Modèles profonds auto-explicables à base de prototypes en vision par ordinateur : application à l'imagerie cellulaire

Martin Blanchard

N° d'ordre NNT : 2025UJMO0061

THÈSE de DOCTORAT
DE L'UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Membre de l'Université de Lyon

Ecole Doctorale N°488
SIS - Sciences Ingénierie Santé

Spécialité de doctorat : Image, vision, signal

Soutenue publiquement le 8 décembre 2025, par :

Martin BLANCHARD

Modèles profonds auto-explicables à base de prototypes en vision par ordinateur : application à l'imagerie cellulaire

---

Devant le jury composé de :

Ronan SICRE, Chaire de professeur junior, IRIT, Université de Toulouse III	Rapporteur
Stefan DUFFNER, Professeur des universités, LIRIS, INSA Lyon	Rapporteur
Laure TOUGNE RODET, Professeure des universités, LIRIS, Université Lumière Lyon 2	Présidente du jury
Christophe DUCOTTET, Professeur des universités, Université Jean Monnet Saint-Etienne	Directeur de thèse
Damien MUSELET, Professeur des universités, L3I, Université de La Rochelle	Co-encadrant de thèse
Olivier DELEZAY, Ingénieur de recherche, Université Jean Monnet Saint-Etienne	Co-encadrant de thèse

Résumé

L'apprentissage profond a récemment connu un développement rapide. De l'analyse d'images au traitement automatique du langage naturel, les champs d'application couvrent de nombreux domaines. Cependant, malgré leurs performances, l'adoption de ces technologies est freinée dans les contextes où les enjeux sont grands : justice, finance, médecine etc. En cause : le manque de compréhension de leur fonctionnement, les réseaux de neurones étant souvent des boîtes noires. Il est difficile de déterminer le raisonnement qui mène à leurs résultats, et les récents progrès n'ont fait que renforcer le besoin de méthodes interprétables. Dans ce contexte, l'interprétabilité d'une méthode qui établirait un diagnostic médical est donc cruciale.

Nos travaux ont pour but de créer une méthode d'analyse d'images de cellules cultivées in vitro, avec une population témoin négatif, et une autre population exposée à une substance d'intérêt ; afin de déterminer automatiquement et sans connaissance préalable, les biomarqueurs d'une atteinte cellulaire due à la substance. La difficulté réside d'une part dans l'absence de connaissances a priori, obligeant à travailler dans un cadre faiblement supervisé, sans étiquetage individuel des cellules mais seulement avec une information globale sur le traitement des cellules ; et d'autre part dans le bruit de ces pseudo-labels. Des cellules exposées au traitement peuvent ne pas réagir et rester saines, tandis que des cellules témoins peuvent s'être dégradées naturellement ou par les manipulations subies. Les cellules qui serviront à notre étude sont des cellules endothéliales de la barrière hémato-encéphalique, que nous avons exposées à des doses croissantes de différents facteurs. Les noyaux cellulaires et l'actine du cytosquelette ont été imagés au microscope en immunofluorescence.

Nous avons exploré les méthodes de classification d'images explicables utilisant des prototypes, c'est-à-dire des représentants typiques de chacune des classes du problème, lesquels servent à déterminer la classe de l'image courante. Parmi ces méthodes, nous avons sélectionné ProtoPNet [27] et l'avons adapté à notre contexte afin d'obtenir des prototypes mettant en lumière des structures cellulaires caractéristiques de nos deux classes. Nous obtenons des prototypes décrivant une structure

d'actine sous forme de filaments pour les cellules saines, et une texture à l'aspect plus granulaire pour les cellules affectées, qui semble traduire une modification du contenu en actine. Ces résultats ont donné lieu à une publication dans la conférence ICIP 2024 [12].

Ces premiers travaux ont aussi motivé le besoin d'une méthode plus souple, avec une visualisation des prototypes plus robuste. Nous proposons pour cela une architecture auto-encodeur variationnel (VAE), pour reconstruire les images liées aux prototypes, utilisant un mélange de Gaussiennes comme a priori pour modéliser les données du problème. Notre modèle baptisé ProtoGMVAE apprend des prototypes plus divers et plus représentatifs de la distribution des données du problème. Ses performances de classification sont légèrement inférieures, mais ce compromis est acceptable au regard des gains en interprétabilité, lesquels sont le véritable but de nos travaux. Ce sont les structures typiques qui nous intéressent, plus qu'une classification complète des cellules. Ces travaux ont été acceptés pour publication à la conférence internationale WACV 2026 [13].

Enfin, nous avons essayé d'appliquer notre méthode ProtoGMVAE à nos images de cellules. Le passage à un contexte d'étude plus réaliste que les jeux de données utilisés comme points de comparaison standards aux autres méthodes nécessite un ajustement des paramètres du modèle et de l'architecture employée. Nous proposons plusieurs adaptations de l'architecture de notre modèle ProtoGMVAE, afin de répondre aux problématiques de ce nouveau cadre. Nous étudions enfin en détail l'explication fournie après apprentissage sur notre base de cellules.

Remerciements

Je tiens en premier lieu à remercier Ronan Sicre et Stefan Duffner d'avoir accepté de rapporter cette thèse, et je leur souhaite une bonne lecture de ce manuscrit. Je veux aussi remercier Laure Tougne Rodet d'avoir accepté le rôle d'examinatrice. J'espère que nos échanges lors de la soutenance seront riches et constructifs.

Je veux aussi remercier tous mes encadrants pendant cette thèse et même avant : Christophe, Damien et Olivier. Ils ont toujours su me guider tout en me laissant beaucoup d'autonomie et de liberté. Merci particulièrement à Christophe et Damien pour leurs conseils en matière d'apprentissage profond et d'analyse d'images. Je n'avais pratiquement jamais travaillé avec des réseaux de neurones auparavant mais grâce à eux j'ai pu apprendre et m'adapter. Damien, je te souhaite le meilleur pour ton nouveau poste à La Rochelle. Merci aussi à Olivier pour tes conseils en matière de biologie cellulaire. J'avais oublié le mot cytoplasme avant cette thèse, donc ton expertise était vraiment cruciale. Merci aussi pour ta patience pendant les périodes où nous travaillions à établir notre méthode d'analyse, sans jamais te fournir de résultats sur les cellules. Travailler avec vous trois a été un plaisir et c'est grâce à vous que j'ai pris le goût de la recherche.

Merci à Zhiguo He pour le prêt du microscope et à Corantin Maurin pour nous avoir aidé à nous en servir et à développer l'acquisition automatique. Sans vous je n'aurais jamais pu avoir d'aussi belles images. Merci aussi à Sabine Palle avec qui nous avions exploré la piste du microscope confocal. Grâce à vous tous j'ai pu aussi faire des manips physiques et sortir un peu de mon bureau.

Merci à Souley qui a travaillé sur la fusion des tranches d'images. Je n'aurais sans doute pas eu le temps de m'investir autant que tu l'as fait pendant ton stage. Merci pour ton excellent travail et je te souhaite bonne chance pour ta thèse.

À mes compagnons d'infortune du bureau E108, merci d'avoir accepté un intrus de l'équipe image, mais j'étais quand même là avant. Je vous souhaite à tous bonne chance pour la fin de vos thèses, surtout Thibaud et Richard qui soutiennent aussi cette année (ou presque), et merci d'avoir toujours été des super collègues. Merci aussi à Isidore, Cyril et Tristan, mes vrais collègues de l'équipe image, vous avez

été encore plus super.

Un grand merci à tous mes copains, de la MLUD, de Télécon ou d'ailleurs. Il faudra fêter ça dignement après la soutenance et je compte sur vous. Un merci spécial à Tristan et Léo avec qui je suis désormais co-auteur en plus d'être ami.

Pour terminer, merci à ma famille, même si ça n'a pas toujours été facile de comprendre ce que je faisais, vous allez voir, tout est plus clair dans le manuscrit.

Merci à tous !

Table des matières

Résumé i
Remerciements iii
Table des figures xiii
Liste des tableaux xiv

1 Introduction **1**
1.1 Contexte général 2
1.2 Contexte biologique 4
1.3 Objectifs 5
1.4 Contributions 7
1.5 Organisation du document 8

2 Étude bibliographique **9**
2.1 Introduction 10
2.2 Notions préalables 10
2.2.1 Architecture des réseaux de neurones 10
2.2.2 Entraînement d'un réseau de neurones 14
2.3 Explications post-hoc 16
2.3.1 Cartes d'attribution 16
2.3.2 Descriptions textuelles et VLMs 21
2.4 Modèles profonds intrinsèquement interprétables 23
2.4.1 Approches par prototypes 26
2.5 Applications dans un contexte biologique 29
2.6 Résumé 31

3 Jeu de données **33**
3.1 Introduction 34
3.2 Culture cellulaire 34
3.3 Marquage et acquisition 35
3.4 Prétraitement 37
3.4.1 Débruitage 37

3.4.2 Incorporation de l'information des tranches 37
3.5 Conclusion 41

4 Explicabilité de ProtoPNet pour l'imagerie cellulaire 43
4.1 Introduction 45
4.2 Méthodologie 47
4.2.1 Formulation du problème 47
4.2.2 Notre approche 48
4.3 Présentation de ProtoPNet 49
4.3.1 Principe 49
4.3.2 Entraînement 50
4.3.3 Visualisation des prototypes 51
4.4 Adaptations proposées 52
4.4.1 Segmentation et masquage 52
4.4.2 Contrôle du champ réceptif 53
4.4.3 Contrôle des prototypes 54
4.5 Résultats expérimentaux 54
4.5.1 Jeu de données 54
4.5.2 Paramètres expérimentaux 57
4.5.3 Segmentation et influence du masquage 57
4.5.4 Influence du champ réceptif 59
4.5.5 Similarités avec les prototypes 60
4.5.6 Interprétation biologique 61
4.6 Discussion et conclusion 61

5 ProtoGMVAE : Un VAE avec un a priori mélange de Gaussiennes pour l'explicabilité par les prototypes 65
5.1 Introduction 66
5.2 Auto-Encodeur Variationnel 69
5.3 ProtoVAE 71
5.4 ProtoGMVAE 74
5.4.1 Motivations 74
5.4.2 Du VAE au GMVAE 75
5.4.3 Architecture 77
5.5 Résultats Expérimentaux 79
5.5.1 Jeux de données et implémentation 79
5.5.2 Comparaison des performances en classification 79
5.5.3 Espace latent et visualisation des prototypes 80
5.5.4 MNIST avec classes déséquilibrées 83
5.5.5 Nombre de composantes 85
5.6 Évaluation des prototypes 86

5.7 Conclusion 91

6 L'explicabilité de ProtoGMVAE à l'épreuve du réel : application à l'imagerie cellulaire 93
6.1 Introduction 94
6.2 Analyse préliminaire 94
6.2.1 Formulation du problème 94
6.2.2 Application de ProtoVAE 96
6.2.3 Application de PanVAE 97
6.2.4 Application de ProtoGMVAE 97
6.3 Nouvelles stratégies 98
6.3.1 Modifications d'architecture 98
6.3.2 Visualisation des prototypes 99
6.4 Résultats expérimentaux 100
6.4.1 Analyse des prototypes 101
6.4.2 Espaces latents 102
6.5 Conclusion 104

7 Conclusion 106
7.1 Rappel des objectifs 107
7.2 Contributions 107
7.2.1 Jeu d'images de cellules 107
7.2.2 Première analyse des cellules par réseau de neurones interprétable 108
7.2.3 ProtoGMVAE 109
7.2.4 Seconde analyse des cellules à l'aide de ProtoGMVAE 109
7.3 Perspectives 110
7.3.1 Amélioration de ProtoGMVAE 110
7.3.2 Approfondissement de l'analyse des images de cellules 111

Table des figures

1.1 Structure anatomique de la barrière hémato-encéphalique. La paroi de tous les vaisseaux sanguins est formée par une fine monocouche de cellules endothéliales microvasculaires cérébrales spécialisées, reliées entre elles par des jonctions serrées, et qui agissent comme une barrière physique et métabolique. Elles sont entourées par une membrane basale vasculaire, des péricytes, une membrane basale parenchymateuse et des pieds astrocytaires, qui contribuent tous directement ou indirectement à la fonction de la BHE. Figure et légende reprises de [86] 3
1.2 Schéma d'une cellule et de son cytosquelette. Les filaments d'actine sont en vert, les microtubules en rouge, le noyau en bleu et au centre de la cellule, le centrosome en vert. ©Manuel Théry, CEA 4
1.3 Images de cellules endothéliales de la BHE générées lors de notre acquisition de données. Canal rouge : actine, canal bleu : noyaux cellulaires. (a) Cellules non-traitées, (b) Cellules traitées 5

2.1 Schéma de l'architecture d'un MLP à 4 couches cachées et fonctionnement d'un neurone artificiel. Figure reprise de [91] 12
2.2 Fonctionnement d'une convolution entre deux images (entrée et noyau). La valeur d'un pixel en sortie est égale à la somme des pixels de l'image d'entrée pondérée par les valeurs du noyau. Si l'on ne gère pas les effets de bord, en remplissant avec des zéros par exemple, l'image de sortie est plus petite que l'image d'entrée lorsque le noyau centré sur le pixel d'application ne rentre pas entièrement dans l'image. 13
2.3 Illustration du champ réceptif d'un réseau de neurones convolutif. Plus le réseau est profond plus celui-ci va avoir tendance à être grand. 14
2.4 Visualisation des caractéristiques apprises par un modèle entraîné sur ImageNet par la méthode des réseaux déconvolutifs. [125] 17
2.5 Exemples d'images qui activent le plus les classes "oie", "autruche" et "limousine" d'un CNN entraîné sur ILSVRC-2013 (ImageNet) [107] 18

2.6 Schéma de l'architecture utilisée par Zhou et al. [131] pour obtenir leur CAM. Il s'agit d'une architecture d'encodeur CNN classique, se terminant par une opération de pooling moyen global (GAP) avant la couche linéaire de classification. 19
2.7 Schéma du fonctionnement de RISE [92]. On génère une série de masques aléatoires que l'on applique à l'entrée pour connaître l'impact des pixels sur le score de classification. 20
2.8 L'architecture utilisée dans [48] pour générer des explications visuelles à une classification. Des réseaux récurrents LSTM sont entraînés pour produire des descriptions des images d'entrée qui soient discriminantes en exploitant la classe prédite par le CNN. 21
2.9 Exemple de décision de classification par le modèle de [79] (gauche) et de la justification de la décision par rapport à un VLM qui n'utilise pas de descripteurs, ici CLIP (droite) avec la similarité entre l'image et les descripteurs liés à chaque classe renseignée dans les barres. 22
2.10 Les cartes de salience obtenues sur une même image pour deux classes très différentes peuvent être très proches. Dans ce cas l'explication qu'elles apportent est très limitée et se cantonne à la zone de l'image qui active le plus les neurones et pas la manière dont elle les active. Figure issue de [99] 23
2.11 Effet du paramètre B de la méthode B-cos [17] sur des problèmes simples de classification. Au centre, la perte d'entropie binaire croisée en fonction de l'angle du vecteur poids du neurone de classification. Les colonnes latérales sont des visualisations du problème de classification et des poids optimaux (flèches) en fonction de B. Bien que les points soient placés au même endroit, la direction optimale du poids varie fortement en fonction de B. 24
2.12 Architecture d'un réseau de neurones auto-explicable [4]. Le réseau est composé de trois parties : un encodeur qui extrait des concepts interprétables des images d'entrée, un paramétreur qui apprend à générer des scores de pertinence, et une fonction d'agrégation qui combine concepts et pertinence afin de classifier. 25
2.13 Fonctionnement de la méthode des sacs de mots visuels. L'image est représentée par un histogramme de mots visuels piochés dans un dictionnaire, correspondants à tous les éléments visuels qui existent au sein de la base de données d'images. 27
2.14 Explication du raisonnement de ProtoPNet [27] pour une image donnée 29
2.15 Architecture proposée par [128]. 30

2.16 Comparaison des prototypes obtenus par ProtoPNet [27] (haut) et XProtoNet [56] (bas). Les boîtes jaunes montrent les contours des prototypes, les boîtes correspondent à la vérité terrain pour les images positives. XProtoNet apprend des prototypes avec des formes variables ce qui permet de mieux capter les zones importantes. 31

3.1 Exemple d'acquisition d'images automatisée. (a) montre l'un des puits de la plaque de culture et le quadrillage correspond à la zone à imager. Chaque carré correspond à une zone de 2048×2048 pixels dans l'image finale. (b) est le résultat de cette acquisition. Le canal rouge contient l'actine, le bleu contient les noyaux. 36
3.2 Fonctionnement de la pyramide laplacienne. D'abord une pyramide gaussienne est calculée avec un filtrage gaussien pour sous-échantillonner les images successivement avec un facteur 2. Puis par soustraction de deux paliers successifs où l'on sur-échantillonne le palier le plus bas, on obtient le laplacien. Toutes les images sont montrées avec la même taille pour mieux voir le contenu. 39
3.3 Exemples de résultats de fusion de deux images avec des focus différents (a) et (b) par pyramide laplacienne (LP)[22, 23] (c) et Quadtree [10] (d) 40
3.4 Zooms sur les résultats de fusion par (a) pyramide laplacienne (LP)[22, 23] et (b) Quadtree [10] 41

4.1 Le but de ces travaux est d'entraîner un réseau profond avec des labels bruités ("Exposé", "Référence") pour chaque cellule individuelle de manière à pouvoir extraire les structures typiques des cellules saines et affectées. Ces structures sont illustrées par les prototypes, exploités par le réseau durant la tâche de classification. . . 46
4.2 L'architecture proposée basée sur ProtoPNet avec trois adaptations principales : un masque de segmentation comme seconde entrée pour empêcher le réseau d'utiliser le fond des images, un champ réceptif limité pour assurer la localité des résultats, et le contrôle des poids de la couche de classification pour renforcer la diversité des prototypes. 48
4.3 Architecture de ProtoPNet [27] 49
4.4 Formes attendues de (gauche) la moyenne pour chaque population de la similarité maximum avec chacun des prototypes et (droite) la différence entre ces moyennes de similarité entre les populations "Exposée" (en orange) et "Référence" (en bleu). 55

4.5 Exemple de l'activation d'un prototype (b) créé par le modèle sur une image non-segmentée (a). Le patch d'arrière plan active le prototype. Sur une image segmentée (c), l'activation obtenue en utilisant le masque de segmentation(d) est localisée uniquement dans la cellule. 56
4.6 Résultat de segmentation obtenu par Cellpose [110] sur une image de cellules 58
4.7 Comparaison de la visualisation d'un prototype pour une carte d'activation donnée (a) générée en utilisant (b) la plus petite boîte englobant au moins le 95ème pourcentile d'activation, (c) une boîte de la taille du champ réceptif centrée autour de la plus haute valeur d'activation. Résultats obtenus en utilisant VGG-11 comme encodeur. 59
4.8 Différence entre les similarités aux prototypes comme dans la figure 4.4 61
4.9 Mosaïques combinant la visualisation d'un prototype et les 20 patches les plus proches dans la base d'entraînement. (a) montre le résultat pour un prototype de la classe "non-traitée" et (b) montre le résultat pour un prototype de la classe "traitée au TNF-α". 62

5.1 Distribution latente et prototypes appris avec ProtoGMVAE sur la base de données MNIST [67] 67
5.2 Visualisation en 2D de la variété apprise par un VAE entraîné sur MNIST, avec reconstruction des images associées aux points. L'information entre deux points est continue. 69
5.3 Architecture de ProtoVAE [40] 72
5.4 Couverture de l'espace latent par les prototypes après 20 époques d'entraînement de (a) ProtoVAE et (b) PanVAE sur la base de données FashionMNIST [122] avec 5 prototypes pour la classe "sac". En bas : Reconstruction des prototypes. Figure tirée de [61] 74
5.5 Schéma de l'architecture proposée 77
5.6 Colonne de gauche : Projections UMAP de l'espace latent appris sur FashionMNIST, avec les prototypes superposés comme des carrés. Colonne de droite visualisation des prototypes appris. Pour notre approche, les prototypes sont classés par le poids associé le plus élevé dans la couche de classification et numérotés pour faciliter la comparaison. 100 prototypes sont appris pour chacune des méthodes. 81
5.7 Projections UMAP des espaces latents appris avec une classification binaire entre chiffres pairs et impairs en déséquilibrant les classes initiales de MNIST, avec (a) ProtoVAE (prototypes jaunes pour la classe "Impair", bleu foncé pour "Pair"), et (b) notre approche. Les labels détaillés des chiffres sont affichés pour plus de clarté. 83

5.8 Prototypes obtenus avec une classification binaire entre chiffres pairs et impairs en déséquilibrant les classes initiales de MNIST avec (a) ProtoVAE et (b) ProtoGMVAE. Les prototypes de ProtoVAE sont assez mal reconstruits et représentent des formes abstraites pour la plupart d'entre eux. Les prototypes de ProtoGMVAE capturent toute la diversité des chiffres qui existent dans le jeu de données et les reconstruisent correctement. 84
5.9 Pour chaque prototype (axe x), nombres d'images de la base de données de test qui sont plus proches de ce prototype que de n'importe quel autre. 85
5.10 Reconstruction des prototypes appris par ProtoVAE sur les quatre jeux de données étudiés 88
5.11 Reconstruction des prototypes appris par PanVAE sur les quatre jeux de données étudiés 89
5.12 Reconstruction des prototypes appris par ProtoGMVAE sur les quatre jeux de données étudiés 90

6.1 Exemples d'images analysées. (a) montre des images provenant de la population de référence non-traitée, (b) montre des images provenant de la population traitée au TNF-α. 95
6.2 (a) Prototypes appris par ProtoVAE sur les images de cellules. 25 prototypes par classe. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par ProtoVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$ 96
6.3 (a) Prototypes appris par PanVAE sur la base des images de cellules. 10 prototypes par classe. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par PanVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$ 98
6.4 (a) Prototypes appris par ProtoGMVAE sur la base des images de cellules. 50 prototypes au total. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par ProtoGMVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$ 99
6.5 Visualisation des prototypes n'étant associés à aucune population particulière. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype. . 101
6.6 Visualisation des prototypes associés à la population $\mathcal{P}^{\text{Ref}}$. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype. Prototypes classés par ordre décroissant de score de classification. 102

6.7 Visualisation des prototypes associés à la population $\mathcal{P}^{\text{Exp}}$. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype. Prototypes classés par ordre décroissant de score de classification. 103
6.8 Projection UMAP de l'espace latent en $z|x, y$ appris par ProtoGM-VAE sur la base des images de cellules. 104
6.9 Projection UMAP de l'espace latent en $y|x$ appris par ProtoGMVAE sur la base des images de cellules. 105

Liste des tableaux

3.1 Résumé de la taille en pixels des images acquises sur les puits de chaque condition de culture cellulaire 36
3.2 Scores de Brenner moyens obtenus sur l'image reconstruite et temps d'exécution en fonction des différentes méthodes de fusion utilisées sur la base de 30 paires d'images. La pyramide Laplacienne semble à la fois la plus rapide et la meilleure en terme de focus. 38

4.1 Précisions obtenues avec les différentes configurations expérimentales 60

5.1 Comparaison du taux de classification correcte obtenu sur 4 jeux de données par différentes méthodes dont la nôtre. Pour toutes les méthodes, 50 prototypes sont utilisés pour MNIST et SVHN, et 100 pour FashionMNIST et CIFAR10. 80
5.2 Précision et perte de reconstruction obtenues par notre modèle sur FashionMNIST en fonction du nombre de composantes K dans le mélange de gaussiennes. 86
5.3 Distance moyenne entre une image d'entrée et le prototype le plus proche, mesurée dans l'espace visuel 87
5.4 Moyenne des variances intra-classe des prototypes, mesurée dans l'espace visuel. Les valeurs marquées d'une étoile (*) ont été calculées après avoir retiré les prototypes qui n'étaient pas reconstruits correctement. 87

1

Introduction

Outline

1.1 Contexte général 2
1.2 Contexte biologique 4
1.3 Objectifs 5
1.4 Contributions 7
1.5 Organisation du document 8

1.1 Contexte général

L'intelligence artificielle prend une place de plus en plus grande dans notre société. Les récents progrès des modèles génératifs, que ce soit les grands modèles de langage pour le texte, ou les modèles de diffusion pour les images, ont permis de montrer au grand public les capacités de ces modèles profonds. Avant cela, les réseaux de neurones avaient déjà révolutionné l'analyse d'images par ordinateur, en atteignant des scores de classification très élevés sur des challenges comme ImageNet [100].

Mais si les performances de ces méthodes ne sont plus à démontrer, plusieurs problèmes se posent néanmoins pour leur adoption à plus grande échelle. Parmi ces problèmes, celui de l'interprétabilité des résultats produits par un réseau de neurones est un des plus importants. Dès lors que l'on souhaite utiliser ces technologies dans des cas d'application concrets, il devient crucial d'être sûr que les résultats sont fondés, et pas seulement une hallucination, ou basés sur un biais au sein des données.

En Europe, le RGPD [1] introduit la notion de "droit à une explication" dans le cadre des décisions prises par des algorithmes. Plus récemment, l'AI Act [2] entré en vigueur en 2024, reprend cette notion et impose que les décisions prises par des intelligences artificielles soient transparentes dans les domaines dits "à hauts risques". Cette exigence entraîne le besoin de nouveaux algorithmes et modèles plus adaptés aux besoins spécifiques des domaines aux enjeux importants comme la médecine, mais aussi la finance ou la justice, et plus généralement partout où ces nouvelles technologies pourraient avoir un impact majeur sur la vie des gens.

La question de comprendre le raisonnement des réseaux de neurones, ainsi que leurs résultats, a toujours été au cœur des recherches autour de l'intelligence artificielle [30, 5, 111]. C'est pour répondre à ces besoins que le domaine de l'intelligence artificielle explicable (eXplainable Artificial Intelligence, abrégé en XAI) a naturellement vu son développement accompagner celui de l'apprentissage profond. L'exigence de transparence ne se traduit pas toujours par des algorithmes exposant clairement leur raisonnement interne, mais plutôt par des explications renforçant la confiance des utilisateurs dans leur décision. En matière d'explicabilité, on peut distinguer plusieurs niveaux, dépendants de plusieurs facteurs, comme l'audience à qui l'explication est destinée, le contexte d'utilisation, la législation ou à quel point le domaine est critique [78]. On distingue généralement deux catégories d'approches explicables : les explications post-hoc, générées après l'entraînement d'un modèle ; et les modèles auto-explicables, pour lesquels les explications font partie intégrante du modèle et de l'entraînement.

Nos travaux s'inscrivent dans ce cadre d'intelligence artificielle auto-explicable. Le but est de proposer une architecture de réseau de neurones capable d'expliquer ses décisions en matière de classification d'images. Ce modèle sera appliqué à des données d'imagerie cellulaire obtenues par microscopie en immunofluorescence, avec des cellules séparées en deux populations : l'une servant de témoin et l'autre exposée à un traitement. Le but est de se servir des explications fournies par le modèle pour identifier et comprendre les changements structurels induits par le traitement. Cette thèse est menée conjointement par le laboratoire Hubert Curien, pour son expertise en analyse d'images et en apprentissage profond, et le laboratoire Sainbiose qui apporte lui ses connaissances en biologie cellulaire.

Nous nous sommes concentrés sur des méthodes dont l'objectif est à la fois de conforter les décisions qui pourraient être prises sur la base des données étudiées, mais aussi de découvrir des motifs, des structures caractéristiques qui représentent fidèlement les différentes classes représentées. Notre but est d'obtenir une méthode qui puissent s'appliquer à de nombreux contextes, pas seulement en biologie cellulaire.

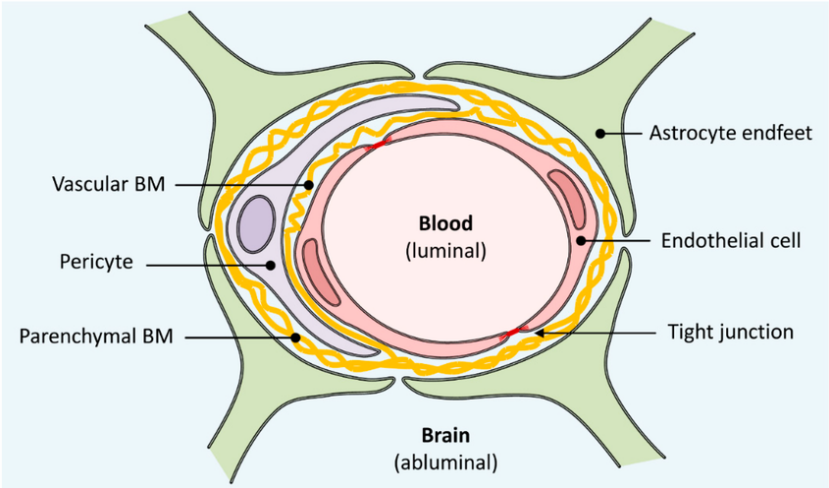


FIGURE 1.1 – Structure anatomique de la barrière hémato-encéphalique. La paroi de tous les vaisseaux sanguins est formée par une fine monocouche de cellules endothéliales microvasculaires cérébrales spécialisées, reliées entre elles par des jonctions serrées, et qui agissent comme une barrière physique et métabolique. Elles sont entourées par une membrane basale vasculaire, des péricytes, une membrane basale parenchymateuse et des pieds astrocytaires, qui contribuent tous directement ou indirectement à la fonction de la BHE. Figure et légende reprises de [86]

1.2 Contexte biologique

Nos travaux s'intéressent à l'étude de la barrière hémato-encéphalique (BHE). Comme son nom l'indique, la BHE assure l'étanchéité entre le flux sanguin dans les vaisseaux et le compartiment cérébral, le protégeant ainsi de diverses molécules toxiques présentes dans le sang [14].

La BHE est composée de différent types de cellules (Figure 1.1), dont les cellules endothéliales, qui constituent le corps des vaisseaux sanguins. Ces cellules sont reliées entre elles par des protéines de jonction serrée qui assurent une certaine imperméabilité. Ces protéines sont fortement associées au cytosquelette des cellules (filaments d'actine) [70].

Nous avons concentré notre analyse sur le cytosquelette et particulièrement l'actine. L'actine se trouve sous forme de filaments souples, répartis en périphérie de la cellule (Figure 1.2). C'est ce qui permet aux cellules de se déformer et de se déplacer. Le choix de l'actine se base sur plusieurs raisons. Tout d'abord c'est une protéine présente dans toutes les cellules du corps. Si notre approche fonctionne avec l'endothélium de la BHE, alors on peut penser qu'elle fonctionne sur n'importe quel autre type de cellule. Deuxièmement, l'actine joue un rôle crucial dans le maintien de la forme de la cellule [33], et toute modification dans la structure filamentaire pourrait mener à une perte fonctionnelle de la BHE. Les jonctions serrées sont également très importantes puisque leur ouverture est l'une des premières étapes qui peuvent mener à l'agression des cellules cérébrales. C'est pourquoi le contrôle de ces jonctions, ainsi que des changements dans l'architecture cellulaire (cytosquelette) est particulièrement pertinent pour évaluer la vulnérabilité des cellules neurales.

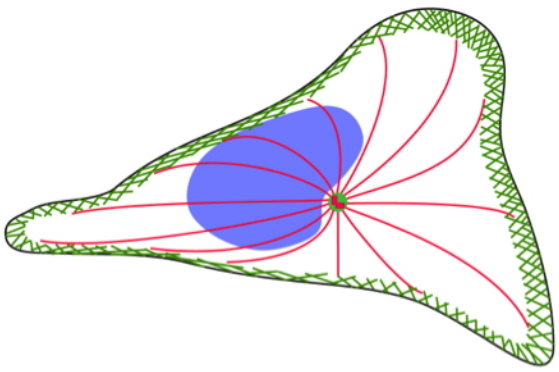
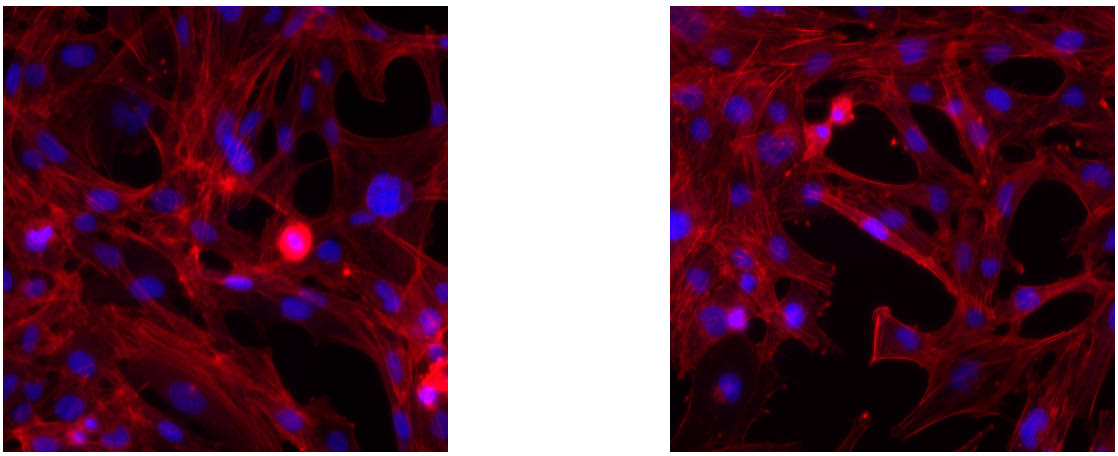


FIGURE 1.2 – Schéma d'une cellule et de son cytosquelette. Les filaments d'actine sont en vert, les microtubules en rouge, le noyau en bleu et au centre de la cellule, le centrosome en vert. ©Manuel Théry, CEA

Beaucoup de travaux liés à l'explicabilité se contentent d'appliquer une méthode à des bases de données classiques d'analyse d'images. Il nous semblait important d'aller au-delà de ce cadre assez artificiel, éloigné des difficultés que l'on peut rencontrer lorsque l'on étudie des cas réels, où de nombreux paramètres ne sont pas maîtrisés, ni maîtrisables.

Dans l'application des réseaux de neurones et de l'apprentissage profond à des problématiques liées à la biologie ou à la santé, la plupart des études se concentrent avant tout sur la détection de pathologies par la classification d'images, et peu s'intéressent à la découverte automatisée de structures. En outre, les images analysées proviennent souvent de bases de données publiques, rassemblant des radios, des scans IRM etc. mais il n'existe pas de base de données ouverte adaptée à notre étude.

La figure 1.3 montre le type d'images sur lesquelles nous souhaitons travailler. Ces images sont issues de notre propre base d'images, que nous avons générée pendant la thèse. Nous reviendrons en détail sur ces images dans les chapitres suivants.



(a) (b)

FIGURE 1.3 – Images de cellules endothéliales de la BHE générées lors de notre acquisition de données. Canal rouge : actine, canal bleu : noyaux cellulaires. (a) Cellules non-traitées, (b) Cellules traitées

1.3 Objectifs

L'objectif de cette thèse est de concevoir une architecture de réseau de neurones capable d'expliquer ses décisions pour la classification d'images. La classification n'est en réalité ici qu'un prétexte afin de découvrir les informations, sous formes

de structures ou de motifs, qui sont typiques d'une classe. Ce sont ces structures qui nous intéressent avant tout, la classification sert à confirmer leur spécificité à l'une des classes du problème.

Le but est d'appliquer ce modèle à des données d'imagerie des cellules de la barrière hémato-encéphalique, obtenues par microscopie en immunofluorescence et séparées en deux populations, l'une servant de témoin et l'autre exposée à un traitement. Nous pourrions ensuite nous servir des explications fournies par le modèle pour comprendre les changements structurels induits par le traitement.

On peut décomposer cet objectif global en trois points que nous souhaitons atteindre :

- Concevoir un modèle neuronal capable de fournir une explication sur la classification d'images.
- Disposer d'une méthode d'identification des structures visuelles spécifiques des données analysées.
- Avoir un processus d'explication du résultat de classification basé sur les structures spécifiques identifiées, qui permette de tirer une information biologique sur l'état des cellules.

Mais plusieurs contraintes limitent notre champ d'action. L'idée à terme étant d'utiliser la méthode développée pour étudier des substances dont l'effet est inconnu, il est impossible d'utiliser des connaissances a priori pour guider l'apprentissage du modèle. La seule information à notre disposition est l'exposition des cellules au traitement, ou non, qui servira de label global pour toutes les cellules d'une population de la culture. Dans ce contexte, l'annotation individuelle des images de cellules pour indiquer celles qui auraient réagi positivement au traitement est inenvisageable, de par l'absence d'information préalable donc, mais aussi par l'aspect chronophage de la tâche. Cela nous contraint donc à des stratégies d'apprentissage non-supervisées (où le réseau de neurones apprend seul, sans indication) ou tout du moins faiblement supervisées (où les informations données au réseau de neurones sont limitées).

Cette absence d'annotations individuelles des cellules apporte une autre difficulté. Pendant la culture cellulaire, certaines des cellules qui ne sont pas traitées, afin de servir de témoins négatifs pour l'analyse, peuvent mourir naturellement ou à la suite des différentes manipulations pour la culture et le marquage préalable à l'acquisition des images. Ces cellules auront donc une apparence de cellule endommagée, mais durant l'entraînement de notre réseau de neurones celles-ci seront quand même désignées comme des cellules saines, puisque nous nous servons de l'information globale sur l'exposition des cellules. A l'inverse, des cellules qui ont

été exposées au traitement peuvent ne pas y réagir et rester saines. En définitive, les informations apportées par les labels globaux sont imparfaites et cela rejoint des problématiques comme l'apprentissage à l'aide de labels bruités.

1.4 Contributions

Les travaux de cette thèse, financés par une subvention publique de l'Agence Nationale de la Recherche (ANR) dans le cadre du plan d'investissement « France 2030 », dont la référence est EUR MANUTECH SLEIGHT - ANR-17-EURE-0026, ont permis d'apporter les contributions suivantes :

- Une base d'images de cellules endothéliales de la barrière hémato-encéphalique, séparées en différentes populations : témoins, traitées au TNF-α (Facteur de Nécrose Tumorale α) et traitées à l'IL-1β (InterLeukine-1β), à des doses croissantes (1 ng/mL, 10ng/mL, 50ng/mL et 100ng/mL)
- Une première analyse des images à l'aide de ProtoPNet [27], qui a permis de mettre en lumière des structures typiques des cellules saines et des cellules affectées par le traitement au TNF-α à la dose maximale. Les résultats obtenus ont été publiés et présentés à la conférence ICIP 2024 [12].
- Une nouvelle architecture de réseau de neurones auto-explicable pour la classification d'images, basée sur un auto-encodeur variationnel, baptisée ProtoGMVAE, utilisant un mélange de gaussiennes pour modéliser les données, et se servant des différentes composantes du mélange comme prototypes dans le cadre de l'explicabilité. Cette nouvelle méthode propose des performances en classification comparables aux autres méthodes d'explicabilité par les prototypes avec des architectures auto-encodeur variationnel, tout en apprenant des prototypes de meilleure qualité, améliorant ainsi l'interprétabilité. Ces travaux ont été acceptés pour publication à la conférence internationale WACV 2026 [13].
  Le code de ProtoGMVAE est disponible publiquement à l'adresse suivante : https://github.com/martin-blcd/ProtoGMVAE
- Une seconde analyse des images cellulaires, cette fois-ci à l'aide de ProtoGMVAE. Nous avons pu mettre en lumière certains défauts de ProtoGMVAE lorsqu'il est appliqué à un cas d'étude plus réaliste, et trouver de nouvelles stratégies pour montrer l'applicabilité de notre méthode. Cette analyse a permis de confirmer les premiers résultats obtenus à l'aide de ProtoPNet.

1.5 Organisation du document

Le reste de cette thèse est organisé en six chapitres :

- Le chapitre 2 présente une étude bibliographique sur le domaine de l'intelligence artificielle explicable. Nous présenterons les catégories d'explications communément admises et les méthodes les plus utilisées actuellement. Nous ferons également un tour d'horizon des applications de ces méthodes dans un contexte biomédical se rapprochant du nôtre.
- Le chapitre 3 est un court chapitre qui présente les données cellulaires que nous avons récoltées et les pré-traitements utilisés pour rendre les images plus facilement exploitables dans la suite de nos opérations.
- Le chapitre 4 propose de se concentrer sur l'une des principales méthodes auto-explicables identifiée au chapitre 2, ProtoPNet [27], et d'étudier son application à notre problème biologique. Nous présenterons les différentes modifications que nous avons dû faire afin d'adapter le modèle à nos données cellulaires et au type d'explication que nous souhaitons.
- Le chapitre 5 introduit une autre approche par prototype : ProtoVAE [40]. Celle-ci est basée sur une architecture de réseau de neurones appelées auto-encodeur variationnel (en anglais Variational Auto-Encoder ou VAE) que nous détaillerons. Puis nous proposerons une extension de ProtoVAE, ProtoGMVAE utilisant des mélanges de Gaussiennes pour mieux modéliser les données d'apprentissage.
- Le chapitre 6 sera dédié à l'application de ProtoGMVAE à notre cas d'étude bio-cellulaire. Nous mettrons notre méthode à l'épreuve d'un contexte réel, et nous proposerons des adaptations pour mieux répondre à la problématique spécifique de ces images, pour montrer comment ProtoGMVAE peut être utilisé pour des vrais cas d'études.
- Enfin, le chapitre 7 sera une conclusion sur les travaux de cette thèse, avec plusieurs propositions pour d'éventuelles futures recherches.

2

Étude bibliographique

Outline

2.1 Introduction 10
2.2 Notions préalables 10
2.2.1 Architecture des réseaux de neurones 10
2.2.2 Entraînement d'un réseau de neurones 14
2.3 Explications post-hoc 16
2.3.1 Cartes d'attribution 16
2.3.2 Descriptions textuelles et VLMs 21
2.4 Modèles profonds intrinsèquement interprétables . . . 23
2.4.1 Approches par prototypes 26
2.5 Applications dans un contexte biologique 29
2.6 Résumé 31

2.1 Introduction

Dès l'introduction des CNNs en vision par ordinateur, comprendre leurs décisions a été un enjeu crucial. Les modèles comptant souvent plusieurs milliers de paramètres, ils sont des "boîtes noires" dont le fonctionnement est trop complexe pour être simplement décodé. De nombreuses méthodes ont vu le jour afin de faire la lumière sur le processus interne des modèles menant à la décision finale. Traditionnellement [114, 3, 8, 6, 129, 103], on classe ces méthodes en deux grandes catégories : les méthodes dites "post-hoc" qui visent à expliquer un modèle "boîte noire" après qu'il ait été entraîné, en analysant les caractéristiques apprises par celui-ci ; et les méthodes qui consistent à construire dès le départ un modèle intrinsèquement interprétable. Ici, il est question de mitiger la complexité des modèles pour rendre leur fonctionnement compréhensible par un humain, ou de rendre localement interprétable chacune ses parties [43]. On parle dans ce second cas de "boîte transparente" ou de "boîte de verre", en opposition aux modèles "boîtes noires".

Nous présenterons d'abord les principales méthodes d'explicabilité post-hoc pour la vision par ordinateur, puis nous explorerons les modèles intrinsèquement explicables. Toutes ces méthodes s'appliquent spécifiquement à la classification d'images.

Il existe un débat sur la différence entre les termes "interprétable" et "explicable" [21, 97], et ces derniers sont souvent utilisés de façon interchangeable dans la littérature. Comme aucun consensus clair ne s'est encore dégagé sur leur utilisation, nous utiliserons nous aussi ces termes indifféremment dans ce document.

2.2 Notions préalables

Avant de s'intéresser aux méthodes d'explicabilité, il est nécessaire de présenter certaines définitions et notions liées aux réseaux de neurones. Celles-ci permettront de mieux comprendre les enjeux et les leviers utilisés pour expliquer le fonctionnement et les décisions d'un réseau de neurones. Nous verrons les éléments classiques qui composent un réseau de neurones mais aussi comment se déroule l'entraînement.

2.2.1 Architecture des réseaux de neurones

Neurone artificiel

La base d'un réseau de neurones est évidemment le neurone. Un neurone artificiel est modélisé par une fonction prenant une ou plusieurs entrées auxquelles sont attachées des poids. C'est la somme pondérée de ces entrées à laquelle est ajoutée

un biais qui constitue la sortie du neurone, ensuite passée dans ce que l'on appelle un fonction d'activation f, qui est une fonction non-linéaire. Mathématiquement, le comportement d'un neurone s'écrit :

$$y = f(\sum_{i=1}^{N}(w_i \cdot x_i) + b) \tag{2.1}$$

avec w_i le poids associé à l'entrée x_i du neurone et b le biais du neurone.

Réseau de neurones entièrement connecté

C'est ce type de neurone artificiel qui va être utilisé dès la fin des années 50 dans ce qui peut être considéré comme l'un des premiers réseaux de neurones : le Perceptron [98]. Le Perceptron est un classifieur linéaire binaire doté d'une seule couche de neurones. En combinant plusieurs de ces couches et en connectant la sortie de chaque neurone d'une couche à l'entrée de tous les neurones de la couche suivante on obtient un réseau entièrement connecté, aussi appelé FCN (pour Fully Connected Network) ou MLP (Multi-Layer Perceptron). La figure 2.1 reprend le fonctionnement d'un neurone artificiel et illustre l'architecture d'un MLP. Ce type de réseaux permet de résoudre des problèmes plus complexes qu'un simple Perceptron mais ils ne sont pas très adaptés aux tâches de vision par ordinateur. En effet, chaque pixel de l'image est une entrée du réseau et cela demande donc un nombre très élevé de neurones pour traiter des images avec une grande définition.

Réseau de neurones convolutif

Un autre type de réseau de neurones plus adapté à la vision par ordinateur est le réseau de neurones convolutif. Avec ce type de réseau, l'image en entrée n'est plus aplatie pour être traitée pixel par pixel mais on effectue des convolutions entre l'image et un noyau (aussi appelé filtre) plus petit dont les coefficients (que l'on désigne aussi comme des poids) sont appris. L'opération de convolution consiste à déplacer ce noyau sur toute l'image pour la filtrer et en obtenir une nouvelle en sortie. Mathématiquement, la valeur d'un pixel aux coordonnées m, n dans l'image de sortie est obtenue par :

$$w * x(m, n) = \sum_{i=-a}^{a}\sum_{j=-b}^{b} w(i, j)x(m - i, n - j) \tag{2.2}$$

avec w le noyau de convolution, $x(m, n)$ le pixel de l'image d'entrée aux coordonnées m, n, ainsi que $2a + 1$ et $2b + 1$ les dimensions du noyau de convolution.

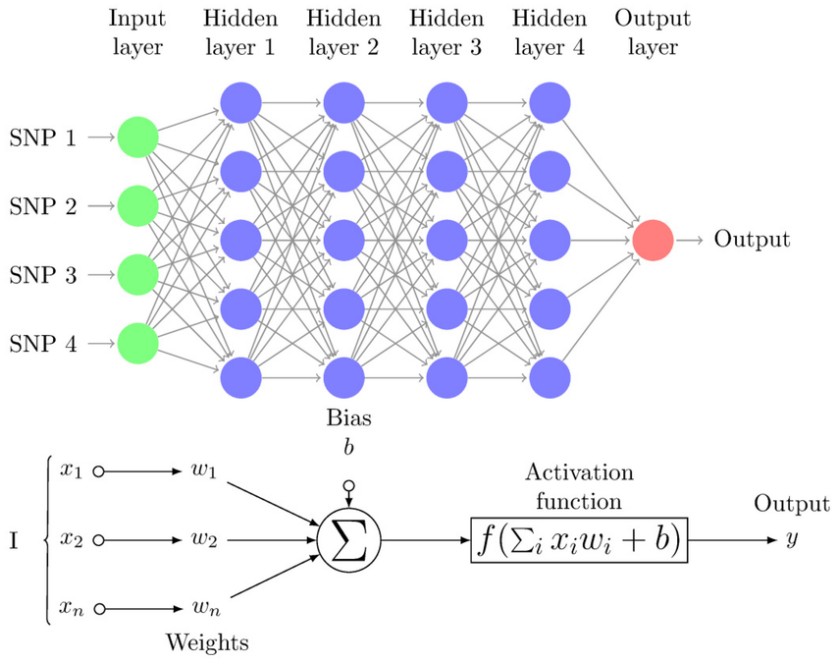


FIGURE 2.1 – Schéma de l'architecture d'un MLP à 4 couches cachées et fonctionnement d'un neurone artificiel. Figure reprise de [91]

Afin de mieux comprendre comment fonctionne la convolution entre une image et un noyau, la figure 2.2 illustre son principe. En appliquant le noyau au niveau du pixel rouge sur l'image d'entrée, on multiplie chaque pixel dans le carré rouge par le poids correspondant dans le noyau. La somme de ces valeurs est alors inscrite dans l'image de sortie. En déplaçant le noyau sur chaque pixel de l'image d'entrée on obtient une nouvelle image en sortie. Celle-ci est plus petite que l'image initiale car sur les bords, le noyau ne peut pas rentrer entièrement et on ne conserve que les pixels où celui-ci rentre entièrement. On peut toutefois résoudre ce problème de bord en considérant comme des 0 les pixels qui se trouveraient en dehors de l'image d'entrée, ou avec d'autres valeurs choisies judicieusement.

Les couches convolutives d'un réseau de neurones possèdent plusieurs noyaux, qui servent à produire autant d'images en sortie. Ces images de sortie sont appelées cartes d'activation ou carte d'attributs (en anglais feature map). En général, dans un réseau convolutif le nombre de ces cartes et de filtres augmente à mesure que le réseau devient plus profond. En échange de cette augmentation de dimensions informationnelles, on réduit souvent les dimensions spatiales pour condenser l'information et ne pas faire exploser la taille des données circulant dans le réseau.

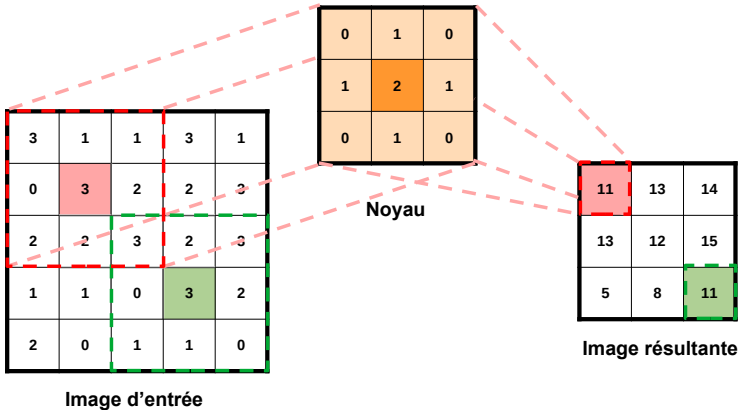


FIGURE 2.2 – Fonctionnement d'une convolution entre deux images (entrée et noyau). La valeur d'un pixel en sortie est égale à la somme des pixels de l'image d'entrée pondérée par les valeurs du noyau. Si l'on ne gère pas les effets de bord, en remplissant avec des zéros par exemple, l'image de sortie est plus petite que l'image d'entrée lorsque le noyau centré sur le pixel d'application ne rentre pas entièrement dans l'image.

Champ réceptif

Dans un réseau convolutif, à mesure que l'on empile les convolutions successives, l'une prenant en entrée la sortie de l'autre, la quantité d'information de l'espace pixel qui est utilisée augmente. Prenons l'exemple de la figure 2.3. Le pixel clair central dans la sortie de la couche 3 est obtenu en convoluant un noyau de taille 3×3 pixels avec la sortie de la couche 2. Et la sortie de la couche 2 est obtenue en convoluant un noyau de taille 3×3 pixels avec les données de la couche 1. Au final, tous les pixels des données de la couche 1 ont une influence sur le pixel central dans la couche 3. Ce carré de pixels qui contribuent à un pixel donné de la sortie finale est appelé champ réceptif.

Couche de pooling

Comme nous l'expliquions dans la partie sur les couches convolutives, on a tendance à augmenter le nombre de filtres et à réduire la taille des cartes d'activation plus le réseau devient profond. La réduction de la taille des cartes se fait typiquement par un type de couche que l'on appelle couche de pooling. On en distingue deux : le pooling moyen et le pooling maximal. Leur fonctionnement ressemble à celui d'une couche convolutive puisque ces couches possèdent des noyaux qui seront convolués à l'image d'entrée. Mais les poids de ceux-ci ne sont pas appris mais

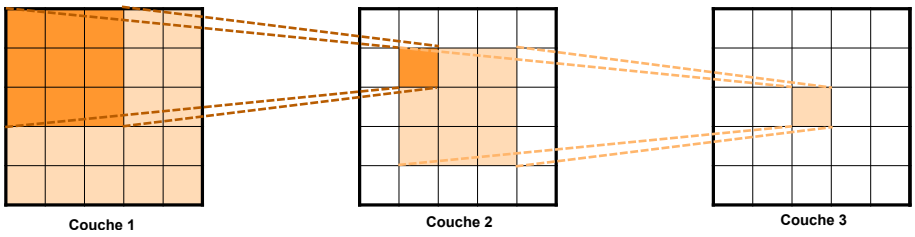


FIGURE 2.3 – Illustration du champ réceptif d'un réseau de neurones convolutif. Plus le réseau est profond plus celui-ci va avoir tendance à être grand.

sont fixes puisqu'ils servent simplement à calculer une moyenne ou un maximum sur les pixels de l'entrée. La réduction de la taille de la carte en sortie se fait via un mécanisme nommé convolution à pas. Au lieu de déplacer le noyau d'un seul pixel lorsque l'on parcourt l'image, on va effectuer des pas plus grands. En se déplaçant de deux pixels au lieu d'un par exemple, on va diviser par deux la taille de la carte d'activation. Parfois on effectue aussi ce que l'on appelle un pooling global. Il s'agit alors non plus de calculer une carte mais de réduire l'image d'entrée à une seule valeur, le maximum ou la moyenne selon le pooling choisi.

Espace latent

Une autre notion importante est celle d'espace latent. Chaque couche d'un réseau de neurones transforme les données qu'elle a en entrée et les projette dans un nouvel espace. Ce que l'on appelle espace latent est l'espace avant la couche de sortie. Cet espace latent est celui qui est optimisé par l'entraînement pour résoudre la tâche souhaitée. Pour un classifieur par exemple, l'espace latent va permettre de séparer au mieux les données avant la classification. Les informations importantes extraites des données d'entrée sont condensées dans l'espace latent, il est donc particulièrement intéressant à analyser pour essayer de comprendre le fonctionnement du réseau de neurones, et de nombreuses méthodes d'explicabilité évoquées dans ce chapitre se servent de cet espace.

2.2.2 Entraînement d'un réseau de neurones

Pour optimiser les poids des différentes couches d'un réseau de neurones on utilise pendant l'entraînement ce que l'on appelle une fonction de perte ou de coût. Cette fonction est une sorte d'objectif qui dépend de la sortie du réseau, et que l'entraînement aura pour but de minimiser pour un ensemble de données d'entrée en fonction des paramètres du réseau, qui seront mis à jour régulièrement.

$$\tilde{\theta} = \text{argmin } L(\theta) \tag{2.3}$$

avec $\tilde{\theta}$ les paramètres mis à jour du réseau, θ les paramètres du réseau et L la fonction de perte correspondant à l'ensemble des données d'entrée.

Pour mettre à jour les paramètres, une des méthodes d'optimisation les plus courantes est la descente de gradient stochastique (en anglais Stochastic Gradient Descent ou SGD). Comme son nom l'indique, cette méthode implique de calculer le gradient de la fonction de perte par rapport aux paramètres du réseau, puis on fait un "pas" dans la direction du gradient et on actualise les paramètres. On répète cette opération jusqu'à convergence ou jusqu'à ce qu'une condition d'arrêt spécifique soit atteinte. Pour une couche donnée i, on peut écrire la mise à jour des paramètres :

$$\tilde{\theta}_i = \theta_i - \eta \, \frac{\partial L}{\partial \theta_i} \tag{2.4}$$

où η est le taux d'apprentissage, la taille du pas dans la mise à jour des paramètres. Évaluer la fonction de perte sur l'ensemble des échantillons d'apprentissage peut se révéler très coûteux. On procède donc itérativement en tirant aléatoirement un sous-ensemble de données d'apprentissage appelé batch. Le calcul du gradient se fait grâce à la rétro-propagation, une application de la règle de dérivation en chaîne aux réseaux de neurones. On décompose le gradient à travers le réseau en une suite de gradients locaux liés aux couches successives. Il est donc important que la fonction de perte ainsi que toutes les opérations calculées dans le réseau soient dérivables afin de pouvoir calculer le gradient. Ce sont souvent les mêmes fonctions de perte qui sont employées pour résoudre certains problèmes. Par exemple pour un problème de classification, on utilise souvent l'entropie croisée. On peut aussi ajouter des termes supplémentaires pour régulariser l'objectif, du moment que ceux-ci sont dérivables.

En résumé, l'entraînement d'un réseau de neurones se déroule comme suit :

- Un ensemble d'images est donné en entrée du réseau et on effectue toutes les opérations pour obtenir les sorties associées.
- On calcule la fonction de perte avec la sortie du réseau.
- On calcule le gradient de la perte.
- On rétro-propage le gradient en partant de la dernière couche vers la première pour mettre à jour les paramètres du réseau.

Lorsque que l'on a considéré toutes les images de la base à l'aide des différents batchs, on a atteint la fin de ce que l'on appelle une époque (ou epoch en anglais).

L'entraînement d'un réseau de neurones prend en général plusieurs époques et on repasse donc les images plusieurs fois en entrée.

2.3 Explications post-hoc

En matière d'explicabilité, les méthodes post-hoc sont très courantes. Elles ont souvent l'avantage de pouvoir s'appliquer à de nombreux modèles déjà entraînés afin de rendre plus claires leurs décisions, ou de pouvoir facilement être adaptées à différentes architectures sans avoir besoin de les modifier.

2.3.1 Cartes d'attribution

Les cartes d'attribution, aussi appelées cartes de salience, sont des images qui visent à expliquer pour un échantillon d'entrée donné (une image) quelles régions ont participé le plus à la décision finale du modèle. Elles fournissent souvent une visualisation sous la forme d'une carte de chaleur superposée à l'image d'entrée.

Une des premières tentatives de produire des cartes d'attribution pour comprendre le fonctionnement d'un CNN est l'utilisation d'un second réseau dit "déconvolutif" [125, 126]. L'idée est de construire un second réseau de neurones, non pas pour extraire les caractéristiques visuelles à partir des pixels de l'image, mais pour reconstruire les pixels d'une image à partir des caractéristiques qui sont activées dans le réseau. Cela permet de reconstruire pour chaque neurone dans le CNN de base une image qui représente la caractéristique visuelle qui l'active. Bien qu'encombrante, cette solution a permis de mieux comprendre ce qu'apprenaient les CNN et notamment, Zeiler et al. ont pu montrer que plus on s'enfonce profondément dans les couches d'un réseau, plus les caractéristiques apprises par les neurones sont des concepts "haut-niveau", comme une roue, un œil ou une patte [125]. A l'inverse, les premières couches du réseau apprennent des concepts plus simples et plus géométriques : des rayures, des courbes ou des quadrillages (Figure 2.4).

Une autre idée qui germa rapidement fut celle de renverser le problème d'optimisation lié au réseau. Au lieu de réduire la perte pour la classification, on cherche à maximiser l'activation des neurones en utilisant les pixels de l'image d'entrée comme paramètres. En répétant cette opération pour chacune des classes des données, on peut visualiser les caractéristiques visuelles qui activent le plus le réseau vers chacune des classes. Erhan et al. [36] avaient introduit cette idée pour les réseaux entièrement connectés puis Simonyan et al. [107] l'étendirent aux réseaux convolutifs. Si cette approche permet de comprendre sur quelles caractéristiques le réseau se base pour classifier, les images générées sont très abstraites et difficilement lisibles (Figure 2.5).

FIGURE 2.4 – Visualisation des caractéristiques apprises par un modèle entraîné sur ImageNet par la méthode des réseaux déconvolutifs. [125]

FIGURE 2.5 – Exemples d'images qui activent le plus les classes "oie", "autruche" et "limousine" d'un CNN entraîné sur ILSVRC-2013 (ImageNet) [107]

Dans le même papier [107], Simonyan et al. introduisent également la notion de salience de classe pour une image spécifique. Le principe est assez simple et part de l'hypothèse que l'amplitude de la dérivée du score de classification pour une classe, par rapport à l'image d'entrée, indique les pixels de l'image qui ont le moins besoin de changer pour changer le plus le score de la classe. Cette dérivée peut facilement être calculée via la rétro-propagation.

La carte d'activation de classe (en anglais Class Activation Map, ou CAM) [131] est l'une des approches les plus connues et son principe ressemble à celui de la carte de salience de classe. Les CAM sont produites pour une image d'entrée donnée et une classe donnée et permettent de voir quels pixels de l'image produisent les activations les plus fortes des neurones du réseau. Zhou et al. partent d'une classe donnée et regardent les poids des neurones de la couche linéaire de classification vers cette classe. Dans leur architecture comme dans la plupart des architectures convolutives pour la classification d'images, l'entrée de cette dernière couche linéaire est simplement la valeur moyenne des pixels pour chaque carte d'activation en sortie des couches convolutives. On a donc un neurone par canal dans la dernière couche convolutive. Le poids associé à chaque neurone correspond à l'importance de la caractéristique visuelle encodée par le canal convolutif. En multipliant par le poids appris dans la couche linéaire le canal correspondant de la carte d'activation, puis en faisant la somme de tous les canaux, on obtient une carte qui montre l'importance des régions de l'image vis-à-vis de la classe sélectionnée. Il suffit de suréchantillonner cette carte à la même taille que l'image d'entrée pour visualiser ces régions. La figure 2.6 illustre ce fonctionnement. Cette technique ne peut être réutilisée qu'avec des architectures CNN similaires.

Une autre variante de cette approche également très utilisée est la méthode Grad-

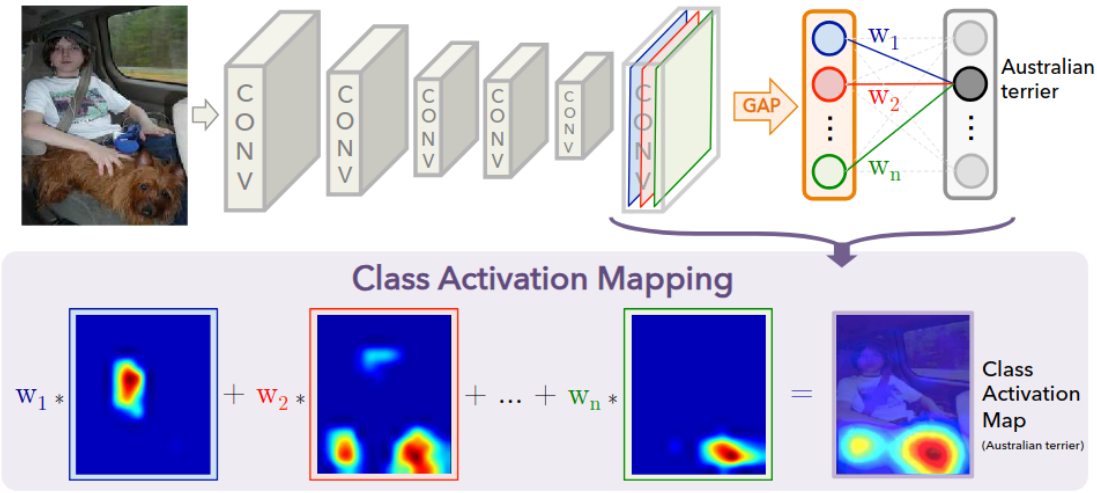


FIGURE 2.6 – Schéma de l'architecture utilisée par Zhou et al. [131] pour obtenir leur CAM. Il s'agit d'une architecture d'encodeur CNN classique, se terminant par une opération de pooling moyen global (GAP) avant la couche linéaire de classification.

CAM [105]. Celle-ci étend le principe de CAM a d'autres architectures CNN, intégrant des couches linéaires ou avec des entrées multi-modales. Selvaraju et al. généralisent l'approche développée dans CAM en utilisant la rétropropagation du gradient à partir du score de sortie associé à une classe pour déterminer des poids qui multiplieront les cartes d'activation en sortie de l'encodeur CNN. Ces poids ont un sens similaire à ceux calculés pour CAM, mais la manière dont ils sont calculés permet d'utiliser Grad-CAM avec n'importe quelle architecture à base de CNN.

L'approche LRP [9], Layer-wise Relevance Propagation, pousse l'idée de rétro-projeter les activations encore plus loin. Avec Grad-CAM, on ne l'utilise que pour retourner à la sortie des couches convolutives, puis on suréchantillonne la carte d'activation obtenue pour retrouver la même taille que l'image d'entrée et pouvoir superposer les deux. Dans LRP, on pousse la rétropropagation jusqu'au niveau des images d'entrée, ce qui donne une carte très détaillée avec une valeur de pertinence directement calculée pour chacun des pixels et non plus interpolée, un peu à la manière de [107]. Mais l'un des inconvénients de LRP est que la règle utilisée pour rétro-projeter les activations peut donner des résultats instables numériquement. Les auteurs ont donc introduit deux nouvelles manières de calculer la pertinence [82], mais ces deux nouvelles formules font chacune appel à un hyper-paramètre dont il faut déterminer empiriquement une valeur satisfaisante pour obtenir une carte d'attribution finale peu bruitée.

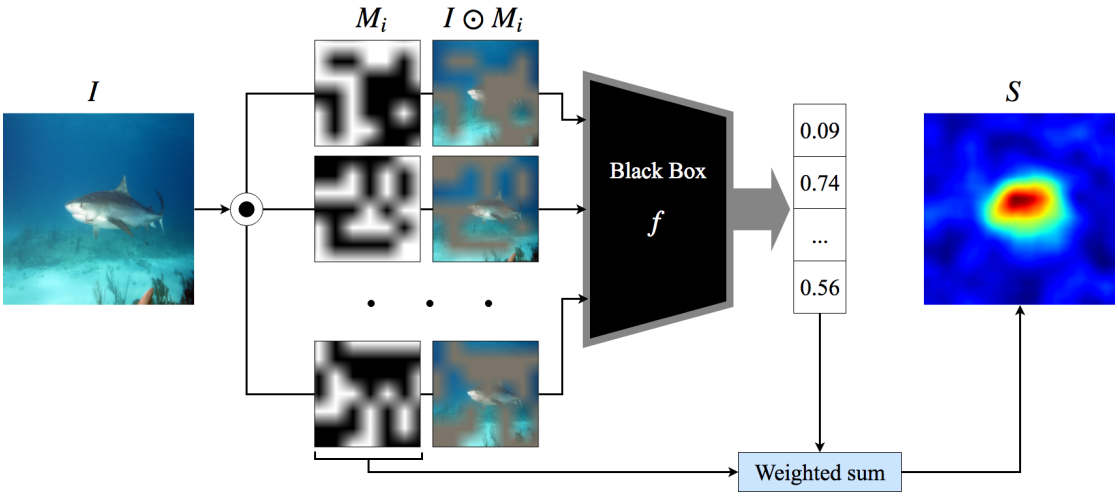


FIGURE 2.7 – Schéma du fonctionnement de RISE [92]. On génère une série de masques aléatoires que l'on applique à l'entrée pour connaître l'impact des pixels sur le score de classification.

Parmi les méthodes qui génèrent des cartes d'attributions, on peut aussi noter les méthodes par perturbation telles que RISE [92] et D-RISE [93], sa variante pour les détecteurs d'objets. Pour une image donnée en entrée, on génère un certain nombre de masques aléatoires que l'on applique à l'image d'entrée. On récupère les scores de classe obtenus pour chaque version masquée de l'image et ceux-ci nous donnent une idée de l'importance des pixels qui ont été masqués pour la classification. On peut alors reconstruire une carte de salience en calculant la somme pondérée par les scores de tous les masques (Figure 2.7). Étant donné que l'on effectue seulement des passes en avant dans un réseau pré-entraîné, le coût en calcul de cette méthode est relativement restreint par rapport aux autres, et elle fonctionne pour tout type de réseau de neurones.

Une autre méthode par perturbation populaire est LIME [96]. LIME se base sur un découpage de l'image en super-pixels, c'est-à-dire des patchs contigus de pixels similaires. Un modèle linéaire est ensuite entraîné avec des versions perturbées de l'image de départ où certains super-pixels sont absents, pour essayer de trouver l'importance de chacun de ces super-pixels pour la classification.

Enfin, une autre méthode d'explication post-hoc couramment utilisée sont les valeurs SHAP (SHapley Additive exPlanation) [74]. C'est une méthode qui puise son inspiration dans la théorie des jeux et la manière de mesurer les effets de la collaboration entre les différents agents. Pour la vision par ordinateur et la classification en particulier, les caractéristiques visuelles remplacent les agents et on mesure la contribution de chacune des caractéristiques vers la prédiction donnée

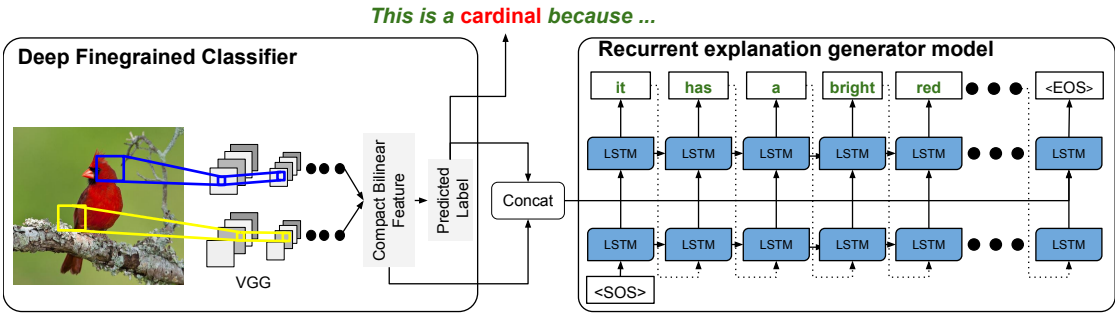


FIGURE 2.8 – L'architecture utilisée dans [48] pour générer des explications visuelles à une classification. Des réseaux récurrents LSTM sont entraînés pour produire des descriptions des images d'entrée qui soient discriminantes en exploitant la classe prédite par le CNN.

par le modèle pour une entrée donnée. On part de la prédiction virtuelle dans le cas où on n'aurait connaissance d'aucune des caractéristiques puis on ajoute une par une les caractéristiques pour connaître leur influence (positive ou négative). Comme l'ordre dans lequel on étudie les caractéristiques a une influence dans le résultat final lorsqu'on étudie un modèle linéaire (comme un réseau de neurones), les valeurs SHAP finales sont obtenues en calculant une moyenne sur tous les ordres d'étude possibles.

2.3.2 Descriptions textuelles et VLMs

Si les cartes d'attribution visent à produire une explication visuelle de la classification, il existe d'autre méthodes qui visent à produire une explication en langage naturel. De nombreux réseaux de neurones ont été entraînés pour donner une description textuelle du contenu d'une image [29, 55, 77, 124]. A partir de là est née l'idée de combiner description et classification afin de mieux comprendre ce que le réseau voyait dans l'image pour la classifier [48]. La figure 2.8 montre l'architecture utilisée par Hendricks et al [48]. Ils emploient d'abord un CNN spécialisé dans la classification d'images, ici un réseau type VGG [108], puis ils transmettent les caractéristiques visuelles apprises par ce dernier à deux réseaux récurrents LSTM [50] entraînés à décrire le contenu d'une image. Mais en plus des caractéristiques visuelles, ces deux LSTM reçoivent aussi la classe prédite par le CNN. Cela leur sert notamment pendant leur entraînement, où ils sont poussés à générer des descriptions qui soient discriminantes par rapport à la tâche de classification.

Les récents progrès en traitement du langage naturel grâce aux architectures transformeurs [116] ont donné lieu à l'avènement des LLM (Large Language Model). Similairement, les architectures transformeurs visuels (ViT) [35] ont donné nais-

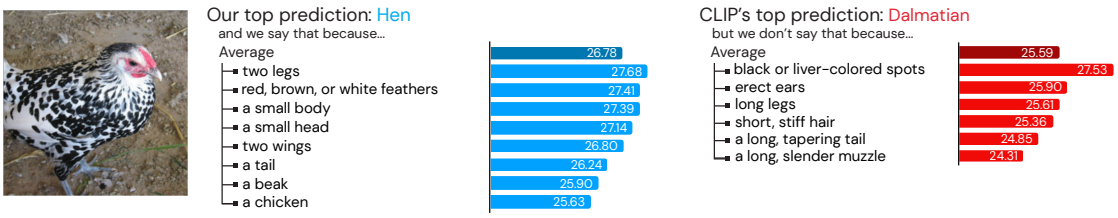


FIGURE 2.9 – Exemple de décision de classification par le modèle de [79] (gauche) et de la justification de la décision par rapport à un VLM qui n'utilise pas de descripteurs, ici CLIP (droite) avec la similarité entre l'image et les descripteurs liés à chaque classe renseignée dans les barres.

sance aux modèles VLM (Vision-Language Models) partageant des espaces de représentation joints pour les images et le texte, comme CLIP [94], VisualBERT [68] ou ALIGN [53]. La capacité de ces derniers à manier à la fois le texte et les images a permis un net progrès dans les tâches de légendage d'image ou les questions-réponses visuelles (VQA) [7].

Certains travaux ont mis à profit ces capacités de description des images pour obtenir des explications des décisions de classification, comme [79]. Leur idée est d'utiliser les grands modèles de langage pour obtenir des descripteurs textuels de l'apparence visuelle de chacune des classes du problème de classification. Ces descripteurs ayant eux-mêmes une position dans l'espace latent joint entre images et texte, ils peuvent servir d'auxiliaires à la classification. De plus, grâce à la mesure de similarité entre l'image et chacun des descripteurs, on peut connaître quels éléments ont fait pencher la balance en faveur d'une classe plutôt qu'une autre. Ce fonctionnement est illustré dans la figure 2.9, où le modèle classifie une image comme étant une poule là où CLIP classe l'image comme étant un dalmatien.

Si les performances des VLMs sont souvent très bonnes, leur utilisation est limitée par plusieurs facteurs. Leurs performances dans une tâche donnée dépendent souvent des données utilisées pour leur entraînement[130]. La plupart des modèles disponibles sont très généralistes et leur application dans des domaines spécifiques nécessite un fine-tuning souvent supervisé qui peut être coûteux, ou bien des données annotées supplémentaires qui ne sont pas toujours disponibles, particulièrement dans le domaine médical. Des tentatives ont été réalisées pour entraîner des VLMs spécialisés dans l'imagerie biomédicale, comme [16], mais ce modèle a été entraîné majoritairement sur de l'imagerie à rayons X et la question de la généralisation à d'autre modalités d'imagerie se pose.

Mais il existe un autre problème, partagé par les méthodes basées sur les VLMs comme par les autres méthodes d'explicabilité post-hoc. Rudin [99] explique que

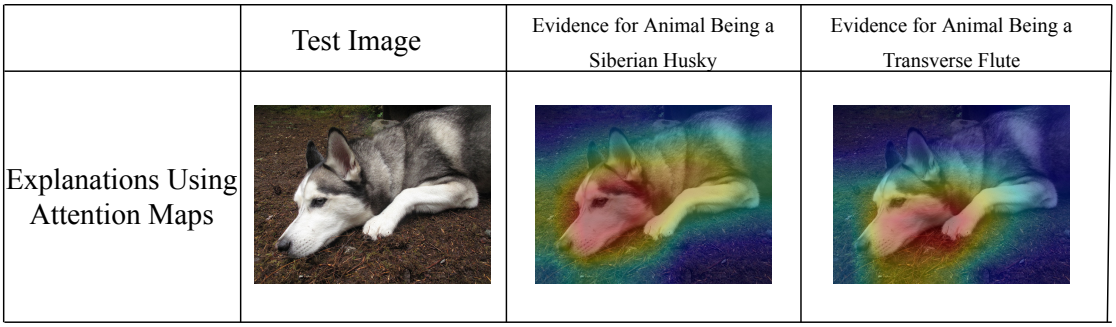


FIGURE 2.10 – Les cartes de salience obtenues sur une même image pour deux classes très différentes peuvent être très proches. Dans ce cas l'explication qu'elles apportent est très limitée et se cantonne à la zone de l'image qui active le plus les neurones et pas la manière dont elle les active. Figure issue de [99]

tenter d'expliquer des modèles "boîte noire" est à même de perpétuer certains problèmes, à savoir que les explications peuvent ne pas être fidèles à ce que le modèle calcule réellement, ou ne pas être assez détaillées pour refléter véritablement le fonctionnement interne du modèle. Par exemple les cartes d'attribution ne montrent que les éléments sur lesquels le modèle s'est basé, pas la manière dont il a utilisé l'information. Ce problème est illustré dans la figure 2.10, où des zones similaires de l'image sont mises en lumière par les cartes de salience pour deux classes très différentes. Pour pallier ces défauts, Rudin préconise de créer des modèles interprétables par conception.

2.4 Modèles profonds intrinsèquement interprétables

Les méthodes pour entraîner des modèles profonds intrinsèquement interprétables sont moins nombreuses à cause des contraintes qu'elles imposent. Il est souvent nécessaire de construire de nouvelles architectures pour répondre au besoin d'explicabilité au lieu de reprendre des architectures de l'état de l'art.

Certains travaux ont essayé de contourner ces contraintes et d'adapter le fonctionnement des CNN pour les rendre interprétables. Parmi ces travaux, on peut citer [127]. Dans cet article, Zhang et al. proposent une nouvelle fonction de perte pour l'entraînement des CNNs qui serait appliquée à chacun des filtres des couches convolutives. L'idée est de pousser chacun des filtres à encoder une partie d'objet spécifique, appartenant à une seule classe ; et de ne s'activer que pour une seule partie de l'objet. En contraignant les filtres de cette manière, l'idée est d'obtenir dans les filtres des caractéristiques visuelles très précises, quitte à ce que ceux-ci ne s'activent que très rarement, afin que ces caractéristiques soient vraiment

spécifiques à un type d'objet.

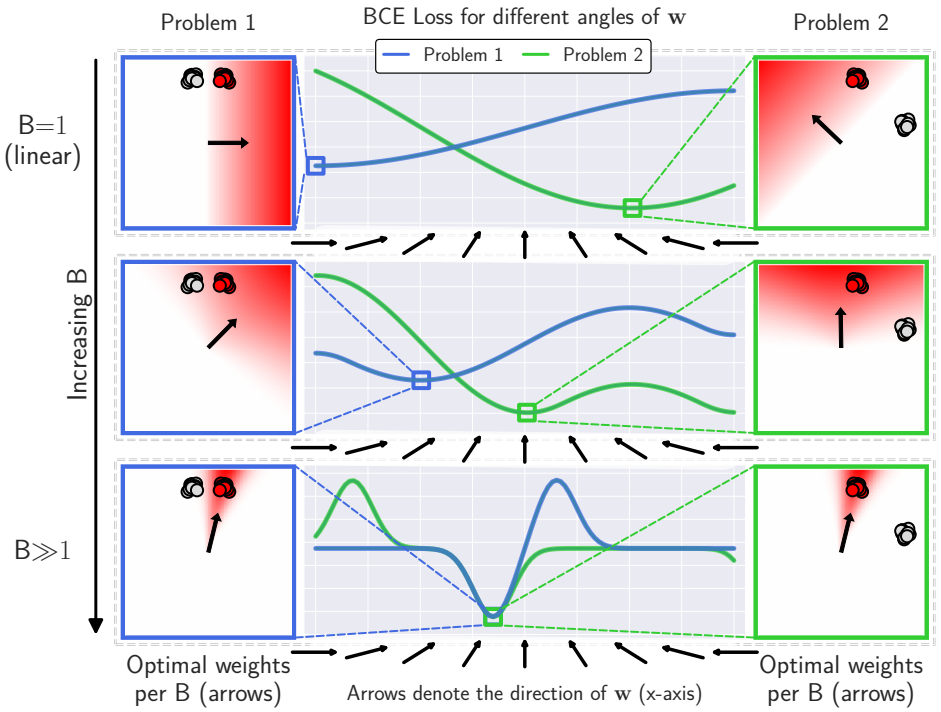


FIGURE 2.11 – Effet du paramètre B de la méthode B-cos [17] sur des problèmes simples de classification. Au centre, la perte d'entropie binaire croisée en fonction de l'angle du vecteur poids du neurone de classification. Les colonnes latérales sont des visualisations du problème de classification et des poids optimaux (flèches) en fonction de B. Bien que les points soient placés au même endroit, la direction optimale du poids varie fortement en fonction de B.

Dans la même veine d'adapter les CNNs traditionnels, Böhle et al. [17] ont proposé une solution simple pour transformer des réseaux de neurones classiques en une version explicable. Leur méthode s'applique aux réseaux denses comme aux réseaux convolutifs. Tout repose sur les transformations linéaires calculées dans les réseaux. Si dans la section 2.2.1 nous écrivions un produit simple entre un seul poids et une seule entrée pour un neurone, dans le cas plus courant où ceux-ci sont multiples, pour obtenir une seule valeur en sortie du neurone, on calcule un produit scalaire entre le vecteur des poids transposé et le vecteur des entrées. Böhle et al. proposent donc une nouvelle transformation baptisée B-cos, qui modifie légèrement cette transformation linéaire basique. Tout d'abord ils ramènent la norme du poids du neurone à 1 pour limiter la valeur d'activation en sortie, puis ils appliquent un paramètre B comme exposant pour la valeur du cosinus. Cet exposant joue un rôle clé car il va accentuer l'effet du cosinus, c'est-à-dire qu'en

dehors d'un angle restreint entre l'entrée du neurone et le poids, les activations produites seront très faibles sinon nulles, comme l'illustre la figure 2.11. Cela va permettre d'aligner précisément les poids avec les entrées et donc d'obtenir en fin d'entraînement des caractéristiques apprises qui sont pertinentes pour résoudre le problème d'optimisation.

En opposition aux méthodes post-hoc, Alvarez-Melis et Jaakkola [4] définissent les propriétés qui font un réseau de neurones auto-explicable (SENN : Self-Explainable Neural Network), ainsi qu'une stratégie générale d'entraînement pour de tels modèles pour une tâche de classification. L'idée est d'apprendre des concepts haut niveau qui serviront de base pour interpréter la décision, plutôt que de baser l'interprétation sur les pixels d'entrée de l'image comme avec la plupart des cartes d'attribution. La figure 2.12 montre l'architecture proposée, composée de trois parties. Le réseau extrait les concepts des images et donne un score de pertinence à ceux-ci. La combinaison de ces deux informations permet alors de classifier une image et l'explication est assez intuitive, dès lors que les concepts sont visualisés et que l'on connaît la pertinence qui leur est associée.

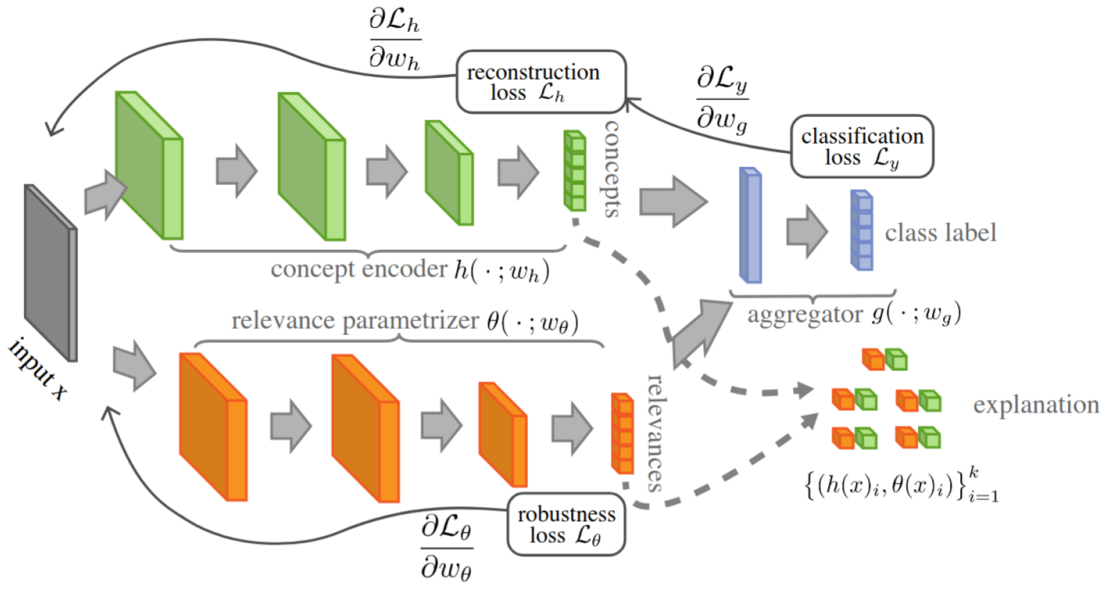


FIGURE 2.12 – Architecture d'un réseau de neurones auto-explicable [4]. Le réseau est composé de trois parties : un encodeur qui extrait des concepts interprétables des images d'entrée, un paramétreur qui apprend à générer des scores de pertinence, et une fonction d'agrégation qui combine concepts et pertinence afin de classifier.

Koh et al. proposent une idée similaire avec les Concept Bottleneck Models [62]

(Modèles à goulot d'étranglement de concepts), où un encodeur apprend à prédire la présence d'un ensemble de concepts haut-niveau prédéfinis et annotés dans les images d'entraînement. On peut ensuite apprendre une tâche de classification en se basant uniquement sur les concepts prédits pour une image. L'analyse de l'influence des concepts dans la prédiction finale permet de comprendre lesquels sont importants pour la tâche.

D'autres se sont inspirés de méthodes qui existaient avant l'arrivée de l'apprentissage profond, et notamment de la méthode "sac de mots visuels" [66, 121]. Avec cette méthode on réduit une image comme une somme de contributions de composantes visuelles ou de descripteurs locaux de l'image, définis dans un dictionnaire ou vocabulaire visuel, sans tenir compte de leur position dans l'image. Cette somme constitue donc un "sac" dans lesquels les composantes sont désordonnées, et sert d'entrée à un classifieur linéaire. Celui-ci assignera un poids à chacune des composantes pour classifier et il suffit donc de regarder quelles composantes portent les plus grands poids pour connaître les zones de l'image qui ont le plus impacté la classification. Plusieurs travaux ont donc adapté cette approche à une utilisation avec des réseaux de neurones, soit en utilisant les caractéristiques extraites par un encodeur [89, 88], soit en contrôlant la taille du champ réceptif de l'encodeur pour que celui-ci travaille sur des patches de l'image, à la manière d'une approche par descripteur locaux [19]. Pour notre étude, ce type d'approche est particulièrement intéressant car le dictionnaire visuel est constitué de manière non-supervisée. Les composantes sont déterminées par des algorithmes de clustering ou par d'autres objectifs d'optimisation qui visent avant tout à représenter toute la diversité des structures visuelles qui existent afin d'obtenir une représentation sous forme de sacs de mots visuels qui soit fidèle au contenu initial de l'image.

2.4.1 Approches par prototypes

Parmi les approches intrinsèquement interprétables, les approches par prototypes se rapprochent de celles utilisant des sacs de mots visuels, les prototypes remplaçant les éléments du dictionnaire visuel. Les prototypes constituent un ensemble réduit d'échantillons des données permettant de décrire simplement et efficacement l'ensemble du jeu de données. L'utilisation des prototypes se faisait initialement pour des tâches de clustering de données [71, 65], avec un fonctionnement similaire à un algorithme comme les k-moyennes [73], où au lieu d'utiliser les centroïdes pour déterminer des clusters, on utiliserait un sous-ensemble bien choisi d'éléments du jeu de données.

De même qu'il est possible d'entraîner un classifieur à partir des données clusterisées avec les k-moyennes, Tibshirani et al. ont proposé d'utiliser les prototypes

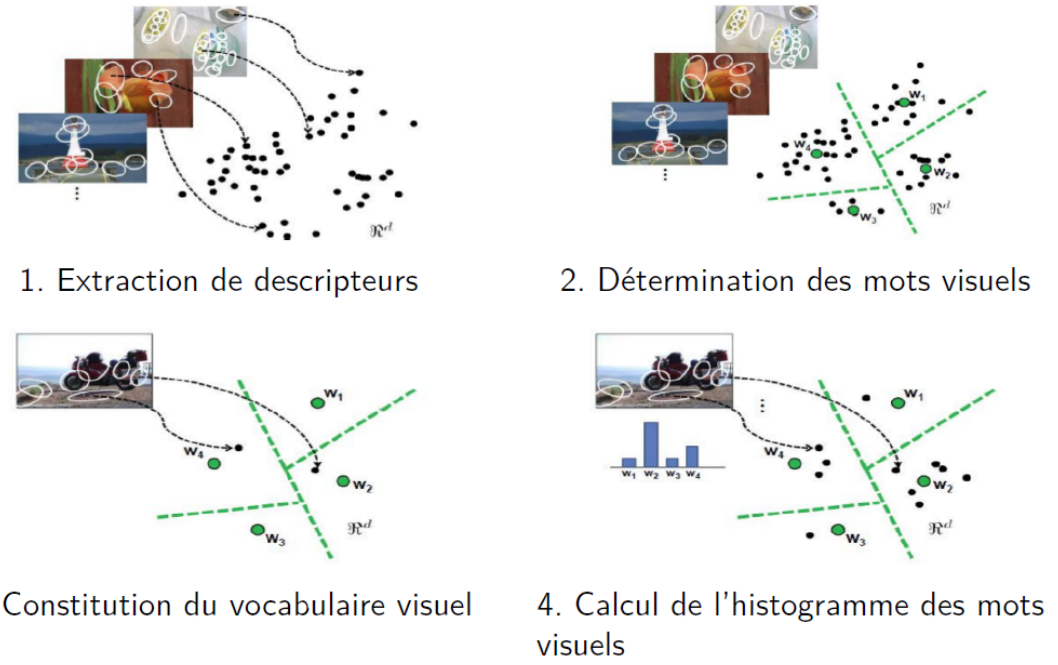


FIGURE 2.13 – Fonctionnement de la méthode des sacs de mots visuels. L'image est représentée par un histogramme de mots visuels piochés dans un dictionnaire, correspondants à tous les éléments visuels qui existent au sein de la base de données d'images.

servant à faire un clustering de l'espace pour obtenir une classification interprétable [11]. Les réseaux de neurones couplés à des prototypes furent utilisé pour des tâches comme le clustering [39], la quantification vectorielle (VQ) ou le few-shot learning (apprentissage avec peu de données, typiquement quelques exemples seulement) [109] et aussi bien sûr pour la classification interprétable.

De façon intéressante, Alvarez-Melis et Jaakkola [4] proposaient dans leur définition des SENN que les concepts appris pour la classification puissent être représentés sous forme de prototypes. Leur approche peut être vue comme une généralisation des approches par prototypes à une classe plus large de réseaux de neurones.

Parmi les travaux définissant des approches basées sur les prototypes, on peut citer [69] qui introduisirent l'idée d'avoir un réseau de neurones dont la classification se fait sur un modèle de raisonnement à partir de cas, c'est-à-dire en se basant sur des exemples, des cas déjà connus. Les prototypes servent ici d'exemples à partir desquels raisonner, en mesurant une distance ou une similarité dans l'espace latent. Si une image d'entrée est proche d'un des prototypes alors le modèle la

classifiera dans la même classe que le prototype. Ce mode de fonctionnement fut repris par les mêmes auteurs dans un second papier intitulé "This looks like that" (ceci ressemble à cela) [27], un titre qui reflète bien ce fonctionnement se basant sur des prototypes servant d'exemples. La figure 2.14 montre le fonctionnement de cette approche où plusieurs zones de l'image sont mises en avant et comparées à des parties similaires d'images servant de prototypes. Cette publication ouvrira la voie à de nombreux modèles s'en inspirant et essayant de pallier ses limitations. On peut séparer en deux catégories les modèles utilisant des prototypes : ceux dont les prototypes sont à la taille d'une image d'entraînement (image-prototypes), et ceux dont les prototypes sont des patches d'images (patch-prototypes). Le modèle de Li et al. appartient à la première catégorie, tandis que ProtoPNet appartient à la seconde catégorie. En général les approches prototypiques qui utilisent des architectures auto-encodeur ou auto-encodeur variationnel [40, 61] entrent dans la catégorie image-prototypes, ces architectures reconstruisant des images de même taille que celles en entrée. Leur avantage est que la visualisation des prototypes est assez naturelle puisque n'importe quel point de l'espace latent peut être reconstruit grâce au décodeur.

Les approches patch-prototypes explorent souvent différentes manières de visualiser les prototypes qui ont été appris par le modèle. ProtoPNet retourne la zone d'une image d'entrée qui donne la plus forte activation d'un prototype grâce à un sur-échantillonnage de la carte d'activation. Carmichael et al. [26] proposent une autre solution à la visualisation en utilisant le champ réceptif, ce qui permet de s'assurer que les pixels qui donnent la visualisation ont bien servi à produire l'activation. Toujours dans l'idée d'améliorer la visualisation des prototypes, des méthodes d'explication post-hoc ont été adaptées, comme le LRP [82] qui devient le PRP (pour Prototype Relevance Propagation) [41]. Pour mieux comprendre ce qu'un prototype représente, Ma et al. [75] proposent de les apprendre non plus seulement comme des points de l'espace latent, mais comme des boules, avec un rayon le plus compact possible. Au moment de la visualisation, tous les patches tombant à l'intérieur de la boule forment la visualisation du prototype. Avec plus d'exemples à regarder, le concept décrit par le prototype devient alors plus facile à interpréter. Récemment, les prototypes ont aussi été mis à profit pour tenter de rendre plus interprétable les décisions des transformeurs visuels [112].

Il existe encore d'autres propositions d'approches par prototypes, jouant aussi sur la manière d'apprendre les prototypes. Dans le chapitre 4, nous verrons plus en détail le fonctionnement de ProtoPNet, et nous reviendrons sur certains de ses dérivés. Dans le chapitre 5, nous nous pencherons sur des approches image-prototypes, particulièrement ProtoVAE [40] et les méthodes s'en inspirant pour combler ses manques.

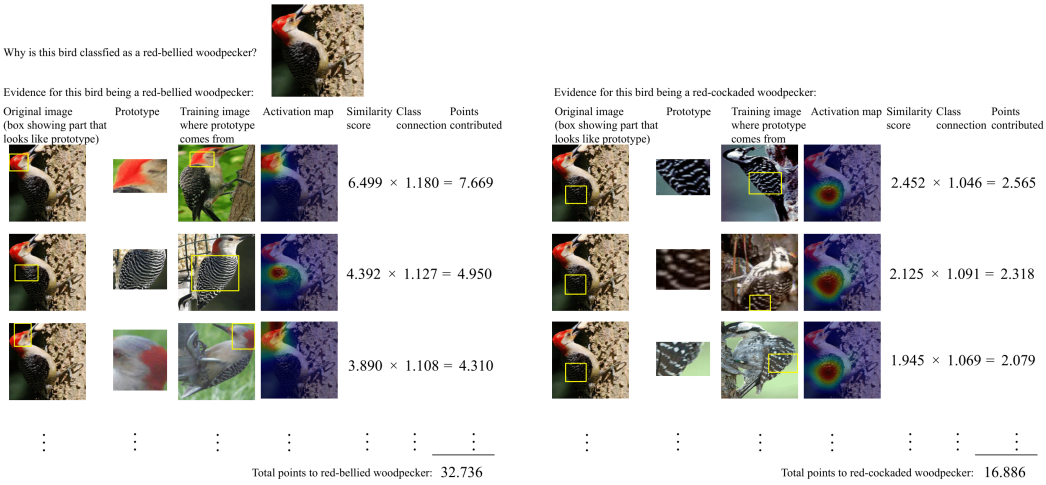


FIGURE 2.14 – Explication du raisonnement de ProtoPNet [27] pour une image donnée

2.5 Applications dans un contexte biologique

Les réseaux de neurones ont rapidement été employés pour l'analyse d'images médicales, leurs capacités en classification ou en segmentation étant très intéressantes pour analyser les volumes de données toujours plus grands générés par le développement de l'imagerie numérique. Très tôt dans ces applications, le manque d'interprétabilité fut cité comme un frein à la généralisation de l'utilisation l'apprentissage profond [76].

Les enquêtes plus récentes sur l'explicabilité [84, 114] montrent que de nombreuses utilisations de l'apprentissage machine et de l'apprentissage profond dans un contexte biomédical sont centrées sur la détection de biomarqueurs par la classification d'images. En association avec des méthodes CAM et Grad-CAM, la classification binaire entre des images témoins "normales" et des images issues de patients atteints d'une pathologie permet d'identifier sur les images quelles zones sont porteuses d'une information quant à la pathologie. Parmi ces études, on peut citer [95] qui emploie ce principe pour identifier différentes pathologies à partir de radiographies du thorax. Un peu plus récemment, Panwar et al. [87] ont utilisé une approche similaire pour détecter les cas de Covid-19 sur le même type d'images.

Si beaucoup de travaux utilisent CAM et Grad-CAM, on sait que ces cartes peuvent parfois être imprécises dans la localisation d'élément. Zhang et al. [128] utilisent une approche basée sur un GAN et un classifieur CNN, illustrée dans la figure 2.15. L'encodeur-décodeur sert de partie générative au GAN, et est entraîné à enlever les biomarqueurs sur les images représentant des anomalies. Le discriminateur est

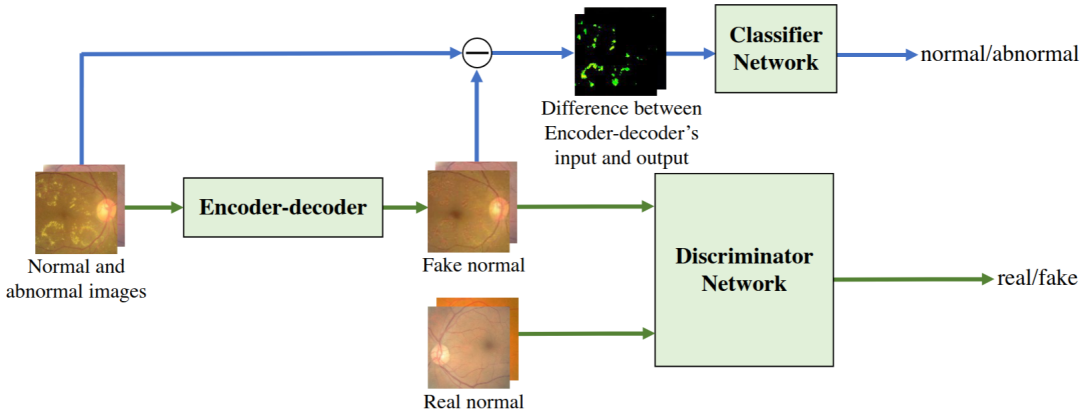


FIGURE 2.15 – Architecture proposée par [128].

lui entraîné à distinguer ces images modifiées d'images réellement normales. Pour renforcer les capacités du générateur, le classifieur est entraîné à déterminer si une image présentait initialement des anomalies en se basant sur la différence entre l'image en entrée du générateur et sa modification en sortie. Zhang et al. comparent la localisation des biomarqueurs qu'ils obtiennent par cette méthode et ce qu'ils obtiennent avec CAM et Grad-CAM, et constatent que la localisation est plus précise et plus en accord avec les annotations des experts avec leur méthode.

Concernant les modèles intrinsèquement interprétables, ProtoPNet, qui est la source d'inspiration principale pour nos travaux, a été utilisé à plusieurs reprises dans des applications biomédicales. On peut citer Mohammadjafari et al. [81] qui ont appliqué ProtoPNet à des scans IRM cérébraux pour détecter la maladie d'Alzheimer et essayer de trouver des patterns prototypiques de la maladie. Carloni et al. [25] ont quant à eux appliqué ProtoPNet sur des images de mammographies pour détecter des cancers. Enfin on peut citer Kim et al. [56] qui ont proposé XProtoNet pour les radiographies thoraciques, un modèle dérivé de ProtoPNet pour mieux localiser et contrôler la taille et la forme des prototypes appris. La figure 2.16 montre bien la différence entre les prototypes obtenus avec ProtoPNet et ceux de XProtoNet et comment ceux-ci apportent une analyse plus fine.

Au-delà de la simple performance en classification, la qualité des explications est importante. Certaines méthodes peuvent produire des explications qui ne sont pas simples à comprendre pour qui n'est pas du domaine. Dans une étude, Evans et al. [37] comparent plusieurs méthodes d'explicabilité appliquées à de l'immunohistologie et font passer un questionnaire en ligne adressé principalement à des pathologistes et des chercheurs en pathologie. Il ressort du questionnaire que les approches par prototypes sont les plus intuitives à comprendre et sont parmi celles qui génèrent le plus de confiance dans les décisions prises par le modèle. Ces ré-

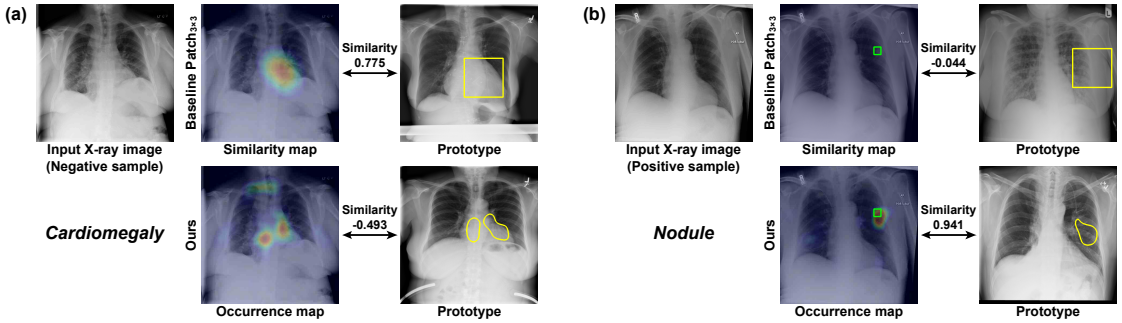


FIGURE 2.16 – Comparaison des prototypes obtenus par ProtoPNet [27] (haut) et XProtoNet [56] (bas). Les boîtes jaunes montrent les contours des prototypes, les boîtes correspondent à la vérité terrain pour les images positives. XProtoNet apprend des prototypes avec des formes variables ce qui permet de mieux capter les zones importantes.

sultats nous confortent dans le choix des approches par prototypes pour mener à bien nos travaux.

2.6 Résumé

On distingue généralement les méthodes d'interprétabilité des réseaux de neurones en deux catégories : les méthodes post-hoc et les modèles intrinsèquement interprétables. Les méthodes post-hoc ont l'avantage de s'appliquer à des modèles déjà entraînés, n'altérant pas les performances obtenues avec des architectures issues de l'état de l'art. Cependant ces méthodes sont souvent moins précises et ne renseignent pas nécessairement sur la manière dont le modèle utilise les informations d'entrée.

Les VLMs sont une option prometteuse. Les explications textuelles peuvent être une aide précieuse pour comprendre ce que le modèle voit dans une image. Mais le texte n'est pas forcément adapté pour décrire des structures visuelles assez abstraites ou très spécifiques comme ce que l'on peut trouver dans des images biomédicales. De plus le fonctionnement interne des modèles reste une boîte noire malgré la production du texte en sortie.

Les modèles intrinsèquement interprétables sont à privilégier pour s'assurer que les explications générées sont fidèles à la réalité. Parmi ces approches, les approches par prototypes ont un fonctionnement assez intuitif, basé sur un raisonnement par analogie où une image ressemble à une autre, permettant à un grand nombre de personnes de comprendre le raisonnement du réseau de neurones.

ProtoPNet [27] est un réseau de neurones qui implémente des prototypes et classifie sur la base d'une mesure de similarité entre l'image d'entrée et des prototypes appris automatiquement pour maximiser la performance de classification. C'est cette approche qui a retenu notre attention et à partir de laquelle nous avons décidé de travailler puisque ces prototypes appris devraient permettre d'identifier les changements structurels au sein des cellules qui distinguent les cellules saines des cellules atteintes, le tout dans un cadre explicable et transparent.

3

Constitution d'une jeu d'images de cellules

Outline

3.1 Introduction 34
3.2 Culture cellulaire 34
3.3 Marquage et acquisition 35
3.4 Prétraitement 37
3.4.1 Débruitage 37
3.4.2 Incorporation de l'information des tranches 37
3.5 Conclusion 41

3.1 Introduction

L'une des premières étapes préalables à l'analyse des données cellulaires est la construction d'un jeu d'images. Pour cela, nous avons cultivé des cellules endothéliales de la barrière hémato-encéphalique in vitro, réparties en plusieurs populations, que nous avons ensuite traitées avec plusieurs substances d'intérêt avant de les imager en immunofluorescence au microscope.

Dans ce chapitre nous allons détailler successivement les étapes de ce processus de constitution de nos données : en commençant par la culture cellulaire, puis le marquage fluorescent des cellules avant l'acquisition des images au microscope, et en terminant par le pré-traitement des images en vue de les analyser avec les modèles que nous avons mis au point.

3.2 Culture cellulaire

Les cellules utilisées sont des cellules endothéliales HBEC-5i (ATCC, Manassas, VA, USA). Elles ont été cultivées dans des plaques 24 puits (Dominique Dutscher, Strasbourg, France), dans du milieu de Eagle modifié par Dulbecco F12 (DMEM/F12) avec 10% de sérum de veau fœtal, 1% de solution pénicilline streptomycine et 40µg/mL de supplément de croissance des cellules endothéliales (ECGS). La culture se fait dans un incubateur à 37°C avec une atmosphère à 5% de CO2.

Sauf mention contraire, tous les réactifs ont été achetés auprès de Sigma-Aldrich (Saint-Quentin Fallavier, France).

Après la culture des cellules, nous avons pu procéder au traitement de celles-ci. Deux substances d'intérêt ont été utilisées : le TNF-α (Facteur de Nécrose Tumorale α) et l'IL-1β (InterLeukine-1β). Ces deux protéines sont des cytokines inflammatoires, c'est-à-dire qu'elles sont sécrétées par les cellules immunitaires pour répondre à une agression et favorisent une réaction inflammatoire. Le TNF-α est connu pour mener à des altérations dans l'organisation des complexes de jonction [117] et pourrait donc être à l'origine d'une disruption de la BHE. Dès lors, cette cytokine pourrait représenter un témoin positif intéressant pour la désorganisation du tissu endothélial.

Pour tester les effets de ces deux cytokines sur nos cellules, nous avons exposé les puits de notre plaque de culture à des doses croissantes de TNF-α et d'IL-1β. Les doses s'échelonnent de 1 ng/mL à 100 ng/mL avec des concentrations intermédiaires à 10 et 50 ng/mL. Pour chacune des doses, les cellules ont été exposées pendant 72 heures avant de procéder au marquage fluorescent.

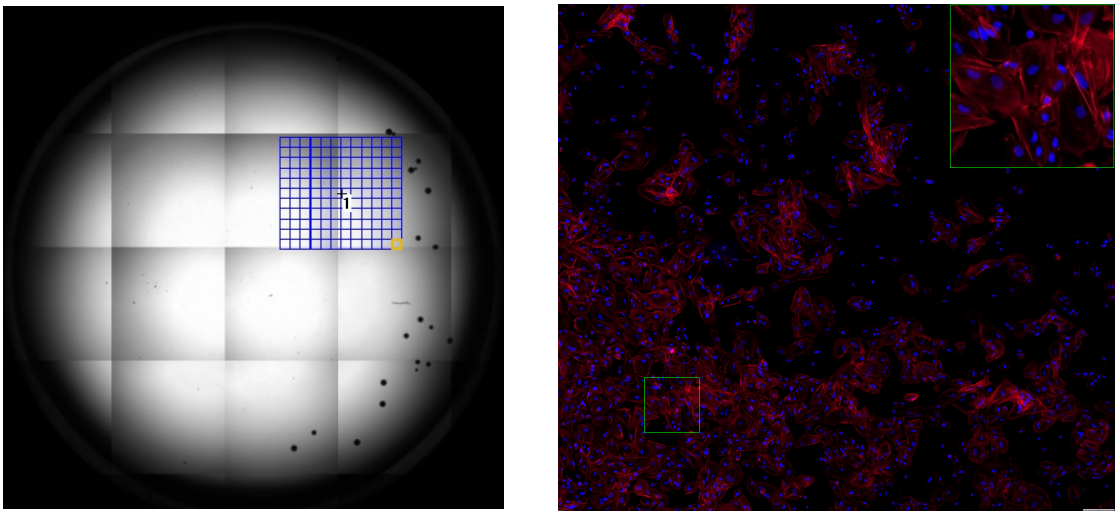
3.3 Marquage et acquisition

Après le traitement des cellules, nous avons procédé au marquage de celles-ci. Les cellules ont d'abord été fixées avec une solution de paraformaldéhyde à 4% dans du PBS (Phosphate-Buffered Saline, en français tampon phosphate salin) pendant 20 minutes. Après deux rinçages au PBS, les cellules ont été incubées avec de la phalloïdine-iFluor555 (Abcam, Cambridge, UK) pour le marquage du cytosquelette, du 4',6-diamidino-2-phénylindole dihydrochloride (DAPI ; Life Technologies, Carlsbad, CA, USA) pour le marquage des noyaux. Un marquage de la ZO-1 (Zonula Occludens 1), l'une des protéines participant aux jonctions serrées de la BHE (cf Section 1.2), avec une solution d'anticorps polyclonaux anti-ZO-1 de lapin a aussi été effectué. Cependant, ce marquage n'a pas donné de bonnes réponses visuelles et nous ne l'avons pas imagé.

L'acquisition des images s'est faite avec un microscope inversé en épifluorescence (IX81 ; Olympus, Tokyo, Japon. Celle-ci a été automatisée via le logiciel SoftImaging (SoftImaging System GmbH, Munster, Allemagne) afin de pouvoir capturer tous les puits de la plaque de culture. Nous avons aussi automatisé la prise d'images avec un focus en différents points de l'épaisseur de la cellule, afin d'être certains d'obtenir au moins une image nette, mais aussi d'avoir des informations sur la surface de la cellule ainsi qu'en son sein. A chaque prise de vue, cinq "tranches" sont prises dans l'épaisseur de la cellule, espacées de 1.75µm, pour une épaisseur totale de 7µm, le tout en grossissement $\times 40$. Il faut ensuite doubler ce nombre puisque nos images contiennent deux canaux : le canal rouge pour le contenu en actine de la cellule, et le canal bleu pour les noyaux. Le marquage vert du ZO-1 n'a pas été imagé. Le temps d'exposition pour le canal rouge est de 155.3ms et de 93.41ms pour le canal bleu.

Malgré l'automatisation du processus, l'acquisition des images est très longue. Pour une plaque entière, avec une zone d'acquisition par puits de taille similaire à celle présentée dans la figure 3.1 (a), il aura fallu près d'une dizaine d'heures. Cette durée peut poser des problèmes. A mesure que le temps passe, les fluorophores utilisés perdent en intensité. Cette variabilité des données introduite par la durée élevée de l'acquisition ne peut pas être maîtrisée et cause une incertitude supplémentaire.

Chacune des conditions de culture a été répétée sur au moins 2 puits de la plaque. Le tableau 3.1 référence la taille des images globales générées sur chacun des puits.



(a) (b)

FIGURE 3.1 – Exemple d'acquisition d'images automatisée. (a) montre l'un des puits de la plaque de culture et le quadrillage correspond à la zone à imager. Chaque carré correspond à une zone de 2048×2048 pixels dans l'image finale. (b) est le résultat de cette acquisition. Le canal rouge contient l'actine, le bleu contient les noyaux.

Population	Taille de l'image puits 1	Taille de l'image puits 2
TNF-α 1ng/mL	20478×18635	16792×16792
TNF-α 10ng/mL	18635×16792	20478×20478
TNF-α 50ng/mL	18635×16792	18635×20478
TNF-α 100ng/mL	20478×18635	18635×18635
IL-1β 1ng/mL	18635×18635	16792×16792
IL-1β 10ng/mL	18635×16792	14948×14948
IL-1β 50ng/mL	18635×16792	16792×16792
IL-1β 100ng/mL	18635×18635	16792×16792
Non traitées	18635×18635	20478×18635

TABLE 3.1 – Résumé de la taille en pixels des images acquises sur les puits de chaque condition de culture cellulaire

3.4 Prétraitement

Les données acquises au microscope sont des images brutes, dans des formats inexploitables par nos réseaux de neurones. Afin de les rendre utilisables, nous avons opéré différentes étapes de pré-traitement. Tout d'abord, du fait de leur large taille (environ 18000×18000 pixels), la première étape consiste à les redécouper en images plus petites qui seront plus faciles à manipuler. Nous avons donc découpé toutes les images brutes en carrés de 2048×2048 pixels.

3.4.1 Débruitage

Tout d'abord chacune de ces images est normalisée pour réduire l'impact des différences de luminosité et notamment les petits débris ou poussières qui ont tendance à créer des tâches saturées.

Ensuite, les images acquises en fluorescence sont entachées de plusieurs sources de bruit liées à la chaîne d'acquisition. Un filtre médian a été utilisé pour diminuer l'influence du bruit lié aux débris pouvant se trouver dans les puits de culture, puis un filtrage gaussien a été appliqué pour lisser légèrement l'image.

Ces deux étapes ont été réalisées à l'aide du logiciel Fiji [104], une distribution open-source de ImageJ2 incluant plusieurs plugins pratiques pour l'analyse d'images scientifiques, et capable de manipuler des images dans le format du logiciel d'acquisition.

3.4.2 Incorporation de l'information des tranches

Une fois les images nettoyées, nous avons fait le choix de combiner l'information des différentes "tranches" de focus en une seule image pour garder une analyse d'images en 2D et non en 3D. L'analyse en 3D serait plus coûteuse en temps et en ressources.

L'idée est donc de récupérer l'information nette dans chacune des tranches et de reconstruire une seule image qui amalgamerait toutes ces informations de manière optimale. Pour savoir si une image est nette ou non, nous avons utilisé la mesure de focus définie par Brenner et al. [20] comme :

$$\sum_{i=0}^{M-1} \sum_{j=0}^{N-3} (I(i, j+2) - I(i, j))^2 \tag{3.1}$$

Cette fonction mesure simplement la somme des carrés des dérivées premières horizontales dans l'image. Malgré sa simplicité, cette fonction donne parmi les meilleurs

Méthode de fusion	Score de Brenner	Durée d'exécution (s)
Pyramide Laplacienne (LP) [22, 23]	126.9972	0.0106
Quadtree [10]	122.4904	0.3812
Guided Filter (GF) [45]	122.8041	0.2239
Curvelet Transform (CT) [44]	122.5111	1.2751
Nonsubsampled contourlet transform (NSCT) [123]	123.1100	3.2305

TABLE 3.2 – Scores de Brenner moyens obtenus sur l'image reconstruite et temps d'exécution en fonction des différentes méthodes de fusion utilisées sur la base de 30 paires d'images. La pyramide Laplacienne semble à la fois la plus rapide et la meilleure en terme de focus.

résultats pour mesurer le focus d'une image d'après une étude comparative de Mir et al. [80], c'est pourquoi nous l'avons choisie.

Grâce à cette mesure nous allons pouvoir déterminer quelles sont les deux images les plus nettes dans la pile de cinq images qui représentent un même champ. Empiriquement, nous n'avons remarqué aucune différence visuelle, ni aucune amélioration du score de Brenner sur l'image reconstruite lorsque l'on combinait plus de deux images. Cela peut vouloir dire qu'une partie de l'information est redondante, ou bien que certaines tranches sont de toute façon trop floues pour être exploitées.

Pour combiner les deux meilleures images nous avons testé plusieurs méthodes, dont les résultats en termes de score de Brenner sur la reconstruction et en temps d'exécution sont reportés dans le tableau 3.2. Les méthodes testées sont issues d'une étude de Liu et al.[72] comparant plusieurs techniques de fusion d'images avec des focus différents. L'étude recense et compare de nombreuses approches issues de l'état de l'art, divisées en plusieurs catégories : les méthodes restant dans le domaine spatial, les méthodes utilisant un domaine de transformation et les méthodes utilisant des réseaux de neurones. Notre choix s'est arrêté sur celles offrant le meilleur compromis entre performances de fusion, temps d'exécution et besoin d'ajustement à nos données. Nous ne détaillerons ici que la méthode de la pyramide Laplacienne [22, 23] puisque c'est elle qui, dans notre cas, donne les meilleurs résultats pour la durée d'exécution la plus faible.

La pyramide laplacienne est basée sur une décomposition multi-échelle de l'image. Dans un premier temps, on crée une pyramide gaussienne ou chaque image au palier k est une version sous-échantillonnée de l'image au palier $k-1$ avec un facteur 2, obtenue en filtrant l'image au palier $k-1$ par convolution avec un filtre gaussien 5×5 et un pas de convolution de 2 pixels. Pour obtenir la pyramide

laplacienne, on calcule la différence entre l'image au palier k et l'image au palier $k+1$ sur-échantillonnée à la même taille que l'image k. L'obtention de cette pyramide laplacienne est résumée visuellement dans la figure 3.2.

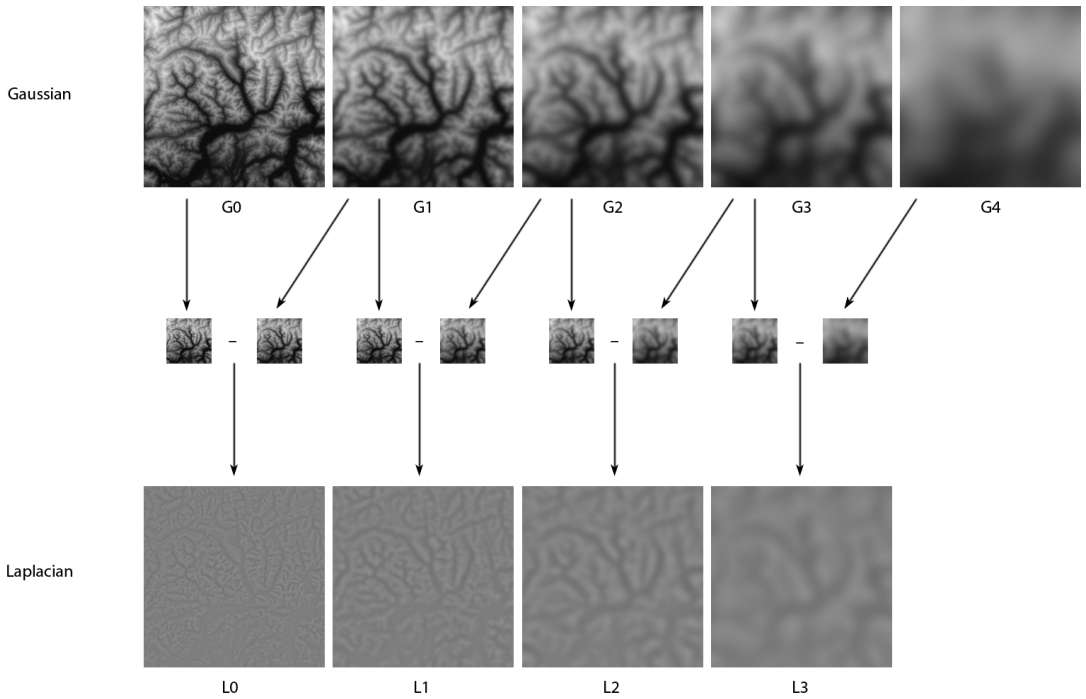
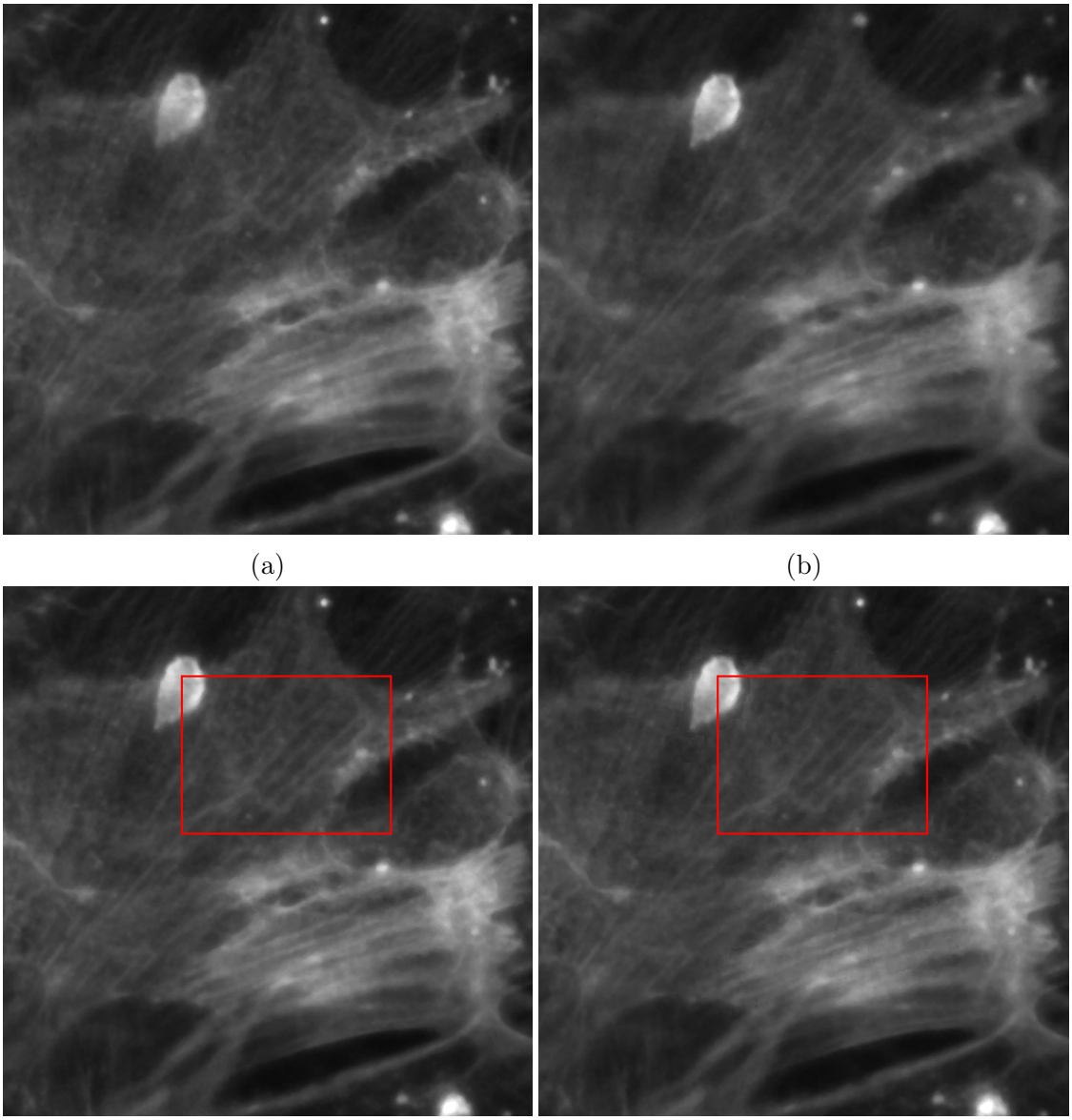


FIGURE 3.2 – Fonctionnement de la pyramide laplacienne. D'abord une pyramide gaussienne est calculée avec un filtrage gaussien pour sous-échantillonner les images successivement avec un facteur 2. Puis par soustraction de deux paliers successifs où l'on sur-échantillonne le palier le plus bas, on obtient le laplacien. Toutes les images sont montrées avec la même taille pour mieux voir le contenu.

Pour fusionner les images, on calcule la pyramide laplacienne de chacune d'entre elles. Ensuite pour chaque palier de la pyramide, on sélectionne la version dont l'intensité totale est la plus élevée. On n'a donc plus qu'une seule image par palier. Pour reconstruire une image on somme simplement tous les paliers sur-échantillonnés à la taille initiale. Un exemple de fusion de deux images avec la pyramide laplacienne et la méthode Quadtree est présenté dans les figures 3.3 et 3.4. La différence n'est presque pas perceptible, mais en regardant le zoom dans la figure 3.4, on peut voir que la reconstruction de Quadtree a produit un artefact dans le coin supérieur gauche, autour de la tâche blanche. La reconstruction obtenue par LP n'a pas de défaut de ce type.



(a) (b)

(c) (d)

FIGURE 3.3 – Exemples de résultats de fusion de deux images avec des focus différents (a) et (b) par pyramide laplacienne (LP)[22, 23] (c) et Quadtree [10] (d)

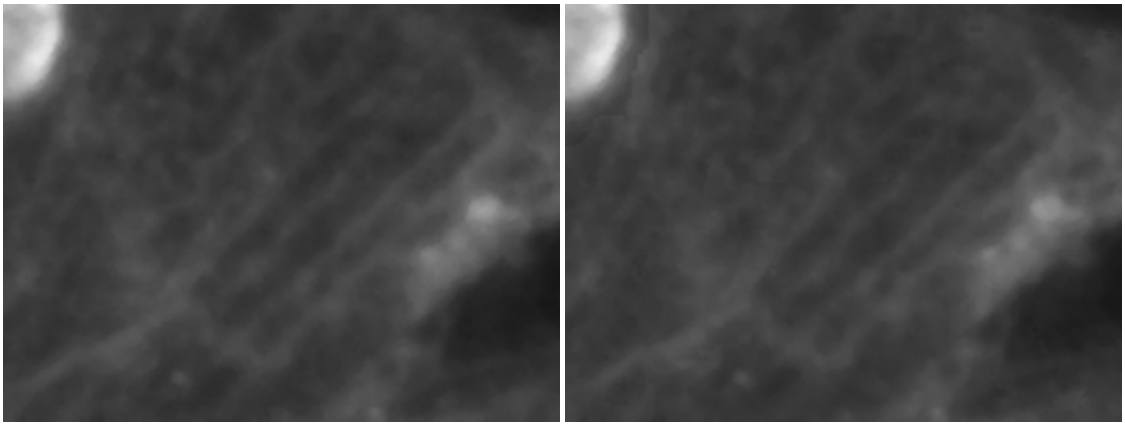


FIGURE 3.4 – Zooms sur les résultats de fusion par (a) pyramide laplacienne (LP)[22, 23] et (b) Quadtree [10]

3.5 Conclusion

Dans ce chapitre nous avons vu comment nous avons constitué un jeu de données d'images de cellules qui nous servira tout au long des travaux d'analyse de cette thèse. Les cellules qui nous intéressent, l'endothélium de la BHE, ont été cultivées in vitro dans des plaques de culture en verre. Elles ont ensuite été exposées à deux substances d'intérêt : le TNF-α et l'IL-1β. Ce sont deux cytokines produites par le système immunitaire et qui mènent à une inflammation. Le TNF-α est connu pour être à l'origine de dommages dans les complexes de jonction des cellules et pourrait donc agir de manière négative sur l'endothélium. L'IL-1β étant une protéine similaire, elle pourrait elle aussi affecter nos cellules.

Les images que nous avons obtenues ici sont une base pour les analyses menées dans les prochains chapitres. Elles seront retravaillées pour mieux correspondre aux besoins de ces analyses. Par exemple, dans le chapitre 4, nous appliquons une segmentation à nos images pour obtenir des cellules individuelles. Dans le chapitre 6, nous re-découpons les images de taille 2048x2048 en images plus petites pour pouvoir mener une analyse sur des motifs plus proches de textures. Il est difficile de donner le nombre de cellules total, mais dans chaque puits ce sont plusieurs milliers de cellules que nous avons pu capturer.

Du fait du processus d'acquisition, des caractéristiques particulières sont introduites à chaque étape. Malgré une exposition au traitement, les cellules peuvent ne pas répondre et rester saines. Inversement, les cellules qui n'ont pas été traitées et doivent servir de témoins négatifs pour l'analyse peuvent quant à elles mourir du fait des manipulations ou simplement du temps qui passe. L'acquisition des

images est très longue et pendant cette durée l'intensité des fluorophores va diminuer, introduisant des variations de contraste entre les cellules prises en photo au début et celles qui l'ont été à la fin. Le fond de la plaque de culture n'est pas parfaitement plat et bien que l'endothélium est monocouche, son épaisseur varie d'un point à un autre. Il faut donc bien régler le focus du microscope, d'autant qu'avec de forts grossissements, même une très faible variation peut rendre l'image floue.

Tous ces éléments appellent d'autant plus à analyser les images avec une méthode interprétable, qui détaille les facteurs qui ont mené à sa décision. Cela permet de s'assurer que ce ne sont pas les potentiels biais introduits qui permettent au classifieur de reconnaître de quelle population les cellules sont issues.

Malgré toutes ces spécificités et les problématiques liées, nous avons pu constituer une base contenant de nombreuses images et avec une qualité suffisante pour mener à bien notre étude.

4

Explicabilité de ProtoPNet pour l'imagerie cellulaire

Outline

4.1 Introduction 45
4.2 Méthodologie 47
4.2.1 Formulation du problème 47
4.2.2 Notre approche 48
4.3 Présentation de ProtoPNet 49
4.3.1 Principe 49
4.3.2 Entraînement 50
4.3.3 Visualisation des prototypes 51
4.4 Adaptations proposées 52
4.4.1 Segmentation et masquage 52
4.4.2 Contrôle du champ réceptif 53
4.4.3 Contrôle des prototypes 54
4.5 Résultats expérimentaux 54
4.5.1 Jeu de données 54
4.5.2 Paramètres expérimentaux 57
4.5.3 Segmentation et influence du masquage 57
4.5.4 Influence du champ réceptif 59
4.5.5 Similarités avec les prototypes 60
4.5.6 Interprétation biologique 61

4.6 Discussion et conclusion 61

4.1 Introduction

Détecter une pathologie le plus tôt possible est essentiel pour réussir à la traiter. En particulier, cette détection peut être basée sur l'analyse de cellules biologiques pour détecter si certaines ont une apparence spécifique ("dégradée" ou "saine"). Les réseaux de neurones convolutifs sont des outils très puissants pour la classification de cellules biologiques à partir d'une base de données annotée. Malheureusement, cette solution a deux inconvénients majeurs : les données doivent être précisément annotées et les décisions du réseaux de neurones sont rarement clairement expliquées à l'expert du domaine, ce qui est pourtant essentiel pour la validation humaine des résultats.

Dans ce chapitre nous proposons une solution pour extraire et afficher des structures qui sont typiques de cellules affectées ou saines pour un traitement donné. Nous allons partir de la base d'images de cellules présentée dans le chapitre précédent dans laquelle les cellules ont été ont été cultivées in vitro. Nous allons nous concentrer sur deux populations : une population saine et une population exposée à un traitement au TNF-α à une dose de 100ng/mL. Notre but est de développer un modèle qui permette d'analyser les cellules et de présenter à un expert en biologie des "prototypes" représentant les cellules saines et un autre groupe de prototypes représentant les cellules affectées par le traitement afin qu'il puisse déduire de ces prototypes les marqueurs biologiques d'une atteinte induite par le traitement.

La première difficulté réside dans le fait que nous n'avons pas de connaissances a priori sur les dégradations structurelles que les cellules peuvent subir en lien avec le traitement. Même si l'on sait que le TNF-α peut être nocif pour nos cellules comme nous l'expliquions dans la section 3.2, l'objectif final est d'appliquer la méthode pour analyser des substances dont on ne sait pas si elles sont toxiques ou non. Il faut donc mettre au point une méthode qui ne se serve pas d'informations préalables sur la structure pour parvenir à une conclusion.

Le deuxième problème qui s'impose à nous est le fait qu'une partie des cellules d'une population exposée au traitement peuvent résister à celui-ci et encore être saines. A l'inverse, certaines cellules d'une population qui n'a pas été traitée et doit servir de témoin négatif peuvent avoir été endommagées naturellement ou par les manipulations lors de l'acquisition des images. Dès lors celles-ci ne présentent plus les structures typiques des cellules saines. Comme nous n'utilisons pas de connaissances a priori et parce que cela serait particulièrement long à faire, il est impossible de labelliser une à une les images de cellules que nous voulons étudier. Nous nous contentons donc d'utiliser un label global au niveau de la population : celle-ci a-t-elle été traitée ou non. En ce sens, ces travaux peuvent être rapprochés d'autres travaux utilisant des labels bruités. La figure 4.1 illustre l'idée derrière les

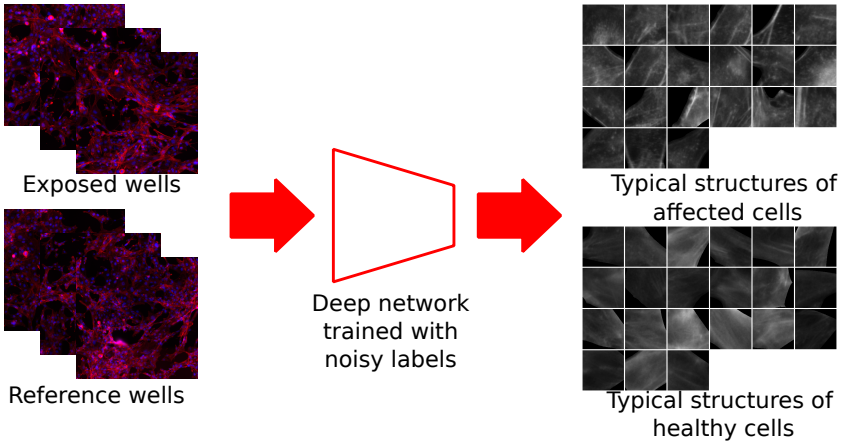


FIGURE 4.1 – Le but de ces travaux est d'entraîner un réseau profond avec des labels bruités ("Exposé", "Référence") pour chaque cellule individuelle de manière à pouvoir extraire les structures typiques des cellules saines et affectées. Ces structures sont illustrées par les prototypes, exploités par le réseau durant la tâche de classification.

travaux de ce chapitre.

Pour atteindre nos objectifs, nous tirons parti des résultats prometteurs obtenus par les approches par prototypes pour l'apprentissage profond auto-explicable, comme ProtoPNet [27]. Comme nous l'avons expliqué dans le chapitre 2, ces approches permettent d'obtenir des résultats de classification très bons grâce à l'apprentissage profond tout en offrant des explications de leurs décisions qui sont intuitives et faciles à comprendre. En effet ces méthodes extraient des exemples typiques de chaque classe, sur lesquels elles se basent pour classifier à partir d'une mesure de distance ou de similarité, de la même manière qu'un humain raisonnerait par analogie pour analyser quelque chose qu'il n'a jamais vu auparavant.

ProtoPNet a été testé sur la base de données Caltech-UCSD Birds-200-2011 [118] représentant des oiseaux de 200 classes différentes ainsi que la base de données Stanford Cars [63] contenant près de 200 classes de voitures différentes. ProtoPNet a aussi été utilisé dans un contexte biologique comme nous l'avons montré dans la section 2.5, cela concernait des scans IRM ou des radiographies. Dans le contexte spécifique de la biologie cellulaire nous montrons ici que plusieurs améliorations sont nécessaires pour tirer pleinement parti de ce réseau de neurones. Tout d'abord nous créons et donnons en entrée du réseau des masques de segmentation qui permettront de guider le réseau pour éviter que celui-ci ne s'intéresse à l'arrière-plan des images. Ensuite nous proposons de limiter la taille du champ réceptif du réseau pour s'assurer que l'encodeur extraie des caractéristiques locales. Enfin

nous contraignons le classifieur pour pousser le réseau à utiliser tous les prototypes à sa disposition pour classifier et représenter la diversité des structures au sein des cellules. Tous ces points seront discutés en détail dans les sections suivantes.

Ce chapitre est organisé comme suit :

- Exposition du problème de manière formelle qui servira à poser les notations pour le reste des explications
- Description de la méthodologie employée, dont le fonctionnement de ProtoPNet et les modifications que nous avons apportées pour mieux répondre à notre problématique
- Présentation des résultats expérimentaux et comparaison de notre approche avec la configuration par défaut de ProtoPNet.
- Analyse des défauts liés à ProtoPNet et proposition de solutions pour les résoudre, ainsi que les limitations liées à cette approche.

4.2 Méthodologie

4.2.1 Formulation du problème

Nous conduisons une expérience biologique pour investiguer comment un produit spécifique influence les caractéristiques visuelles d'une population de cellules. Nous avons deux groupes : un groupe référence, dénoté par $\mathcal{P}^{\text{Ref}}$, qui n'a pas été exposé au produit étudié, et un groupe expérimental, dénoté par $\mathcal{P}^{\text{Exp}}$, qui lui y a été exposé. Initialement, les deux groupes de cellules sont cultivés dans des plaques dédiées puis traités avec un agent fluorescent, selon la méthode décrite dans le chapitre 3. Chaque image peut contenir plusieurs cellules, lesquelles sont donc segmentées individuellement. L'image d'une cellule unique i est formellement représentée par I_i, et le masque de segmentation correspondant est noté M_i, encodé comme une image binaire avec les mêmes dimensions spatiales que I_i.

Notre objectif principal est de déterminer si les cellules montrent des changements visuels dus à l'exposition au produit. Pour y parvenir, nous assignons un label, Y_i, à chacune des cellules i, où $Y_i = 0$ indique qu'il n'y a aucun changement visuel et $Y_i = 1$ indique une altération visuelle. Notre tâche peut donc être considérée comme un problème de classification supervisée, avec comme données d'entrée $X_i = \{I_i, M_i\}$ pour $i \in \mathcal{P}^{\text{Ref}} \cup \mathcal{P}^{\text{Exp}}$, et la prédiction du label Y_i comme objectif.

Cependant, d'un point de vue biologique, nous savons que toutes les cellules dans $\mathcal{P}^{\text{Exp}}$ ne montrent pas de changements visuels ; certaines restent non-affectées.

Inversement, certaines cellules dans $\mathcal{P}^{\text{Ref}}$ peuvent montrer des caractéristiques visuelles similaires à celles des cellules affectées. De plus, déterminer le vrai label Y_i est pratiquement infaisable car nous manquons de connaissances a priori sur les caractéristiques visuelles des cellules affectées. Ainsi, notre entraînement n'est donc que faiblement supervisé, puisque nous nous basons sur des labels imparfaits dérivés de la population à laquelle appartiennent les cellules, qui peuvent être faux à l'échelle d'une seule cellule. Ces labels sont notés : $\widetilde{Y}_i = \delta(i \in \mathcal{P}^{\text{Exp}})$. Pendant la phase de test en revanche, notre but est de prédire le vrai label Y_i associé à la cellule.

Étant donnés les vrais labels inconnus, notre objectif est d'analyser les décisions prises par le système de classification. Cette analyse remplit deux rôles : tout d'abord, évaluer la qualité de la prédiction, et deuxièmement, identifier les structures visuelles qui sont indicatrices de l'apparence visuelle des cellules affectées. Spécifiquement, nous souhaitons obtenir des structures prototypiques qui remplissent trois critères :

- I/ Caractériser les changements intra-cellulaires
- II/ Représenter des textures locales
- III/ Servir de marqueur visuel de l'état biologique de la cellule

4.2.2 Notre approche

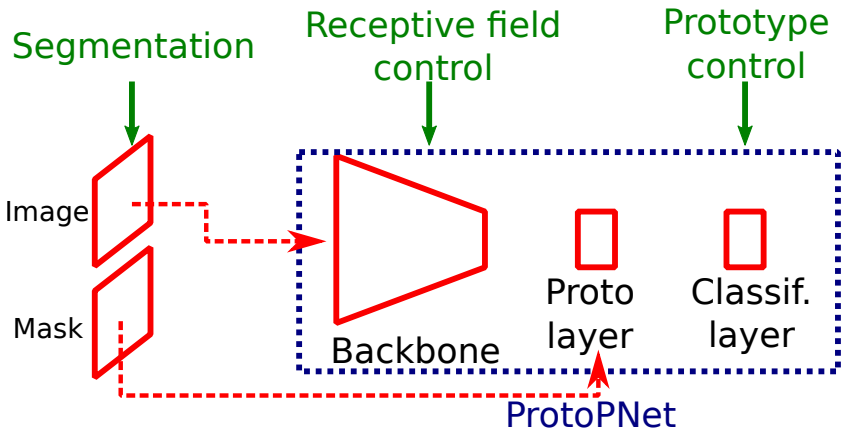


FIGURE 4.2 – L'architecture proposée basée sur ProtoPNet avec trois adaptations principales : un masque de segmentation comme seconde entrée pour empêcher le réseau d'utiliser le fond des images, un champ réceptif limité pour assurer la localité des résultats, et le contrôle des poids de la couche de classification pour renforcer la diversité des prototypes.

Notre modèle tire parti de ProtoPNet, l'architecture basée sur les prototypes proposée par Chen et al. [27], comme illustré dans la figure 4.2. En plus de cette architecture, nous proposons trois contributions importantes :

- Nous appliquons une segmentation à l'image d'entrée dans le but d'extraire des cellules individuelles et les masques de segmentation associés. Le masque est donné en entrée à la couche des prototypes. Ce masque doit empêcher les prototypes de représenter l'arrière-plan noir des images.
- Nous contrôlons la taille du champ réceptif en choisissant comme encodeur l'architecture la plus adaptée pour obtenir des visualisations des prototypes plus facilement interprétables.
- Nous gelons la couche de classification afin de contrôler l'activation des prototypes sur les classes et améliorer la diversité des structures représentées par ceux-ci.

Nous commençons donc, dans la section 4.3, par présenter les détails de l'architecture de ProtoPNet, son entraînement et la visualisation des prototypes appris. Ensuite, dans la section 4.4, nous présentons les adaptations que nous avons proposées.

4.3 Présentation de ProtoPNet

4.3.1 Principe

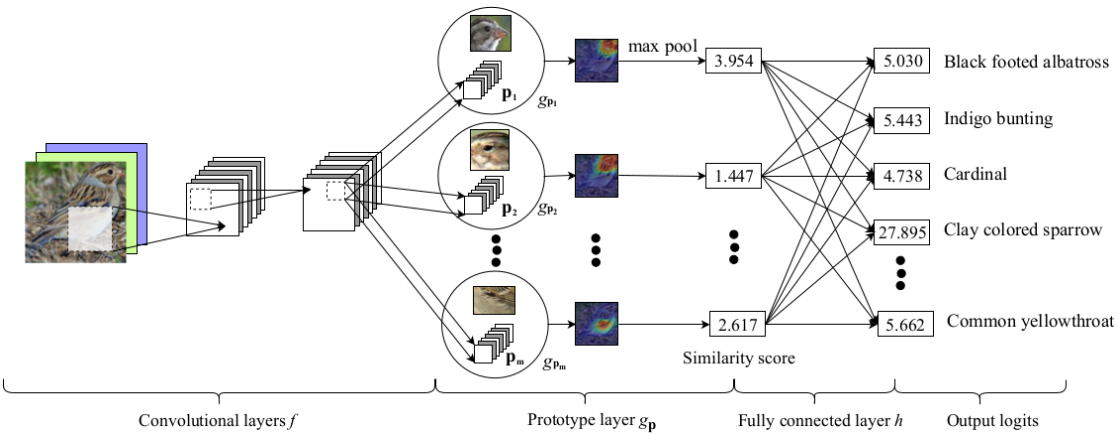


FIGURE 4.3 – Architecture de ProtoPNet [27]

ProtoPNet est composé de trois parties (Figure 4.3) : un encodeur convolutif f, une couche de prototypes g et une couche linéaire h pour la classification. Dans

les expériences de la publication originelle, Chen et al. utilisent des encodeurs typiques issus de l'état de l'art : VGG-16, VGG-19 [108], ResNet-34, ResNet-152 [46], DenseNet-121 et DenseNet-161 [52] ; avec des poids pré-entraînés sur ImageNet [31]. Deux couches convolutives avec un noyau 1×1 sont ajoutées en sortie de l'encodeur avec des activations ReLU. Le rôle de l'encodeur est d'extraire les caractéristiques visuelles pertinentes des images d'entrée I et les transformer en une carte de features Z. Ensuite, cette carte de features Z est donnée en entrée à la couche de prototypes. Cette couche contient autant de neurones que l'on souhaite apprendre de prototypes. Chaque prototype est de dimension identique à un élément de la carte de features Z, représentant une région de l'image d'entrée appelée patch. Il est possible de prendre des prototypes plus grands, correspondant à plusieurs patches de Z mais Chen et al. n'explorent pas cette piste. La couche de prototypes calcule une similarité entre chacun des prototypes qu'elle garde en mémoire et tous les patches de la feature map (ou bien tous les ensembles de patches contigus de taille similaire au prototype le cas échéant) avec la formule suivante :

$$g_{p_j} = \max_{z \in \text{patches(Z)}} \log\left(\frac{d_{j,z}^2 + 1}{d_{j,z}^2 + \epsilon}\right) \tag{4.1}$$

avec g_{p_j} le neurone associé à un prototype donné p_j, z un patch de la feature map Z et $d_{j,z}$ la distance L^2 entre le prototype p_j et z.

On a donc en sortie une valeur par prototype, correspondant à la valeur de similarité maximale entre ce prototype et les patches de Z. La couche linéaire finale multiplie ces valeurs de similarité calculées précédemment par une matrice de poids w_h pour obtenir en sortie des logits pour chacune des classes, qui sont ensuite passés dans une fonction softmax pour être transformés en probabilités. Cette couche de classification est initialisée de manière à associer les prototypes à une classe, en prenant une valeur 1 pour le poids entre un prototype et la classe à laquelle il est censé appartenir, et -0.5 vers les autres classes.

4.3.2 Entraînement

L'entraînement de ProtoPNet est décomposé en trois étapes. D'abord toutes les couches sauf h sont entraînées en utilisant la fonction de perte suivante :

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \text{CrsEnt}(h \circ g_{p_j} \circ f(X_i), \widetilde{Y}_i) + \lambda_1 \,\text{Clst} + \lambda_2 \,\text{Sep} \tag{4.2}$$

où CrsEnt désigne la perte d'entropie croisée et n est la taille du jeu de données d'entraînement. Clst et Sep sont définis comme :

$$\begin{aligned} \text{Clst} &= \frac{1}{n}\sum_{i=1}^{n} \min_{p_j \in P_{\widetilde{Y}_i}} \min_{z \in \text{patches}(f(X_i))} d_{j,z}^2 \\ \text{Sep} &= -\frac{1}{n}\sum_{i=1}^{n} \min_{p_j \notin P_{\widetilde{Y}_i}} \min_{z \in \text{patches}(f(X_i))} d_{j,z}^2 \end{aligned} \tag{4.3}$$

Le terme de clustering Clst pousse chaque image d'entraînement à avoir au moins un de ses patches proche d'au moins un prototype de la bonne classe dans l'espace latent. Le terme de séparation Sep pousse chaque image d'entraînement à avoir tous ses patches éloignés de tous les prototypes des mauvaises classes dans l'espace latent.

La deuxième étape de l'algorithme d'entraînement est l'étape de projection des prototypes. Pendant la première étape, la couche des prototypes était initialisée avec des points aléatoires dans l'espace latent qui sont ensuite déplacés pour minimiser la fonction de perte.

Chaque prototype p_j est projeté sur le patch d'une image d'entrée de la même classe le plus proche dans l'espace latent, dans le but d'être capable de comprendre ce que ce point de l'espace latent veut dire visuellement. Cette étape de projection est définie comme :

$$p_j \leftarrow \arg\min_{z \in \mathcal{Z}_j} d \tag{4.4}$$

avec $\mathcal{Z}_j$ l'ensemble de tous les patches de la même classe que p_j. Comme cette opération déplace les prototypes qui avaient été optimisés pendant l'étape précédente pour minimiser la perte et maximiser la précision en classification, les performances chutent après cette seconde étape.

Il y a donc une troisième étape qui consiste simplement à optimiser seulement la dernière couche de classification en figeant le reste du réseau afin de compenser la perte de précision induite par l'étape de projection. Ces trois étapes peuvent être répétées afin d'obtenir de meilleurs résultats.

La procédure d'entraînement de ProtoPNet est résumée par l'algorithme 1.

4.3.3 Visualisation des prototypes

Pendant l'étape de projection de l'entraînement, chaque prototype est poussé vers le patch d'une image de la base d'entraînement le plus proche dans l'espace latent. Pour trouver à quel patch dans l'espace pixel original il correspond, Chen et al.

Algorithme 1 Entraînement de ProtoPNet

Initialisation : $w_f \leftarrow$ poids pré-entraînés sur ImageNet ou 0 ; $\forall j : p_j \leftarrow$ valeurs aléatoires ; $\forall k, j : w_h^{(k,j)} \leftarrow 1$ si $p_j \in P_k$, $w_h^{(k,j)} \leftarrow -0.5$ si $p_j \notin P_k$; époque $t_0 = 0$;
Tant que $t_0 < t_{\text{fin}}$ ET NON(convergence) **Faire**
  Pour $t = 0$ à $N_{\text{étape 1}}$ **Faire**
    $w_f \leftarrow \nabla_{w_f} \mathcal{L}(X, \tilde{Y})$
    $P \leftarrow \nabla_P \mathcal{L}(X, \tilde{Y})$
  Fin Pour
  $t_0 \leftarrow t_0 + N_{\text{étape 1}}$
  Pour Tout $p_j \in P$ **Faire**
    $p_j \leftarrow \arg\min_{z \in \mathcal{Z}_j} d$
  Fin Pour
  Geler f ; Geler g ;
  Pour $t = 0$ à $N_{\text{étape 3}}$ **Faire**
    $w_h \leftarrow \nabla_{w_h} \text{CrsEnt}(h \circ g \circ f(X), \widetilde{Y})$
  Fin Pour
  Dégeler f ; Dégeler g ;
Fin Tant que

passent l'image dont il provient en entrée du réseau, puis suréchantillonnent la carte de similarité en sortie du neurone correspondant au prototype g_{p_j}, à la même taille que l'image d'entrée. Chen at al. définissent alors comme visualisation du prototype le rectangle le plus petit qui inclut au moins le 95ème pourcentile de toutes les valeurs de similarité dans la carte.

4.4 Adaptations proposées

Dans cette section, nous allons introduire les différentes modifications que nous avons apportées à ProtoPNet pour mieux répondre au problème formulé dans la section 4.2.1. Pour guider le réseau sur les structure internes aux cellules, nous mettons tout d'abord en place un masquage du fond. Puis, nous modifions la méthode utilisée pour visualiser les prototypes en nous servant de l'information du champ réceptif de l'encodeur convolutif. Enfin, nous contrôlons les poids de la couche de classification pour forcer le réseau à apprendre des prototypes pertinents.

4.4.1 Segmentation et masquage

Comme nous venons de l'expliquer, nous voulons nous assurer que les prototypes représentent des structures intra-cellulaires. Pour y parvenir, nous proposons d'uti-

liser une segmentation pour travailler sur des cellules individuelles. Le masque de segmentation peut ensuite être utilisé pour restreindre les activations du modèle sur les zones non-significatives. Pour cela on ajoute à la carte de distances entre un prototype p_j et tous les patches $z \in Z$ calculée à l'intérieur de g_{p_j} le masque binaire de segmentation M_z sous-échantillonné à la taille de la carte, et dont les pixels sont à 1 lorsque qu'ils ne correspondent pas à une zone d'intérêt. Cela permet d'augmenter artificiellement la distance entre l'arrière-plan ou les zones inutiles et les prototypes, réduisant ainsi la possibilité d'apprendre un prototype qui leur ressemble. Mathématiquement, cette opération est calculée de la manière suivante :

$$d_{j,z} = ||z - p_j||_2 + aM_z \tag{4.5}$$

où M_z est la valeur du masque de segmentation sous-échantillonné aux mêmes coordonnées que z, et a est un nombre arbitrairement grand.

4.4.2 Contrôle du champ réceptif

Ici nous proposons de limiter la taille du champ réceptif afin de renforcer la localité des structures représentées par les prototypes. Un grand champ réceptif va incorporer de l'information venant d'un grand patch de l'espace pixel dans un seul patch latent. Il devient alors très dur de comprendre ce que la similarité entre le prototype et ce patch latent veut véritablement dire dans l'espace pixel.

Un champ réceptif plus petit va aider avec ce problème de localité puisque l'ensemble des pixels correspondants dans l'espace d'entrée sera plus petit. Les caractéristiques profondes apprises par l'encodeur convolutif seront plus à même de décrire des textures, plutôt que des structures avec un faible niveau d'abstraction comme il serait d'ordinaire le cas.

Comme nous l'évoquions dans la section 4.3, ProtoPNet est prévu pour fonctionner avec différents encodeurs convolutifs communs. La plupart d'entre eux possèdent des champs réceptifs plus grands que les images que nous souhaitons analyser. Cela pose donc évidemment le problème de localité dont nous venons de parler.

Mais nous souhaitons également contrôler le champ réceptif du réseau pour éviter un autre écueil de ProtoPNet : la visualisation des prototypes. Carmichael et al. [26] proposent de limiter la visualisation au rectangle de pixels de l'image d'entrée correspondant au champ réceptif, toujours centré autour de l'activation la plus forte pour le prototype. Cela permet de s'assurer que la visualisation des prototypes n'est pas impactée par la méthode de suréchantillonnage utilisée pour remonter à l'espace pixel de départ, et surtout, que l'on étudie bien tous les pixels qui ont contribué à l'activation du prototype. C'est cette méthode de visualisation que nous allons utiliser dans nos expériences.

4.4.3 Contrôle des prototypes

Avec nos données, ce que nous nous attendons à obtenir en matière d'activation des prototypes est illustré dans la figure 4.4. Pour chaque prototype, nous extrayons la valeur de similarité maximum de chaque image de l'ensemble d'entraînement. Ensuite nous calculons la valeur moyenne de ces maxima de similarité pour les deux populations séparées $\mathcal{P}^{\text{Ref}}$ et $\mathcal{P}^{\text{Exp}}$ pour obtenir le graphe de gauche sur la figure 4.4. Le graphe de droite représente la différence entre les deux distributions de gauche.

Pour les cellules de $\mathcal{P}^{\text{Ref}}$, la similarité maximale avec les prototypes "Référence" devrait être élevée et très faible avec les prototypes "Exposée". Pour $\mathcal{P}^{\text{Exp}}$ nous nous attendons à avoir une similarité haute avec les prototypes "Exposée" et une similarité modérément élevée avec les prototypes "Référence". Cette dissymétrie théorique vient du fait que nos classes "Exposée" et "Référence" sont imparfaites et que des cellules qui ont été exposées au traitement vont rester saines et donc avoir la même apparence que les cellules de "Référence" sur toute ou partie de leur surface.

Nous proposons de geler les poids de la dernière couche de classification avec des valeurs à 1 pour les poids reliant un prototype à la classe à laquelle il est censé appartenir et -0.5 pour la classe opposée, comme cela est fait à l'initialisation. Empiriquement nous nous sommes rendus compte que durant l'entraînement de nombreux poids de la couche avaient tendance à diminuer jusqu'à 0, rendant les prototypes associés inutiles pour la classification et ceux-ci sont donc très peu mis à jour et ne deviennent plus représentatifs de structures typiques d'une population. En fixant les poids, nous nous attendons à ce que le modèle soit forcé d'apprendre des prototypes qui ont du sens puisque ceux-ci pèseront nécessairement dans la classification. Les valeurs déséquilibrées entre la classe correcte et la mauvaise classe permettent que des prototypes "Référence" qui se trouveraient dans des cellules de la classe "Exposée" ne faussent pas la classification.

4.5 Résultats expérimentaux

4.5.1 Jeu de données

Le jeu de données que nous avons utilisé pour mener nos expériences est constitué à partir des images que nous avons acquises et que nous avons décrites dans le chapitre 3. Il s'agit donc d'images de cellules endothéliales de la barrière hémato-encéphalique avec un canal rouge contenant l'information liée à l'actine et un canal bleu contenant les noyaux.

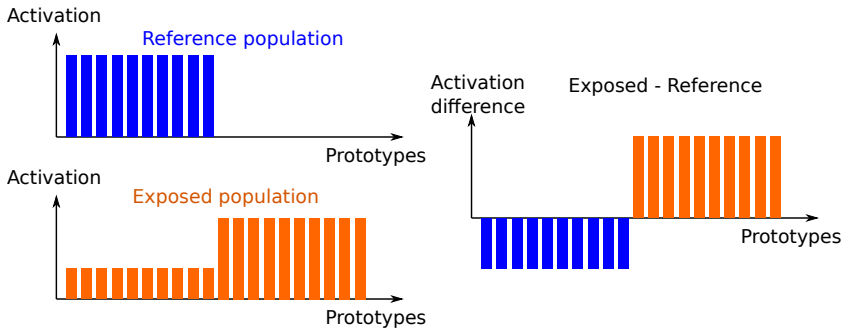
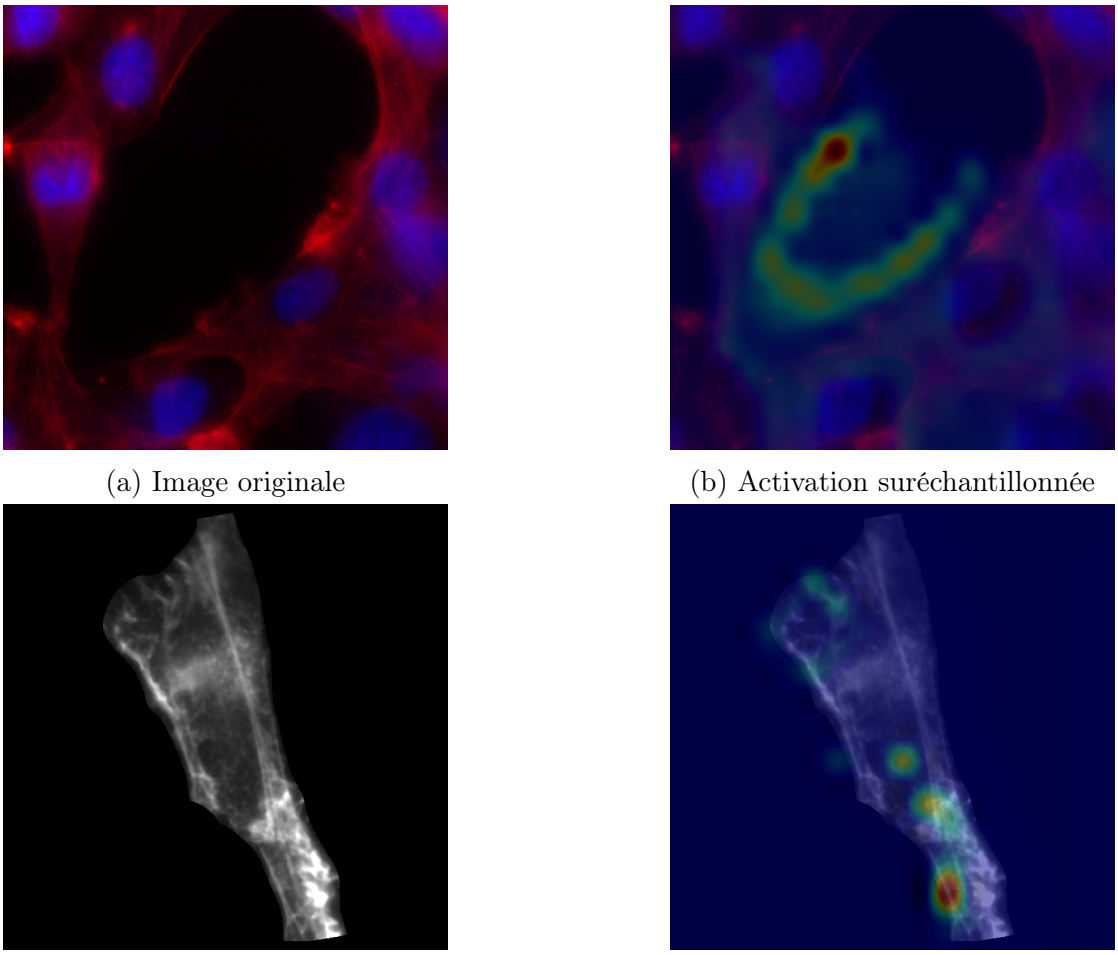


FIGURE 4.4 – Formes attendues de (gauche) la moyenne pour chaque population de la similarité maximum avec chacun des prototypes et (droite) la différence entre ces moyennes de similarité entre les populations "Exposée" (en orange) et "Référence" (en bleu).

Nous sommes partis des images en 2048×2048 pixels et nous avons opéré une segmentation pour travailler avec des images contenant seulement une cellule. La méthode de segmentation est expliquée dans la section 4.5.3. Après la segmentation, nous avons trié les images obtenues. Certaines contenaient des débris plutôt que des cellules et nous avons aussi essayé d'enlever les images contenant des cellules incomplètes car situées au bord d'une image initiale. Cet effet de bord pourrait être limité en travaillant avec des images plus grandes lors du pré-traitement mais nous avons tout de même plus de 3000 images de cellules avec lesquelles travailler ce qui est suffisant pour notre application. Nous avons aussi ramené toutes les images de cellules uniques à la même taille pour faciliter certains calculs dans le réseau. Nous avons fixé cette taille à 650×650 pixels, ce qui correspondait à la taille des plus grandes cellules. Pour ne pas déformer les plus petites images, nous avons simplement rempli le fond avec du noir jusqu'à obtenir la taille désirée.

Dans les expériences de ce chapitre, seules les populations non-traitées et la population traitée au TNF-α à la dose 100ng/mL ont été étudiées. De même, le canal bleu n'est pas utilisé en dehors de la segmentation et le réseau de neurones travaille donc uniquement sur des images en niveaux de gris qui correspondent au canal rouge de l'actine.

Les données sont séparées en un jeu d'entraînement et un jeu de test contenant chacun 50% des images de départ, soit plus d'un millier de cellules. Le jeu d'entraînement a été augmenté avec les mêmes opérations que celles utilisées pour ProtoPNet, à savoir des rotations aléatoires entre $-15°$ et $+15°$, un retournement horizontal avec une probabilité de 50%, du cisaillement (shear) aléatoire avec une intensité maximale de 10 à gauche ou à droite et des transformations projectives (skew) sans rotation. Le package Python Augmentor [15] a été utilisé pour calcu-



(a) Image originale (b) Activation suréchantillonnée

(c) Image originale (d) Activation suréchantillonnée

FIGURE 4.5 – Exemple de l'activation d'un prototype (b) créé par le modèle sur une image non-segmentée (a). Le patch d'arrière plan active le prototype. Sur une image segmentée (c), l'activation obtenue en utilisant le masque de segmentation(d) est localisée uniquement dans la cellule.

ler ces transformations. Toutes les transformations ont été effectuées de manière statique avant l'entraînement et sont donc fixes pour toutes les expériences.

4.5.2 Paramètres expérimentaux

Nous avons basé nos expériences sur le code Python fourni par Chen et al. [27] qui a servi pour leurs expériences avec ProtoPNet. Ce code utilise la bibliothèque PyTorch [90], une bibliothèque logicielle très courante pour créer et entraîner des réseaux de neurones en Python.

Nous avons choisi de créer 10 prototypes pour chaque classe. Les facteurs λ_1 et λ_2, qui multiplient respectivement les termes Clst et Sep de la fonction de perte utilisée pour l'étape 1 de l'entraînement du modèle, ont été réglés à 0.8 et 0.08, valeurs recommandées dans la publication originelle. Le réseau a été entraîné pendant 30 époques, avec une projection des prototypes toutes les 10 époques, suivie d'une optimisation de la dernière couche pendant 10 époques également. L'algorithme d'optimisation utilisé est Adam [57] avec un taux d'apprentissage de 10^{-4}. Ces paramètres resteront les mêmes pour toutes les expériences que nous mèneront dans ce chapitre. En utilisant ces paramètres par défaut donnés dans [27], couplés à un encodeur DenseNet-121 qui donne les meilleurs résultats dans le comparatif donné dans cette publication, et sans segmentation, nous obtenons une précision de 78.93%. Nous considérons cette valeur comme notre référence et examinerons l'influence de nos modifications sur celle-ci. Un récapitulatif des différentes précisions obtenues avec chacune des adaptations est proposé dans le tableau 4.1.

4.5.3 Segmentation et influence du masquage

Pour segmenter nos images, nous avons utilisé l'algorithme Cellpose [110], et plus précisément le modèle fourni pré-entraîné sur des images de cytosquelettes, en utilisant le canal rouge de l'actine comme canal principal et le canal bleu des noyaux comme canal secondaire. Le paramètre de rayon des cellules détecté automatiquement a été utilisé car la taille des cellules est assez variable d'une image à l'autre. La figure 4.6 montre les résultats de segmentation pour une image de notre base de données. Ce résultat n'est pas optimal car nous n'avons pas ré-entraîné leur modèle sur nos données, mais les performances nous semblaient suffisantes pour effectuer le reste des opérations correctement. De plus, nos travaux se concentrent sur la classification interprétable et non sur la segmentation et il nous semblait chronophage de développer une meilleure méthode de segmentation. Nous reviendrons sur ces problèmes et les améliorations possibles dans la section 4.6.

La figure 4.5 montre une comparaison des cartes d'activation obtenues par notre modèle en utilisant une entrée non-segmentée et l'activation obtenue avec une en-

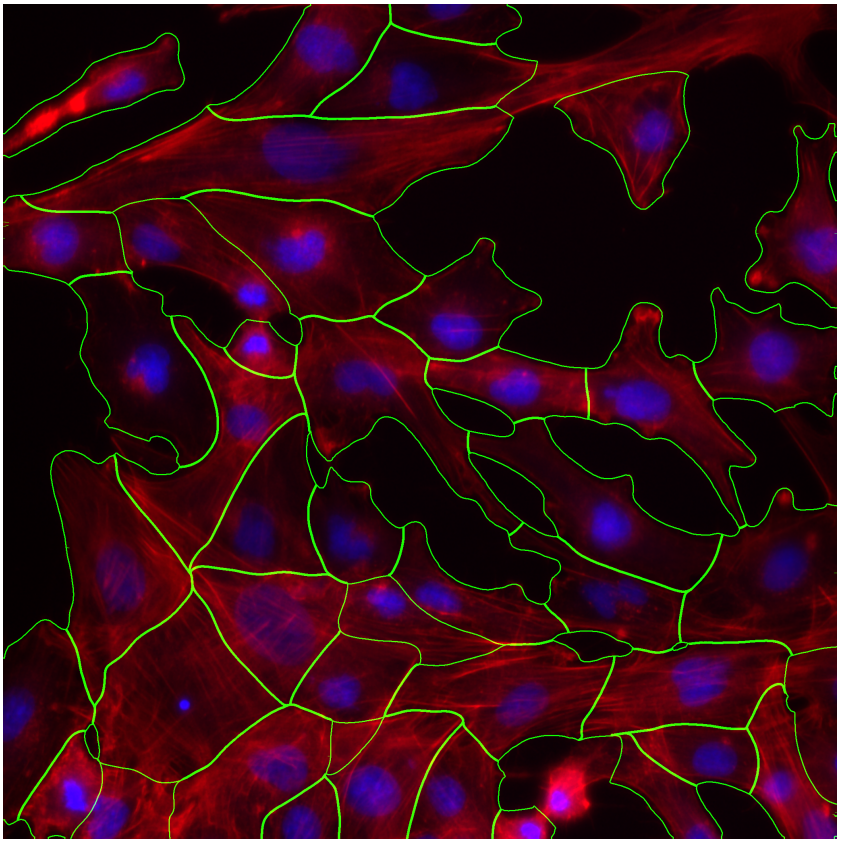
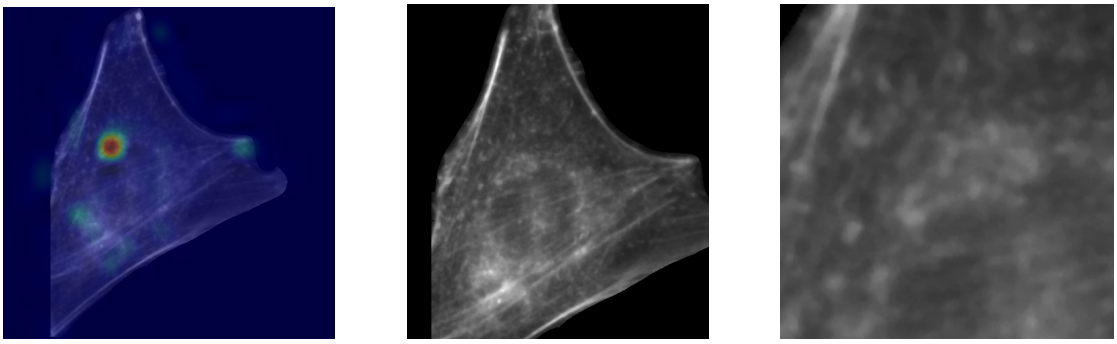


FIGURE 4.6 – Résultat de segmentation obtenu par Cellpose [110] sur une image de cellules

trée segmentée et en utilisant le masque pour conditionner les cartes de distances calculées par la couche des prototypes à rester à l'intérieur de la cellule. Nous pouvons voir que la version du modèle qui n'utilise pas la segmentation (ligne du haut) a créé un prototype qui ressemble à l'arrière-plan noir. Si ce prototype a peut-être permis de classifier, l'information qu'il apporte n'est pas pertinente dans notre analyse. Il nous renseigne sur le fait que les images de cellules traitées contiennent typiquement plus de fond. Cela se comprend car les cellules endommagées par le traitement auront sans doute plus de mal à rester accrochées à leur support et pourraient être arrachées pendant les manipulations pour le marquage fluorescent. Ou bien cela peut être lié à de mauvaises manipulations plus globalement. Le prototype ne contient donc pas d'information biologique intra-cellulaire et le prototype ne permet pas de déduire le type de changement que les cellules ont pu subir. En utilisant la segmentation (ligne du bas), on voit que cette fois-ci la carte de similarité ne prend des valeurs non-nulles qu'à l'intérieur de la cellule. En visualisant le prototype, nous pourrons alors tenter de comprendre quelle structure



(a) (b) (c)

FIGURE 4.7 – Comparaison de la visualisation d'un prototype pour une carte d'activation donnée (a) générée en utilisant (b) la plus petite boîte englobant au moins le 95ème pourcentile d'activation, (c) une boîte de la taille du champ réceptif centrée autour de la plus haute valeur d'activation. Résultats obtenus en utilisant VGG-11 comme encodeur.

est représentée et quelle information biologique elle porte.

En plus d'améliorer l'interprétabilité des prototypes, la segmentation permet également d'améliorer la classification. De notre performance référence à 78.93% nous passons à 83.93%.

4.5.4 Influence du champ réceptif

Pour valider nos hypothèses sur l'effet du champ réceptif nous avons testé un encodeur convolutif différent de notre configuration de référence. Là où nos expériences précédentes utilisaient DenseNet-121, nous utilisons désormais VGG-11 lui aussi pré-entraîné sur ImageNet. Le champ réceptif de DenseNet-121 est plus grand que nos images d'entrée, tandis que VGG-11 possède un champ réceptif de 150×150 pixels.

La figure 4.7 montre que la méthode initiale pour la visualisation des prototypes proposée par ProtoPNet (a) n'est pas adaptée à notre cas. La carte de similarité possède des pics à plusieurs endroits de la cellule, ce qui n'est pas étonnant puisqu'une même structure peut se retrouver à plusieurs endroits de celle-ci. Seulement, en essayant d'englober jusqu'au 95ème pourcentile des valeurs de similarité dans un seul rectangle, la visualisation ne va pas réussir à mettre en lumière une seule région et plutôt représenter toute la cellule (b). On n'obtient donc pas d'information pertinente avec cette méthode de visualisation. A la place, nous proposons d'utiliser comme visualisation d'un prototype un rectangle de la taille du champ réceptif du réseau centré autour de la valeur maximale de similarité dans la carte

	Référence	Segmentation et Masque	Dernière couche figée
VGG-11	68.79	77.51	89.75
DenseNet-121	78.93	83.93	86.53

TABLE 4.1 – Précisions obtenues avec les différentes configurations expérimentales

suréchantillonnée (c).

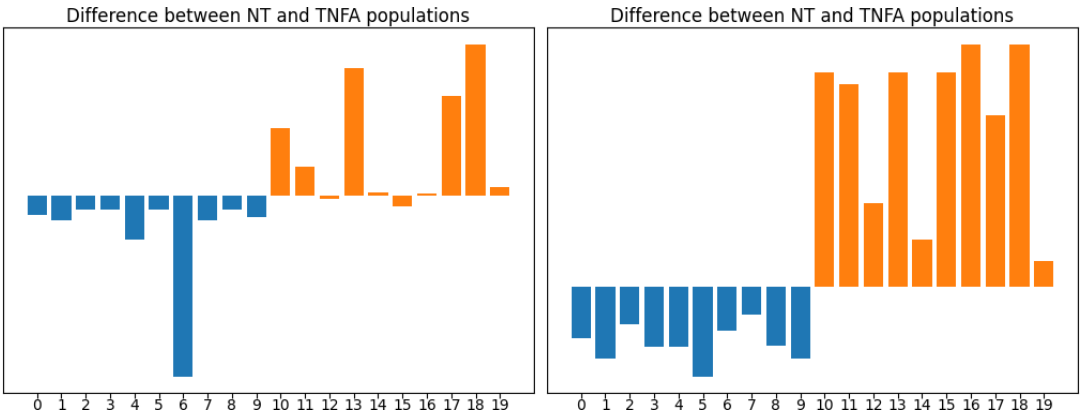
Avec VGG-11, on distingue bien mieux le type de structure que le prototype encode étant donné que la taille du champ réceptif est réduite. Si l'on avait gardé DenseNet-121 comme encodeur, ce type de visualisation n'aurait pas donné de résultat exploitable puisqu'il aurait renvoyé l'image entière et n'aurait pas mis en valeur une texture ou une structure en particulier.

Quant à la performance en classification, notre configuration utilisant VGG-11, combinée à la segmentation et au masquage obtient seulement 77.51% de précision. C'est légèrement inférieur à ce que nous obtenons en référence avec DenseNet-121, mais le gain d'interprétabilité nous paraît être un compromis satisfaisant pour une perte de précision somme toute minime.

4.5.5 Similarités avec les prototypes

Nous avons décrit les similarités idéales auxquelles nous nous attendions dans la section 4.4.3. Ici nous montrons la différence entre les moyennes des similarités maximales pour les deux populations comme dans la figure 4.4 (b). La figure 4.8 montre le résultat obtenu en optimisant la dernière couche de classification comme cela est fait durant l'étape de 3 l'entraînement de ProtoPNet (a) et le résultat que nous obtenons en figeant les poids de cette couche à leur valeur initiale (b). Nous pouvons observer que la méthode originale produit des distributions assez clairsemées avec seulement quelques prototypes qui portent une ressemblance avec les structures cellulaires. En figeant la dernière couche nous obtenons une distribution plus dense qui semble traduire que nous parvenons à capturer plus de diversité de textures existantes. Cette propriété nous est particulièrement utile pour essayer de comprendre tous les changements que pourraient subir les cellules et ne pas se concentrer exclusivement sur un seul marqueur.

Enfin, du point de vue de la classification, la précision grimpe à 89.75% en combinant toutes nos modifications, soit la meilleure performance que nous ayons obtenue jusque là, tout en fournissant des prototypes plus divers et plus facilement interprétables.



(a) Poids non-figés (b) Poids figés

FIGURE 4.8 – Différence entre les similarités aux prototypes comme dans la figure 4.4

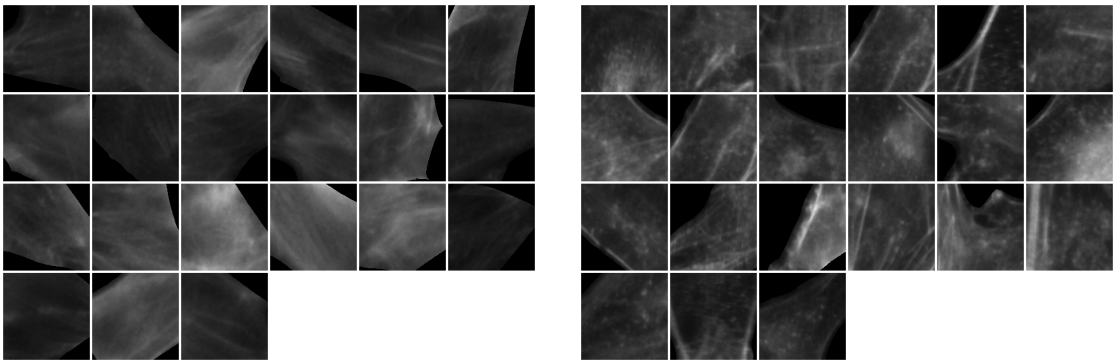
4.5.6 Interprétation biologique

Une seule instance d'un prototype n'est souvent pas suffisante pour comprendre pleinement la structure biologique que celui-ci représente. Pour mieux comprendre ce que les textures des prototypes montrent, nous avons parcouru la base d'entraînement pour récupérer les 20 images qui activent le plus chacun des prototypes et appliqué notre méthode de visualisation sur chacune d'entre elles pour obtenir 20 patches de la taille du champ réceptif. En combinant ensemble ces patches dans une sorte de mosaïque, il devient plus simple de comprendre le motif capturé par le prototype.

Sous cette forme de mosaïque, nous pouvons voir deux types de textures se dégager dans la figure 4.9. La première est plutôt uniforme avec des filaments d'actine faiblement contrastés (a). A l'inverse, la seconde texture est bien plus contrastée et montre une texture à l'aspect granulaire avec des filaments d'actine assez marqués, surtout sur les bords des cellules. D'un point de vue biologique, ces deux textures semblent décrire une modification de l'organisation de l'actine au sein de la cellule, qui pourrait être liée à une altération de la fonction des cellules endothéliales et à une rupture de la barrière hémato-encéphalique.

4.6 Discussion et conclusion

Nous avons créé un modèle, basé sur ProtoPNet, qui assure l'explicabilité de son processus de raisonnement. Nous l'avons adapté à un problème d'imagerie cellulaire puisqu'il extrait des structures typiques à partir des images des populations,



(a) Cellules non-traitées (b) Cellules traitées au TNF-α

FIGURE 4.9 – Mosaïques combinant la visualisation d'un prototype et les 20 patches les plus proches dans la base d'entraînement. (a) montre le résultat pour un prototype de la classe "non-traitée" et (b) montre le résultat pour un prototype de la classe "traitée au TNF-α".

en mettant à profit les performances en classification de l'apprentissage profond, malgré un cadre d'apprentissage faiblement supervisé. Le modèle fournit des éléments visuels qui caractérisent les structures intra-cellulaires avec des textures locales, qui peuvent ensuite être analysées par un expert dans le domaine de la biologie. Cela permet d'apprendre les biomarqueurs de la dégradation associée à un produit donné, dans notre cas le TNF-α.

Plusieurs points restent à améliorer. En premier lieu, l'étape de segmentation est source d'erreurs. En plus de ne pas forcément découper correctement les cellules, leur représentation de façon individuelle, alors qu'elles appartiennent à un tapis, crée artificiellement des bords de cellules dont la géométrie ne reflète pas toujours la réalité. On pourrait améliorer la qualité de la segmentation en ré-entraînant le modèle de Cellpose avec quelques exemples que nous aurions segmentés à la main, mais cela pose deux problèmes. Premièrement cela nuit au côté automatisé de l'analyse puisqu'il faut qu'un expert intervienne. Deuxièmement nous aimerions travailler sans connaissances a priori, or on peut considérer que savoir segmenter les cellules est un forme de connaissance. Il faut donc trouver une autre manière d'améliorer la segmentation. Une idée simple est de ne plus fonctionner avec des cellules uniques sur les images mais de garder des images de la couche cellulaire et de ne segmenter que ce qui appartient à l'arrière plan. Le masque de segmentation binaire alors obtenu ne distinguerait que les zones représentant des cellules, indifféremment du nombre de cellules sur l'image, des zones de fond où aucune information pertinente biologiquement ne peut être extraite. C'est une approche qui est souvent utilisée lorsque l'on étudie des tissus [24].

Une autre piste d'amélioration est une meilleure utilisation du canal bleu des noyaux. Pour l'instant nous ne l'utilisons que pour effectuer notre segmentation, mais nous savons que les noyaux peuvent contenir des informations pertinentes pour détecter les biomarqueurs de certaines atteintes [47].

Enfin il existe des améliorations liées à notre architecture et à ProtoPNet lui même (par exemple, l'effet du nombre de prototypes appris n'a pas été étudié en profondeur), mais aussi des choix liés au fonctionnement intrinsèque du modèle. Les prototypes sont pour l'instant liés à une classe par les poids utilisés dans la couche de classification. Mais on pourrait imaginer avoir des prototypes qui soient partagés, puisque les structures typiques des cellules peuvent se retrouver aussi chez certaines cellules labellisées "Exposée". ProtoPShare [102] adopte cette approche en fusionnant les prototypes issus de classes différentes mais qui représentent la même information visuelle. Dans un autre genre, ProtoPool [101] crée un "pot commun" de prototypes et chaque classe possède des emplacements qui sont remplis avec des prototypes du pot commun afin de classifier les images. Un autre modèle dérivé de ProtoPNet, ProtoTree [83] permet également de raisonner de manière négative sur les prototypes, c'est-à-dire de classifier par rapport à l'absence d'un prototype. Si cela peut paraître comme un raisonnement intuitif, le problème posé est que l'absence d'un prototype ne renseigne pas vraiment sur le contenu de l'image. Aussi, malgré les poids figés dans la couche de classification, la diversité obtenue parmi les prototypes reste assez faible. Certains se ressemblent fortement et rien n'empêche vraiment tous les prototypes d'une classe d'être similaires. On peut penser que dans les problèmes multi-classes étudiés par Chen et al. certaines classes sont très proches et que pour améliorer les performances de classification le modèle apprend naturellement des prototypes variés. Mais notre problème est une classification binaire et un seul prototype par classe pourrait théoriquement suffire. Or nous voulons essayer de déterminer toutes les structures qui existent au sein des cellules, pas seulement une.

L'entraînement par séquences du modèle est aussi un problème. En effet, la deuxième étape de l'entraînement qui consiste à projeter les prototypes (qui sont à ce stade des points abstraits dans l'espace latent du modèle) vers les patches latents issus d'images d'entrée les plus proches, réduit nécessairement la précision de la classification puisque les points abstraits avaient été optimisés pour la classification. Même si Chen et al. montrent que cette diminution de précision est bornée, elle appelle quand même à une troisième étape d'entraînement pour compenser cette perte, ce qui demande du temps supplémentaire et pose la question d'une méthode qui n'aurait pas besoin de cette projection.

Mais toutes ces approches dérivées de ProtoPNet ne résolvent pas un des problèmes majeurs de la méthode : la visualisation des prototypes. La solution uti-

lisée, le sur-échantillonnage des cartes d'activation, peut ne pas représenter les patches réellement importants pour la classification [26, 41]. Notre méthode étant assez similaire il est probable qu'elle aussi tombe dans cet écueil. Cela appelle donc à visualiser les prototypes d'une autre manière. Une solution naturelle serait d'utiliser une architecture auto-encodeur. C'est cette approche qu'ont utilisée Li et al [69] dans une publication qui pré-date ProtoPNet et qui a servi de base à son développement. Gautam et al. vont un peu plus loin et proposent ProtoVAE [40], qui utilise un auto-encodeur variationnel afin de garder une meilleure cohérence entre l'espace visuel initial et l'espace latent du réseau. ProtoVAE est l'inspiration de la nouvelle itération de notre modèle que nous allons détailler dans le chapitre suivant.

5

ProtoGMVAE : Un VAE avec un a priori mélange de Gaussiennes pour l'explicabilité par les prototypes

Outline

5.1 Introduction 66
5.2 Auto-Encodeur Variationnel 69
5.3 ProtoVAE 71
5.4 ProtoGMVAE 74
5.4.1 Motivations 74
5.4.2 Du VAE au GMVAE 75
5.4.3 Architecture 77
5.5 Résultats Expérimentaux 79
5.5.1 Jeux de données et implémentation 79
5.5.2 Comparaison des performances en classification 79
5.5.3 Espace latent et visualisation des prototypes 80
5.5.4 MNIST avec classes déséquilibrées 83
5.5.5 Nombre de composantes 85
5.6 Évaluation des prototypes 86
5.7 Conclusion 91

5.1 Introduction

Dans le chapitre précédent, nous avons montré que ProtoPNet permettait d'obtenir des prototypes qui décrivent bien certaines structures présentes dans les cellules, mais souffre de plusieurs problèmes. D'abord, il y a un problème de diversité dans les prototypes appris. Souvent, plusieurs prototypes sont visuellement identiques et comme le nombre de prototypes est fixé avant l'apprentissage, on "perd" un prototype qui aurait pu servir à décrire autre chose. L'autre problème de ProtoPNet est lié à la visualisation des prototypes. En effet, avant l'étape de projection des prototypes pendant l'entraînement, les prototypes ne sont que des points abstraits dans l'espace latent. Dans ce chapitre, nous proposons d'utiliser une architecture en auto-encodeur et plus particulièrement en auto-encodeur variationnel comme cela est fait dans ProtoVAE [40].

ProtoVAE permet effectivement de régler le problème de la visualisation des prototypes en épargnant une étape de projection qui fait baisser les performances du modèle. Mais l'utilisation de ProtoVAE soulève d'autres questions, au premier rang desquelles le lien entre les prototypes appris et la structure des données du problème. Par exemple, les prototypes ont tendance à se regrouper dans de petites régions de l'espace latent et échouent à représenter toute la diversité de motifs qui existent. Gautam et al.[40] ont donc proposé une nouvelle contrainte pour la fonction de perte pour rendre les prototypes orthogonaux entre eux dans l'espace latent, afin d'améliorer leur diversité. Dans une autre publication s'inspirant de ProtoVAE, Kjaersgaard et al. [61] proposent une autre contrainte pour maximiser la couverture par les prototypes du volume occupé par les images d'une classe, toujours dans le but d'augmenter la diversité.

La distribution des données dans l'espace latent peut parfois être très complexe et difficile à modéliser avec un VAE simple. Nous proposons dans cette partie d'utiliser un VAE ayant comme a priori un mélange de gaussiennes. Le mélange pourra décrire la distribution des données dans l'espace latent et il sera possible de définir les prototypes comme le centre de chaque composante de ce mélange. De cette manière, les prototypes deviennent des points clés dans l'espace latent qui aident à expliquer la distribution des données. De plus, l'entraînement de bout en bout d'un VAE rend possible l'utilisation du décodeur pour visualiser les prototypes, sans avoir besoin de l'étape de visualisation.

Dans ce chapitre nous allons étudier la pertinence des prototypes appris, ainsi que leur diversité, et tenter de montrer que ces deux points sont liés à la manière dont ceux-ci modélisent la distribution des données d'apprentissage. Après avoir étudié le fonctionnement de ProtoVAE nous proposerons donc de modéliser cette distribution avec un mélange de gaussiennes, et d'associer un prototype à chacun des

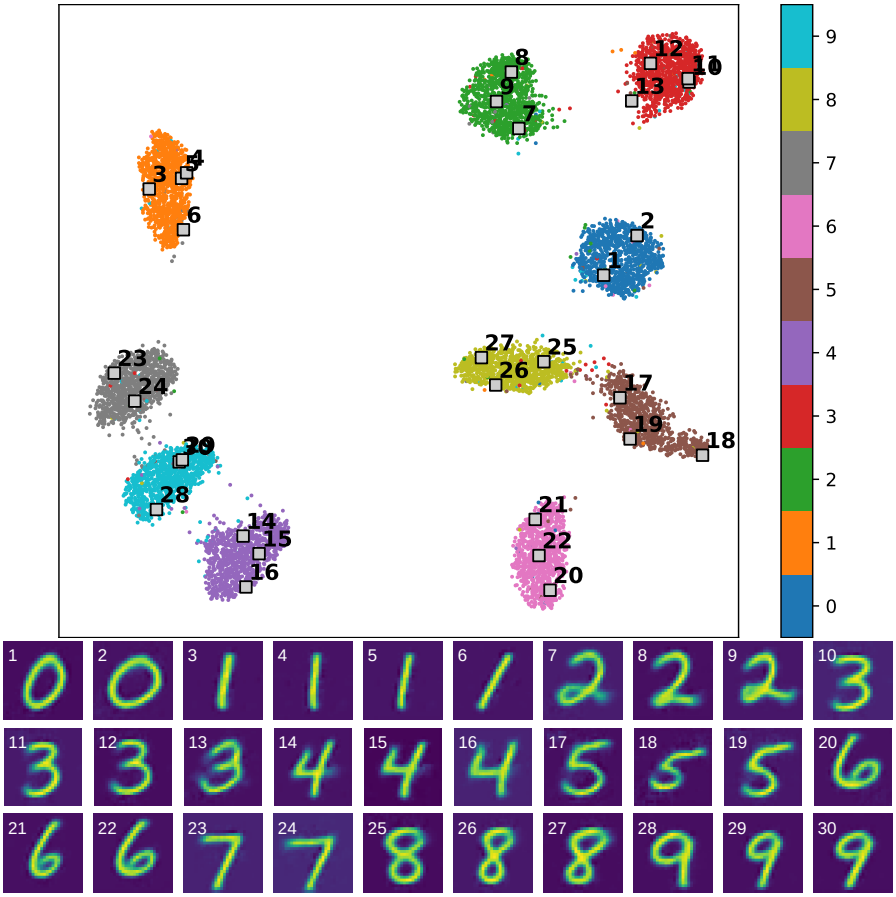


FIGURE 5.1 – Distribution latente et prototypes appris avec ProtoGMVAE sur la base de données MNIST [67]

centres des composantes de ce mélange. En parallèle, nous utiliserons la position relative de chacun des échantillons d'entrée par rapport aux prototypes pour prédire sa classe. Ainsi nous renforcerons leur pouvoir discriminant pour la tâche de classification. Avec ces prototypes notre but est à la fois d'expliquer la distribution des données dans l'espace latent et de fournir une classification interprétable des données.

Il est nécessaire de clarifier la définition de prototype que nous utiliserons ici. Contrairement à ceux obtenus avec ProtoPNet, les prototypes de ProtoVAE ou de ProtoGMVAE ne correspondent pas à un échantillon d'entraînement, puisque le décodeur du réseau permet de reconstruire une image à partir de n'importe quel point dans l'espace latent. Ensuite, dans ProtoGMVAE, nos prototypes n'appartiendront pas à une classe précise. Une affectation pourra être effectuée après l'entraînement dans le but de comprendre comment chacun des prototypes pèse

dans la classification mais aucune restriction sur une classe particulière ne leur est appliquée. L'intuition derrière ce changement est que dans un problème, plusieurs classes peuvent partager des caractéristiques. Si on reprend le jeu de données Caltech-UCSD Birds-200-2011 [118] contenant des images d'oiseaux et initialement utilisé par ProtoPNet, plusieurs espèces d'oiseaux ont des traits communs. Même si ces traits ne permettent pas d'isoler une espèce précise, ils permettent néanmoins de réduire les possibilités et informent sur la morphologie de l'oiseau. Enfin, une troisième différence avec le chapitre précédent est qu'ici nos prototypes sont des images entières et non plus seulement une partie d'une image. Le décodeur étant entraîné à reconstruire correctement l'image d'entrée, il ne peut reconstruire que des images de taille fixe.

La figure 5.1 illustre la distribution latente des images de la base de données MNIST [67] composée d'images de chiffres manuscrits et la position des 30 prototypes appris avec ProtoGMVAE. Ces derniers sont représentés en gris puisqu'ils n'appartiennent à aucune classe précise. Cependant on voit bien que le modèle a spontanément appris des exemples de toutes les classes et que ceux-ci couvrent des cas divers, par exemple les prototypes numéro 7 et 8 montrent deux manières distinctes de tracer un 2 avec une boucle ou non. Aussi, les prototypes couvrent une large partie des nuages de points de chacune des classes.

L'adoption d'un mélange de gaussiennes comme a priori pour l'espace latent d'un auto-encodeur variationnel a déjà été proposée dans différentes publications [32, 106] qui utilisent le terme de GMVAE pour Gaussian Mixture Variational Autoencoder. Ces modèles sont utilisés pour des tâches diverses comme le clustering [54, 42, 115] ou la classification [38], mais à notre connaissance, il existe peu de tentatives d'inscrire ces approches GMVAE dans le cadre de l'intelligence artificielle explicable. D'autre part, le détail des calculs théoriques dans ce cadre n'est pas toujours clair et certaines de ces solutions ne sont pas basées sur de vraies mixtures de gaussiennes comme Rui Shu le montre [106]. Dans la section 5.4.2 nous allons essayer de détailler clairement les hypothèses et ce qui en découle pour déduire la fonction de perte et ses différents termes.

Dans la suite de ce chapitre nous allons donc commencer par revenir sur le fonctionnement d'un VAE, puis nous étudions ProtoVAE, ses motivations et ses limitations avant d'introduire notre nouveau modèle ProtoGMVAE. Nous montrons à travers de nombreuses expérimentations son fonctionnement et nous le comparons à ProtoVAE pour mettre en évidence les avantages que notre méthode offre.

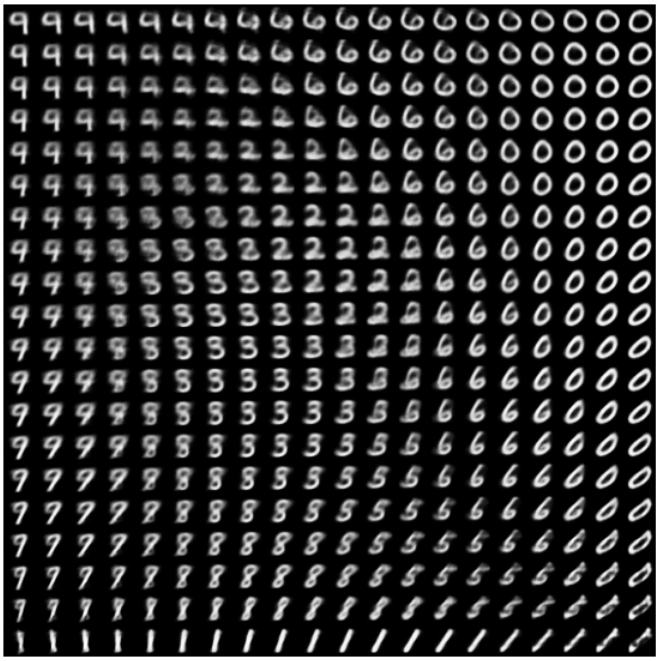


FIGURE 5.2 – Visualisation en 2D de la variété apprise par un VAE entraîné sur MNIST, avec reconstruction des images associées aux points. L'information entre deux points est continue.

5.2 Auto-Encodeur Variationnel

Dans le chapitre précédent, notre modèle basé sur ProtoPNet comportait un encodeur dont le rôle était de transformer les images en une représentation plus compacte au niveau des dimensions spatiales mais en augmentant le nombre de canaux. Un auto-encodeur combine un encodeur comme celui-là et un décodeur, dont le rôle est l'inverse, c'est-à-dire qu'il doit permettre de reconstruire l'image initiale à partir de sa représentation latente. L'inconvénient de l'auto-encodeur classique est que cet espace latent peut ne pas être continu ou très difficile à interpréter. Avec un VAE en revanche, l'espace latent appris est continu et régularisé ce qui permet de facilement interpoler des informations entre deux points (Figure 5.2).

L'architecture auto-encodeur variationnel a été proposée en 2013 par Kingma et Welling [60]. Initialement, elle a été mise au point afin de résoudre la tâche d'inférence variationnelle, c'est-à-dire de trouver la distribution postérieure $p(z|x)$ sur les variables latentes z associées à un ensemble de données $X = \{x_i\}_{i=1}^{N}$, contenant N échantillons i.i.d. d'une variable x qui peut être continue ou discrète, dans le

but de représenter la distribution latente de l'ensemble de données. On fait l'hypothèse que les données sont générées par un processus aléatoire qui implique une variable aléatoire continue et non observée z. Le processus est découpé en deux étapes : d'abord une valeur $z^{(i)}$ est générée à partir d'une distribution a priori $p_\theta(z)$; ensuite une valeur $x^{(i)}$ est tirée d'une distribution conditionnelle $p_\theta(x|z)$. Dans ce scénario on ne connaît ni les paramètres θ de ce processus, ni les valeurs des variables latentes $z^{(i)}$. Avec le théorème de Bayes, on peut écrire le problème :

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \tag{5.1}$$

où $p_\theta(z|x)$ est la distribution postérieure que l'on cherche à obtenir, $p_\theta(x|z)$ la vraisemblance, $p_\theta(z)$ la probabilité a priori et $p_\theta(x)$ est la vraisemblance marginale ou preuve.

Pour résoudre l'équation 5.1, il faudrait calculer la vraisemblance marginale $p_\theta(x)$ mais ce calcul est très complexe voire insoluble dans la plupart des cas. Pour résoudre ce problème, un modèle de reconnaissance $q_\phi(z|x)$, paramétrisé par un ensemble de paramètres ϕ, est introduit pour servir d'approximation de $p_\theta(z|x)$. La nouvelle tâche est donc de rendre le plus proches possible ces deux distributions. Pour cela, on utilise la divergence de Kullback-Leibler (aussi appelée KL-divergence) comme fonction objectif à minimiser :

$$\begin{aligned} D_{KL}[q_\phi(z|x)||p_\theta(z|x)] &= \int_z q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z|x)} dz \\ &= \mathbb{E}_{q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}] \end{aligned} \tag{5.2}$$

En remplaçant $p_\theta(z|x)$ par sa définition donnée dans l'équation 5.1, on peut réécrire l'équation précédente (5.2) :

$$\begin{aligned} D_{KL}[q_\phi(z|x)||p_\theta(z|x)] =& \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(x|z) \\ & - \log p_\theta(z)] + \log p_\theta(x) \end{aligned} \tag{5.3}$$

En réécrivant de cette manière, on fait à nouveau apparaître la vraisemblance marginale $p_\theta(x)$ dont le calcul est insoluble, mais il est toujours possible de minimiser la KL-divergence en minimisant le terme d'espérance. Ce terme peut lui aussi être minimisé au travers d'une autre fonction, la ELBO (Evidence Lower Bound), qui s'exprime ainsi :

$$\begin{aligned} ELBO &= -\mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(x|z) - \log p_\theta(z)] \\ &= -D_{KL}[q_\phi(z|x)||p_\theta(z)] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \end{aligned} \tag{5.4}$$

La ELBO étant l'opposé du terme d'espérance dans l'équation 5.3, on va chercher à maximiser celle-ci pour minimiser l'espérance.

On peut aussi comprendre ce problème en passant par la maximisation de la vraisemblance marginale. La vraisemblance marginale dans l'équation 5.1 peut être vue comme la probabilité que l'échantillon x ait été généré par le modèle, indépendamment des variables latentes z utilisées. On souhaite donc que cette valeur soit la plus élevée possible. En pratique, on cherche plutôt à maximiser la log-vraisemblance marginale, puisque les propriétés du logarithme font que maximiser la log-vraisemblance revient à maximiser la vraisemblance, et que pour des distributions exponentielles comme la loi normale, la log-vraisemblance est plus facile à optimiser. On va donc chercher à maximiser la log-vraisemblance marginale :

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + ELBO(\theta, \phi; x) \tag{5.5}$$

La KL-divergence étant une fonction positive, la log-vraisemblance est forcément supérieure ou égale à la ELBO, d'où son nom (en français : limite inférieure de la preuve) puisqu'elle fournit une borne inférieure à la vraisemblance marginale, aussi appelée preuve. Pour rendre notre modèle le plus vraisemblable possible, on va donc essayer de maximiser cette borne inférieure.

L'entraînement d'un VAE se fait en utilisant $-$ELBO comme fonction de perte (on cherche à minimiser cette fonction , d'où le signe $-$ pour maximiser la ELBO). Dans les formulations précédentes, $q_\phi(z|x)$ correspond à l'encodeur de notre réseau de neurones et $p_\theta(x|z)$ correspond au décodeur. Le premier terme de la ELBO en $-D_{KL}$ pousse $q_\phi(z|x)$ à être proche de l'a priori $p_\theta(z)$ choisi. La forme de cet a priori est libre mais celle la plus communément utilisée est la gaussienne $z \sim \mathcal{N}(\mu, \sigma^2)$, au point que le terme VAE sous-entend souvent l'utilisation d'un a priori gaussien et que l'on précise si c'est un autre a priori qui est employé. Le second terme quant à lui est une erreur de reconstruction négative qui pousse le décodeur à reconstruire correctement l'image x à partir des variables latentes z associées.

5.3 ProtoVAE

Notre approche étant directement inspirée par ProtoVAE [40], nous allons détailler les motivations des auteurs de ce modèle ainsi que son fonctionnement.

Comme nous l'avons déjà dit précédemment, ProtoVAE cherche à régler le problème de ProtoPNet lié à la visualisation des prototypes en utilisant l'architecture VAE [60] que nous venons de détailler. Ce cadre est intéressant puisqu'il permet

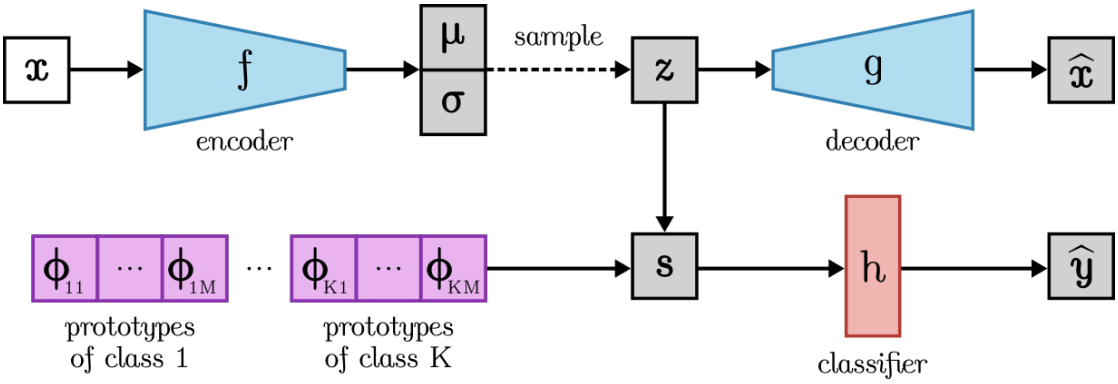


FIGURE 5.3 – Architecture de ProtoVAE [40]

de retourner naturellement dans l'espace visuel d'entrée, sans l'approximation de l'étape de projection.

Le fonctionnement de ProtoVAE est résumé dans le schéma de l'architecture de la figure 5.3. Une image x est donnée en entrée du réseau et passe d'abord par un encodeur f qui transforme cette image en une distribution gaussienne de moyenne μ et d'écart-type σ. De cette distribution est tirée une réalisation z. La valeur z est donnée au décodeur qui va reconstruire une image $\hat{x}$ qui doit être la plus proche possible de x suite à l'entraînement. Parallèlement à la reconstruction, z est aussi comparé à chacun des j-èmes prototypes de la classe k, Φ_{kj} pour obtenir un vecteur de similarité s. C'est sur la base de cette mesure de similarité, donnée en entrée du classifieur h, que la prédiction $\hat{y}$ sur la classe à laquelle appartient l'image x est faite. Tous les prototypes Φ_{kj} peuvent aussi être reconstruits via le décodeur g afin de visualiser ce qu'ils représentent.

Les auteurs de ProtoVAE proposent d'associer une gaussienne avec chacun des prototypes en utilisant "*un mélange de VAEs partageant le même réseau de neurones chacun avec un a priori gaussien centré sur l'un des prototypes*". Par conséquent, pour obtenir leur fonction de perte, ils multiplient le terme de KL-divergence de la ELBO 5.4 par une mesure de similarité (la même qu'utilisée dans ProtoPNet 4.1) dans l'espace latent entre l'échantillon courant et les prototypes de la bonne classe, normalisée par la somme de ces similarités :

$$\sum_{k=1}^{C}\sum_{j=1}^{M} y_i(k)\frac{s_i(k,j)}{\sum_{l=1}^{M} s_i(k,l)} D_{KL}[\mathcal{N}(\mu_i,\sigma_i)||\mathcal{N}(\Phi_{kj},I_d)], \tag{5.6}$$

où C est le nombre de classes, M le nombre de prototypes par classe, y_i la classe de l'échantillon courant, s_i la mesure de similarité entre l'échantillon courant et un prototype donné, et Φ_{kj} le j-ème prototype de la classe k.

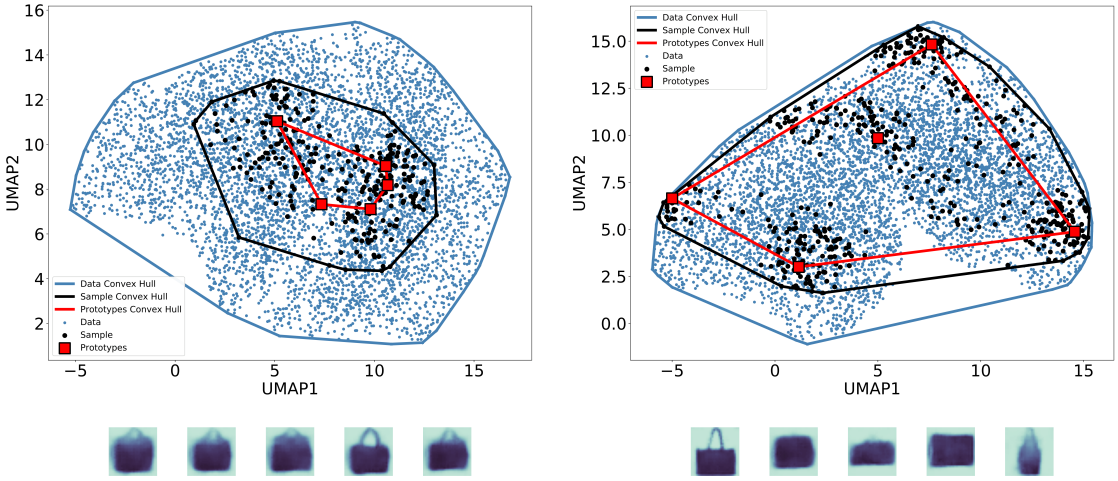
Il nous semble que le but de cette fonction de perte est de donner plus de poids aux prototypes qui sont les plus proches de l'échantillon d'entraînement. Mais quand l'échantillon ressemble au prototype, la similarité s_i est élevée et la KL-divergence est faible. La multiplication de ces deux termes pourrait avoir un effet contradictoire néfaste pour l'apprentissage. De plus, la mesure de similarité utilisée est la même que ProtoPNet. Chen et al. n'ont pas de motivations précises quant au choix de cette fonction et Gautam et al. n'ont pas non plus étudié son rôle. D'autres travaux s'inspirant de ProtoPNet ou ProtoVAE utilisent à la place la similarité cosinus [34, 119, 120] et soulèvent une interrogation quant à la définition de la mesure de similarité à utiliser.

Un autre point important à discuter est la diversité des prototypes. ProtoVAE définit la diversité d'un modèle auto-explicable [4] comme la propriété de ne pas contenir d'information redondante dans les concepts (prototypes) de l'espace latent. Gautam et al. se reposent sur la classification pour assurer la diversité inter-classes, et sur l'orthonormalisation des vecteurs de prototypes d'une même classe dans l'espace latent pour la diversité intra-classe. Le problème de cette régularisation est qu'elle ne parvient pas à couvrir la vraie diversité d'une classe, comme le montrent Kjaersgaard et al[61]. Ils introduisent donc un nouveau modèle PanVAE qui utilise une nouvelle régularisation basée sur une perte volumétrique :

$$\mathcal{L}_{vol} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{|G_k|^{\frac{1}{2}}} \tag{5.7}$$

avec G_k est la matrice de Gram donnée par $G_k = \Phi_k^T \Phi_k$ c'est-à-dire le carré de la matrice des prototypes de chaque classe avec les vecteurs de prototypes Φ_{kj} comme colonnes. $|G_k|^{\frac{1}{2}}$ mesure le volume du parallélotope formé par les vecteurs colonnes des prototypes dans l'espace latent.

Cette nouvelle régularisation pousse les prototypes à couvrir une plus grande partie du volume occupé par les échantillons de la classe dans l'espace latent. La figure 5.4 montre bien que l'enveloppe convexe formée par les prototypes couvre une plus grande partie de l'enveloppe convexe totale de la classe. Avec la reconstruction des prototypes en parallèle, on voit que cela se traduit aussi par une plus grande diversité visuelle des prototypes. Cependant en voyant l'emplacement des vecteurs de prototypes dans l'espace latent, on peut se poser la question de si cette nouvelle régularisation ne va pas pousser les prototypes sur les bords de la classe, poussant ainsi ces derniers à représenter des cas marginaux qui ne sont pas les plus représentatifs de la classe. On voit déjà que le prototype le plus à droite dans les reconstructions de (b) n'est pas très net et diffère assez des autres.



(a) (b)

FIGURE 5.4 – Couverture de l'espace latent par les prototypes après 20 époques d'entraînement de (a) ProtoVAE et (b) PanVAE sur la base de données FashionMNIST [122] avec 5 prototypes pour la classe "sac". En bas : Reconstruction des prototypes. Figure tirée de [61]

5.4 ProtoGMVAE

5.4.1 Motivations

Maintenant que les limitations de ProtoVAE ont été détaillées dans la section précédente, nous proposons ProtoGMVAE, une nouvelle méthode qui utilise :

1. Une architecture **VAE** pour une visualisation fidèle des concepts appris par notre modèle
2. Un **Mélange de Gaussiennes** comme a priori pour notre VAE afin d'obtenir plus de flexibilité dans la modélisation de la distribution des données.
3. Des **prototypes qui n'appartiennent à aucune classe spécifique** pour couvrir la vraie diversité des données de tout le jeu d'apprentissage sans contrainte trop restrictive.
4. Des **probabilités** pour classifier, plutôt qu'une mesure de similarité dont le choix et les effets ne sont pas complètement justifiés.

Deux architectures GMVAE concurrentes ont été proposées en 2016 : celle proposée par Dilokthanakul et al. [32] et celle de Jiang et al. [54]. Ces deux architectures étaient utilisées pour du clustering non-supervisé. La nôtre est basée sur la propo-

sition de Rui Shu [106], qui est une réponse au modèle proposé par Dilokthanakul et al., soulignant certaines complexités non nécessaires pour obtenir un GMVAE. En se basant sur les travaux de Kingma et al. [58] qui avaient proposé des modèles pour le clustering semi-supervisé, Shu aboutit à ce qu'il appelle le "vrai GMVAE". Si ces travaux ne font l'objet que d'une note de blog et n'ont pas été publiés formellement, la formulation du "vrai GMVAE" est en fait similaire à celle développée par Jiang et al.

Notre modèle est basé sur celui de Shu, auquel nous avons ajouté un classifieur. ProtoGMVAE est entraîné de bout en bout, afin que les composantes du mélange de gaussiennes puissent à la fois donner une représentation robuste dans l'espace latent et classifier les images précisément.

5.4.2 Du VAE au GMVAE

Pour passer du fonctionnement d'un VAE tel qu'on l'a vu dans la section 5.2 au GMVAE, on ne peut pas simplement remplacer l'expression de l'a priori gaussien $p_\theta(z)$ par un mélange de gaussienne dans la ELBO 5.4. En effet, cela imposerait de calculer la KL-divergence entre deux mélanges de gaussiennes, puisque notre encodeur $q_\phi(z|x)$ serait construit de manière à produire un GMM lui aussi. Or il n'existe pas de solution analytique à ce calcul et on ne peut que chercher une approximation. Pour résoudre ce problème, on peut introduire une seconde variable latente y, qui représente les poids des composantes dans le mélange de gaussiennes associé à l'entrée x. Si on ré-exprime le théorème de Bayes avec cette seconde variable, la nouvelle tâche d'inférence dans ce scénario devient :

$$p_\theta(y, z|x) = \frac{p_\theta(x|y, z)p_\theta(y, z)}{p_\theta(x)} \tag{5.8}$$

Nous allons définir deux a priori. Le premier est sur la variable y que nous allons choisir comme une variable catégorielle sur les K composantes du mélange, avec des probabilités uniformes pour chacune. Ensuite, nous allons prendre un a priori gaussien pour $z|y$.

$$\begin{aligned} y &\sim Cat(1/K) \\ z|y &\sim \mathcal{N}(\hat{\mu}(y), \hat{\sigma}^2(y)) \end{aligned} \tag{5.9}$$

Si on écrit l'expression de z marginalisé par rapport à y à partir des deux a priori précédents, on voit que celle-ci est bien celle d'un GMM, mais l'ajout de la variable y va nous permettre de ne pas avoir à calculer de terme ne dépendant que de z.

$$z \sim \sum_k Cat(1/K)\ \mathcal{N}(\hat{\mu}(y), \hat{\sigma}^2(y)) \tag{5.10}$$

Reprenons à partir de l'équation 5.8 et dérivons nos calculs comme nous l'avions fait pour le VAE dans la section 5.2. A nouveau, $p_\theta(x)$ n'est pas calculable et nous allons introduire un modèle de reconnaissance $q_\phi(y, z|x)$ pour approximer la vraie distribution a posteriori $p_\theta(y, z|x)$. En suivant les mêmes étapes que pour le VAE pour obtenir l'équation 5.2, on peut écrire la KL-divergence entre le modèle de reconnaissance et la distribution postérieure avec nos nouvelles notations :

$$D_{KL}[q_\phi(y, z|x)||p_\theta(y, z|x)] = \mathbb{E}_{q_\phi(y,z|x)}[\log \frac{q_\phi(y, z|x)}{p_\theta(y, z|x)}] \tag{5.11}$$

Là encore, nous pouvons réécrire l'expression de la divergence de Kullback-Leibler en remplaçant l'a posteriori $p_\theta(y, z|x)$ par sa définition de l'équation 5.8.

$$\begin{aligned} D_{KL}[q_\phi(y, z|x)||p_\theta(y, z|x)] =& \mathbb{E}_{q_\phi(y,z|x)}[\log q_\phi(y, z|x) - \log p_\theta(x|y, z) - \log p_\theta(y, z)] \\ &+ \log p_\theta(x) \end{aligned} \tag{5.12}$$

A partir de cette équation on peut écrire la nouvelle ELBO, qui correspond à l'opposé du terme d'espérance. Puis on applique la règle de Bayes sur $q_\phi(y, z|x)$ et $p_\theta(y, z)$ pour simplifier l'expression.

$$ELBO_{GMVAE} = -\mathbb{E}_{q_\phi(y,z|x)}[\log q_\phi(y, z|x) - \log p_\theta(x|y, z) - \log p_\theta(y, z)] \tag{5.13a}$$

$$= \mathbb{E}_{q_\phi(y,z|x)}[\log \frac{p_\theta(y)}{q_\phi(y|x)} + \log \frac{p_\theta(z|y)}{q_\phi(z|x, y)} + \log p_\theta(x|y, z))] \tag{5.13b}$$

$$= -D_{KL}[q_\phi(y|x)||p_\theta(y)] - D_{KL}[q_\phi(z|x, y)||p_\theta(z|y)] + \mathbb{E}_{q_\phi(y,z|x)}[\log p_\theta(x|y, z)] \tag{5.13c}$$

Pour plus de clarté, réécrivons la perte, qui est égale à $-ELBO_{GMVAE}$:

$$\mathcal{L}_{GMVAE} = \underbrace{D_{KL}[q_\phi(y|x)||p_\theta(y)]}_{\text{bleu}} + \underbrace{D_{KL}[q_\phi(z|x, y)||p_\theta(z|y)]}_{\text{rouge}} - \underbrace{\mathbb{E}_{q_\phi(y,z|x)}[\log p_\theta(x|y, z)]}_{\text{vert}} \tag{5.14}$$

La partie bleue est donc une KL-divergence qui va pousser la variable y prédite par l'encodeur à suivre l'a priori uniforme de $p_\theta(y)$. Similairement, la partie rouge va pousser la variable $z|y$ à suivre l'a priori gaussien de $p_\theta(z|y)$. Enfin, la partie verte est une erreur de reconstruction négative entre la sortie $\hat{x}$ et l'entrée x. Dans cette expression, on voit qu'aucun terme ne dépend que de la variable latente z. L'introduction de la seconde variable y a permis d'exprimer la ELBO sous une forme où chaque élément peut être calculé, rendant l'entraînement et l'inférence

bien plus simples. Mais l'introduction de cette seconde variable a un impact sur l'architecture du réseau qui n'est plus tout à fait la même que celle proposée dans ProtoVAE, et que nous allons décrire dans la section suivante.

5.4.3 Architecture

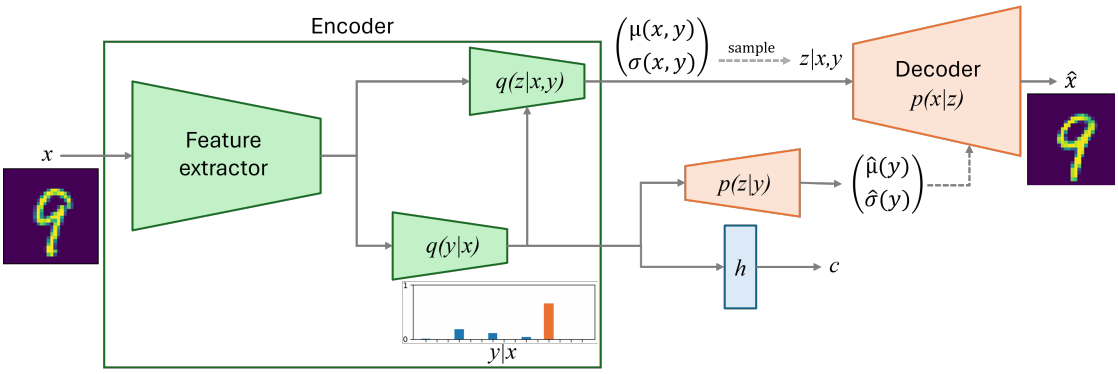


FIGURE 5.5 – Schéma de l'architecture proposée

La figure 5.5 montre une vue d'ensemble de notre architecture. Nous envoyons une image x en entrée de l'encodeur q paramétré par un ensemble de paramètres apprenables ϕ. Cet encodeur est constitué d'un extracteur de features partagé par deux branches : $q_\phi(y|x)$ qui prédit un vecteur K-dimensionnel, contenant les coefficients du mélange de gaussiennes associé à l'image d'entrée ; et $q_\phi(z|x,y)$ qui prédit la distribution latente $\mathcal{N}(\mu(x,y), \sigma^2(x,y))$ étant donné l'image et la composante du mélange. μ et σ sont deux fonctions apprises par $q_\phi(z|x,y)$ qui visent à prédire la moyenne et l'écart type de la distribution latente. Ensuite, la distribution latente $y|x$ est utilisée pour la classification par une couche linéaire non-biaisée h pour produire des logits sur l'ensemble de toutes les classes $\mathcal{C}$. Un échantillon $z|x,y$ est tiré de la distribution apprise par $q_\phi(z|x,y)$ et est passé au décodeur pour la reconstruction.

Le décodeur p, lui aussi paramétré par un ensemble de paramètres apprenables θ, est divisé en deux parties. $p_\theta(x|z)$ reconstruit une image étant donné l'échantillon latent $z|x,y$. La deuxième partie, $p_\theta(z|y)$ prédit une distribution normale $\mathcal{N}(\hat{\mu}(y), \hat{\sigma}^2(y))$ uniquement à partir du vecteur latent $y|x$. Comme pour l'encodeur, $\hat{\mu}$ et $\hat{\sigma}$ sont des fonctions apprises par $p_\theta(z|y)$. Cette seconde partie du décodeur remplit deux rôles : elle permet de calculer la KL-divergence gaussienne de la perte (partie rouge de l'équation 5.14) ; et elle permet également de visualiser n'importe quel point de l'espace latent sans avoir besoin d'une image d'entrée mais en fournissant une composante y du mélange qui peut être choisie arbitrairement. Cette

utilisation pour la visualisation est celle qui est mise à profit afin de visualiser chacune des composantes $k \in \{k_j\}_{j=1}^{K}$ du mélange, en utilisant chaque $\hat{\mu}_k$, la moyenne prédite par $p_\theta(z|y)$ avec un vecteur encodé au format one-hot y_k comme entrée, en guise de variable $z|x, y$ qui peut ensuite être décodée à travers $p_\theta(x|z)$ pour obtenir une visualisation de la composante. Nous considérons les composantes du mélange comme nos prototypes.

Nous avons décidé de classifier en utilisant seulement la variable latente $y|x$ pour deux raisons. D'abord cela utilise la nature probabiliste de l'architecture pour remplacer la mesure de similarité ou de distance dans l'espace latent, tout en donnant quand même des informations sur quels prototypes ressemblent à l'image d'entrée. Ensuite, l'espace latent $z|x, y$ est utilisé pour la reconstruction. La classification et la reconstruction peuvent être deux tâches antagonistes car deux images similaires peuvent parfois être issues de deux classes différentes. Accomplir ces deux tâches dans le même espace pourrait compliquer l'entraînement. A la place, l'espace latent $y|x$ agit comme un espace dual pour la classification et permet de préserver la structure liée à l'aspect visuel de l'espace $z|x, y$.

Pour entraîner toute l'architecture, y compris la couche de classification, nous utilisons la fonction de perte suivante :

$$\mathcal{L}_{PGMVAE} = \mathcal{L}_{GMVAE} + \mathcal{L}_{pred} \tag{5.15}$$

où $\mathcal{L}_{GMVAE}$ est repris de l'équation 5.14, et $\mathcal{L}_{pred} = \frac{1}{N}\sum_{i=1}^{N} CE(h(y_i), c_i)$ est la perte d'entropie croisée entre la classe prédite par le réseau $h(y_i)$ et la vérité terrain c_i associée à l'échantillon d'entrée x_i. Comme aucune association explicite entre une composante et une classe n'est faite avant l'entraînement, nous utilisons les poids appris par la couche de classification, qui est linéaire et non-biaisée, pour obtenir une indication des prototypes auxquels les éléments d'une classe ressemblent.

Fonctionnellement, $\mathcal{L}_{GMVAE}$ est calculée comme une espérance, comme dans l'équation 5.13b. Pour chaque composante k du mélange et le vecteur encodé au format one-hot correspondant y_k, nous calculons $q_\phi(z|x, y_k)$ et $p_\theta(z|y_k)$, les gaussiennes associées à l'entrée x et y_k, ainsi que $p_\theta(x|y, z)_k$ l'erreur de reconstruction entre l'image d'entrée x et sa reconstruction en utilisant un échantillon tiré de $q_\phi(z|x, y_k)$. Cette erreur est calculée comme une entropie croisée binaire, dans le cas d'une image binaire, et comme une erreur quadratique moyenne dans le cas d'une image à valeurs réelles.

$$\mathcal{L}_{GMVAE} = \sum_{k=1}^{K} q_\phi(y|x)_k \left(\log \frac{q_\phi(z|x, y_k)}{p_\theta(z|y_k)} + \log q_\phi(y|x)_k + \log p_\theta(x|y, z)_k - \log 1/K \right) \tag{5.16}$$

5.5 Résultats Expérimentaux

5.5.1 Jeux de données et implémentation

Nous avons utilisé 4 jeux de données communs pour évaluer ProtoGMVAE : MNIST [67] qui contient des images de chiffres manuscrits en niveaux de gris de taille 28×28 pixels, FashionMNIST [122] dont les images sont dans un format identique à MNIST, mais qui contiennent des images d'articles de mode répartis en 10 classes, CIFAR-10 [64] qui contient des images RGB de 10 classes différentes allant d'animaux domestiques comme sauvages à des véhicules de différents types (voitures, avions, bateaux), en taille 32×32 pixels ; et SVHN [85] dont le format est identique à CIFAR-10 mais dont les images contiennent des chiffres écrits sur des plaques postales.

Pour tous les jeux de données, nous avons utilisé pour l'extracteur de features et le décodeur $p_\theta(x|z)$ des architectures convolutives similaires à celles que ProtoVAE utilise pour s'assurer une comparaison équitable. $q_\phi(z|x,y)$ est composé de 3 couches denses avec des fonctions d'activation ReLU, suivies par une couche dense pour la prédiction de $\mu(x,y)$ et une seconde en parallèle pour $\sigma(x,y)$. $q_\phi(y|x)$ contient deux couches denses suivies par un softmax pour obtenir des probabilités. $p_\theta(z|y)$ est composé d'une couche linéaire pour $\hat{\mu}(y)$ et une autre en parallèle pour $\hat{\sigma}(y)$. Pour les jeux de données CIFAR-10 et SVHN, nous avons utilisé de l'augmentation de données avec les mêmes opérations aléatoires que celles utilisées par ProtoVAE : des retournements horizontaux, des recadrages et des petites variations des paramètres de couleurs. Le code pour répliquer nos expériences est disponible à l'adresse suivante : `https://github.com/martin-blcd/ProtoGMVAE`. Le code est basé sur une implémentation PyTorch [59] du GMVAE défini par Shu [106].

5.5.2 Comparaison des performances en classification

Le tableau 5.1 montre les résultats de classification obtenus avec ProtoVAE [40], PanVAE [61], ProtoPNet [27] et ProtoGMVAE pour comparaison. Nous avons utilisé les paramètres par défaut fournis dans les codes publics pour ProtoVAE et PanVAE. Les chiffres de taux de classification correcte pour ProtoPNet sont repris des expériences de ProtoVAE. Nous notons que toutes les méthodes basées sur des VAE obtiennent des résultats similaires sur les jeux de données testés et de meilleurs résultats que ProtoPNet. Parmi ces méthodes basées sur des VAE, notre approche obtient un taux légèrement inférieur, mais cela semble un compromis acceptable par rapport aux gains en interprétabilité qu'elle offre. Il apparaît que notre solution structure l'espace d'une manière telle que des images visuellement

Dataset	MNIST	FMNIST	SVHN	CIFAR10
ProtoPNet	94.7	85.4	88.6	67.8
ProtoVAE	**99.4**	**91.9**	**92.2**	**84.6**
PanVAE	99.3	91.7	91.8	83.9
ProtoGMVAE	99.1	91.6	90.0	82.0

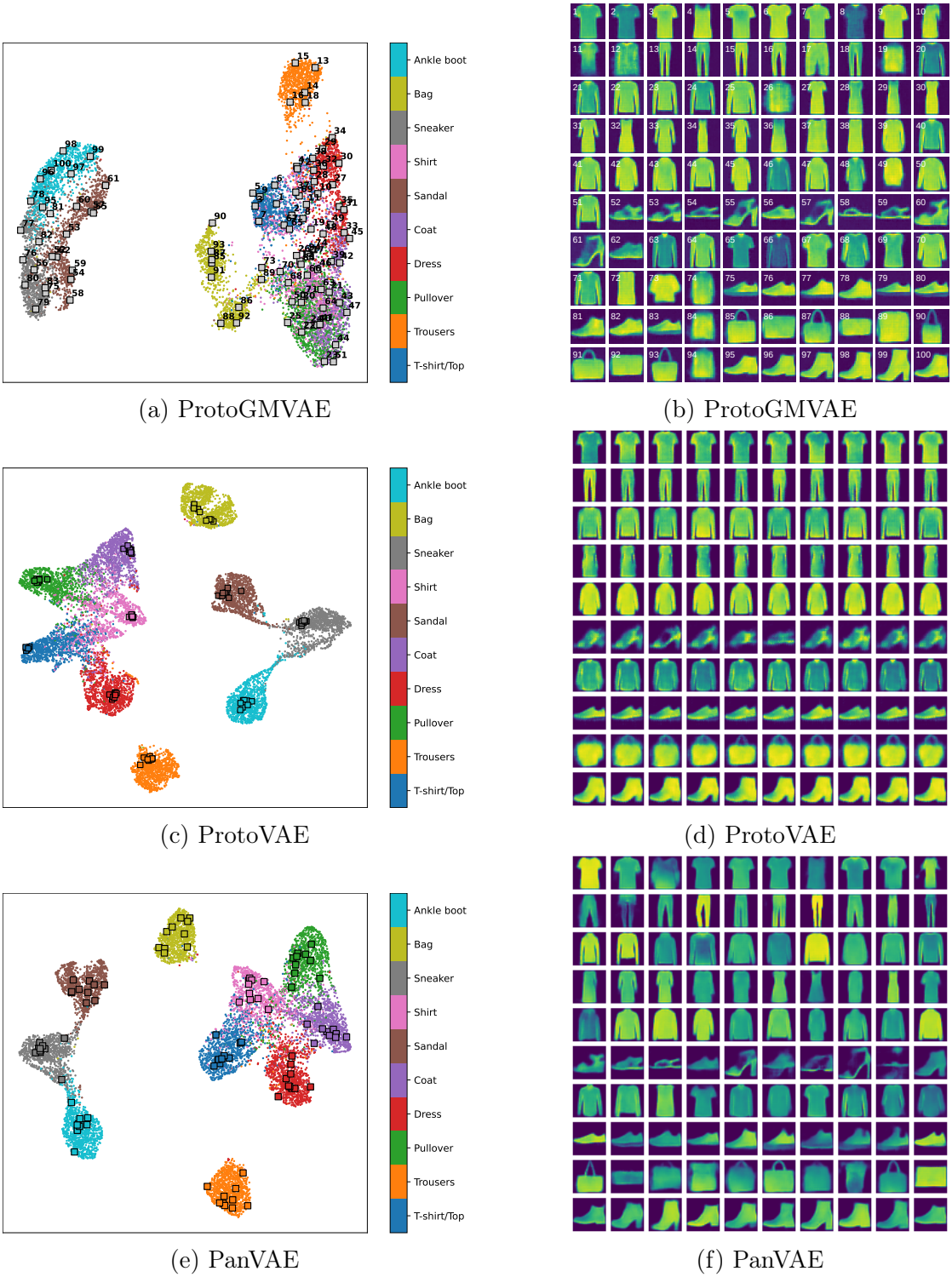
TABLE 5.1 – Comparaison du taux de classification correcte obtenu sur 4 jeux de données par différentes méthodes dont la nôtre. Pour toutes les méthodes, 50 prototypes sont utilisés pour MNIST et SVHN, et 100 pour FashionMNIST et CIFAR10.

similaires sont proches l'une de l'autre. Évidemment, lorsque deux catégories sont similaires, il est plus difficile pour notre approche de pousser les points de l'espace latent correspondant loin les uns des autres. ProtoVAE et PanVAE n'ont pas cette contrainte et sont donc capable de créer des clusters plus discriminants dans l'espace latent. Néanmoins, nous montrerons qu'en créant un espace latent déconnecté de l'apparence visuelle, les reconstructions des prototypes ne sont plus pertinentes et leurs modèles perdent leur capacité d'explicabilité.

5.5.3 Espace latent et visualisation des prototypes

Afin d'évaluer qualitativement la pertinence de nos prototypes, nous comparons tout d'abord les espaces latents appris avec notre méthode avec ceux de ProtoVAE et PanVAE sur les base de données FashionMNIST [122]. Les résultats sont montrés dans la colonne de gauche de la figure 5.6. Nous pouvons voir que 4 clusters principaux apparaissent pour chacune des méthodes : un cluster avec les pantalons, un avec les chaussures, un avec les sacs et un dernier cluster avec le reste des articles. C'est un bon clustering des données par rapport à l'aspect visuel de chacun des groupes. Cependant, on peut remarquer que la répartition des prototypes est très différente selon les méthodes. De façon intéressante, nos prototypes sont dispersés au sein de tous les clusters de chaque classe. Malgré l'utilisation d'un terme d'orthogonalité dans leur perte pour renforcer la diversité intra-classe, les prototypes appris par ProtoVAE sont concentrés dans les mêmes zones tandis que certains prototypes de PanVAE sont plus au bord des clusters des classes.

Cela devient encore plus problématique quand on visualise les reconstructions des prototypes dans la colonne de droite de la figure 5.6. Avec ProtoGMVAE nous obtenons toute la diversité qui existe au sein du jeu de données, avec quasiment tous les prototypes reconstruits clairement. Par exemple, pour la classe "T-shirt/Haut", nous obtenons toute une gamme de tons, du sombre au clair. Le cas le plus parlant



(a) ProtoGMVAE (b) ProtoGMVAE

(c) ProtoVAE (d) ProtoVAE

(e) PanVAE (f) PanVAE

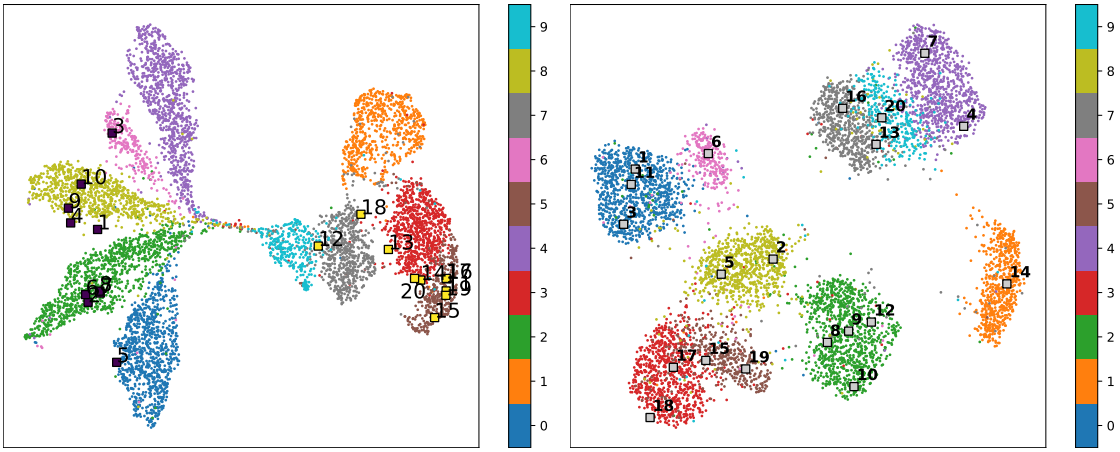
FIGURE 5.6 – Colonne de gauche : Projections UMAP de l'espace latent appris sur FashionMNIST, avec les prototypes superposés comme des carrés. Colonne de droite visualisation des prototypes appris. Pour notre approche, les prototypes sont classés par le poids associé le plus élevé dans la couche de classification et numérotés pour faciliter la comparaison. 100 prototypes sont appris pour chacune des méthodes.

est la classe "Sandale". Elle va des tongs aux sandales à talon, créant ainsi beaucoup de diversité intra-classe. ProtoGMVAE capture des exemples de nombreuses formes de sandales existant au sein de la classe, tandis que ProtoVAE ne reconstruit pas correctement les prototypes, qui ressemblent à une sorte de moyenne de toutes les images de sandales.

PanVAE souffre d'un autre type de problème. Il pousse certains des prototypes vers les bords des clusters, et ces derniers tendent donc à représenter des cas limites. Bien que cela aide à capturer la diversité de la classe, les prototypes peuvent n'être représentatifs que d'une poignée d'images et ne pas être de bons représentants de la classe dans son ensemble. Les auteurs de PanVAE proposent néanmoins un moyen d'éliminer des prototypes comme ceux-là après l'entraînement, mais cela semble être une manière de contourner les effets indésirables du terme de diversité de leur perte plutôt qu'une vraie solution. Cependant, plusieurs de leurs prototypes restent au centre des clusters, probablement car ceux situés au bord maximisent déjà la contrainte de volume. Leur méthode ne semble donc pas vraiment bénéficier de l'utilisation de plus de prototypes pour obtenir une plus grande diversité.

Les espaces latents de la figure 5.6 montrent aussi que ProtoVAE et PanVAE ne savent pas comment gérer des points qui pourraient appartenir à plusieurs classes. Cela est particulièrement visible avec les classes "Pullover", "Manteau" et "Chemise". Certaines images sont laissées loin de tout prototype. Un autre avantage à avoir des prototypes qui ne soient pas spécifiques à une classe précise est d'allouer plus de prototypes pour les clusters où cela est vraiment utile. La classe "Pantalon" montre moins de diversité et est très différente des autres classes visuellement et est donc plus simple à classifier. ProtoGMVAE n'accorde que 6 prototypes à ce cluster, et en laisse plus pour les données plus complexes comme les robes par exemples, qui existent dans de nombreuses formes, couleurs et longueurs.

Comme nous l'avons dit dans la sous-section précédente, l'un des inconvénients majeurs de ProtoGMVAE est sa difficulté à classifier correctement quand les données sont plus complexes et plus entremêlées visuellement, comme pour SVHN et CIFAR-10. Le tableau 5.1 montrait que nous sommes moins précis que les autres méthodes basées sur des VAE. Cette précision moindre est aussi reflétée dans les prototypes que nous sommes capables de reconstruire. Dans la section 5.6, où nous mesurons la qualité de nos prototypes, nous montrons également les reconstructions obtenues par toutes les méthodes. En particulier, la figure 5.12 montre les reconstructions de ProtoGMVAE sur toutes les bases de données testées. On peut voir qu'il est difficile de vraiment distinguer les objets représentés pour le jeu CIFAR-10, ou que la forme des chiffres de SVHN n'est pas toujours claire.



(a) ProtoVAE (b) Notre approche

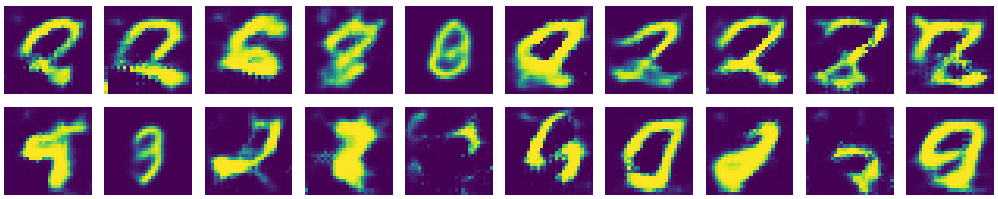
FIGURE 5.7 – Projections UMAP des espaces latents appris avec une classification binaire entre chiffres pairs et impairs en déséquilibrant les classes initiales de MNIST, avec (a) ProtoVAE (prototypes jaunes pour la classe "Impair", bleu foncé pour "Pair"), et (b) notre approche. Les labels détaillés des chiffres sont affichés pour plus de clarté.

5.5.4 MNIST avec classes déséquilibrées

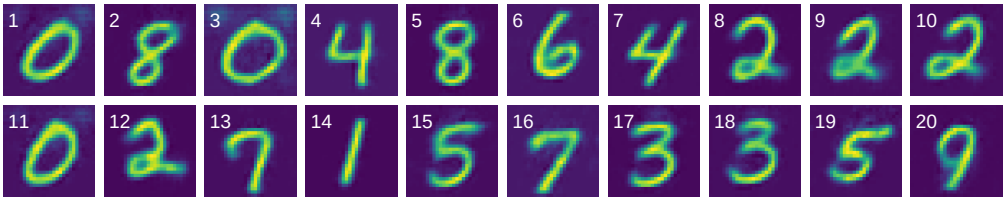
Pour mieux évaluer les performances de notre méthode, nous avons réalisé des expériences avec une classification binaire des chiffres du jeu de données MNIST entre les chiffres pairs et impairs, avec des proportions déséquilibrées aléatoirement de chaque chiffre. Le but de cette configuration est de mettre en évidence la capacité de notre approche à capturer toute la diversité d'une catégorie dans une tâche de classification. En effet, dans ce contexte il y a 5 chiffres par classe avec une distribution déséquilibrée, et nous attendons d'une approche auto-explicable qu'elle donne au moins un prototype par chiffre pour représenter tous les cas qui existent dans une classe. Les architectures utilisées dans ces expériences sont identiques à celles utilisées pour la configuration MNIST normale, détaillée dans la sous-section 5.5.1, avec seulement 20 prototypes au total.

À partir des espaces latents de la figure 5.7 on peut voir que nos prototypes sont répartis dans tous les groupes, tandis que ProtoVAE tend à concentrer ses prototypes aux extrémités des clusters. Ce comportement pourrait être lié à la priorité donnée à la classification par ProtoVAE. Comme les chiffres pairs et impairs sont poussés de chaque côté de l'espace latent, placer les prototypes aux extrémités maximise la distance avec la mauvaise classe et facilite la classification.

Lorsque l'on regarde les prototypes dans la figure 5.8 on peut remarquer un manque



(a) ProtoVAE



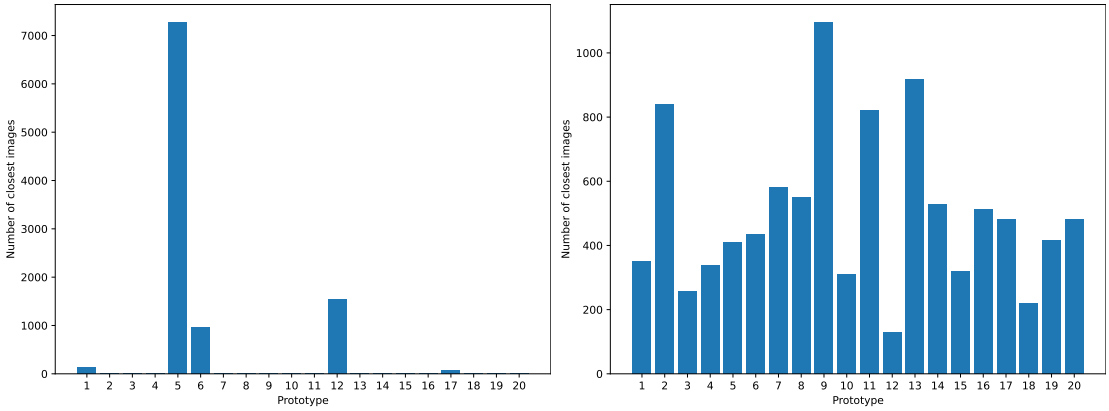
(b) ProtoGMVAE

FIGURE 5.8 – Prototypes obtenus avec une classification binaire entre chiffres pairs et impairs en déséquilibrant les classes initiales de MNIST avec (a) ProtoVAE et (b) ProtoGMVAE. Les prototypes de ProtoVAE sont assez mal reconstruits et représentent des formes abstraites pour la plupart d'entre eux. Les prototypes de ProtoGMVAE capturent toute la diversité des chiffres qui existent dans le jeu de données et les reconstruisent correctement.

de diversité et une mauvaise reconstruction avec ProtoVAE. Notre approche cependant parvient à représenter tous les chiffres sans même utiliser un terme de diversité explicite dans la perte. Un autre élément intéressant est que bien qu'ils soient proches dans l'espace latent, certains prototypes de ProtoVAE sont loins les uns des autres dans l'espace visuel. Par exemple les prototypes numéro #14 et #20 sur la deuxième ligne de la figure 5.8 (a) sont très différents et pourtant lorsque l'on regarde sur la figure 5.7 (a) les deux sont très proches. Cela reflète une déconnexion entre les deux espaces qui peut être néfaste à une classification interprétable.

Comme les projections UMAP peuvent distordre la vraie structure de l'espace latent, nous avons réalisé une autre expérience pour montrer que ProtoVAE n'utilise pas correctement ses prototypes pour décrire les données dans l'espace latent. Pour chaque prototype, nous regardons combien d'images d'entrée sont plus proches de ce prototype que de n'importe quel autre. Pour ProtoVAE nous utilisons simplement la distance entre l'image projetée dans l'espace latent et le prototype. Pour notre approche, nous générons le vecteur y associé à une image et regardons quelle composante porte la probabilité maximum.

La figure 5.9 montre ces distributions pour (a) ProtoVAE et (b) ProtoGMVAE.



(a) ProtoVAE (b) ProtoGMVAE

FIGURE 5.9 – Pour chaque prototype (axe x), nombres d'images de la base de données de test qui sont plus proches de ce prototype que de n'importe quel autre.

Avec ProtoVAE seuls quelques prototypes concentrent toutes les images autour d'eux, tandis que notre méthode distribue les images plus uniformément autour de chaque prototype. Cela veut dire que certains prototypes de ProtoVAE sont situés dans des régions de l'espace latent qui ne ressemblent à aucune image d'entrée du jeu de données.

5.5.5 Nombre de composantes

L'un des paramètres les plus importants dans notre approche est le nombre K de composantes dans le mélange de gaussiennes, qui est le nombre de prototypes. Cela impacte directement la durée d'entraînement car la plupart des calculs sont effectués composante par composante. De plus le nombre de composantes dicte la taille des couches linéaires utilisées dans $q_\phi(z|x,y)$, $q_\phi(y|x)$ et $p_\theta(z|y)$.

Nous avons donc analysé les variations de la performance en classification et en reconstruction en fonction de différentes valeurs de K. Le tableau 5.2 compile ces résultats. On peut voir que l'impact sur la classification est quasi inexistant, tandis que pour la reconstruction, le modèle gagne visiblement en qualité à mesure que le nombre de prototypes augmente. Cela s'explique sans doute par une meilleure couverture des différents cas de figure au sein de la base données offerte par le plus grand nombre de prototypes. La classification n'en bénéficie pas forcément car il est toujours possible de classifier simplement par proximité d'un cluster ou d'un autre, indépendamment du fait que celui-ci soit correctement et entièrement décrit par les prototypes.

Un autre aspect qui n'est pas reflété ici, est l'amélioration de la diversité des

K	20	50	100	150
Précision	91.5	91.3	91.6	91.5
Reconstruction	16.5	15.8	15.7	15.2

TABLE 5.2 – Précision et perte de reconstruction obtenues par notre modèle sur FashionMNIST en fonction du nombre de composantes K dans le mélange de gaussiennes.

prototypes. En effet, en allouant plus de prototypes pour décrire les données, il est logique que plus de versions d'une même classe soit captées par le modèle.

5.6 Évaluation des prototypes

Les précédentes expériences ont mis en avant de manière visuelle et qualitative les avantages des prototypes appris par ProtoGMVAE par rapport aux autres méthodes. Ici nous proposons de mesurer objectivement les différences entre les approches. Pour cela nous utiliserons deux mesures. Tout d'abord, nous mesurons la distance moyenne entre une image d'entrée et le prototype le plus proche. Comme ProtoVAE et PanVAE ont des espaces latents de dimensions différentes au nôtre, et comme notre architecture varie de manière assez importante par rapport à la leur, les espaces pourraient ne pas donner des différences de distances significatives. Pour contourner ce problème, nous mesurons les distances dans l'espace visuel, qui est aussi l'espace le plus intuitif pour l'interprétation des résultats. Nous calculons un écart quadratique moyen entre l'image d'entrée et la reconstruction du prototype prédit par notre modèle avec la plus forte probabilité, ou bien avec la reconstruction du prototype le plus proche dans l'espace latent dans le cas de ProtoVAE et PanVAE. Une faible distance moyenne signifie que chaque échantillon d'entrée a un prototype qui lui ressemble visuellement, et donc que les prototypes couvrent uniformément toute la distribution des données. Les résultats obtenus sont renseignés dans le tableau 5.3. Sur chaque jeu de données, notre méthode obtient une distance plus faible que ProtoVAE et PanVAE. Cela renforce un peu plus l'idée que l'espace latent qu'ils apprennent est déconnecté de l'espace visuel, ce qui gêne l'interprétabilité de leur résultats.

Ensuite, pour évaluer la diversité des prototypes, nous proposons une autre mesure : la variance des prototypes d'une même classe, également évaluée dans l'espace visuel. Intuitivement, une forte variance intra-classe signifie que les prototypes reflètent différentes instances de la classe, ou tout du moins qu'ils capturent les variations qui existent au sein de cette classe. Les mesures de variance obtenues

Dataset	MNIST	FMNIST	SVHN	CIFAR10
ProtoVAE	0.0086	0.0076	0.0031	0.0042
PanVAE	0.0067	0.0071	0.0033	0.0044
ProtoGMVAE	**0.0041**	**0.0023**	**0.0027**	**0.0031**

TABLE 5.3 – Distance moyenne entre une image d'entrée et le prototype le plus proche, mesurée dans l'espace visuel

Dataset	MNIST	FMNIST	SVHN	CIFAR10
ProtoVAE	0.0525	0.0448	0.0006	0.0069
PanVAE	0.0574	0.0659	0.0022*	0.0076
ProtoGMVAE	**0.0751**	**0.0791**	**0.0057**	**0.0110**

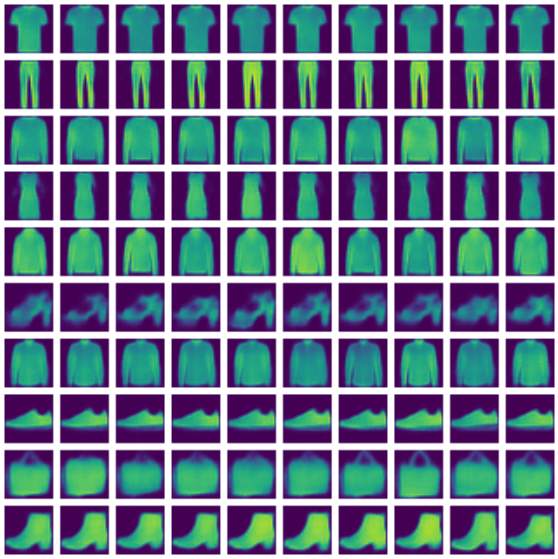
TABLE 5.4 – Moyenne des variances intra-classe des prototypes, mesurée dans l'espace visuel. Les valeurs marquées d'une étoile (*) ont été calculées après avoir retiré les prototypes qui n'étaient pas reconstruits correctement.

sont reportées dans le tableau 5.4.

Encore une fois, ProtoGMVAE obtient de meilleurs résultats que les autres méthodes sur tous les jeux de données. Cela montre que même sans utiliser un terme de diversité explicite, comme un terme d'orthogonalité pour ProtoVAE, ou une perte volumétrique dans le cas de PanVAE, notre méthode arrive tout de même à inclure plus des différentes versions d'un objet d'une classe donnée. Ces résultats confirment ce que nous avions identifié qualitativement dans la figure 5.6. Les prototypes de ProtoVAE sont très similaires les uns aux autres et ne parviennent pas à représenter toutes les données. PanVAE s'en sort mieux et obtient des prototypes plus divers mais ils sont encore assez loin des images d'entrée, du point de vue visuel. Notre méthode combine à la fois des prototypes divers et une modélisation fidèle de la distribution des données.

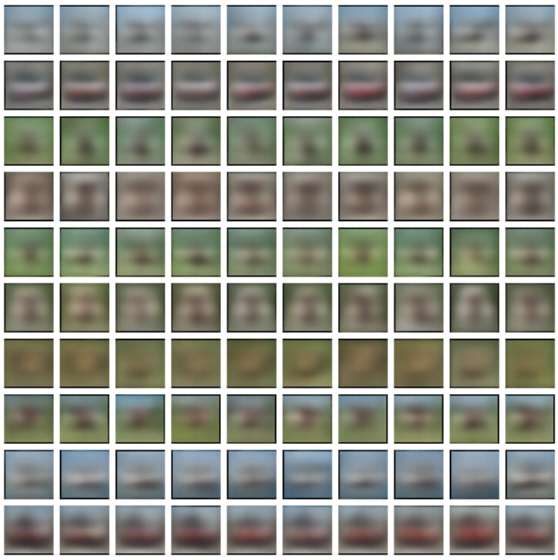
Les résultats doivent être nuancés, particulièrement pour la variance. Dans leur publication, les auteurs expliquent un critère pour détecter et éliminer les prototypes qui sont en dehors de la distribution, un effet secondaire de leur perte volumétrique. Ils enlèvent les prototypes qui ne donnent une similarité maximale pour aucune des images d'entraînement. Mais ils ne donnent aucun détail sur la manière dont ils appliquent cette élimination, et quand ils l'appliquent. Le code qu'ils fournissent n'effectue pas cette élimination par défaut. Cela veut dire que les prototypes qui sont assez mal reconstruits, comme dans la Figure 5.6f par exemple, auraient sans doute été éliminés pendant la phase de tri des prototypes

(a) MNIST



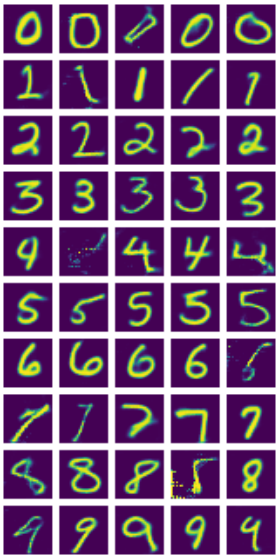
(b) FashionMNIST

(c) SVHN

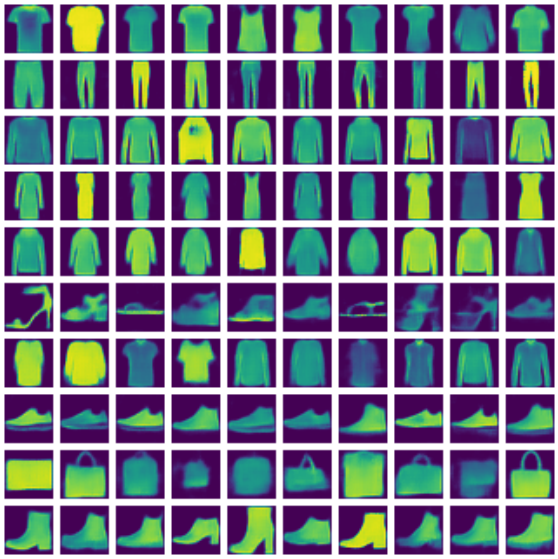


(d) CIFAR10

FIGURE 5.10 – Reconstruction des prototypes appris par ProtoVAE sur les quatre jeux de données étudiés



(a) MNIST

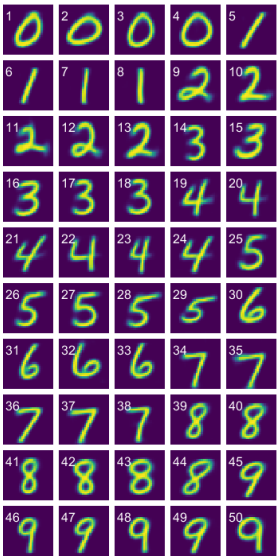


(b) FashionMNIST

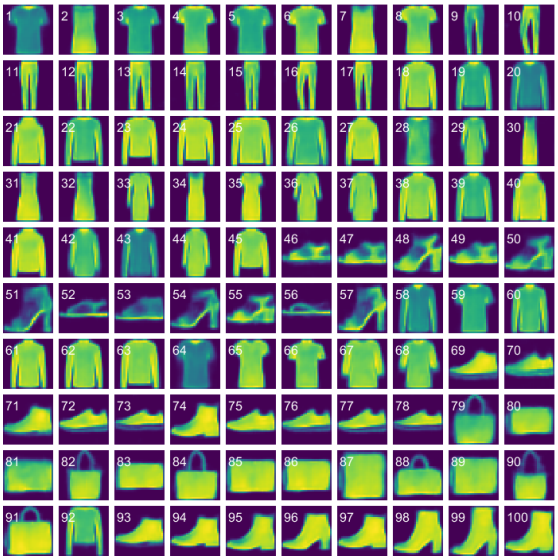
(c) SVHN

(d) CIFAR10

FIGURE 5.11 – Reconstruction des prototypes appris par PanVAE sur les quatre jeux de données étudiés

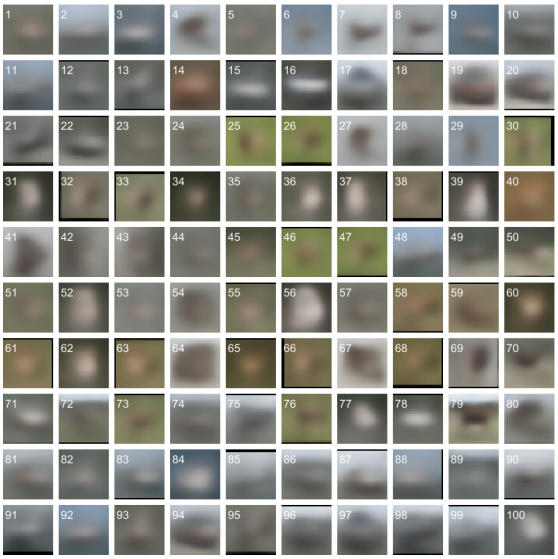


(a) MNIST



(b) FashionMNIST

(c) SVHN



(d) CIFAR10

FIGURE 5.12 – Reconstruction des prototypes appris par ProtoGMVAE sur les quatre jeux de données étudiés

hors-distribution.

Pour nos calculs d'évaluation, l'absence d'élimination n'a pas d'impact sur la distance entre une image et le prototype le plus proche. Pour la variance, en revanche, les prototypes hors-distribution pourraient augmenter les résultats. Cet effet est particulièrement visible pour le jeu de données SVHN, où certains prototypes ne représentaient aucune information utile, mais ne présentait simplement que du bruit, comme on peut le voir dans la figure 5.11c. La variance mesurée était de 0.0265, bien plus élevée qu'aucune autre méthode ou jeu de données. Le score dans le Tableau 5.4 a été recalculé sans ces prototypes hors-distribution. Nous avons également calculé les scores de variance et de distance obtenus par PanVAE en utilisant une des conditions de déclenchement de la phase d'élimination que les auteurs proposent dans leur code, mais nous ne savons pas si celle-ci correspond exactement à celle qu'ils ont utilisée dans leur papier. Ces résultats sont similaires à ceux que nous avons obtenus dans les Tableaux 5.3 et 5.4 et ne changent pas les conclusions que nous avons écrites un peu plus tôt.

Les prototypes obtenus avec ProtoVAE, PanVAE et ProtoGMVAE sur les quatre jeux de données sont donnés dans les figures 5.10, 5.11 et 5.12 respectivement.

5.7 Conclusion

Dans ce chapitre, nous avons proposé ProtoGMVAE, une méthode basée sur les VAE, utilisant un mélange de gaussiennes comme a priori pour modéliser fidèlement la distribution des données dans l'espace latent. Nous utilisons les composantes du mélange comme des prototypes non spécifiques à une classe et classifions les échantillons d'entrée de manière interprétable grâce aux probabilités d'appartenances prédites pour chaque composante.

Nous démontrons que ProtoGMVAE produit des prototypes divers sans avoir besoin d'un terme explicite poussant la diversité dans la fonction de perte. Les performances de classification obtenues sont semblables aux autres approches basées sur les prototypes utilisant des VAE testées, ProtoVAE et PanVAE.

Nous illustrons que notre approche est une bonne solution pour la classification intrinsèquement interprétable au travers de l'analyse des espaces latents appris et en les comparant à ceux de ProtoVAE et PanVAE. Ces expériences montrent que nos prototypes sont plus uniformément répartis et que ceux-ci restent pertinents même dans des cas où les données d'apprentissage sont déséquilibrées. Nous étayons ces illustrations par des mesures objectives, de variance visuelle intra-classe des prototypes et de distance moyenne entre les images de test et les prototypes, là aussi dans l'espace visuel. ProtoGMVAE obtient de meilleurs résultats dans tous

les scénarios.

Nous remarquons que le mélange gaussien tend à structurer l'espace latent de manière à ce qu'il soit fortement corrélé avec l'apparence visuelle des échantillons. Il s'agit d'une caractéristique intéressante qui soutient l'explicabilité de notre modèle, mais qui ne nous permet pas de surpasser les alternatives en termes de précision de classification. Nos travaux futurs se concentreront sur ce point.

6

L'explicabilité de ProtoGMVAE à l'épreuve du réel : application à l'imagerie cellulaire

Outline

6.1 Introduction 94
6.2 Analyse préliminaire 94
6.2.1 Formulation du problème 94
6.2.2 Application de ProtoVAE 96
6.2.3 Application de PanVAE 97
6.2.4 Application de ProtoGMVAE 97
6.3 Nouvelles stratégies 98
6.3.1 Modifications d'architecture 98
6.3.2 Visualisation des prototypes 99
6.4 Résultats expérimentaux 100
6.4.1 Analyse des prototypes 101
6.4.2 Espaces latents 102
6.5 Conclusion 104

6.1 Introduction

Le chapitre précédent introduisait ProtoGMVAE, notre approche explicable pour la découverte automatisée de structures typiques. Nous l'avons testée sur plusieurs jeux de données classiques et les résultats obtenus étaient encourageants. Désormais, nous allons appliquer notre méthode à nos images de cellules pour tenter de découvrir des marqueurs biologiques d'une atteinte cellulaire causée par un traitement.

Pour notre application, reconstruire une image de cellule entière n'a pas beaucoup d'intérêt puisque l'on cherche des motifs typiques plutôt que des cellules typiques. Nous avons donc fait le choix de retravailler notre jeu de données et de ne plus utiliser de segmentation comme dans le Chapitre 4, mais de simplement découper les images des puits pour générer de petites images représentant des textures intracellulaires. Nous reviendrons plus en détail sur les données analysées plus loin dans ce chapitre.

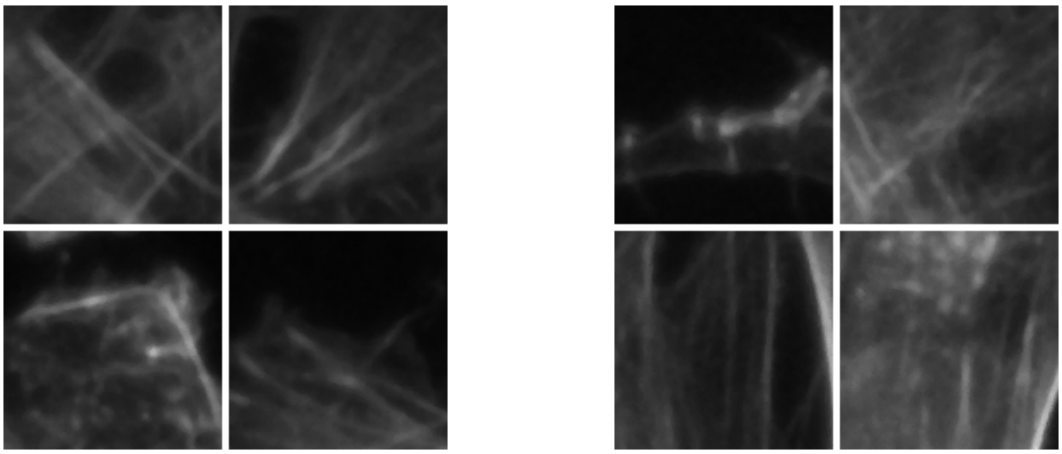
Le principal défi pour appliquer ProtoGMVAE à ces images est lié à leur complexité. En effet, les jeux de données comme MNIST [67], FashionMNIST [122], SVHN [85] ou CIFAR-10 [64] représentent des images naturelles, où l'information est centrée et orientée de manière similaire. Pour nos cellules, l'orientation est aléatoire et n'a pas d'importance sémantique, et certains motifs peuvent représenter des bords de cellules, l'information n'est alors contenue que dans une partie de l'image, qui n'est pas nécessairement centrée.

Dans ce nouveau chapitre, nous allons commencer par mettre en lumière les difficultés liées à un cas d'étude réel qu'est celui des cellules endothéliales de la BHE, en appliquant ProtoGMVAE tel que nous l'avions défini précédemment, ainsi que les méthodes auxquelles nous nous comparions : ProtoVAE [40] et PanVAE [61]. Nous verrons ensuite en détail les stratégies et les changements de paramètres que nous allons employer pour tenter de répondre aux problématiques spécifiques à ce cas d'étude. Enfin nous analyserons les résultats et les prototypes obtenus grâce à ces stratégies et nous conclurons sur l'applicabilité de ProtoGMVAE dans un contexte d'étude concret.

6.2 Analyse préliminaire

6.2.1 Formulation du problème

Le problème que nous cherchons à résoudre ici est assez similaire à celui que nous avions posé dans la section 4.2.1. Nous avons deux groupes de cellules : un groupe



(a) $\mathcal{P}^{\text{Ref}}$ (b) $\mathcal{P}^{\text{Exp}}$

FIGURE 6.1 – Exemples d'images analysées. (a) montre des images provenant de la population de référence non-traitée, (b) montre des images provenant de la population traitée au TNF-α.

de référence $\mathcal{P}^{\text{Ref}}$ qui n'est pas traité, et un groupe expérimental $\mathcal{P}^{\text{Exp}}$ exposé au traitement d'intérêt. Dans ce cas, il s'agit du TNF-α à une concentration de 100 ng/mL.

L'objectif est d'identifier les potentiels changements des cellules suite au traitement. Les labels auxquels nous avons accès sont imparfaits, ils ne renseignent que sur la provenance d'une image, à savoir la population $\mathcal{P}^{\text{Ref}}$ ou $\mathcal{P}^{\text{Exp}}$, mais les cellules de $\mathcal{P}^{\text{Ref}}$ peuvent être endommagées et les cellules $\mathcal{P}^{\text{Exp}}$ peuvent ne pas répondre au traitement.

La différence avec le chapitre 4, se situe dans le type d'images que nous analysons. Nous n'utilisons plus de segmentation pour avoir des cellules individuelles, et ce pour deux raisons. La première est que la segmentation n'est jamais parfaite et peut introduire des erreurs. Elle crée aussi artificiellement des effets de bord puisque les cellules sont organisées en tapis, et que séparer les cellules introduit des frontières nettes qui n'existent pas en réalité. Deuxièmement, ProtoGMVAE reconstruit les prototypes à la même taille que les images d'entrée et non comme des patches, comme c'était le cas pour ProtoPNet. Or, nous souhaitons identifier des textures et des structures intra-cellulaires. Travailler avec des images de cellules entières devient alors inutile.

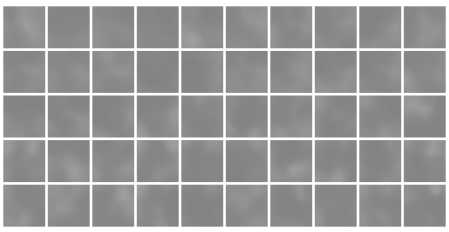
Nous avons donc choisi de découper les images des puits de culture en une multitude de petites images de taille 128×128 pixels, qui chacune représente alors une information de texture. A l'entrée du réseau, les images sont sous-échantillonnées

d'un facteur 2 pour se rapprocher de la taille des images que nous avions étudiées précédemment, sans perdre trop d'information. Nous avons également rajouté des augmentations avec des retournements horizontaux et verticaux aléatoires, pour renforcer l'apprentissage puisque l'orientation des textures n'apporte pas d'information. Comme les images des puits complets contiennent des zones noires, sans aucune cellule, un certain nombre des images re-découpées étaient entièrement noires. Nous avons donc effectué un tri, en définissant un seuil sur l'énergie des images, pour enlever celles ne contenant aucune information. La figure 6.1 montre des exemples des images ainsi obtenues. Au final, nous avons 25000 images, avec environ 13000 images pour $\mathcal{P}^{\text{Ref}}$ et 12000 images pour $\mathcal{P}^{\text{Exp}}$.

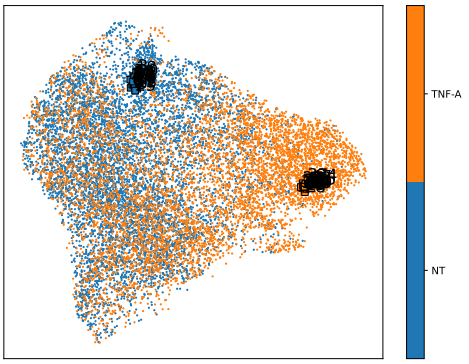
6.2.2 Application de ProtoVAE

Dans un premier temps, nous avons appliqué ProtoVAE [40] à cette nouvelle problématique cellulaire pour souligner la difficulté qu'ont la plupart des méthodes à s'adapter à un contexte d'étude réaliste.

La figure 6.2a montre les prototypes que ProtoVAE a appris. Tous ne représentent que du bruit et aucune information ne peut être tirée de ces derniers. Même entre les deux classes, il n'y a pas de différence notable. Pour la classification, ProtoVAE obtient un taux de classification correcte de 71.37% après entraînement.



(a)



(b)

FIGURE 6.2 – (a) Prototypes appris par ProtoVAE sur les images de cellules. 25 prototypes par classe. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par ProtoVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$

De la même manière que dans le chapitre précédent, on peut voir dans l'espace latent (Figure 6.2b) que tous les prototypes se situent au même endroit et ne vont

pas se répartir dans tout l'espace pour représenter toute la diversité de structures qui existe dans les données.

6.2.3 Application de PanVAE

De la même manière que pour ProtoVAE, nous avons également appliqué PanVAE [61] à nos images de cellules. Là encore, la méthode n'arrive pas à obtenir des prototypes qui encodent une véritable information (Figure 6.3a). Côté classification, les résultats sont mêmes pires que pour ProtoVAE, puisqu'on obtient seulement 68.77% de taux de classification correcte.

Il est important de souligner que PanVAE ne supporte pas très bien d'avoir un grand nombre de prototypes par classe. Nous n'avons utilisé que 20 prototypes au total, car en passant à 50 comme pour les autres ProtoVAE ou ProtoGMVAE, l'apprentissage rencontrait des problèmes et les résultats obtenus étaient encore moins bons. Cela semble lié à leur perte volumétrique, dont les valeurs tendent à exploser avec un grand nombre de prototypes.

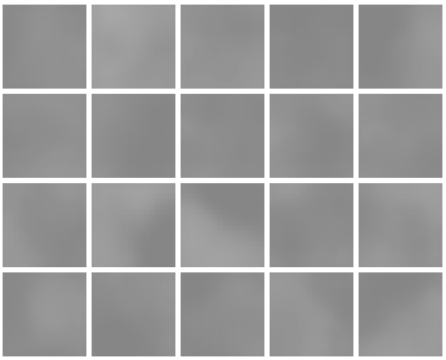
En s'intéressant à l'espace latent appris (Figure 6.3b), on remarque que les prototypes sont un peu plus étalés, mais que la région où les structures semblent appartenir aux deux classes est pratiquement dépourvue de prototypes. Cela est sans doute dû au rattachement des prototypes à une classe précise dans la méthode. On semble également distinguer un cluster au centre droit, plutôt rattaché à la classe $\mathcal{P}^{\text{Ref}}$ qui n'est pas représenté, ainsi que le petit nuage de points à gauche.

6.2.4 Application de ProtoGMVAE

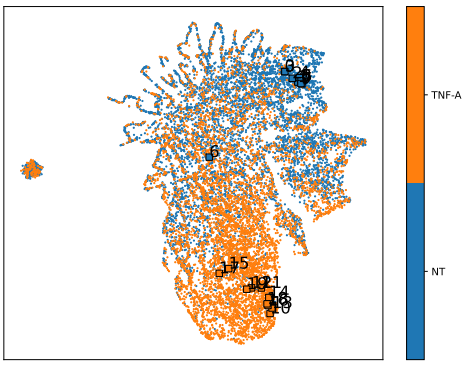
Les résultats préliminaires obtenus par ProtoGMVAE, sont un peu meilleurs que ceux des deux méthodes précédentes. Tout d'abord, même s'ils ne représentent pas de détails, les prototypes montrent tout de même une certaine diversité visuelle, et une information d'intensité lumineuse (Figure 6.4a). Ensuite, les performances de classification sont elles aussi supérieures à ProtoVAE et PanVAE, puisque nous obtenons 76.38 % de taux de classification correcte.

Mais l'analyse de l'espace latent (Figure 6.4b) montre que notre méthode aussi rencontre un problème de modélisation des données. Les prototypes ne sont pas bien répartis et l'ensemble des données semble avoir été modélisé comme une seule gaussienne.

Ce que montre cette analyse préliminaire est qu'aucune méthode ne semble capable de correctement représenter les données. Cela pose évidemment un problème pour l'apprentissage et la visualisation des prototypes. Même la tâche de classification



(a)



(b)

FIGURE 6.3 – (a) Prototypes appris par PanVAE sur la base des images de cellules. 10 prototypes par classe. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par PanVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$

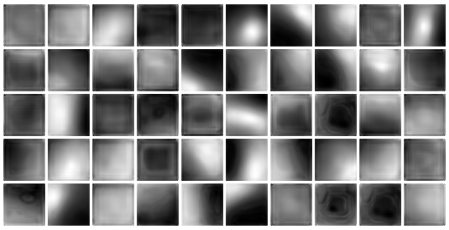
n'est pas pas complètement résolue et l'interprétation est alors impossible et aucune information pertinente ne peut être extraite des images.

6.3 Nouvelles stratégies

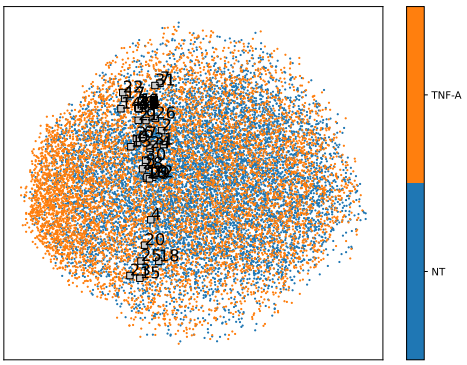
A la lumière des résultats précédents, il paraît impératif d'adapter notre méthode. Dans cette section nous allons détailler les différentes modifications pour parvenir à obtenir une bonne classification des images, et des prototypes dont il est possible de tirer une information biologique.

6.3.1 Modifications d'architecture

Dans le chapitre 5, afin que la comparaison avec ProtoVAE et PanVAE soit équitable, nous avions fait le choix de n'utiliser comme eux qu'une seule couche linéaire pour classifier les images. Désormais, cette exigence n'est plus nécessaire et nous pouvons utiliser un classifieur plus conséquent pour tenter de gagner en performances de classification. Pour retrouver l'information d'appartenance à une classe des prototypes, nous ne pouvons plus utiliser directement les poids associés à une classe puisqu'il y a plusieurs couches de neurones. En revanche, nous pouvons générer des vecteurs one-hot pour chacun des prototypes, que nous donnons en entrée du classifieur. Ce sont les logits en sortie de celui-ci qui nous disent maintenant vers quelle classe penchent les prototypes.



(a)



(b)

FIGURE 6.4 – (a) Prototypes appris par ProtoGMVAE sur la base des images de cellules. 50 prototypes au total. (b) Projection UMAP de la base d'images de test dans l'espace latent appris par ProtoGMVAE. NT : $\mathcal{P}^{\text{Ref}}$, TNF-A : $\mathcal{P}^{\text{Exp}}$

De la même manière, le choix de l'architecture de l'encodeur et du décodeur est plus libre. Les images étant plus grandes, nous avons fait le choix d'ajouter des couches convolutives et de pooling pour réduire la taille des cartes d'activations obtenues en sortie de notre encodeur et extraire des caractéristiques plus fines au sein des images.

6.3.2 Visualisation des prototypes

La reconstruction des prototypes obtenue dans l'analyse préliminaire n'est pas très précise et ne permet pas de voir des détails de la structure de l'actine des cellules. C'est un effet connu des VAE [18] s'expliquant par plusieurs facteurs. D'abord, la minimisation de la divergence de Kullback-Leibler dans la ELBO n'empêche pas $p_\theta(x)$ d'avoir une variance plus élevée que la distribution des données originales. Cette variance accrue amène donc à reconstruire des images plus diverses que les données originales, et inévitablement aussi des images floues. L'autre cause des reconstructions floues est le terme de perte utilisée pour la reconstruction. Dans le cas où l'a priori choisi est gaussien, le terme $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ dans la ELBO (cf Section 5.2) peut être simplifié comme une erreur quadratique moyenne (MSE). La MSE va accorder plus de poids aux basses fréquences de l'image et aux zones plus lumineuses. Cela conduit à la perte des hautes fréquences et donc des détails des images. Ce n'est pas un problème qui peut être simplement réglé, puisqu'il découle de la formulation mathématique du VAE.

Des approches comme celle de Hou et al. [51] proposent de remplacer la MSE par d'autres fonctions de perte, comme une perte perceptuelle calculée comme une distance entre la représentation latente des images générée par le VAE et celle générée par un réseau pré-entraîné pour la classification d'images. Cela permet de réduire les différences perceptuelles et sémantiques entre deux images dont les représentations latentes sont proches. Mais en remplaçant la MSE, on n'optimise plus réellement la ELBO, et la formulation n'est donc plus exactement celle d'un VAE.

Afin de contourner le problème de la reconstruction, nous avons choisi de revenir à une visualisation indirecte des prototypes. En générant les vecteurs $y|x$ associés à chacune des images de la base de test, nous pouvons déterminer celles qui activent le plus fortement chacun des prototypes. Nous gardons les 10 images qui activent le plus les prototypes afin de mieux comprendre le motif représenté par ces derniers.

Même si nous ne nous servons plus directement de l'encodeur après entraînement, il est important de le conserver, ainsi que la perte de reconstruction, puisqu'ils participent la formulation générale de la tâche d'optimisation du VAE. Sans ces composantes, nous avons pu constater dans nos expériences que tous les prototypes perdent un lien avec l'espace visuel d'entrée et que leur emplacement dans l'espace latent devient indifférent et identique pour tous.

6.4 Résultats expérimentaux

Dans cette section, nous allons analyser les résultats obtenus par ProtoGMVAE en appliquant les nouvelles stratégies définies précédemment. Notre intérêt se porte principalement sur la nouvelle visualisation des prototypes et leur capacité à extraire des informations structurelles sur les cellules, ainsi que sur la structuration de notre espace latent.

Comme nous l'expliquions dans la section 6.3.1, nous avons renforcé certaines parties de ProtoGMVAE. L'encodeur et le décodeur comportent désormais cinq couches convolutives contre seulement trois précédemment. Le nombre de couches denses du classifieur a été porté à trois. Le reste de l'architecture est inchangé. Le nombre de prototypes est fixé à 50. Le modèle est entraîné avec l'optimiseur Adam [57] avec un taux d'apprentissage de 10^{-4}. La taille des mini-batches est de 64 images. Nous avons également gardé les augmentations par retournements horizontaux et verticaux utilisées dans l'analyse préliminaire. Le détail des paramètres d'entraînement est disponible dans le code de ProtoGMVAE : https://github.com/martin-blcd/ProtoGMVAE.

6.4.1 Analyse des prototypes

Nous avons donc extrait de la base de test les 10 images qui activent le plus chacun des prototypes. Comme nous l'expliquions, dans la section 6.3.1, pour identifier la classe dont ils se rapprochent le plus, nous avons classifié les vecteurs one-hot correspondants. Après examen des scores de classification, nous avons pu déterminer les prototypes liés à la population $\mathcal{P}^{\text{Ref}}$, ceux liés à la population $\mathcal{P}^{\text{Exp}}$, et quelques prototypes qui ne sont pas particulièrement typiques d'une classe. Chaque ligne de la figure 6.5 montre les 10 images qui activent le plus chacun des 8 prototypes non-spécifiques, dont le score de classification ne dépasse pas 70% vers une classe. Tout d'abord, on peut remarquer que sur chaque ligne, les 10 images partagent bien un aspect similaire. Cela nous rassure sur la capacité des prototypes à encoder une information visuelle, malgré la reconstruction qui n'est pas nette. Le point commun entre ces prototypes intermédiaires semble être qu'ils encodent principalement du bruit, ou bien une saturation de l'intensité lumineuse liée à des défauts lors de l'acquisition, comme des impuretés ou des bulles d'air qui se sont glissées dans les cultures cellulaires. Le fait que ces prototypes ne penchent pas particulièrement vers une classe précise est assez logique puisque ces défauts ne sont pas liés à des paramètres biologiques propres aux cellules.

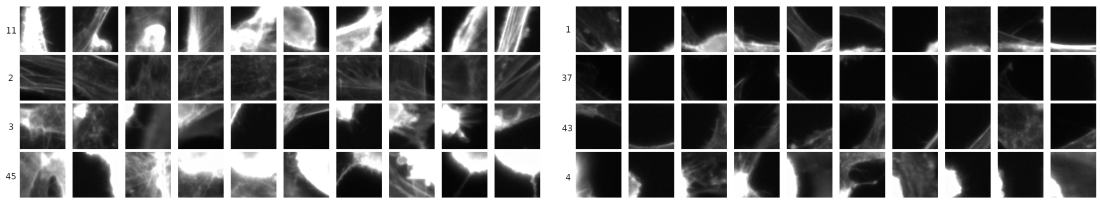


FIGURE 6.5 – Visualisation des prototypes n'étant associés à aucune population particulière. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype.

La figure 6.6 reprend les visualisations des 18 prototypes qui sont liés à la classe des cellules non-traitées, classés par probabilité d'appartenance décroissante. Beaucoup de ces prototypes montrent des structures d'actine sous forme de filaments, répartis de manière uniforme (n°5,28,25,15,8). Certains prototypes sont cependant assez surprenants, et semblent représenter une position dans l'image (n°49), ou des bords de cellules (n°24). Ces deux motifs ne paraissent pourtant pas liés à une information biologique particulière.

Dans la figure 6.7, ce sont les visualisations des prototypes de cellules traitées qui sont représentées. On retrouve là aussi des textures montrant de l'actine filamentaire (n°20), mais on voit également apparaître certaines textures plus granuleuses (n°48,42,30,14), que l'on avait déjà identifiées dans notre analyse du chapitre 4.

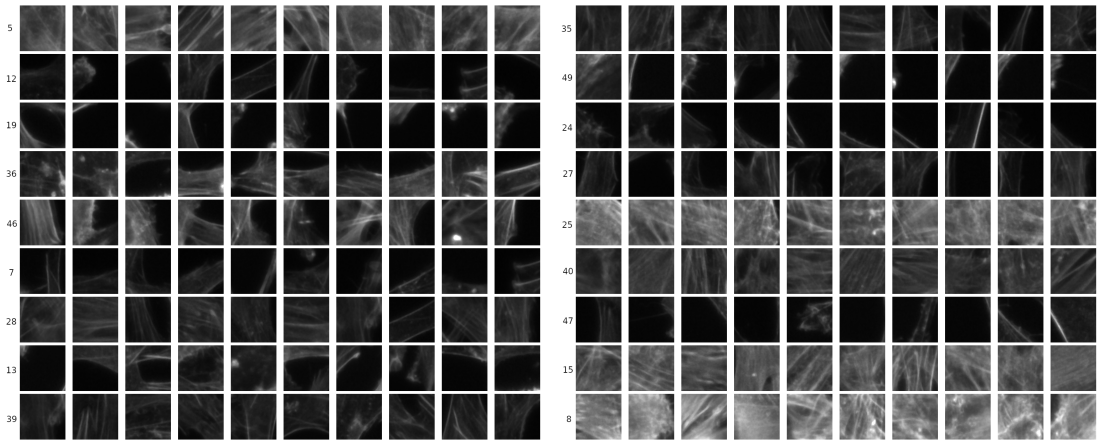


FIGURE 6.6 – Visualisation des prototypes associés à la population $\mathcal{P}^{\text{Ref}}$. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype. Prototypes classés par ordre décroissant de score de classification.

Tout comme pour les cellules de la population de référence, on trouve certains prototypes qui encodent du bruit de fond ou des morceaux de cellule très petits (n°34,9,44) ou là encore une saturation d'intensité (n°0). Proportionnellement, il y a plus d'images présentant du fond dans la population $\mathcal{P}^{\text{Exp}}$ que dans la population $\mathcal{P}^{\text{Ref}}$, il est donc plus logique d'y retrouver ce genre de prototypes. On peut aussi souligner que les images de la figure 6.7, particulièrement la colonne de droite, paraissent plus contrastées que celles de la figure 6.6. Il peut s'agir là d'une vraie information pertinente au point de vue biologique, dans le cas où cela serait lié à une modification de la structure d'actine, ne laissant que certains filaments intacts, lesquels seraient plus lumineux que le reste de la cellule à l'image. Il pourrait aussi s'agir d'un simple biais sur les images, car comme nous l'expliquions dans le chapitre 3, de nombreux paramètres biologiques et optiques de l'acquisition peuvent varier d'un puits à l'autre.

6.4.2 Espaces latents

L'autre point qui restait à améliorer est la structuration de notre espace latent. La figure 6.8 nous montre l'espace latent que nous avons appris avec notre encodeur plus conséquent. On voit que les prototypes sont mieux répartis qu'auparavant, avec tout de même des zones plus denses. Mais il semble que les données soient toujours structurées comme une seule gaussienne. Le modèle atteint pourtant 82.92% de classification correcte, grâce au classifieur amélioré. Ce taux élevé est difficile à comprendre en regardant la position des images et des prototypes dans l'espace

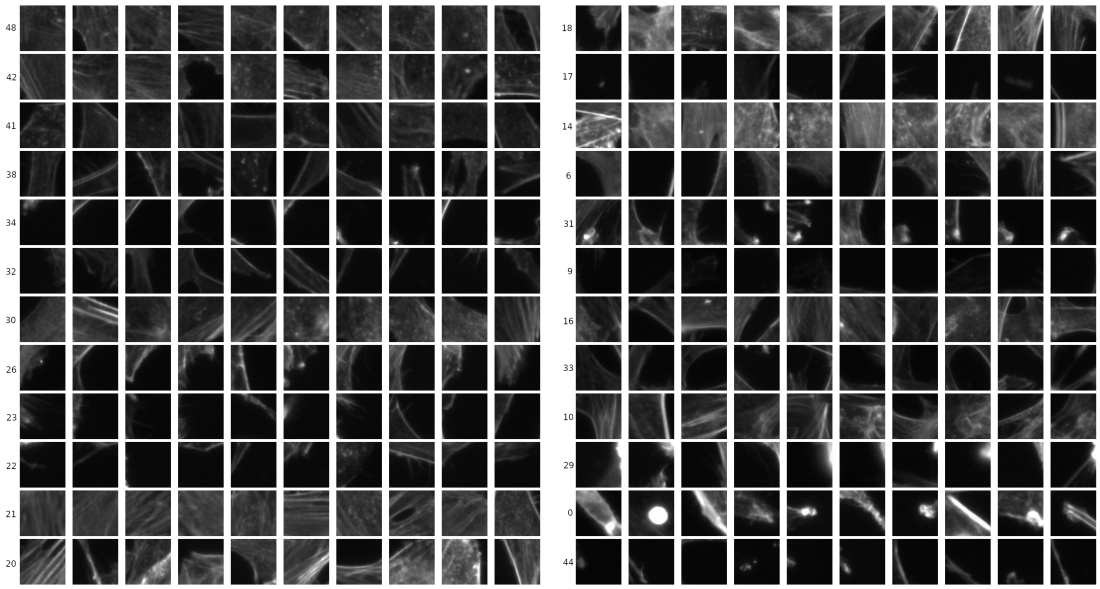


FIGURE 6.7 – Visualisation des prototypes associés à la population $\mathcal{P}^{\text{Exp}}$. Chaque ligne correspond à un prototype et est constituée des 10 images de la base de test activant le plus ce prototype. Prototypes classés par ordre décroissant de score de classification.

latent.

Pour mieux comprendre les résultats de classification, nous pouvons observer un second espace latent, celui associé à notre seconde variable latente $y|x$. Rappelons que la distribution $y|x$ exprime le poids de chaque composante du mélange pour l'image d'entrée x. C'est ce vecteur qui est utilisé pour prendre la décision de classification et c'est celui-ci que nous pouvons représenter. La figure 6.9 montre ce second espace latent, et révèle une toute autre structuration de nos données. Dans cet espace, chaque dimension correspond à un prototype et la position d'une image sur la dimension correspond au pourcentage d'activation du prototype. Si la séparation des données n'est pas nette, on distingue cependant des zones de l'espace où une classe est plus représentée que l'autre. Il faut se rappeler que c'est bien dans cet espace que nous effectuons notre classification, il est donc logique que celui-ci reflète mieux la distinction que notre modèle fait entre les différentes classes.

On peut aussi se demander pourquoi les images semblent ne pas être très bien organisées en $z|x, y$. Cela pourrait avoir un lien avec la forme des distributions $y|x$ à partir desquels nous classifions. Actuellement, plusieurs prototypes peuvent être conjointement activés, et rien ne contraint les distributions associées aux proto-

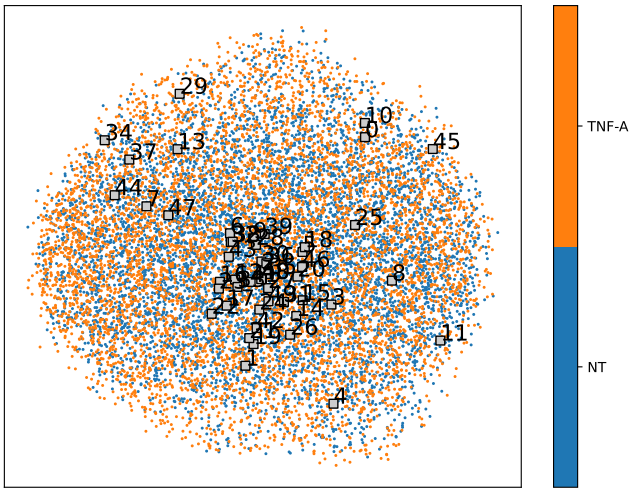


FIGURE 6.8 – Projection UMAP de l'espace latent en $z|x, y$ appris par ProtoGMVAE sur la base des images de cellules.

types à ne pas se chevaucher dans l'espace latent $z|x, y$. Il pourrait donc y avoir un effet où les distributions $y|x$ sont bien discriminantes, mais les modes leur correspondant dans l'espace $z|x, y$ se chevauchant, les frontières de décision deviennent alors invisibles.

6.5 Conclusion

Cette nouvelle analyse des images de cellules à l'aide de ProtoGMVAE nous a permis de mieux adapter son fonctionnement à un contexte réel. Nous avons pu identifier certaines limitations dans la formulation initiale de notre modèle : prototypes illisibles, et espace latent mal structuré.

Les adaptations de l'architecture, avec un encodeur-décodeur plus profond, n'ont pas suffi à régler les problèmes de structuration de l'espace. Les images ne semblent pas s'organiser de façon discriminante pour la tâche de classification, mais les prototypes sont mieux répartis dans tout l'espace, et couvrent une large diversité de structures. Ces modifications ont également permis de dégager une hypothèse, celle du chevauchement des modes en $z|x, y$, qui pourrait être à l'origine de la confusion dans l'espace latent, et permettre dans de futurs travaux d'améliorer encore notre méthode. Il serait aussi intéressant de regarder l'écart type des composantes du

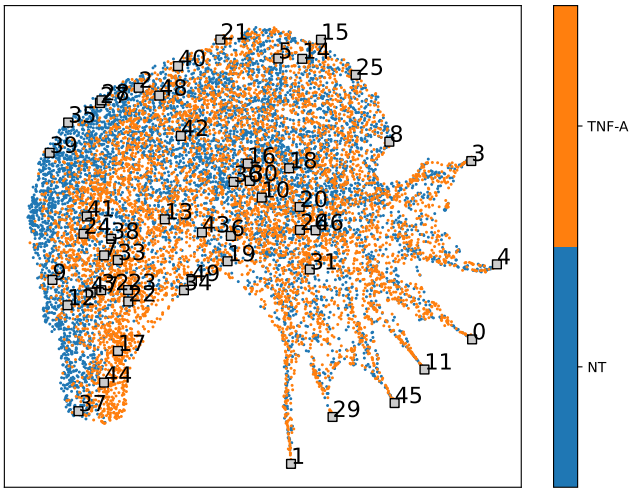


FIGURE 6.9 – Projection UMAP de l'espace latent en $y|x$ appris par ProtoGMVAE sur la base des images de cellules.

mélange pour vérifier si certains sont très élevés et recouvrent une large partie de l'espace latent.

Du côté de la visualisation des prototypes, la formulation de notre problème d'optimisation fait apparaître deux variables latentes $z|x, y$ et $y|x$. La reconstruction des prototypes au travers de la première ne permettant pas de voir correctement les détails, nous avons pu nous rabattre sur la seconde variable et identifier les images de la base de test qui activent le plus chacun des prototypes. Grâce à cette nouvelle méthode de visualisation, nous avons pu mettre en lumière des structures typiques de nos deux populations de cellules, $\mathcal{P}^{\text{Ref}}$ et $\mathcal{P}^{\text{Exp}}$, qui semblent indiquer la présence d'actine non-filamentaire dans les cellules ayant répondu au traitement au TNF-α. Ces résultats sont en accord avec nos travaux du chapitre 4 et nous confortent dans l'idée que notre méthode est capable de fonctionner dans des applications réelles, avec une complexité élevée.

7

Conclusion

Outline

7.1 Rappel des objectifs 107
7.2 Contributions 107
7.2.1 Jeu d'images de cellules 107
7.2.2 Première analyse des cellules par réseau de neurones interprétable 108
7.2.3 ProtoGMVAE 109
7.2.4 Seconde analyse des cellules à l'aide de ProtoGMVAE . 109
7.3 Perspectives 110
7.3.1 Amélioration de ProtoGMVAE 110
7.3.2 Approfondissement de l'analyse des images de cellules . 111

7.1 Rappel des objectifs

L'enjeu de l'interprétabilité des réseaux de neurones est au cœur des réflexions actuelles. C'est un des vecteurs de réticence à leur adoption, particulièrement dans des domaines où l'impact sur la vie des personnes peut être important, notamment la santé. Nos travaux avaient pour but de mettre au point une méthode pour la classification d'images, qui serait capable d'apporter une explication de ses décisions. L'idée était qu'au travers de cette classification, notre réseau de neurones apprenne les structures caractéristiques qui définissent les différentes classes des données d'entraînement. Ces structures pourraient alors servir de base à l'interprétation des résultats.

Les principales difficultés étaient liées au type de données que nous souhaitions analyser. Nous avions en ligne de mire l'analyse des cellules endothéliales de la BHE, traitées par différentes substances d'intérêt. A terme, nous souhaitions pouvoir étudier des substances dont les effets sur les cellules puissent être inconnus. Notre méthode ne pouvait donc pas se servir de connaissances préalables, par exemple, pas d'étiquetage individuel précis des images de cellules, seulement une indication globale de l'exposition au traitement. Ces labels globaux sur les populations sont imparfaits, car pendant la culture cellulaire, il se peut que les cellules exposées au traitement ne réagissent pas à ce dernier, tandis que dans la population témoin, non-exposée, certaines cellules peuvent avoir été endommagées par les différentes manipulations ou sont mortes naturellement.

Nous avons exploré les méthodes d'explicabilité par prototypes, qui semblaient le mieux répondre à nos attentes et permettre de remplir nos objectifs.

7.2 Contributions

7.2.1 Jeu d'images de cellules

Dans un premier temps, nous avons constitué notre propre jeu de données. Toute la culture cellulaire a été faite au laboratoire Sainbiose. Plusieurs populations ont été cultivées avec une exposition à des doses croissantes de deux produits : le TNF-α et l'IL1-β, deux cytokines inflammatoires. Les deux populations ont ensuite été imagées en immunofluorescence, toujours au laboratoire Sainbiose. Deux composantes cellulaires ont été marquées et imagées : les noyaux et l'actine du cytosquelette. Nous avons pris des images en faisant varier le point de focus du microscope, pour compenser les variations éventuelles de niveau, mais aussi pour avoir des informations venant du coeur des cellules, ainsi que de leur surface. Pour intégrer toutes ces informations en une seule image et éviter un traitement sur toute une pile, nous

avons utilisé la méthode de la pyramide Laplacienne pour fusionner les pixels nets des différentes tranches. Les détails de ces opérations sont décrits dans le chapitre 3.

7.2.2 Première analyse des cellules par réseau de neurones interprétable

Dans le chapitre 4, nous avons effectué une première analyse de nos images de cellules en utilisant ProtoPNet, une approche de classification explicable utilisant des prototypes. ProtoPNet apprend des structures typiques de chacune des classes et classifie les images d'entrée sur la base de leur similarité à ces prototypes. Pour adapter ProtoPNet à notre contexte biomédical, nous avons dû effectuer certaines modifications. Nous avons travaillé avec des images ne comportant que des cellules individuelles, en appliquant Cellpose, un algorithme de segmentation d'images spécialisé pour l'imagerie cellulaire. Nous avons ré-employé les masques de segmentation obtenus comme secondes entrées du réseau de neurones pour empêcher les activations dans le fond des images, qui pouvaient apporter des information utiles à la classification (les cellules traitées ayant tendance à se décoller de leur support de culture car affaiblies), mais peu pertinentes du point de vue de la biologie car n'indiquant pas l'état de la cellule. Nous avons également changé la méthode initiale de visualisation des prototypes, peu adaptée à nos images, en limitant le champ réceptif du réseau de neurones. Enfin nous avons aussi contraint le modèle à apprendre des prototypes pertinents, en gelant les poids de la couche de classification. Ainsi, chacun des prototypes a un impact sur la classification et le modèle est forcé d'apprendre des structures pertinentes. Il apprenait auparavant des motifs souvent inutiles, mais les écartait en passant à zéro les poids correspondant dans la couche de classification pendant l'apprentissage.

Avec ces adaptations, nous avons réussi à mettre en lumière deux structures intéressantes pour les cellules saines et pour les cellules traitées au TNF-α. La première décrit une structure filamentaire de l'actine, attendue pour des cellules saines. La seconde décrit quant à elle une structure d'actine non-filamentaire, avec une texture granuleuse. Cet aspect de l'actine semble indiquer une modification de sa répartition au sein de la cellule, et une potentielle dépolymérisation de l'actine, menant à la perte de la structure cellulaire et à une éventuelle atteinte de l'intégrité de la BHE.

Malgré les résultats encourageants obtenus précédemment, ProtoPNet et notre approche souffrent de plusieurs défauts. La segmentation introduit plusieurs biais, avec les éventuels défauts, mais aussi en créant artificiellement des effets de bord. Les cellules étant organisées en un tapis cellulaire, il est difficile de délimiter préci-

sément leurs contours. Ensuite, l'entraînement de ProtoPNet se fait par séquences alternées : apprentissage des prototypes, puis apprentissage de la classification. Cet entraînement alterné entraîne des pertes de performance indésirables. De plus, les prototypes ne sont liés qu'à une seule classe. C'est une propriété qui n'est pas forcément désirable puisqu'à cause de nos labels imparfaits, les classes peuvent partager des structures similaires. Pour toutes ces raisons, nous avons voulu mettre au point une nouvelle méthode plus adaptée.

7.2.3 ProtoGMVAE

Le chapitre 5 était dédié à la présentation de ProtoGMVAE, une méthode pour l'explicabilité des réseaux de neurones, basée sur les prototypes. ProtoGMVAE reprend une architecture d'auto-encodeur variationnel, en utilisant un mélange de gaussiennes comme a priori, afin de modéliser fidèlement les données d'apprentissage. On peut obtenir une classification basée sur la probabilité d'appartenance de l'image d'entrée aux différentes composantes du mélange. Ces composantes servent alors de prototypes pour interpréter le résultat de classification.

ProtoGMVAE corrige la plupart des défauts de ProtoPNet que nous avions soulevés dans le chapitre précédent. La procédure d'apprentissage se fait désormais de bout-en-bout, sans étape de projections des prototypes qui diminuait les résultats de classification et nécessitait de compenser cette perte par une optimisation du classifieur seul. Nos prototypes ne sont désormais plus rattachés à une classe précise et peuvent être typiques de plusieurs classes si celles-ci partagent des caractéristiques communes. Enfin, la procédure de visualisation des prototypes est simplifiée puisque l'architecture comprend un décodeur, capable de reconstruire des images et de retourner facilement dans l'espace pixel initial.

Nous comparons nos résultats à d'autres méthodes d'explicabilité utilisant des auto-encodeurs variationnels, avec un apprentissage de prototypes : ProtoVAE et PanVAE. Les résultats expérimentaux montrent que malgré une performance légèrement inférieure en classification, ProtoGMVAE apprend des prototypes de meilleure qualité. Ceux-ci sont plus divers et couvrent mieux la distribution des données. De plus, même dans des contextes où les données sont déséquilibrées, ProtoGMVAE parvient à apprendre des prototypes pertinents et représentatifs de tous les éléments des classes.

7.2.4 Seconde analyse des cellules à l'aide de ProtoGMVAE

Enfin, dans le chapitre 6, nous mettons ProtoGMVAE à l'épreuve sur l'analyse des images de cellules. Beaucoup de méthodes d'explicabilité ne sont testées que sur les cas d'études simplifiés que sont les jeux de données classiques d'analyse

d'images. Sans aucune adaptation, ProtoGMVAE n'est pas capable de résoudre correctement la tâche mais notre analyse préliminaire montre que les résultats sont meilleurs que ceux des deux autres méthodes auxquelles nous nous comparons. Nous proposons plusieurs stratégies pour contourner les écueils spécifiques au problème des cellules : visualiser les images de la base de test qui activent le plus les prototypes pour comprendre ce qu'ils représentent ; et utiliser une architecture plus profonde pour l'encodeur-décodeur et pour le classifieur, afin d'apprendre des caractéristiques plus riches et mieux classifier les images, respectivement. Grâce aux modifications apportées, nous sommes parvenus à extraire des prototypes encodant des informations de structures cellulaires similaires à celles que nous avions déterminé dans le chapitre 4 et nous avons rendu ProtoGMVAE plus robuste pour des applications spécifiques.

7.3 Perspectives

7.3.1 Amélioration de ProtoGMVAE

Bien que les résultats de ProtoGMVAE soient prometteurs, notre dernière étude a soulevé plusieurs pistes d'amélioration pour rendre la méthode encore plus robuste et généralisable. La reconstruction semble à la peine lorsque les données sont complexes et détaillées. C'est un problème assez connu des VAE et plusieurs travaux cherchent à améliorer leur capacité à reconstruire des textures et des détails fins dans des images. Les VQ-VAE [113] par exemple améliorent grandement la qualité de reconstruction des images.

Un autre problème identifié des VAE est l'enchevêtrement des dimensions de l'espace latent, c'est-à-dire le fait que les dimensions de l'espace latent peuvent encoder plusieurs informations visuelles se recoupant. Ce problème pourrait être derrière la mauvaise structuration de notre espace latent $z|x, y$. Des alternatives comme le β-VAE [49] ou le β-TCVAE [28] parviennent à découpler les caractéristiques dans l'espace et pourraient nous aider à apprendre de meilleurs espaces latents.

Il y a aussi la piste évoquée dans notre dernier chapitre, du chevauchement des distributions associées aux prototypes dans l'espace latent. En plus de vérifier l'hypothèse que nous avions formulée dans la section 6.4.2, nous pourrions aussi essayer de contrôler la variance de ces distributions. Actuellement, ce paramètre est appris durant l'entraînement du modèle, mais il serait possible de le fixer, soit pour les distributions en sortie de $q_\phi(x, y)$, soit pour les distributions de $p_\theta(z|y)$, ou même pour les deux. Une taille fixe ou limitée pourrait ainsi aider à ce que les distributions couvrent une moins grande partie de l'espace latent, et réduire le risque de chevauchement, qui nuit à l'interprétabilité de cet espace.

Enfin, il pourrait être intéressant de créer une méthode à mi-chemin entre les approches patch-prototypes et les approches image-prototypes. En découpant les images données en entrée du réseau en une série de patches, pour apprendre la reconstruction à cette échelle ci, puis en classifiant l'image initiale avec une fonction d'agrégation des scores de chacun des patches, il serait possible d'obtenir une méthode combinant les aspects avantageux des deux types d'approches.

7.3.2 Approfondissement de l'analyse des images de cellules

Dans notre dernier chapitre, nous nous sommes attelés à l'analyse des images de cellules. Nous ne nous sommes intéressés qu'à un seul traitement et une seule dose parmi toutes les populations que nous avions imagées, principalement par manque de temps. Il pourrait être intéressant de tester toutes les autres populations et de trouver un moyen de quantifier l'atteinte des cellules en fonction de la dose de produit à laquelle elles ont été exposées.

Nous avions également envisagé de faire une nouvelle culture cellulaire, en utilisant les mêmes cellules et les mêmes traitements, dans le but d'avoir une meilleure validation expérimentale des résultats obtenus. Si nous poussons l'analyse des cellules sur toutes les populations que nous avons déjà, alors cette étape de validation supplémentaire serait une bonne manière de solidifier les résultats et les éventuelles conclusions biologiques sur les mécanismes d'action délétères du TNF-α et de l'IL1-β sur nos cellules.

Nous pourrions également étendre l'analyse à d'autres substances d'intérêt. L'une des pistes évoquée au début de la thèse et qui avait motivé le choix des cellules endothéliales de la BHE, était le sérum sanguin de patients atteints de maladies neurodégénératives. Si le sérum avait un effet sur ces cellules, alors il aurait pu s'agir d'un mécanisme d'action potentiel de ce type de maladie. Cela pourrait un jour permettre de diagnostiquer ces maladies avant que les effets les plus graves apparaissent.

Enfin, puisque ProtoGMVAE a pu être adapté pour l'analyse des cellules, la méthode pourrait être généralisée à des images issues de différentes applications réelles, dans des contextes pouvant aller au-delà de la biologie.

Bibliographie

[1] Règlement (ue) 2016/679 du parlement européen et du conseil du 27 avril 2016 relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données, et abrogeant la directive 95/46/ce (règlement général sur la protection des données). http ://data.europa.eu/eli/reg/2016/679/oj, 2016. 2

[2] Règlement (ue) 2024/1689 du parlement européen et du conseil du 13 juin 2024 établissant des règles harmonisées concernant l'intelligence artificielle et modifiant les règlements (ce) n° 300/2008, (ue) n° 167/2013, (ue) n° 168/2013, (ue) 2018/858, (ue) 2018/1139 et (ue) 2019/2144 et les directives 2014/90/ue, (ue) 2016/797 et (ue) 2020/1828 (règlement sur l'intelligence artificielle). http ://data.europa.eu/eli/reg/2024/1689/oj, 2024. 2

[3] Amina Adadi and Mohammed Berrada. Peeking inside the black-box : a survey on explainable artificial intelligence (xai). *IEEE access*, 6 :52138–52160, 2018. 10

[4] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018. ix, 25, 27, 73

[5] Robert Andrews, Joachim Diederich, and Alan B Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6) :373–389, 1995. 2

[6] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence : an analytical review. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 11(5) :e1424, 2021. 10

[7] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa : Visual question ans-

wering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 22

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai) : Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58 :82–115, 2020. 10

[9] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7) :e0130140, 2015. 19

[10] Xiangzhi Bai, Yu Zhang, Fugen Zhou, and Bindang Xue. Quadtree-based multi-focus image fusion using a weighted focus-measure. *Information fusion*, 22 :105–118, 2015. x, 38, 40, 41

[11] Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. 2011. 27

[12] Martin Blanchard, Olivier Delézay, Christophe Ducottet, and Damien Muselet. Delving into the explainability of prototype-based cnn for biological cell analysis. In *2024 IEEE International Conference on Image Processing (ICIP)*, pages 2909–2915. IEEE, 2024. ii, 7

[13] Martin Blanchard, Olivier Delézay, Christophe Ducottet, and Damien Muselet. Protogmvae : A variational auto-encoder with true gaussian mixture prior for prototypical-based self-explainability. *To be published at IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026. ii, 7

[14] Marie Blanchette and Richard Daneman. Formation and maintenance of the bbb. *Mechanisms of development*, 138 :8–16, 2015. 4

[15] Marcus D Bloice, Christof Stocker, and Andreas Holzinger. Augmentor : an image augmentation library for machine learning. *arXiv preprint arXiv :1708.04680*, 2017. 55

[16] Benedikt Boecking, Naoto Usuyama, Shruthi Bannur, Daniel C Castro, Anton Schwaighofer, Stephanie Hyland, Maria Wetscherek, Tristan Naumann, Aditya Nori, Javier Alvarez-Valle, et al. Making the most of text semantics to improve biomedical vision–language processing. In *European conference on computer vision*, pages 1–21. Springer, 2022. 22

[17] Moritz Böhle, Mario Fritz, and Bernt Schiele. B-cos networks : Alignment is all we need for interpretability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10329–10338, 2022. ix, 24

[18] Gustav Bredell, Kyriakos Flouris, Krishna Chaitanya, Ertunc Erdil, and Ender Konukoglu. Explicitly minimizing the blur error of variational autoencoders. *arXiv preprint arXiv :2304.05939*, 2023. 99

[19] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv :1904.00760*, 2019. 26

[20] John F Brenner, Brock S Dew, J Brian Horton, Thomas King, Peter W Neurath, and William D Selles. An automated microscope for cytologic research a preliminary evaluation. *Journal of Histochemistry & Cytochemistry*, 24(1) :100–111, 1976. 37

[21] David A Broniatowski and David A Broniatowski. *Psychological foundations of explainability and interpretability in artificial intelligence*, volume 4. US Department of Commerce, National Institute of Standards and Technology, 2021. 10

[22] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1983. x, 38, 40, 41

[23] Peter J Burt and Edward H Adelson. Merging images through pattern decomposition. In *Applications of digital image processing VIII*, volume 575, pages 173–181. SPIE, 1985. x, 38, 40, 41

[24] Gabriele Campanella, Matthew G Hanna, Luke Geneslaw, Allen Miraflor, Vitor Werneck Krauss Silva, Klaus J Busam, Edi Brogi, Victor E Reuter, David S Klimstra, and Thomas J Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature medicine*, 25(8) :1301–1309, 2019. 62

[25] Gianluca Carloni, Andrea Berti, Chiara Iacconi, Maria Antonietta Pascali, and Sara Colantonio. On the applicability of prototypical part learning in medical images : Breast masses classification using protopnet. In *International Conference on Pattern Recognition*, pages 539–557. Springer, 2022. 30

[26] Zachariah Carmichael, Suhas Lohit, Anoop Cherian, Michael J Jones, and Walter J Scheirer. Pixel-grounded prototypical part networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4768–4779, 2024. 28, 53, 64

[27] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that : deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019. i, ix, x, 7, 8, 28, 29, 31, 32, 46, 49, 57, 79

[28] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018. 110

[29] Xinlei Chen and C Lawrence Zitnick. Mind's eye : A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431, 2015. 21

[30] Randall Davis, Bruce Buchanan, and Edward Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial intelligence*, 8(1) :15–45, 1977. 2

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 50

[32] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv :1611.02648*, 2016. 68, 74

[33] Roberto Dominguez and Kenneth C Holmes. Actin structure and function. *Annual review of biophysics*, 40 :169–186, 2011. 4

[34] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet : An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10265–10275, 2022. 73

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,

Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*, 2020. 21

[36] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3) :1, 2009. 16

[37] Theodore Evans, Carl Orge Retzlaff, Christian Geißler, Michaela Kargl, Markus Plass, Heimo Müller, Tim-Rasmus Kiehl, Norman Zerbe, and Andreas Holzinger. The explainability paradox : Challenges for xai in digital pathology. *Future Generation Computer Systems*, 133 :281–296, 2022. 30

[38] Jhosimar Arias Figueroa. Semi-supervised learning using deep generative models and auxiliary tasks. In *NeurIPS Workshop on Bayesian Deep Learning*, 2019. 68

[39] Stanislav Fort. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv :1708.02735*, 2017. 27

[40] Srishti Gautam, Ahcene Boubekki, Stine Hansen, Suaiba Salahuddin, Robert Jenssen, Marina Höhne, and Michael Kampffmeyer. Protovae : A trustworthy self-explainable prototypical variational model. *Advances in Neural Information Processing Systems*, 35 :17940–17952, 2022. xi, 8, 28, 64, 66, 71, 72, 79, 94, 96

[41] Srishti Gautam, Marina M-C Höhne, Stine Hansen, Robert Jenssen, and Michael Kampffmeyer. This looks more like that : Enhancing self-explaining models by prototypical relevance propagation. *Pattern Recognition*, 136 :109172, 2023. 28, 64

[42] Mahdi Ghorbani, Samarjeet Prasad, Jeffery B Klauda, and Bernard R Brooks. Variational embedding of protein folding simulations using gaussian mixture variational autoencoders. *The Journal of Chemical Physics*, 155(19), 2021. 68

[43] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5) :93 :1–93 :42, 2019. 10

[44] Liqiang Guo, Ming Dai, and Ming Zhu. Multifocus color image fusion based on quaternion curvelet transform. *Optics express*, 20(17) :18846–18860, 2012. 38

[45] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6) :1397–1409, 2012. 38

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 50

[47] Indra Heckenbach, Garik V Mkrtchyan, Michael Ben Ezra, Daniela Bakula, Jakob Sture Madsen, Malte Hasle Nielsen, Denise Oró, Brenna Osborne, Anthony J Covarrubias, M Laura Idda, et al. Nuclear morphology is a deep learning biomarker of cellular senescence. *Nature Aging*, 2(8) :742–755, 2022. 63

[48] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European conference on computer vision*, pages 3–19. Springer, 2016. ix, 21

[49] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae : Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017. 110

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997. 21

[51] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 1133–1141. IEEE, 2017. 100

[52] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 50

[53] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021. 22

[54] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding : An unsupervised and generative approach to clustering. *arXiv preprint arXiv :1611.05148*, 2016. 68, 74

[55] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 21

[56] Eunji Kim, Siwon Kim, Minji Seo, and Sungroh Yoon. Xprotonet : diagnosis in chest radiography with global and local explanations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15719–15728, 2021. x, 30, 31

[57] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014. 57, 100

[58] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014. 75

[59] Diederik P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. https ://github.com/insdout/GMVAE-pytorch, 2014. 79

[60] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013. 69, 71

[61] Rune Kjærsgaard, Ahcène Boubekki, and Line Clemmensen. Pantypes : Diverse representatives for self-explainable models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13230–13237, 2024. xi, 28, 66, 73, 74, 79, 94, 97

[62] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020. 25

[63] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 46

[64] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 79, 94

[65] Ludmila I Kuncheva and James C Bezdek. Nearest prototype classification : Clustering, genetic algorithms, or random search ? *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(1) :160–164, 1998. 26

[66] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006. 26

[67] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998. xi, 67, 68, 79, 94

[68] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert : A simple and performant baseline for vision and language. *arXiv preprint arXiv :1908.03557*, 2019. 22

[69] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes : A neural network that explains its predictions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 27, 64

[70] Stefan Liebner, Rick M Dijkhuizen, Yvonne Reiss, Karl H Plate, Dritan Agalliu, and Gabriela Constantin. Functional morphology of the blood–brain barrier in health and disease. *Acta neuropathologica*, 135 :311–336, 2018. 4

[71] Manhua Liu, Xudong Jiang, and Alex C Kot. A multi-prototype clustering algorithm. *Pattern Recognition*, 42(5) :689–698, 2009. 26

[72] Yu Liu, Lei Wang, Juan Cheng, Chang Li, and Xun Chen. Multi-focus image fusion : A survey of the state of the art. *Information Fusion*, 64 :71–91, 2020. 38

[73] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137, 1982. 26

[74] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 20

[75] Chiyu Ma, Brandon Zhao, Chaofan Chen, and Cynthia Rudin. This looks like those : Illuminating prototypical concepts using multiple visualizations. *Advances in Neural Information Processing Systems*, 36 :39212–39235, 2023. 28

[76] Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology : Challenges and opportunities. *Medical image analysis*, 33 :170–175, 2016. 29

[77] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv :1412.6632*, 2014. 21

[78] Winston Maxwell, Valérie Beaudouin, Isabelle Bloch, David Bounie, Stéphan Clémençon, Florence d'Alché Buc, James Eagan, Pavlo Mozharovskyi, and Jayneel Parekh. Identifying the'right'level of explanation in a given situation. In *Proceedings of the First International Workshop on New Foundations for Human-Centered AI (NeHuAI), Santiago de Compostella, Spain*, page 63, 2020. 2

[79] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. *arXiv preprint arXiv :2210.07183*, 2022. ix, 22

[80] Hashim Mir, Peter Xu, and Peter Van Beek. An extensive empirical evaluation of focus measures for digital photography. In *Digital Photography X*, volume 9023, pages 167–177. SPIE, 2014. 38

[81] Sanaz Mohammadjafari, Mucahit Cevik, Mathusan Thanabalasingam, and Ayse Basar. Using protopnet for interpretable alzheimer's disease classification. In *Canadian AI*, 2021. 30

[82] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation : an overview. *Explainable AI : interpreting, explaining and visualizing deep learning*, pages 193–209, 2019. 19, 28

[83] Meike Nauta, Ron Van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14933–14943, 2021. 63

[84] Sajid Nazir, Diane M Dickson, and Muhammad Usman Akram. Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks. *Computers in Biology and Medicine*, 156 :106668, 2023. 29

[85] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011. 79, 94

[86] Felix Neumaier, Boris D Zlatopolskiy, and Bernd Neumaier. Drug penetration into the central nervous system : pharmacokinetic concepts and in vitro model systems. *Pharmaceutics*, 13(10) :1542, 2021. viii, 3

[87] Harsh Panwar, PK Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, Prakhar Bhardwaj, and Vaishnavi Singh. A deep learning and grad-cam based color visualization approach for fast detection of covid-19 cases using chest x-ray and ct-scan images. *Chaos, Solitons & Fractals*, 140 :110190, 2020. 29

[88] Nikolaos Passalis and Anastasios Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5755–5763, 2017. 26

[89] Nikolaos Passalis and Anastasios Tefas. Neural bag-of-features learning. *Pattern Recognition*, 64 :277–294, 2017. 26

[90] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch : An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 57

[91] Miguel Pérez-Enciso and Laura M Zingaretti. A guide on deep learning for complex trait genomic prediction. *Genes*, 10(7) :553, 2019. viii, 12

[92] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise : Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv :1806.07421*, 2018. ix, 20

[93] Vitali Petsiuk, Rajiv Jain, Varun Manjunatha, Vlad I Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko. Black-box explanation of object detectors via saliency maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11443–11452, 2021. 20

[94] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 22

[95] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet : Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv :1711.05225*, 2017. 29

[96] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you ?" explaining the predictions of any classifier. In *Proceedings of the*

22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016. 20

[97] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *Ieee Access*, 8 :42200–42216, 2020. 10

[98] Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958. 11

[99] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5) :206–215, 2019. ix, 22, 23

[100] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3) :211–252, 2015. 2

[101] Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, pages 351–368. Springer, 2022. 63

[102] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protopshare : Prototypical parts sharing for similarity discovery in interpretable image classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1420–1430, 2021. 63

[103] Zohaib Salahuddin, Henry C Woodruff, Avishek Chatterjee, and Philippe Lambin. Transparency of deep neural networks for medical image analysis : A review of interpretability methods. *Computers in biology and medicine*, 140 :105111, 2022. 10

[104] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji : an open-source platform for biological-image analysis. *Nature methods*, 9(7) :676–682, 2012. 37

[105] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam : Visual explanations

from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 19

[106] Rui Shu. Gaussian mixture vae. https ://ruishu.io/2016/12/25/gmvae/, 2016. 68, 75, 79

[107] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks : Visualising image classification models and saliency maps. *arXiv preprint arXiv :1312.6034*, 2013. viii, 16, 18, 19

[108] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014. 21, 50

[109] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 27

[110] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose : a generalist algorithm for cellular segmentation. *Nature methods*, 18(1) :100–106, 2021. xi, 57, 58

[111] Alan B Tickle, Robert Andrews, Mostefa Golea, and Joachim Diederich. The truth will come to light : Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE transactions on neural networks*, 9(6) :1057–1068, 1998. 2

[112] Hugues Turbé, Mina Bjelogrlic, Gianmarco Mengaldo, and Christian Lovis. Protos-vit : Visual foundation models for sparse self-explainable classifications. *arXiv preprint arXiv :2406.10025*, 2024. 28

[113] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 110

[114] Bas HM Van der Velden, Hugo J Kuijf, Kenneth GA Gilhuijs, and Max A Viergever. Explainable artificial intelligence (xai) in deep learning-based medical image analysis. *Medical image analysis*, 79 :102470, 2022. 10, 29

[115] Yasemin Bozkurt Varolgüneş, Tristan Bereau, and Joseph F Rudzinski. Interpretable embeddings from molecular simulations using gaussian mixture variational autoencoders. *Machine Learning : Science and Technology*, 1(1) :015012, 2020. 68

[116] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 21

[117] Romain Versele, Emmanuel Sevin, Fabien Gosselet, Laurence Fenart, and Pietra Candela. Tnf-α and il-1β modulate blood-brain barrier permeability and decrease amyloid-β peptide efflux in a human blood-brain barrier model. *International Journal of Molecular Sciences*, 23(18) :10235, 2022. 34

[118] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 46, 68

[119] Chong Wang, Yuyuan Liu, Yuanhong Chen, Fengbei Liu, Yu Tian, Davis McCarthy, Helen Frazer, and Gustavo Carneiro. Learning support and trivial prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2062–2072, 2023. 73

[120] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable image recognition by constructing transparent embedding space. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 895–904, 2021. 73

[121] Jutta Willamowski, Damian Arregui, Gabriella Csurka, Christopher R Dance, and Lixin Fan. Categorizing nine visual classes using local appearance descriptors. *illumination*, 17 :21, 2004. 26

[122] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv :1708.07747*, 2017. xi, 74, 79, 80, 94

[123] Bin Yang, Shutao Li, and Fengmei Sun. Image fusion using nonsubsampled contourlet transform. In *Fourth International Conference on Image and Graphics (ICIG 2007)*, pages 719–724. IEEE, 2007. 38

[124] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016. 21

[125] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014. viii, 16, 17

[126] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 international conference on computer vision*, pages 2018–2025. IEEE, 2011. 16

[127] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8827–8836, 2018. 23

[128] Rong Zhang, Shuhan Tan, Ruixuan Wang, Siyamalan Manivannan, Jingjing Chen, Haotian Lin, and Wei-Shi Zheng. Biomarker localization by combining cnn classifier and generative adversarial network. In *International conference on medical image computing and computer-assisted intervention*, pages 209–217. Springer, 2019. ix, 29, 30

[129] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5) :726–742, 2021. 10

[130] Yuhui Zhang, Alyssa Unell, Xiaohan Wang, Dhruba Ghosh, Yuchang Su, Ludwig Schmidt, and Serena Yeung-Levy. Why are visually-grounded language models bad at image classification ? *Advances in Neural Information Processing Systems*, 37 :51727–51753, 2024. 22

[131] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. ix, 18, 19