, \ldots, \underline{k}_{N_l}^{(l)})$$

et $\mathbf{k}^{(l)}\tilde{\star}\underline{f}^{(l-1)} := \left(\underline{k}_1^{(l)}\tilde{\star}\underline{f}^{(l-1)}, \cdots, \underline{k}_{N_l}^{(l)}\tilde{\star}\underline{f}^{(l-1)} \right)$

- équivariance : en prenant $\mathcal{T}_g = \mathcal{U}_g$ et $\mathcal{T}'_g = \mathcal{L}_g$ on a bien :

$$\tilde{\Phi}^{(l)}(\mathcal{U}_g\underline{f}^{(l-1)}) = \mathcal{L}_g\tilde{\Phi}^{(l)}(\underline{f}^{(l-1)})$$

Group Convolution Layer



SE(2) Group Convolution Layer

Couche qui met à jour les features

Group Convolution Layer

- ***Group correlations*** (représentation de SE(2)) :

$$(\underline{K} \star \underline{F})(g) := \sum_{c=1}^{N_c} (\mathcal{L}_g K_c, F_c)_{\mathbb{L}_2(SE(2))} = \sum_{c=1}^{N_c} \int_{SE(2)} K_c(g^{-1} \cdot g') F_c(g') \mathrm{d}g'$$

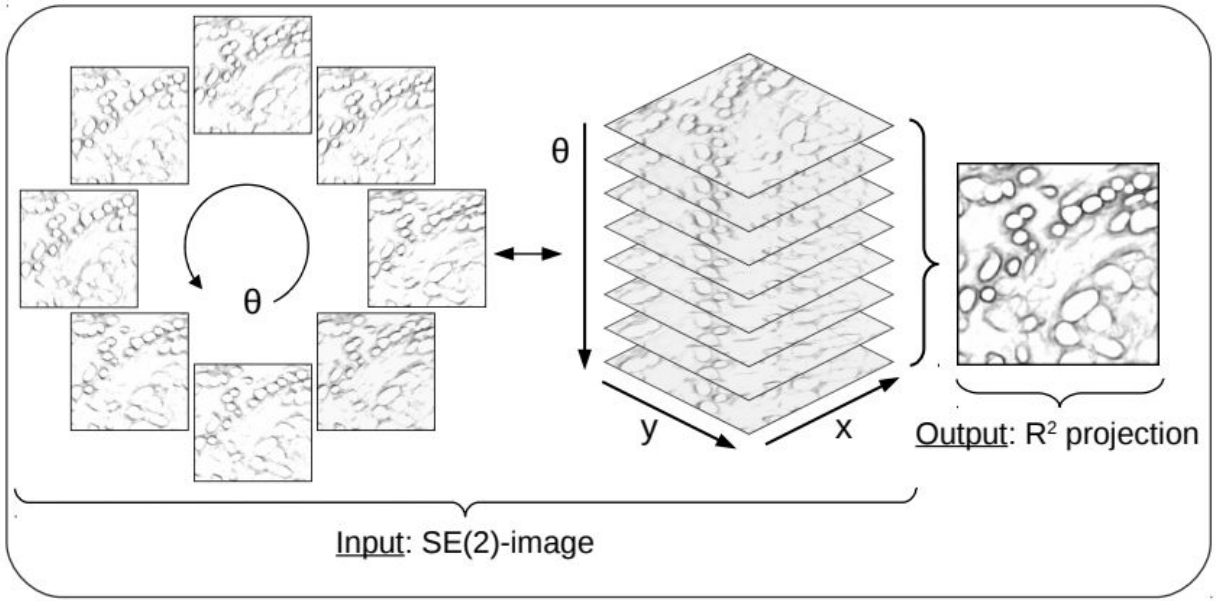
  où $\underline{K}$ est l'ensemble de noyaux (3D) de convolution roto-translatés

- ***Group convolution layer*** : pour $\underline{F}^l \in (\mathbb{L}_2(SE(2))^{N_l}$

$$\underline{F}^{(l)} = \Phi^{(l)}(\underline{F}^{(l-1)}) := \mathbf{K}^{(l)} \star \underline{F}^{(l-1)} := \left(\underline{K}_1^{(l)} \star \underline{F}^{(l-1)} , \dots , \underline{K}_{N_l}^{(l)} \star \underline{F}^{(l-1)} \right)$$

- Rotation des noyaux de convolution de SE(2) obtenus par une rotation planaire et une translation suivant l'axe **θ**.

Couche de projection



Projection Layer

Couche qui projette les nouvelles features suivant l'axe de rotation pour obtenir l'image finale

Couche de projection

On prend la moyenne sur tous les axes d'orientations :

$$\underline{f}^{(l)}(\mathbf{x}) = \mathcal{P}(F^{(l)})(\mathbf{x}) := \underset{\theta \in [0,2\pi)}{\text{mean}} \underline{F}^{(l)}(\mathbf{x}, \theta)$$

⟶ On aurait pû prendre n'importe quel opérateur invariant par permutation pour assurer l'invariance par rotation locale

Discrétisation du groupe SE(2) : cas des images

On modifie les définitions précédentes en posant : $SE(2,N) := \mathbb{R}^2 \rtimes SO(2,N)$

où N est le nombre d'angles de rotation : $\theta_i = \frac{2\pi}{N} i$ avec i compris entre 0 et N-1

- Lifting layer : $\mathbf{k}^{(l)}$ s'applique sur une image 2D avec N_{l-1} canaux et donne une SE(2,N)-image avec N_l canaux : forme n x n x N_{l-1} x N_l
- Group convolution layer : $\mathbf{K}^{(l)}$ s'applique à un ensemble de features N_{l-1} -channels vers un ensemble de features N_l-channels : taille nxnxNxN_{l-1}xN_l
- On obtient les parties spatiales tournées de chaque canal par interpolation bilinéaire.

Experiments & Results

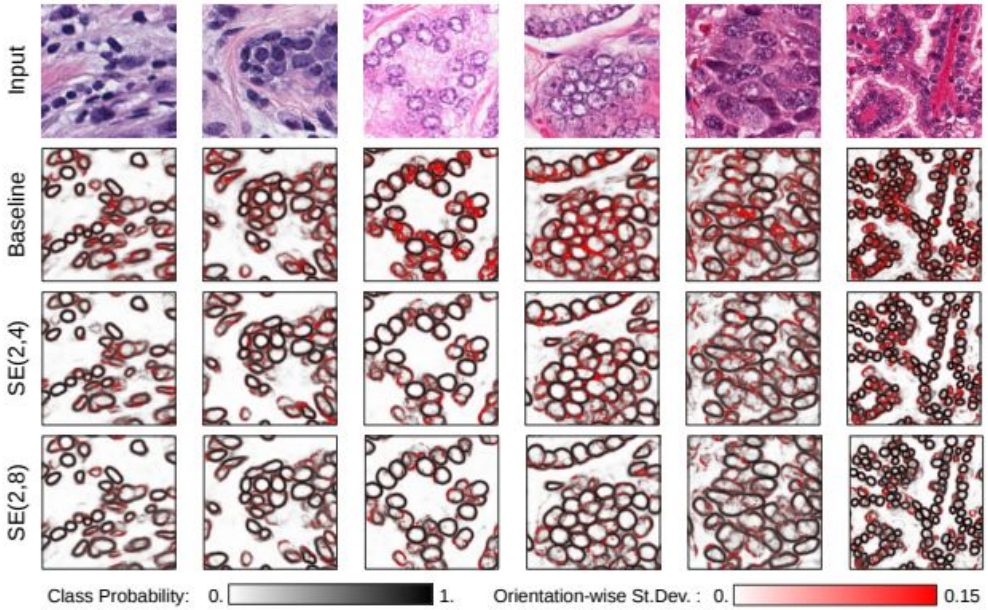


Figure 4: Example of image patches selected from the test set of the nuclei segmentation benchmark (column 1-2: breast tissue, column 3-4: prostate tissue, column 5: kidney tissue, column 6: liver). For each image, and a selection of models, the raw predictions of the nucleus boundary class were computed and stored for the set of rotated inputs using steps of $\pi/8$ rad. Predictions were re-aligned and their means were mapped to gray-scale and the standard deviations of the predictions were mapped to a white-to-red color scale. The overlap of these statistics is shown below each original image. Selected models are the best obtained models that were trained without reduced data regime over repeats (based on their F_1-score).

Conclusion

- Meilleure performance sur les images hispathologiques avec le G-CNN qu'avec des CNNs utilisant les data augmentation
- Meilleure robustesse aux orientations d'entrées
- De moins bonnes performances avec SE(2,16) comparé au SE(2,8) G-CNN
- Trade-off entre la capacité du réseau et l'angle de résolution choisi

Harmonic Networks: Deep Translation and Rotation Equivariance

Sommaire

Présentation générale

- Recours traditionnel à l'**augmentation de données**
  > Features maps peu interprétables
  > Ne capture **pas** les équivariances locales

- Avec les Harmonic Networks (H-Nets)
  > « **Hard baking** » : représenter des rotations à 360° au sein même des filtres
  > Contrainte sur les filtres, qui appartiennent à la famille des **harmoniques circulaires**

Les harmoniques circulaires

1. Définition

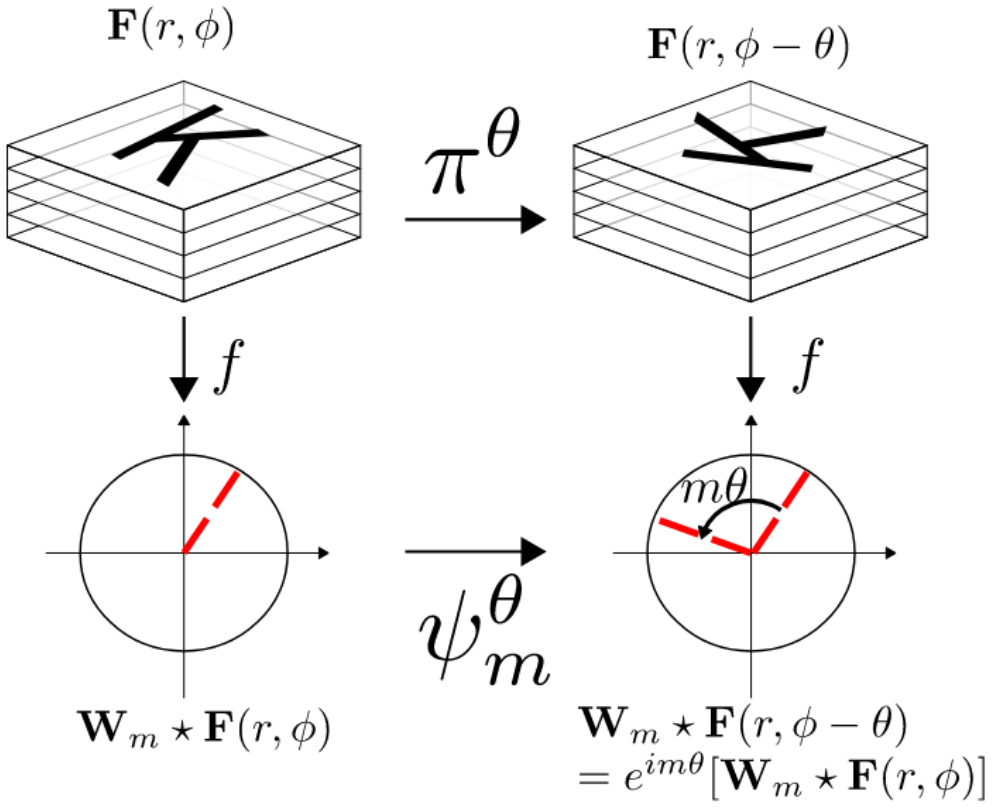
$$\mathbf{W}_m(r,\phi;R,\beta) = R(r)e^{i(m\phi+\beta)}.$$

r,ϕ	Coordonnées spatiales de l'image (polaires)
$m \in \mathbb{Z}$	Ordre de rotation
$\beta \in [0,2\pi)$	Profil radial : contrôle la forme du filtre
$R:\mathbb{R}_+ \to \mathbb{R}$	Terme de phase : détermine la sélectivité à l'orientation du filtre

> Filtres à valeurs complexes
> R et β sont appris pendant l'entraînement

Les harmoniques circulaires

2. Propriété d'équivariance



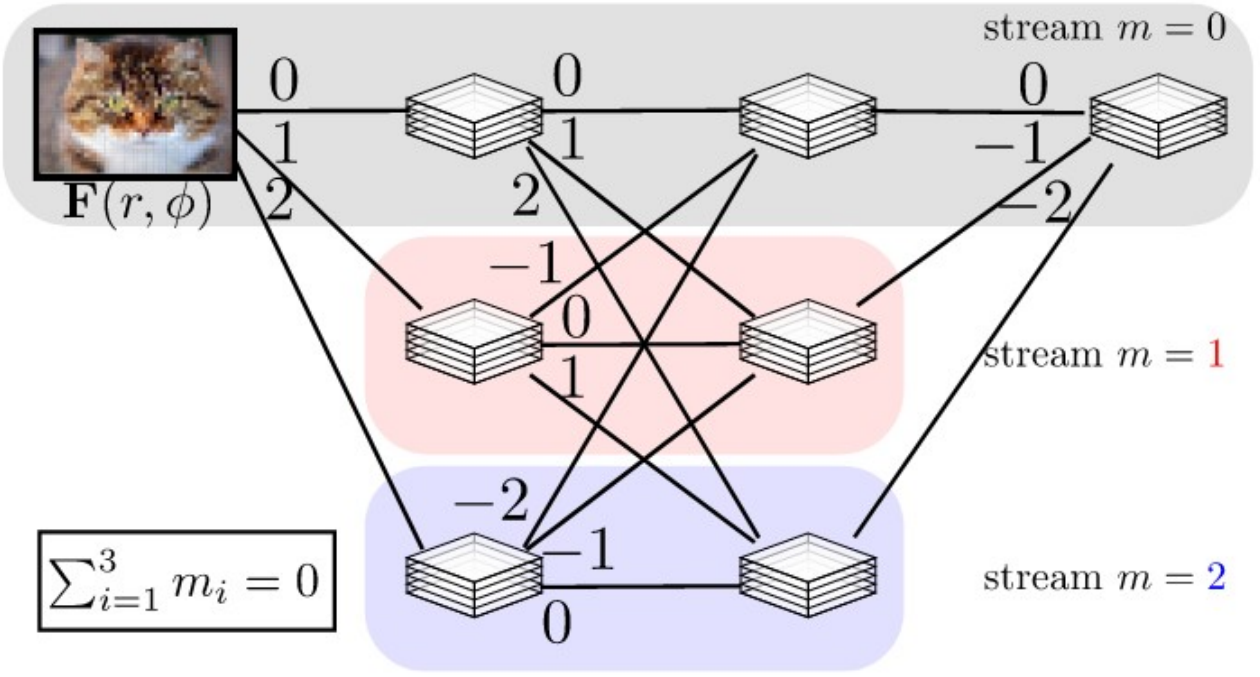
Les harmoniques circulaires

3. Propriétés notables

1. L’enchaînement de deux cross-correlations d’ordres m_1 et m_2 conduit à une réponse d’ordre $m_1 + m_2$

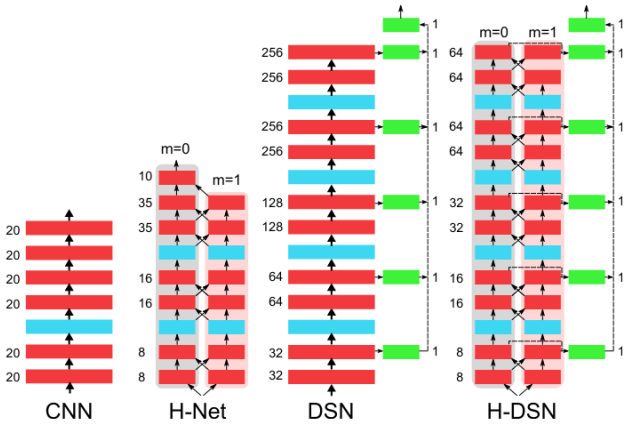
2. **Condition d’équivariance** : la somme de deux réponses d’ordre m reste d’ordre m

Implémentation



Exemple de H-Net à deux couches avec sortie *m=0*

Benchmarks



1. MNIST : Classification

Method	Test error (%)	# params
SVM [18]	11.11	-
Transformation RBM [31]	4.2	-
Conv-RBM [27]	3.98	-
CNN [3]	5.03	22k
CNN [3] + data aug*	3.50	22k
*P*4CNN rotation pooling [3]	3.21	25k
*P*4CNN [3]	2.28	25k
H-Net (Ours)	**1.69**	33k

2. Détection de contours

Method	ODS	OIS	# params
HED, [33]*	0.640	0.650	2346k
HED, low # params [33]*	0.697	0.709	115k
Kivinen et al. [14]	0.702	0.715	-
H-Net (Ours)	**0.726**	**0.742**	116k
CSCNN†, [10]	0.741	0.759	
DeepEdge†, [2]	0.753	0.772	
N^4-Fields†, [8]	0.753	0.769	
DeepContour†, [28]	0.756	0.773	
HED†, [33]	0.782	0.804	2346k
DCNN + sPb†, [15]	**0.813**	**0.831**	

Visualisation des filtres

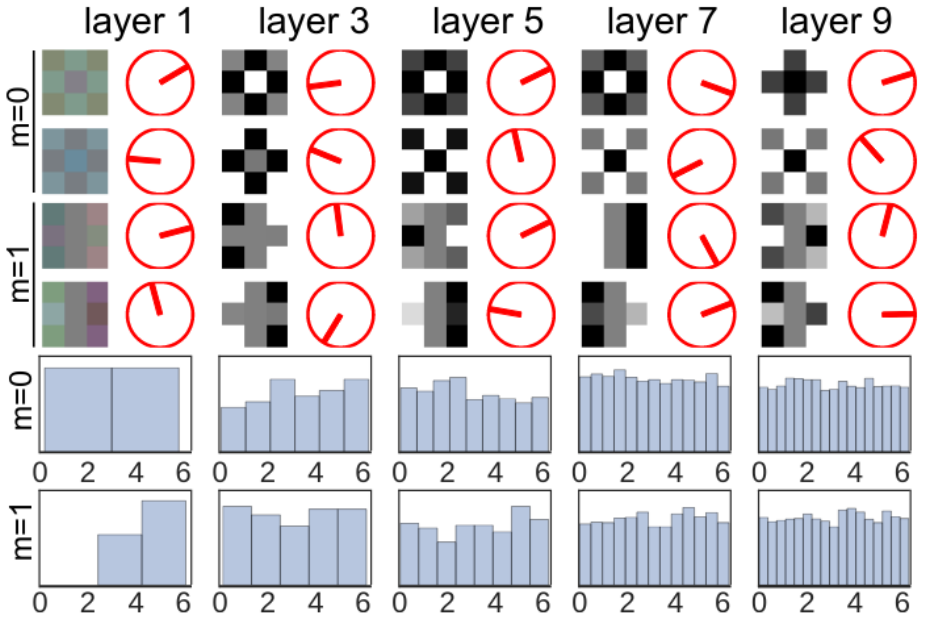


Figure 9. Randomly selected filters and phase histograms from the BSDS500 trained H-DSN. Filter are aligned at $\beta = 0$; and the oriented circles represent phase. We see few filter copies and no blank filters, as usually seen in CNNs. We also see a balanced distribution over phases, indicating that boundaries, and their deep feature representations, are uniformly distributed in orientation.

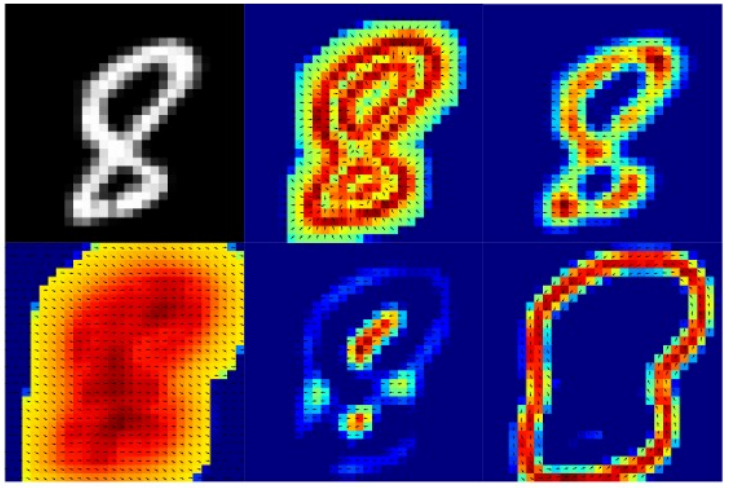


Figure 11. Feature maps from the MNIST network. The arrows display phase, and the colors display magnitude information (jet color scheme). There are diverse features encoding edges, corners, whole objects, negative spaces, and outlines.

Data ablation