Deep Learning pour l'identification de différences phénotypiques visuelles subtiles entre lignées neuronales, modèles de la maladie de Parkinson

Tiphaine Champetier

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Deep Learning pour l'identification de différences phénotypiques visuelles subtiles entre lignées neuronales, modèles de la maladie de Parkinson

Soutenue par
Tiphaine CHAMPETIER
Le 27 Mai 2020

École doctorale n°
École Doctorale Complexité du vivant ED515

Spécialité
Biologie computationnelle

Composition du jury :

M. Auguste GENOVESIO École Normale Supérieure (IBENS)	*Directeur de thèse*
M. Patrick BOUTHEMY Inria	*Président du jury et Rapporteur*
M. Matthias GROSZER Institut du Fer à Moulin	*Rapporteur*
Me Priscille BRODIN Institut Pasteur	*Examinatrice*
Me Elsa ANGELINI Imperial College	*Examinatrice*
Me Gaëlle BONCOMPAIN Institut Curie	*Examinatrice*
M. Peter SOMMER Ksilink	*Invité*

1794
ENS | ksilink

Remerciements

Il faut, paraît-il, tout un village pour élever un enfant... Je crois qu'il n'en faut pas moins pour accoucher d'une thèse. Je souhaiterais par ces mots, remercier mon village de m'avoir porté jusqu'ici.

Merci à Auguste de m'avoir permis de tenter ma chance dans un domaine nouveau pour moi, puis soutenu pendant ces années de découvertes. Je te remercie pour ta patience. Merci à Ksilink pour cette opportunité de thèse. Je remercie Yong-Jun Kwon et Peter pour l'aide que vous m'avez apportée pour comprendre la complexité des expériences menées.

Merci à l'îlot biocomp qui a vu bien des arrivées et des départs durant ces 3 dernières années, mais qui est resté un havre de fraîcheur où je me suis toujours plu à retourner.
Je pense à Alice, Shihav et Sreetama qui m'ont accueillie à la veille de leur départ : que de rires, déjà ! (une version de "savez vous planter les choux" en hindi est en cours de traduction).
Merci à Felipe, tu as notamment été une ouverture sur un autre monde : celui du design Brésilien, une niche exceptionnelle.
Merci à Benoît, pour les pauses escalade, en compagnie de France !
Merci à Toni et Nikita, la fine équipe, professionnels du selfie, à la page sur les réseaux, et toujours prêts à l'écoute.
Merci à Solène et Elise, pour vos discussions du soir et aussi Aurélien et Ayush, j'ai adoré jouer en votre société.
Merci à Amira et Ouardia, pour les moments de qualité passés en votre présence (les autres sont moins quali ;)).
Merci à Maxime pour l'expertise programmation, le smile, et les parties de baby.
Merci à Raphaël pour ton sarcasme, et ta bonne humeur discrète, prête à jaillir à tout moment.
Merci à Mathieu, toujours très ouvert à une tentative de bash-ing et clef de voûte de l'équipe ! Que de rire, encore !
Merci à Alexis, pour ton humour fin, pour ta tranquillité, pour nos discussions tant scientifiques que méditatives (ouai ouai, on a grave médité).
Merci à DD et France pour votre soutien indéfectible, pour m'avoir prouvé que j'étais mauvaise observatrice, et pour nos rires, toujours. Mention spéciale découverte des neurchis et psychologue pour DD. Et mention spéciale coach végétarianisme et féminisme pour France ! !

Merci à la team du 2e, Glouvel, Swann, Lambert, et Elise, toujours un plaisir de partager avec vous des spaghettis bolognaises végétariennes cannelle coriandre.

Merci à Valou, Zeina, Loïc, et Léa, partenaires Ksilink et bien plus encore !
Merci à Pierre pour ton soutien, tes bons conseils de programmation et nos vacances.
Merci à Thomas et les autres, Ludo, Mélo, Manon et Adri, une bouffée d'airs musicaux, hebdomadaire, et salvatrice !
Merci à la team d'ultimate frisbee pour la seconde bouffée d'air hebdomadaire, polluée par les boulettes synthétiques que nous avalions à chaque cut banane. Heureusement, le fleurus nous aidait à digérer le tout.
Merci aux zouzies, Paupy, Amély, Audy, Mishky, Amby, Mélody et Josy et aussi à Cypy, zouzie qui s'ignore.
Merci à ma famille, vous qui m'avez soutenu sans relâche dans mon entreprise de thèse ! Merci à maman pour ta préoccupation constante à mon bien-être et à papa pour ton émerveillement sur chacune de mes maigres réalisations.
Merci à Moby, pour le nouveau chemin que tu ouvres devant moi d'un scoober décidé.
Merci à Adrien, pour Twin Peaks, les chouquettes, les débats, les analyses MBTI, Didier, l'ameublement du QG, le soin porté aux plantes (3e basilic qui a rendu l'âme...), les massages, le coaching de la garagiste, les marinades, bref, notre amitié...

Un grand merci à mes chers relecteurs, Adrien, France et Alexis. Sans vous, ce manuscrit aurait été illisible.

Je suis heureuse d'avoir été aussi bien entourée. L'écriture de ces mots me le révèle enfin : ces trois années passées s'enveloppent déjà d'un doux écrin de mélancolie.

Abstract

High-content screening has experienced a significant growth since the mid-2000s. This technology is of primary interest to the pharmaceutical industry as it allows in principle the discovery of therapeutic molecules for diseases whose molecular pathways are poorly identified. Until now, a measurable cell phenotype must first be identified in order to evaluate the effect of a compound library on it. From an image analysis point of view, treated cells are automatically detected in hundreds of thousands of images and measurements of descriptors allow to finely differentiate effective treatments from a negative control. In collaboration with the companies Ksilink and Sanofi, we have been confronted with a new type of high-content screen to identify compounds effective against Parkinson's disease. This one, presenting images of neurons, made analysis dependent on robust cell segmentation obsolete. In addition, using cell differentiation and genome editing techniques, the controls of the cell model are two neuron cultures, isogenic but for one mutation : the GS2019 mutation that induces the disease. Nevertheless, the heterogeneity of these complex cells and the fine differences between the two isogenic lines do not allow the human eye to identify two distinct phenotypes. In order to allow the automatic detection of differences between phenotypes, we have proposed to use deep learning approaches. This work was mainly divided into two steps. The first step consisted in identifying a network architecture capable of classifying neural images. We learned that neuron cultures show phenotype differences in a very heterogeneous and therefore not systematic way. The second step consisted in proposing methods to explain and interpret subtle differences in phenotype, to ensure that the screening is performed on the basis of a proven difference between phenotypes and not on the basis of a technical bias. Based on the premise that differences between neuronal phenotypes are difficult to visually apprehend between two different images due to the high natural variability of neurons, we propose the idea that transforming the same image from one phenotype to another may represent an interesting approach. Indeed, we show that it is possible to train antagonistic networks to transform an image of a neuron that does not carry the mutation into a neuron that carries it and vice versa. In this way, we have been able to highlight potential assay biases but also what we believe to be true morphological differences related to the pathological mutation that were previously invisible.

Résumé

Le criblage à haut contenu connaît un essor important depuis le milieu des années 2000. Cette technologie est d'un intérêt primordial pour l'industrie pharmaceutique car elle permet en principe la découverte de molécules à visée thérapeutique pour des maladies dont les voies moléculaires sont mal identifiées. Jusqu'à présent un phénotype cellulaire mesurable doit au préalable être identifié afin d'évaluer l'effet d'une banque de composés sur celui-ci. Du point de vue de l'analyse d'images, les cellules traitées sont détectées automatiquement sur des centaines de milliers d'images, et des mesures de descripteurs permettent de différencier finement les traitements effectifs par rapport à un contrôle négatif. En collaboration avec les entreprises Ksilink et Sanofi, nous avons été confrontés à un nouveau type de crible à haut contenu pour identifier des composés efficaces contre la maladie de Parkinson. Celui-ci, présentant des images de neurones, rendait caduque l'analyse dépendant d'une segmentation robuste des cellules. Par ailleurs, grâce à des techniques de différentiation cellulaire et d'édition de génome, les contrôles du modèle cellulaire sont deux cultures de neurones, isogéniques à une mutation près : la mutation G2019S induisant la maladie. Néanmoins, l'hétérogénéité propre à ces cellules de forme complexe, et les différences fines entre les deux lignées isogéniques ne permettent pas à l'oeil humain l'identification de deux phénotypes distincts. Dans le but de permettre la mise en évidence automatique des différences entre phénotypes, nous avons proposé d'employer des approches de deep learning. Ce travail s'est décomposé principalement en deux étapes. La première étape a consisté à identifier une architecture de réseau capable de classifier des images de neurones. Nous avons appris que les cultures de neurones montraient des différences de phénotype de manière très hétérogène, et donc pas systématique. La deuxième étape a consisté à proposer des méthodes pour expliquer et interpréter les différences subtiles de phénotype, ceci afin de s'assurer que le crible soit effectué sur la base d'une différence avérée entre phénotypes et non sur la base d'un biais technique. Partant du principe que les différences entre phénotypes neuronaux entre deux images différentes sont difficiles à appréhender visuellement, du fait de la grande variabilité naturelle des neurones, nous proposons l'idée que la transformation d'une même image d'un phénotype à un autre peut représenter une approche intéressante. En effet, nous montrons qu'il est possible d'entraîner des réseaux antagonistes à transformer une image de neurone non porteur de la mutation en neurone porteur et inversement. De cette façon, nous avons pu mettre en évidence des potentiels biais de l'assay mais également ce que nous pensons être de véritables différences morphologiques liés à la mutation pathologique auparavant invisibles.

Sommaire

Chapitre 1

Introduction

1.1 Les maladies neurodégénératives

1.1.1 Une pandémie

Soigner les maladies neurologiques est un grand défi de notre siècle. Avec une population vieillissante, on observe une augmentation de la prévalence des maladies neurodégénératives. Celles-ci impactent non seulement les malades, qui voient leurs capacités cérébrales et motrices dégradées, mais aussi leur famille. C'est sur celle-ci que retombe la lourde tâche de s'occuper d'un être dépendant que l'on ne reconnaît parfois plus. D'autre part, les coûts pour la société sont énormes avec, pour la maladie d'Alzheimer seule, un montant estimé à 19 milliards d'euros [1] par an (coûts médicaux et d'aide informelle des proches). Pour comparaison, le déficit du système de retraite français devrait atteindre 17 milliards en estimation haute [2].

1.1.2 La réponse pharmaceutique

Les maladies neurodégénératives sont pandémiques. Les démences séniles dont la maladie d'Alzheimer touchent un million de personnes en France, la maladie de Parkinson, 160 000 [3]. Malgré le marché énorme que ces maladies représentent, l'industrie pharmaceutique peine à répondre à la demande de traitement. En effet, la maladie de Parkinson est caractérisée par une dégénérescence des neurones dopaminergiques dans certaines parties du cerveau qui participent au contrôle moteur. Alors qu'au niveau macroscopique, cette dégénérescence est responsable des symptômes moteurs bien connus de la maladie, les tremblements, la rigidité musculaire et la lenteur des mouvements, au niveau microscopique, la dégénérescence neuronale entraîne une chute de la présence de dopamine au sein de ces zones cérébrales. Ainsi, les traitements classiques visent à fournir aux zones du cerveaux touchées par la diminution neuronale des précurseurs de la dopamine, afin que l'activité motrice puisse perdurer. Seulement, l'état du malade se détériore très rapidement lorsque, malgré la compensation dopaminergique, la disparition neuronale est telle que les neurones restant ne suffisent plus à assurer les fonctions vitales. Ces traitements sont dit symptomatiques : ils ne guérissent pas de la maladie, ni la ralentissent. La lente destruction suit son cousr [4].

1.2 Pourquoi n'existe-il pas de médicament curatif à ce jour ?

1.2.1 La stratégie de recherche dite de criblage à haut-débit.

La productivité de la R&D est essentielle au secteur pharmaceutique. Un médicament va mettre aux alentours de 12 ans à être développé à partir de sa découverte, et la propriété intellectuelle qui lui est associée n'est protégée qu'une vingtaine d'années [5]. De la capacité de l'entreprise à produire de nouvelles molécules thérapeutiques va dépendre sa survie.

Depuis les années 1990 et l'avènement de la génomique, l'obsession de la recherche pharmaceutique est la cible thérapeutique [6]. La cible est typiquement une protéine, ayant un rôle clef dans le développement moléculaire de la maladie. On pense également que pour rétablir une physiologie plus saine chez le patient, on pourra modifier son comportement. A cette fin, on cherche à attacher une molécule, futur médicament, sur la cible. Cette molécule va, soit empêcher d'autres molécules de s'attacher à l'endroit où elle se trouve, ou bien obliger la protéine cible à adopter certaines conformations, ou encore augmenter son activité. La méthode la plus populaire de découverte de ces molécules thérapeutiques est le high-throughput screening ou criblage à haut-débit.

Le criblage à haut-débit est une méthode expérimentale, utilisée en biologie fondamentale et en recherche pharmaceutique. Elle permet de réaliser des centaines, milliers, voire millions de tests en parallèle sur des plaques de culture multi-puits, chaque puits étant une expérimentation indépendante non-biaisée par l'expérimentateur. Cette méthode a vu le jour grâce à la robotisation du maniement des plaques de culture et de l'injection des liquides en parallèle.

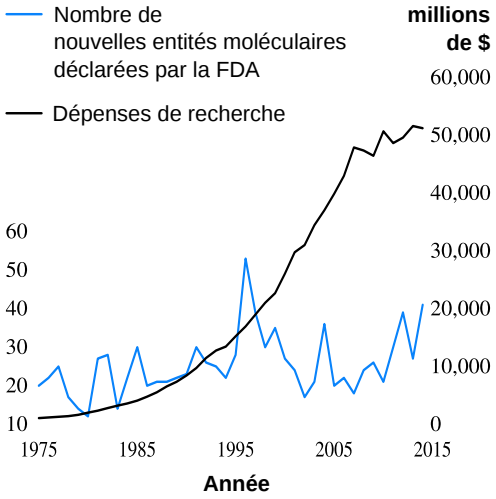


FIGURE 1.1 – **Nombre de nouvelles molécules thérapeutiques approuvés par la FDA (Agence fédérale américaine pour les médicaments) et dépenses de recherche pharmaceutique par année.** Adaptée de [7]

Grâce au criblage haut-débit, les entreprises pharmaceutiques criblent des millions de molécules, dans l'espoir de trouver celle qui aura le mécanisme voulu sur la cible thérapeutique, mais aussi le moins d'effet indésirable. On appelle ce type de crible *target-based* ou orienté cible, car on ne teste pas la molécule dans une cellule, mais seulement au contact de la cible moléculaire.

1.2.2 Augmentation du coût de la recherche et fort taux d'attrition des molécules-candidates tout au long des phases de développement.

Le criblage moléculaire à haut-débit a pris de plus en plus de place dans le paysage de la recherche pharmaceutique. Néanmoins, malgré un investissement exponentiel dans les années 1990-2010, le nombre de nouvelles molécules thérapeutiques a stagné (voir Fig. 1.1). Les potentiels médicaments ne passent généralement pas les stades de test clinique (chez l'homme) pour 2 principales raisons : soit ils ne sont pas assez effi-

caces, soit ils présentent un danger pour la santé. Ce taux d'échec élevé a soulevé de nombreuses hypothèses et remises questions des stratégies adoptées par l'ensemble du secteur. Certains y voit le résultat d'un durcissement du processus de validation des médicaments par les autorités régulatrices dont fait partie la FDA, d'autres le reflet de la translation vers des projets à haut risque mais à potentiel de marché très rémunérateur [8]. En effet, les domaines pour lesquels aucun médicament n'est sur le marché, profitent d'une moins grande concurrence, mais sont aussi ceux les plus à risque. Ces médicaments qui n'ont aucun précédent ciblant la même protéine sont appelées "first-in-class". D'autres encore, pensent que c'est le choix du paradigme de recherche qui est à mettre en cause [6]. En effet, la méthode du criblage à haut débit nécessite de connaître à fond les processus pathologiques au niveau moléculaire afin de pouvoir déterminer la bonne cible thérapeutique. Or, il existe énormément de maladies pour lesquelles le savoir n'est pas complet. Dans cette situation, cibler une protéine en particulier peut être une gageure qui coûte plusieurs centaines de millions avec des chances de réussite très faibles. En effet, rien ne dit que la cible et le mécanisme d'action choisi sont les meilleurs possibles pour modifier le processus pathologique, et ce sans effet secondaire.

Pour ce qui est de la recherche pharmaceutique relative au système nerveux central, et particulièrement des maladies neurodégénératives comme Alzheimer ou la maladie de Parkinson, les causes sont très probablement multifactorielles et complexes. Les mécanismes moléculaires de la maladie ne sont pas non plus compris avec précision [9]. Et parmi les intuitions que l'on peut avoir, quelle cible vaut-il mieux privilégier ?

1.2.3 Alternative de paradigmes ou techniques complémentaires ? Le criblage à haut-débit et le criblage phénotypique.

Le phénotype correspond aux caractéristiques observables d'un organisme, par exemple la forme d'une cellule, la température d'un être humain etc... Avant l'avènement du criblage haut-débit et orienté cible, la recherche pharmaceutique était basée sur des approches phénotypiques. Une molécule est sélectionnée sur le phénotype qu'elle confère aux cellules ou organisme lorsqu'elle est administrée. Dans les années 1990, la robotisation des systèmes de culture, l'apparition des caméras CCD, l'augmentation des capacités de stockage de données, la diversification des fluorophores et l'amélioration des microscopes et leur robotisation, ont permis le développement du criblage à haut-débit dit phénotypique. Au lieu de mesurer l'interaction d'une bibliothèque de molécules sur une molécule cible, on image la réaction de cellules au contact des molécules de la bibliothèque.

Le grand avantage du criblage phénotypique est qu'il n'y a pas d'*a priori* sur le mécanisme moléculaire. On teste les molécules sur des cellules, et on cible un phénotype donné. Dans leur revue de 2011, Swinney et Antony [6] étudient les stratégies qui ont été utilisées pour trouver les nouveaux médicaments de 1999 à 2008 (voir figure 1.2). Dans le cas des petites molécules *small-molecule drugs*, la grande majorité des médicaments "first-in-class" sont trouvées par des méthodes de criblages phénotypiques. Inversement, les méthodes de criblages moléculaires à haut-débit sont

plus efficaces pour trouver des "followers", molécules qui agissent sur la même cible que d'autres médicaments, précédemment approuvés par une agence de régulation.

Les deux avantages de chacune des méthodes apparaissent assez clairement. Tandis que le criblage moléculaire ou orienté cible montre une grande capacité à améliorer l'efficacité des drogues pour un mécanisme moléculaire d'action reconnu comme étant pertinent thérapeutiquement, le criblage phénotypique, de son côté, permet de se soustraire à l'hypothèse biologique trop contraignante que représente la cible thérapeutique lorsque les connaissances scientifiques sont trop incomplètes, ou que les causes sont multifactorielles [6, 10, 11].

1.3 Quel modèle cellulaire pour le criblage phénotypique ?

Les questions centrales au criblage phénotypique sont celles du modèle cellulaire et de l'essai biologique. Selon Moffat et al [10], pour s'assurer du succès d'un crible phénotypique, il faut veiller à ce que la chaîne de traduisibilité du crible soit pertinente. En d'autre termes, l'essai biologique doit être le plus prédictif possible du potentiel des molécules testées. A cette fin, le modèle de la maladie, la lecture de l'essai et la biologie de la maladie chez l'humain doivent partager des mécanismes de base. Par conséquent, quel modèle cellulaire serait meilleur que les propres cellules de patient pour récapituler sa maladie ?

Ksilink, partenaire pour l'élaboration de cette thèse, est un institut de recherche public-privé, expert dans les technique de recherche de médicament "first-in-class" par screening à imagerie haut-débit et donc phénotypique. Ils utilisent des techniques de pointe pour modéliser les maladies.

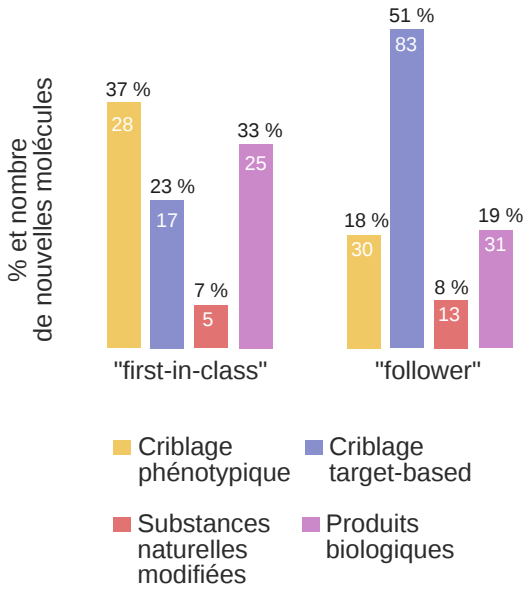


FIGURE 1.2 – **Répartition des nouveaux médicaments découverts entre 1999 et 2008, selon la stratégie de recherche et par type de médicament.** Les médicaments *"first-in-class"* sont les premiers à cibler la protéine sur laquelle ils s'attachent avec leur mécanisme moléculaire d'action propre. Les médicaments dits *"followers"* ne sont pas les premiers dans leur classe, il y a déjà eu des molécules associées à leurs cibles et mécanismes d'action. Au cours de la décade étudiée, le crible cellulaire phénotypique est la méthode qui a le plus réussi pour les médicaments *"first-in-class"*. Quant à la méthode de crible moléculaire à haut-débit, c'est la méthode de référence pour les médicaments *"followers"*. Adaptée de [6]

En vue de réaliser un crible phénotypique de la maladie de Parkinson, l'entreprise a obtenu un modèle cellulaire composé de 2 lignées de neurones isogéniques (possédant le même génome) à l'exception d'une mutation. En effet, les 2 lignées ont été créées à partir de fibroblastes (cellules de peau) d'un patient parkinsonien porteur de

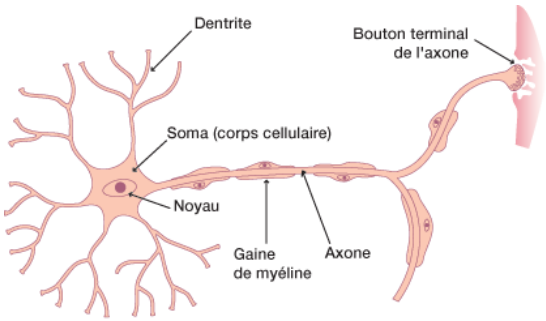


FIGURE 1.3 – **Schéma d'un neurone.**

la mutation G2019S sur le gène codant pour la protéine LRRK2. Grâce à des techniques de reprogrammation cellulaire [12], ces cellules de peau ont ensuite été induites en cellules souches pluripotentes (iPSC pour induced Pluripotent Stem Cells). Les iPSC ont été séparées en 2 groupes, pour l'un d'entre eux, on a recouvert le génotype sain, en annulant la mutation grâce à l'outil d'édition génomique CRISPR-cas9 [13]. Ce génotype est dit *wild-type* ou WT [1]. Par la suite, les 2 lignées d'iPSC sont différenciées en cellules progénitrices de neurones (NPCs pour Neuronal Progenitor Cells).

Les NPCs seront cultivées et différenciées en neurones dopaminergiques. Le schéma, figure 1.3 expose les structures cellulaires d'un neurone. Après la différenciation, la culture est maintenue durant 30 jours. Puis les neurones sont congelés, envoyés à Ksilink, puis décongelés et cultivés pendant 7 jours. Les neurones seront cultivés sur des plaques de cultures contenant 384 puits. Pour réaliser les images des cellules, des marqueurs fluorescents de diverses molécules sont ajoutés (voir figure 1.4). Les 3 marqueurs ciblent : les noyaux des cellules, avec le marqueur DAPI, la tyrosine hydroxylase, spécifique aux neurones dopaminergiques, et enfin l'alpha-synucléine, molécule dont l'agrégation serait une des causes de la maladie. Il a notamment été prouvé que son accumulation a lieu dans des neurones dopaminergiques dérivés d'une patiente porteuse de la même mutation que celle étudiée par l'entreprise Ksilink [14]. Les contrôles négatifs dans cette étude, sont des neurones dopaminergiques dérivés d'iPSCs d'une donneuse en bonne santé, les lignées ne sont donc pas isogéniques.

Dans le cas de criblage à deux contrôles (positif et négatif), dans le même temps que les images sont générées, l'équipe doit s'assurer de pouvoir mesurer les différences de phénotypes cellulaires. Divers paramètres expérimentaux peuvent être testés pour optimiser cette mesure. La robustesse du modèle cellulaire doit être également testée.

1.4 Analyse des cribles à haut-contenu

1.4.1 Analyse traditionnelle

Une expérience de criblage à haut-contenu se traite par analyse automatisée d'images, pour parvenir à traiter les milliers d'images produites. Dans le cas de l'analyse sans deep learning, un phénotype doit être identifié pour pouvoir être détecté par des mesures *ad hoc*. L'analyse d'images débute généralement par une détection des cellules. Des calculs de descripteurs sont ensuite développés spécifiquement pour différencier finement les traitements positifs par rapport au contrôle négatif.

1. Car c'est le génotype de plus grande abondance dans la nature

1.4.2 Problématique de l'analyse du modèle cellulaire de Parkinson

Avec le modèle cellulaire neuronal de la maladie de Parkinson, deux problèmatiques émergent.
Premièrement, les cellules neuronales ne sont pas segmentables : une analyse reposant sur des descripteurs de cellules uniques ne convient donc plus. Deuxièmement, le phénotype n'est pas connu a priori, et l'on aimerait détecter de façon non-biaisée les différences subtiles entre les deux lignées neuronales isogéniques. La détection de ces différences paraît ardue, étant donnée l'hétérogénéité qui caractérise les cultures de neurones.

Les travaux présentés dans ce manuscrit ont proposé de mettre à profit les approches de traitement d'image par apprentissage de réseau profond pour révéler les différences phénotypiques subtiles. Ces différences permettent de différencier les neurones provenant d'un patient Parkinsonien, porteur de la mutation LRRK2-G2019S, ou de neurones à génotype corrigé.

1.5 Plan de la thèse

Pour répondre à ce besoin d'un nouvel outil d'analyse phénotypique, nous avons employé des techniques de classification d'images, de génération de phénotype, ainsi que des outils de clustering. La figure 1.5 présente le schéma récapitulatif de ces techniques et de la façon dont on les a agencées pour obtenir des solutions à la problématique. Cette thèse se développe à travers cinq chapitres. Le Chapitre 2 sera consacré à l'état de l'art. Au sein du Chapitre 3, nous montrerons qu'il est possible de distinguer les phénotypes proches des deux lignées cellulaires, utilisées par nos partenaires industriels de chez Ksilink. Nous y développons un outil de visualisation spatiale de phénotype. Dans le Chapitre 4, nous détournons un outil de transformation d'images pour mettre en lumière des différences subtiles des phénotypes. Au cours du cinquième chapitre, nous étudions la possibilité de représenter les phénotypes des cultures WT et LRRK2-G2019S par des sous-populations définies à partir de descripteurs appris automatiquement.

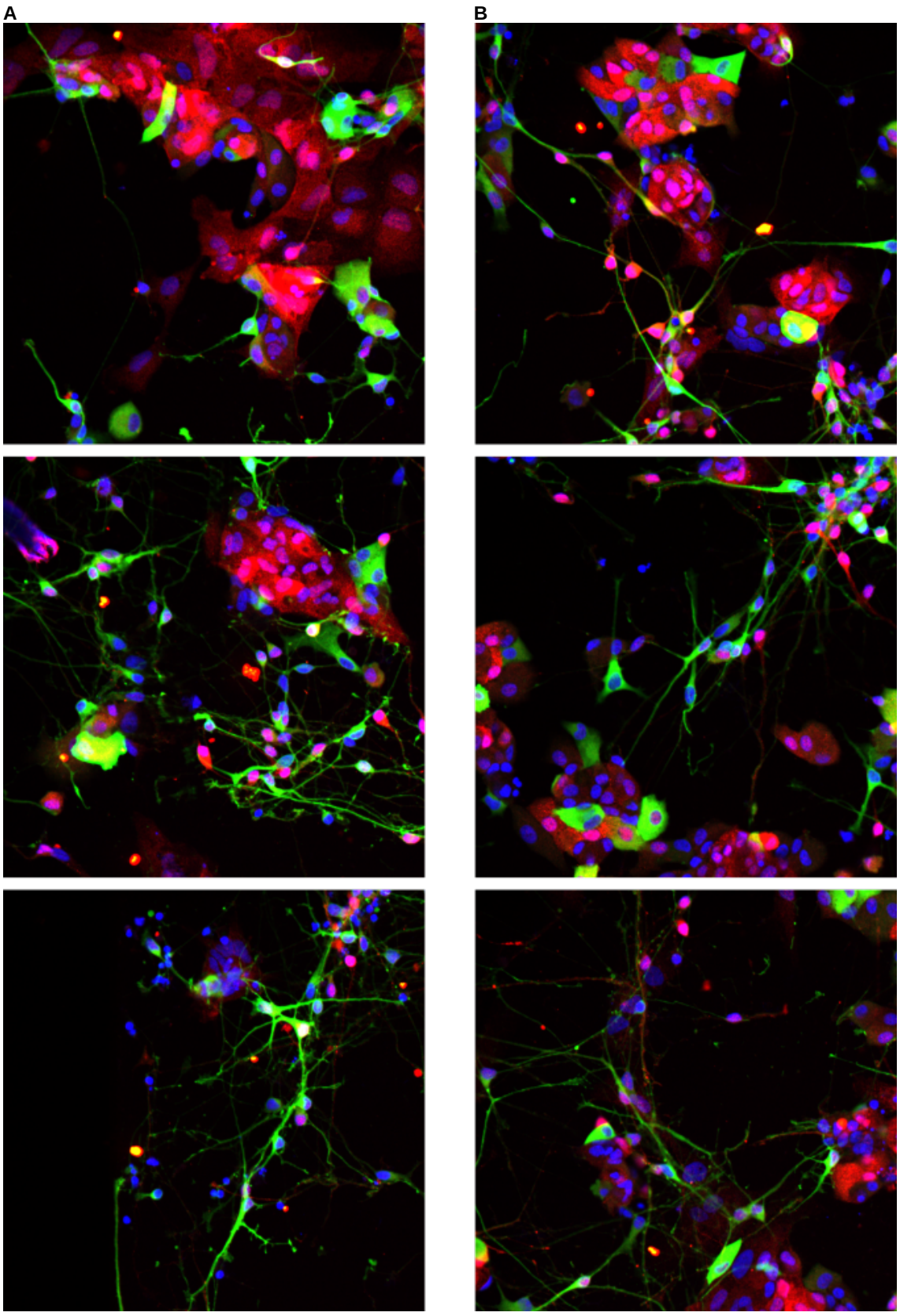


FIGURE 1.4 – **Exemple d'images du dataset de LRRK2.** Images de culture de neurones obtenus par différenciation à partir de cellules souches. Les marqueurs sont : en bleu, DAPI, marqueur du noyau ; en vert, un marqueur de TH, tyrorine hydroxylase, caractéristique des neurones dopaminergiques (les iPSCs n'en ont pas) ; en rouge, un marqueur de l'alpha-synucléine, protéine s'agrégeant dans les neurones en dégénérescence. (A) cellules WT (wild type : la mutation a été corrigée par CRISPR/cas9). (B) cellules porteuses de la mutation G2019S sur le gène LRRK2.

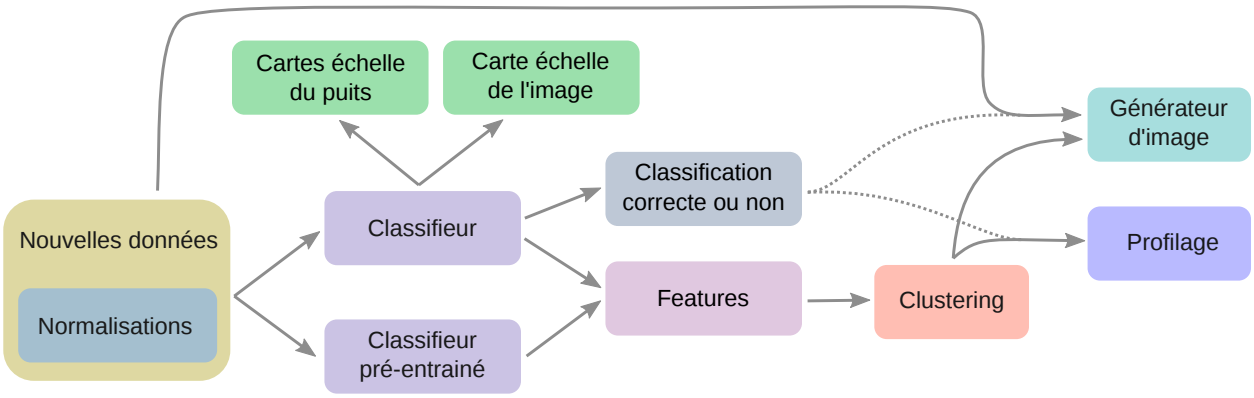


FIGURE 1.5 – **Schéma récapitulatif des divers solutions explorées en vue de la compréhension des phénotypes.**
1. L'exploitation des données commence par la normalisation des images ou des features extraites.
2. La question de la classification des images par classe (contrôle positif et contrôle négatif) a été abordée. Lorsque la classification est réalisée à l'aide de réseaux convolutionnels, différentes analyses de phénotypes spatiaux ont été réalisées : les cartes de phénotypes (blocs verts).
3. On peut extraire des features pour chaque image, les regrouper en cluster en vue de réaliser du profiling.
4. Enfin, j'ai utilisé des outils de génération d'image en vue d'expliquer les différences de phénotype.

Chapitre 2

État de l'art

Dans ce chapitre, on se propose d'abord de donner aux travaux de cette thèse un contexte sociétal et biologique, celui des maladies neuro-dégénératives. Dans le cadre de la maladie de Parkinson, la grande complexité des mécanismes moléculaires explique les difficultés rencontrées dans la recherche d'un traitement. Face à cette complexité, les industriels pharmaceutiques ont développé des stratégies de criblage à haut-contenu. Les modèles cellulaires utilisés dans ces expériences de criblage contiennent des différences phénotypiques subtiles. Pour distinguer ces différences, les méthodes d'apprentissage profond apparaissent comme un outil pertinent, efficace, mais surtout non-biaisé, pour appréhender de larges jeux de données contenant des millions d'images. Les dernières parties de ce chapitre, sont consacrés à la description des outils de génération et de transformation d'images, puis des algorithmes de clustering, qui permettent d'appréhender au mieux l'hétérogénéité cellulaire, par la création de sous-populations au sein du jeu de données.

2.1 La maladie de Parkinson

La maladie de Parkinson affecte 1% des plus de 60 ans en France [15]. C'est une maladie à évolution lente, dont les symptômes apparaissent progressivement sur plusieurs années. Les symptômes classiques sont moteurs et incluent les tremblements, la rigidité et la lenteur des mouvements. Le diagnostic clinique repose sur eux. Ils résultent de la déficience de neurones dans la région de la *susbstantia nigra* ; cette zone cérébrale est responsable du contrôle de l'équilibre et du mouvement. Cependant, avant l'apparition des symptômes moteurs, les patients peuvent développer des symptômes cognitifs et comportementaux comme l'apathie, la somnolence, des troubles de l'humeur mais aussi la constipation et l'anosmie (perte de l'odorat).

Trouver un traitement pour la maladie constitue un double défi. Le premier réside dans la détection des premiers stades de la dégénérescence chez les patients. En effet, lorsque les symptômes moteurs s'installent, la destruction neuronale est déjà bien avancée (près de 80% des neurones de la *substantia nigra* [16]). Pour s'assurer de l'efficacité des traitements, les patients doivent être pris en charge avant d'atteindre ce stade clinique. Le deuxième défi est le traitement curatif de la neuro-dégénérescence. À ce jour, les traitements existants sont symptomatiques. En effet, la destruction des neurones dopaminergiques entraîne une baisse de la production d'un neurotransmetteur essentiel au cerveau, la dopamine. Les traitements existants n'empêchent pas la perte neuronale. En revanche, ils compensent leur disparition en mettant à disposition du cerveau des précurseurs de la dopamine ou bien des molécules *agonistes*, c'est-à-dire ayant le même fonctionnement que la dopamine sur les récepteurs de celle-ci.

Les différentes voies cellulaires sous-jacentes à la maladie sont complexes, s'entremêlent, et parfois s'entretiennent selon des cercles vicieux (voir figure 2.1). Leurs détails moléculaires ne sont d'ailleurs pas parfaitement décryptés. Il existe cependant de nombreuses études *in vivo* et *in vitro* qui viennent supporter les hypothèses exposées ci-dessous [9]. On notera que les résultats obtenus *in vitro* et chez l'animal ne sont pas directement transposables chez l'humain.

La plupart des maladies neuro-dégénératives sont des protéinopathies : le mauvais repliement de certaines protéines entraîne leur accumulation

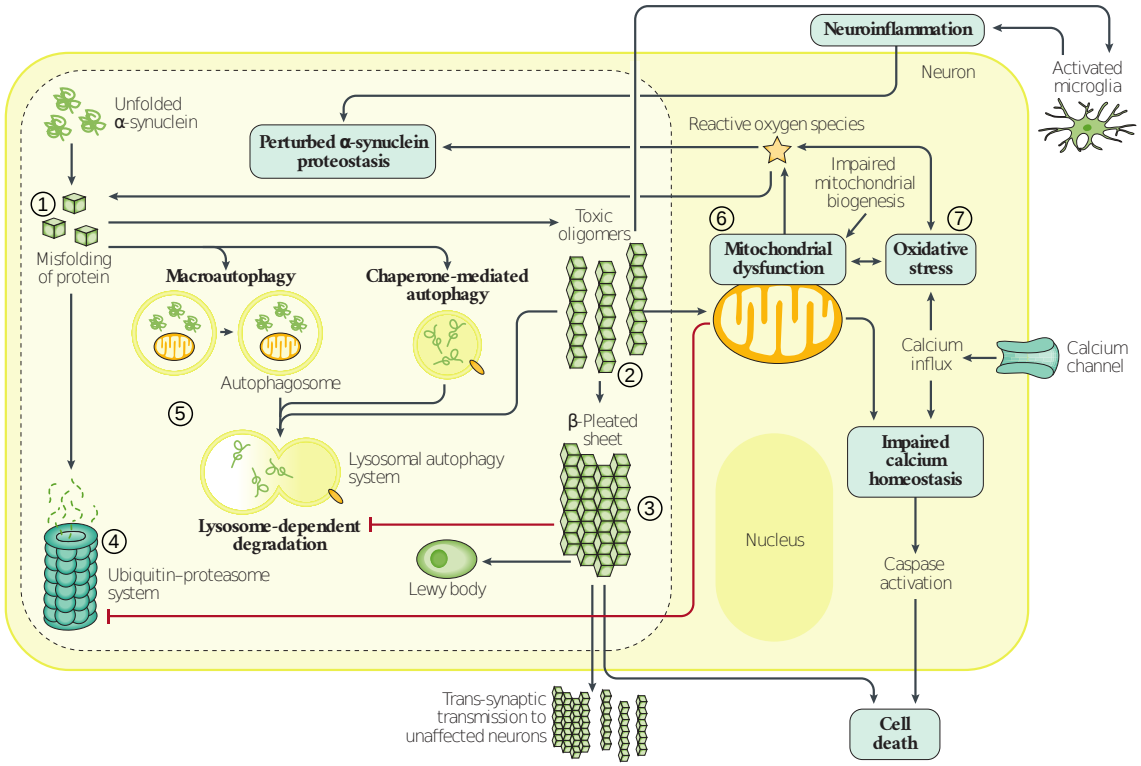


FIGURE 2.1 – **Complexité des mécanismes cellulaires impliqués dans la maladie de Parkinson.** Cette figure a été reproduite depuis *Nature reviews Disease primers* [9].

sous forme d'agrégats toxiques [17]. La maladie de Parkinson est associée à l'agrégation de la protéine alpha-synucléine par suite de son mauvais repliement. Ces agrégats insolubles peuvent prendre quatre formes différentes : des corps de Lewy (structures rondes dans le cytoplasme), des dépôts granuleux plus diffus, des dépôts dans les dendrites et les axones neuronaux, et enfin des structures extra-cellulaires [9].

Les implications de ces agrégats sur les processus cellulaires sont variés.
L'accumulation d'alpha-synucléine devient toxique lorsque les protéines mal repliées (1) s'agrègent en oligomères (2). Ces grandes molécules auto-entretiennent leur accumulation en induisant l'inflammation neuronale qui contribue au mauvais repliement de l'alpha-synucléine.
Les oligomères (2) peuvent également s'associer en feuillets β (3). Ceux-ci peuvent être excrétés et transmis à des cellules voisines, contribuant à la propagation lente de la maladie à travers le cerveau. En outre, les feuillets ou fibrilles (3) ont un effet inhibiteur sur les processus de dégradation des protéines mal repliées et des oligomères, contribuant ainsi par un cercle vicieux à leur accumulation [9]. En effet, pour détruire ces protéines mal repliées et ces oligomères toxiques, la cellule possède l'ubiquitine-protéasome (4) et le processus d'autophagie lysosomale (5). L'accumulation de ces protéines pourrait être reliée au ralentissement de ces deux activités cellulaires. Cette hypothèse est cohérente avec le fait que l'âge est le principal facteur de risque de la maladie et que le vieillissement est un facteur de ralentissement de ces deux activités [9].
Les oligomères (2) peuvent aussi provoquer des dysfonctionnements mitochondriaux (6) qui participeraient au

développement de la maladie. Une de leurs conséquences sur les neurones est une augmentation du stress oxydant (6). Or les neurones dopaminergiques de la *subtantia nigra* y sont particulièrement sensibles [18]. En effet, une grande dépense énergétique est nécessaire pour entretenir leur axone à arborescence fournie. Cette énergie est aussi nécessaire pour maintenir la régulation de leur rythme interne, à base d'oscillations de concentration de calcium. Les neurones dopaminergiques doivent aussi contrer l'effet oxydant de la dopamine ainsi que de ses métabolites. Enfin, la combinaison des anormalités mitochondriales et du stress oxydant peut contribuer à l'épuisement du système autophagie-lysosome (5).

Pour comprendre les mécanismes à l'œuvre dans la pathogenèse, il a notamment été utile de s'intéresser aux formes héréditaires de la maladie. De nombreuses mutations sont associées à la maladie, pour autant, seules 5% à 10% des formes de la maladie sont héritées, ainsi de nombreuses mutations correspondent à des facteurs de risque [4].

L'une de ces mutations est la mutation G2019S sur le gène encodant la protéine LRRK2 (leucine-rich repeat kinase 2). Elle représente 5% des formes familiales de Parkinson et 1 à 2% des formes sporadiques, c'est-à-dire chez un patient n'ayant aucun membre de sa famille atteint par la maladie [19]. Par ailleurs, le phénotype clinique des patients porteurs de la mutation est très similaire aux cas sporadiques non explicable par des mutations.
Biologiquement, elle est associée à l'altération d'une des deux voies de dégradation des protéines et en particulier de l'alpha-synucléine. Cette mutation accélère l'accumulation d'alpha-synucléine dans les fibrilles chez les neurones cultivés et exposés à ceux-ci [20]. Des études suggèrent également que la mutation est liée à des anormalités de fonctionnement de la mitochondrie [21]. C'est sur cette mutation qu'est basé le modèle cellulaire de Ksilink, entreprise partenaire.

2.2 Deep learning pour classification

Le *deep learning* ou apprentissage profond est une sous-division du machine learning ou apprentissage automatique.

2.2.1 Machine learning

Ce dernier consiste à apprendre à un ordinateur à résoudre une tâche définie par l'être humain. Par exemple, l'ordinateur peut apprendre à prédire le prix d'une maison en fonction de ses caractéristiques ou encore à décider si un email est un spam ou non. Dans le premier cas, la valeur à retrouver est continue, c'est un prix, il s'agit donc d'une régression ; dans le second cas, la valeur est catégorielle, on cherche à retrouver un label, c'est une classification. Dans les deux cas, on va s'aider de descripteurs d'"entrée", connus pour chacune des maisons ou des mails, pour prédire la "sortie". On appelle également ces descripteurs des *features*, ce sont des caractéristiques mesurables ou calculables pour chaque élément. Peuvent être des features, pour la maison, le nombre de mètres carrés de sa surface, sa distance au centre-ville, la moyenne des prix des maisons aux alentours, ou bien, pour un mail, les réponses à "est-ce que l'expéditeur est connu/sur les listes de spam ?", "est ce que les mots "carte bancaire" et "erreur en votre faveur" sont dans le corps du mail ?", etc... L'ordinateur va apprendre les pa-

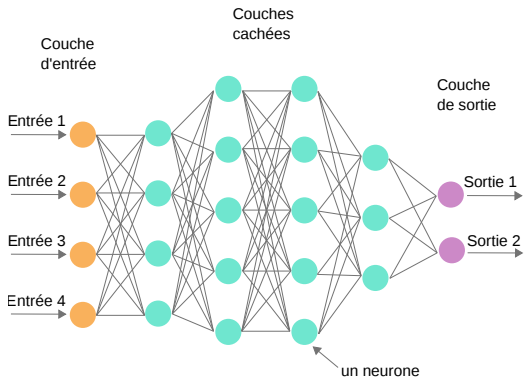


FIGURE 2.2 – **Réseau de neurones simple.** Ce réseau accepte 4 entrées (points orange), qui correspondent à 4 features ou descripteurs. Ces entrées vont être combinées au sein de 4 couches cachées de neurones (points bleus). Les poids (liens gris) qui relient les neurones vont être optimisés pour que les deux sorties (points mauves) correspondent à ce que l'on attend.

ramètres d'un algorithme pour prédire la bonne sortie à partir de ces features. Ces paramètres ne peuvent souvent pas être calculés directement, l'ordinateur va les ajuster itérativement. Pour les deux exemples de classification et de régression précédents, on entraîne les algorithmes avec des labels ou des valeurs-cibles connus pour un ensemble d'échantillons – le prix de la maison ou la catégorie de l'email, spam ou pas. Pour cette raison, ces types d'entraînements sont dits supervisés.
Dans le cas du deep learning, l'algorithme appris est une succession de couches de calculs, un réseau de neurones, qui combinent les caractéristiques ou features d'entrée.

2.2.2 Deep learning

La figure 2.2 présente l'architecture d'un réseau de neurones que l'on appelle perceptron multi-couche.
Dans le cas d'une tâche de classification, les deux sorties seraient les probabilités d'appartenir à deux catégories, par exemple spam ou email légitime. Lorsque le signal passe dans le réseau, on calcule la valeur d'un neurone en réalisant la somme des valeurs des neurones de la couche précédente, pondérée par les poids qui les relient au neurone. Une fois cette somme calculée, on lui applique une fonction de modification non linéaire, historiquement une fonction sigmoïde. Sans cette non-linéarité, l'ensemble du traitement du signal d'entrée par le réseau reviendrait à faire une somme pondérée. Dans ce réseau, chaque neurone est relié à tous les neurones de la couche suivante : ce sont des couches dites fully-connected.

Le terme d'apprentissage profond désigne la révolution qui a permis l'explosion de la taille des réseaux de neurones - autant leur nombre de couches que leur nombre de neurones, de paramètres et donc de calculs - et des progrès importants dans des tâches de haut niveau telles que la reconnaissance d'image ou le traitement automatique du langage. Cette augmentation de la profondeur des réseaux se base sur l'accroissement sans précédant de la puissance de calcul des cartes graphiques des ordinateurs, ainsi que de la massification des données dont on dispose. La démocratisation de l'informatique et d'internet a fait exploser les quantités de photos et textes que l'on peut rassembler.
Pour entraîner ces réseaux aux nombreux paramètres, il est nécessaire d'avoir une grande quantité de données. En effet, parmi les caractéristiques des données d'entrée, l'algorithme doit être capable de distinguer le bruit de l'information, plus exactement, ce qui est généralisable à d'autres données de ce qui est spécifique à une instance. Or, si le nombre de paramètres est trop grand, lors de l'apprentissage, le réseau peut se contenter de relier le bruit de

chaque donnée à sa valeur ou son label cible et, par là, ne pas être capable de généraliser à de nouvelles données. C'est ce que l'on appelle l'overfitting qui se traduirait par l'adaptation excessive de l'algorithme aux données d'entrée. L'overfitting peut être quantifié par la mesure de l'erreur de prédiction. Par exemple, pour des tâches de classification, on mesure la précision avec laquelle l'algorithme prédit les bons labels, l'*accuracy*. Elle rend compte du pourcentage de données bien classées. Pour vérifier que l'on ne fait pas de l'overfitting, on garde une partie des données de côté pendant l'apprentissage pour vérifier que l'algorithme est bien capable de généraliser à des données qu'il ne connaît pas. On a donc un dataset d'entraînement et un dataset de validation.

D'un point de vue algorithmique, l'entraînement du réseau de neurones a lieu en deux étapes répétées jusqu'à convergence ou arrêt de l'entraînement (voir flèches dans la figure 2.2).
La première étape est la prédiction des labels, elle se fait par propagation vers l'avant du signal. Puis on calcule l'erreur faite dans la prédiction (fonction de coût). La seconde étape rétropropage l'erreur. Plus exactement, on rétro-propage le gradient de l'erreur, qui nous indique la façon de modifier les paramètres les uns par rapport aux autres (augmenter ou diminuer leur valeur). On rétro-propage le gradient en faisant le calcul chaîné du gradient de l'erreur par rapport à chaque neurone (dérivée première). Puis, on modifie les paramètres de l'algorithme qui influencent la valeur prise par chacun de ces neurones.
Dans cet algorithme classique de modification des paramètres – appelé Stochastic Gradient Descent (SGD) [22]), la modification est proportionnelle à la fois au gradient de l'erreur du neurone que le paramètre influence et à un taux d'apprentissage, *learning rate*, choisi en fonction de la rapidité avec laquelle on souhaite que le réseau apprenne et de sa stabilité lors de l'apprentissage.

2.2.3 Deep learning pour l'analyse d'images ou réseau convolutionnel de neurones

Les modèles convolutionnels sont des réseaux certes plus profonds mais avec un nombre de paramètres réduits si on compare à d'autres architectures (voir figure 2.2). Le partage des poids permet un apprentissage optimisé en s'assurant la reconnaissance d'un motif quelque soit l'endroit de l'image. De plus la capacité de ces réseaux à extraire eux-mêmes des features est très intéressante car ils rendent obsolètes le calcul de features et leur sélection qui devaient être choisis pour chacune des tâches.

Il existe différents types de réseaux selon que les données sont statiques (images) ou séquentielles (langage, vidéo, enregistrement audio), mais aussi selon la tâche à accomplir : classification ou génération. On parlera dans cette partie de réseaux convolutionnels pour des tâches de classification d'images (2D). Les architectures décrites seront spécifiques à cette tâche de classification, mais les briques constituantes peuvent être utilisées pour d'autres tâches.

A la différence des réseaux de deep learning évoqués précédemment, les réseaux convolutionnels ne prennent pas de features en entrée, mais des images ou tout signal 1D, 2D, 3D ou 4D possédant une certaine cohérence dans chacune des dimensions. Dans un signal à une dimension, par exemple la température à

chaque heure au cours d'une journée, les valeurs successives ont une cohérence, dans l'exemple temporelle. Pour un signal 2D, une image par exemple, un pixel vert aura plus de chance de se voir entouré de pixels verts, etc...
Dans le cas des images, l'entrée est donc composée de la matrice des pixels, voir figure 2.3. Les features ne sont pas données en entrée du réseau, elles sont en fait apprises par celui-ci à travers les couches successives de convolution. On dit que le réseau apprend une représentation de l'instance d'entrée dans un espace à N dimensions. La dimension de la représentation est inférieure à celle de l'espace d'entrée qui est celui des pixels : largeur x longueur x nombre de canaux.

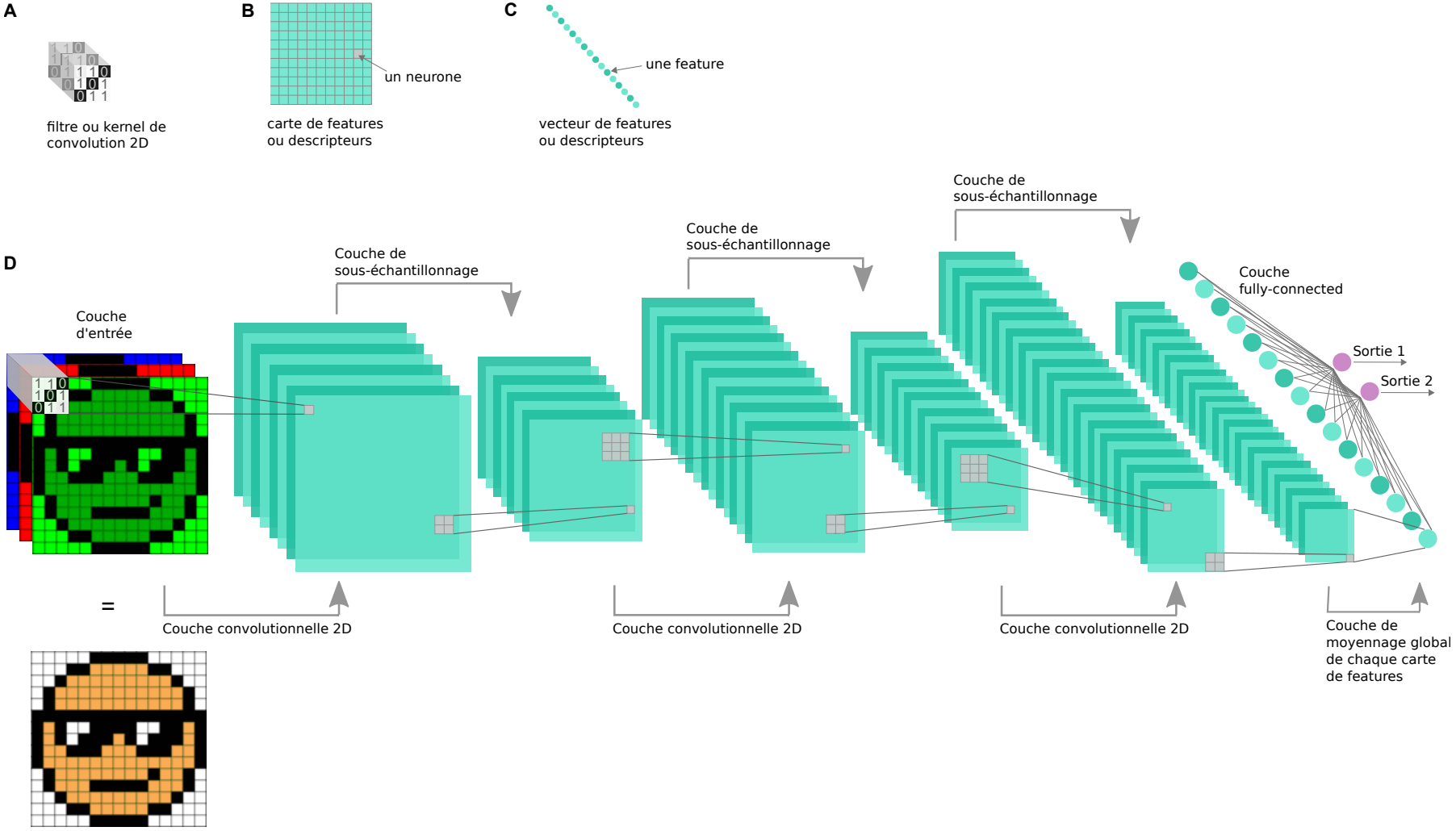

A
filtre ou kernel de convolution 2D
B
un neurone
carte de features ou descripteurs
C
une feature
vecteur de features ou descripteurs
D
Couche d'entrée
Couche de sous-échantillonnage
Couche de sous-échantillonnage
Couche de sous-échantillonnage
Couche fully-connected
Sortie 1
Sortie 2
=
Couche convolutionnelle 2D
Couche convolutionnelle 2D
Couche convolutionnelle 2D
Couche de moyennage global de chaque carte de features

FIGURE 2.3 – **Réseau de neurones convolutionnel.**
Un réseau de neurones convolutionnel est composé de couches de calculs de différentes natures. Lorsque l'on fait passer une image dans chacune de ces couches, le résultat est une matrice 3D (empilement de cartes 2D, en turquoise sur le schéma).
Les couches convolutionnelles sont obtenues en convoluant des filtres 2D (voir **A**) avec le bloc précédent. Pour la première couche de convolution, le bloc précédent est l'image elle-même. Le filtre présenté en **A** est un filtre 3x3 (hauteur x largeur en nombre de pixels) qui détecte les lignes diagonales. Les filtres sont spécialisés dans la détection d'un motif. Par exemple, le filtre A détecte les lignes noires diagonales. La convolution permet de détecter le motif du filtre dans toutes les positions de l'image. Le résultat d'une convolution (**B**) est une carte 2D, constituée de neurones sur une grille. On l'appelle carte de features. La cohérence spatiale est maintenue au sein de chaque carte de features. Chaque valeur de neurone correspond à l'adéquation du filtre vis à vis de l'image à cette position là. Le nombre de cartes de features 2D dans un bloc est égal au nombre de filtres dans la couche de convolution qui le précède. La profondeur d'un filtre correspond au nombre de cartes 2D dans le bloc sur lequel il s'applique (chaque canal du filtre est associé à une carte du bloc). En effet, on combine ainsi les résultats des convolutions des filtres de la couche précédente. On peut visualiser les filtres de la première couche de convolution facilement car ce sont des filtres à 3 canaux correspondant aux canaux RGB de l'image, voir figure 2.5. A contrario, les filtres des couches suivantes ont généralement beaucoup plus de canaux et ne peuvent pas être visualisés facilement.
Entre deux couches de convolution, il est usuel d'intercaler des couches de sous-échantillonnage qui diminue la résolution des cartes 2D. Cela permet aux filtres de convolution suivants d'intégrer une information relative à une plus grande partie de l'image. On peut remplacer ces deux étapes par une couche convolutionnelle sous-échantillonnante (avec un stride de 2).
Dans les réseaux récents, l'avant-dernière couche moyenne l'information de chaque carte en un nombre qui reflétera la valeur de cette feature pour l'image totale. Il résulte de cette opération un vecteur de neurones ou features **C** qui correspond à une représentation de l'image dans un espace à N dimensions, N étant le nombre de features de ce vecteur. Cette étape s'appelle un sous-échantillonage par moyennage global [23].
Enfin, la dernière couche du réseau est fully-connected comme dans le réseau de neurone simple (voir figure 2.2). Elle correspond à l'optimisation de la position d'un (hyper)plan dans l'espace des features pour couper cet espace en deux (s'il y a deux classes). De part et d'autre de ce plan, se trouvent les points correspondant aux images des 2 classes.
À partir des sorties, la fonction softmax [24] permet d'obtenir des probabilités d'appartenance à chaque classe. Elle est utilisée très largement par la communauté pour toutes les tâches de classification.

Les réseaux convolutionnels ont été inspirés par les processus biologiques liés à la vision. L'organisation des réseaux en couches ressemble, en effet, à l'organisation du cortex visuel [27].
De plus, les neurones qui résultent d'une couche de calculs convolutionnels possèdent un champ réceptif, similaire à celui que l'on observe chez les animaux [28]. Effectivement, on peut voir, figure 2.6, le champ réceptif de chaque neurone en vert et en bleu. L'image d'entrée se trouve sur la gauche. Plus l'on s'en éloigne, plus le champ réceptif d'un neurone est grand. En d'autres termes, les neurones des couches profondes du réseau intègre des informations provenant d'une zone plus grande de l'image. Des méthodes arithmétiques existent pour calculer la taille de ce champ récepteur dans les réseaux informatiques plus ou moins complexes [29, 30]. Grâce à ce système, les neurones ont un champ réceptif de plus en plus large quand on se déplace vers les couches profondes du réseau. Les couches successives permettent ainsi de capturer les détails dans les premières couches jusqu'à la scène complète dans les dernières couches, ce qui participe grandement à la puissance de discrimination de ce genre de réseaux.
Enfin, les filtres appris par optimisation, figure 2.5, sont stimulés de la même façon que des neurones situés dans le cortex visuel primaire [31, 32], réagissant à des orientations et des couleurs spécifiques à chacun. Cependant, en comparaison des systèmes de vision humaine, les réseaux de neurones de classification sont connus pour être fortement biaisés vers la reconnaissance des textures plutôt que des formes [33]. Dans la diversité des réseaux dont on parlera par la suite, certains sont un peu moins biaisés que d'autres, AlexNet par exemple [33]. Une récente étude de Sinha et al.. [34] montre que les capacités du réseau sont améliorées en appliquant un pré-traitement des images d'entraînement avec des filtres passe-bas. Les formes sont conservées, les textures moins. Les features apprises sont plus généralisables.
Cette facilité à représenter la texture plutôt que la forme, se reflétera aussi dans la partie consacrée aux réseaux générateurs d'images 2.3 et aux transformations par CycleGAN 2.4.2.

2.2.4 ImageNet et apprentissage par transfert de connaissances

Les réseaux de neurones possèdent un grand nombre de paramètres. Afin de leur permettre d'apprendre des features généralisables, de ne pas faire d'overfitting), il est nécessaire de les entraîner avec un grand nombre d'images. Le dataset ImageNet [35] présenté en 2009, contient plus de 10 millions d'images annotées pour 1000 différentes classes (labels). Ce sont des images dites naturelles : animaux, voitures, avions, objets, etc. Il est rare de disposer d'un dataset aussi grand. Ainsi, en classification avec du deep learning, une pratique très répandue consiste consiste à pré-entrainer un réseau sur le gigantesque dataset ImageNet, *avant* un apprentissage sur un dataset d'intérêt [36]. Ainsi les filtres appris par le réseau (figure 2.5) lorsqu'il est entraîné sur un dataset beaucoup plus conséquent que celui à disposition, peuvent être réutilisés pour extraire des informations sur le nouveau dataset, et qui seront pertinentes. Ce type d'apprentissage serait comparable à celui qu'utilise le cerveau humain pour reconnaître des nouvelles formes, par exemple un nouvel alphabet [37].

L'apprentissage par transfert de connaissance ou *transfer learning*

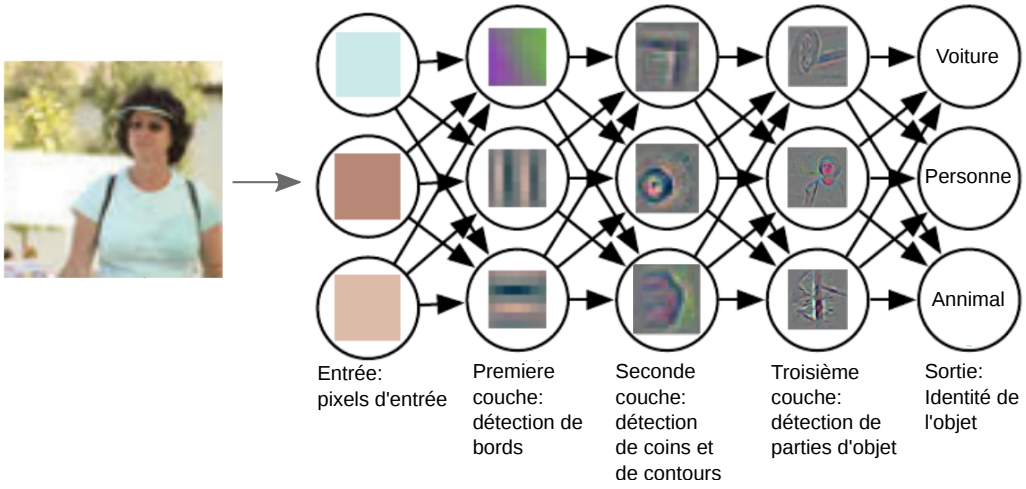


FIGURE 2.4 – **La complexité des motifs détectés par les features augmente lorsqu'on s'enfonce dans le réseau.** Chaque carte de feature détecte certains motifs. On commence par donner au réseau une image aléatoire, on regarde l'activation d'un neurone ou d'une carte de features donnée. Ensuite on rétro-propage le gradient pour que l'image d'entrée soit modifiée de telle sorte que le neurone d'intérêt soit activé plus intensément. Après quelques itérations, on obtient une image d'entrée qui maximise l'activation du neurone d'intérêt. Cela nous permet de visualiser dans l'espace de départ à quoi le neurone est sensible. Figure adaptée de [25]

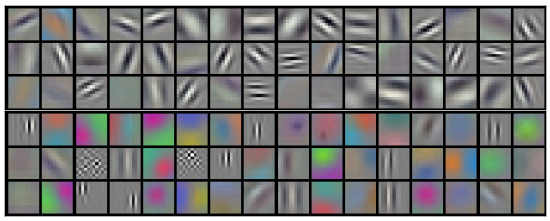


FIGURE 2.5 – **96 filtres de la première couche de convolution du réseau AlexNet [26]** On retrouve des filtres connus en analyse d'image tels que les filtres permettant de détecter des contrastes de couleurs, les bords d'un objet. Figure adaptée de [26]

peut être mis en place de différentes façons. La première, celle de Razavian et al.. [38], est d'utiliser le réseau pré-entraîné comme un extracteur de features. La dernière couche de classification est retirée, et, pour chaque image du nouveau dataset, on calcule le vecteur de features (voir figure 2.3). Un bloc de cartes de features intermédiaires peut être transformé en vecteur de features, dont les éléments sont les moyennes spatiales de chacune des

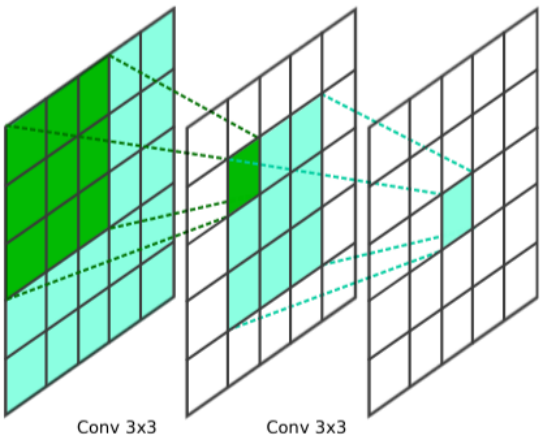


FIGURE 2.6 – **Champs réceptifs de neurones.** Une seule carte de features est représentée par couche pour simplifier la compréhension. Le champ réceptif du neurone bleu de la troisième couche est de 3x3 dans la deuxième couche et de 5x5 dans la première couche.

cartes de features. Une fois les vecteurs extraits pour chaque image, on peut entraîner un classifieur linéaire, comme une machine à vecteurs de support (Support Vector Machine ou SVM). La deuxième façon est d'affiner l'entraînement du réseau pré-entraîné à l'aide du nouveau dataset, en optimisant les paramètres de nouveau [39]. Il est possible de conserver les paramètres des premières couches fixes, afin d'empêcher l'overfitting. En fonction de la similitude du nouveau dataset avec le dataset grande-échelle de pré-entraînement et de la taille du nouveau dataset, on peut choisir entre les diverses possibilités d'entraînement.

Dans un article de décembre 2019, Raghu et al.. [36] ont étudié l'apprentissage par transfert pour des datasets d'imagerie médicale. Ces datasets sont très différents des images naturelles d'ImageNet. Ils comparent l'utilisation de très gros réseaux (23 millions de paramètres) obtenant des scores records sur la classification d'ImageNet, avec de réseaux beaucoup plus simples, le plus petit ayant un million de paramètres. Leurs résultats sont multiples. En cohérence avec l'idée générale autour de l'apprentissage par transfert, le réseau réutilisant les paramètres pré-appris apprend beaucoup plus rapidement la nouvelle tâche qui lui échoit. Par contre, après convergence des apprentissages, les auteurs n'observent pas de différence d'accuracy entre les réseaux pré-entrainés et ceux entraînés de zéro (indépendamment de la taille des réseaux). De plus, la similarité des représentations des images dans des réseaux, pré-entraînés ou non, est quantifiable grâce à un outil de mesure spécifique [40, 41]. Dans les gros réseaux, les représentations sont significativement différentes entre pré-entraînement et entraînement à partir de zéro. Dans les petits réseaux, la différence n'est pas significative. Les petits réseaux pré-entraînés évolueraient rapidement vers une autre représentation, tandis que les gros réseaux auraient une plus grande inertie, inertie plus importante dans les paramètres en début de réseau, une différence d'agilité comparable à celle entre une pirogue et un paquebot.

L'apprentissage par transfert permet d'accélérer le processus d'entraînement. La technique peut être intéressante si le dataset d'intérêt ne possède pas beaucoup de données mais les représentations apprises peuvent être différentes de celles obtenues par entraînement de zéro. La performance de discrimination du réseau n'est a priori pas affectée s'il y a convergence.

2.2.5 Les améliorations de l'apprentissage

De concert avec les avancées technologiques matérielles (cartes graphiques de plus en plus puissantes), les réseaux sont devenus de plus en plus profonds et de plus en plus difficiles à entraîner. On couvrira dans cette partie les différentes innovations architecturales et d'entraînement les plus généralement utilisées aujourd'hui.

Le sous-échantillonnage

Le sous-échantillonnage contribue à élargir le champ réceptif effectif des neurones (voir figure 2.6), sans apprendre de paramètres. Pour sous-échantillonner, on peut ajouter des couches spécialisées de *pooling*. L'opération consiste à rassembler plusieurs valeurs de neurones de la couche d'entrée sous un seul nombre (valeur maximum ou moyenne) [42]. On peut également réaliser des convolutions

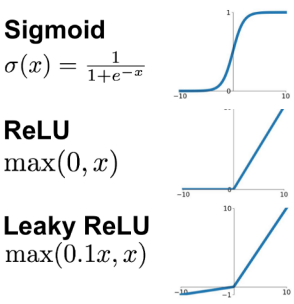


FIGURE 2.7 – **Fonctions non-linéaires d'activation.**

en sautant une position sur 2 (stride égal à 2).
Le sous-échantillonnage permet aussi une invariance à des petites translations et à l'échelle des objets. Par ailleurs, Riesenhuber et al.. [43] montrent que l'opération de pooling qui conserve le maximum plutôt que la moyenne, permet d'obtenir des résultats plus proches de ceux obtenus lors d'expériences chez le singe.

Rectified Linear Units ou ReLU

Un des principaux problèmes pour entraîner de grands réseaux est la disparition du gradient.
On rappelle que, pour optimiser les paramètres du réseau, on calcule l'erreur faite pour chaque classification puis on revient en arrière, couche par couche, pour calculer dans quel sens modifier chaque paramètre : c'est le calcul du gradient (voir figure 2.2). L'utilisation de la fonction de transformation non-linéaire sigmoïde, induit une multiplication du gradient à chaque couche par des valeurs très faibles. La sigmoïde possède, en effet, une dérivée presque nulle sur des intervalles importants (voir figure 2.7). Ainsi, l'amplitude du gradient décroît à mesure de sa rétro-propagation : il disparaît.
Une solution a été d'introduire l'activation Relu [44], dont le gradient est égal à

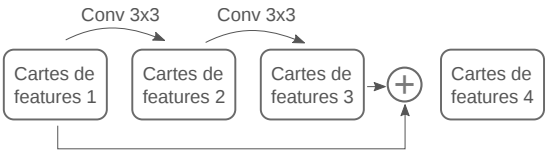


FIGURE 2.8 – **Bloc résiduel.** Après deux couches de convolution, le bloc de cartes de features de départ est additionné avec le bloc de cartes de features d'arrivée.

la valeur du neurone si positive. Il n'y a plus de saturation pour les grandes valeurs. Cela a permis d'entraîner des réseaux plus profonds, plus rapidement, à commencer par AlexNet en 2012 [26]. Améliorant encore les fonctions d'activation ReLU, les leaky ReLU [45] permettent de ne pas faire "mourir" un neurone grâce à son faible gradient dans les valeurs négatives, qui lui permet de continuer à être optimisé et de ne pas stationner à zéro.

Bloc résiduel et ResNet

Une autre façon de parer la disparition du gradient est d'ajouter des liaisons qui sautent des couches de convolution. Le gradient peut alors passer par ces liaisons sans être diminué, puisque le gradient à travers cette liaison directe est de 1. ResNet est le premier réseau à utiliser cette technique et a fait preuve d'énormes avancées en terme d'accuracy sur le dataset d'Imagenet, dépassant même les capacités humaines [46]. Des expériences, précédant ce modèle [22, 26], montraient qu'avec des réseaux de plus en plus profonds, les succès de reconnaissance d'image étaient de plus en plus fameux. He et al. [46], auteurs du ResNet, réussissent à entraîner des réseaux allant de 50 à 152 couches de convolutions avant cela, GoogleNet en 2014 contenait 22 couches de convolutions.

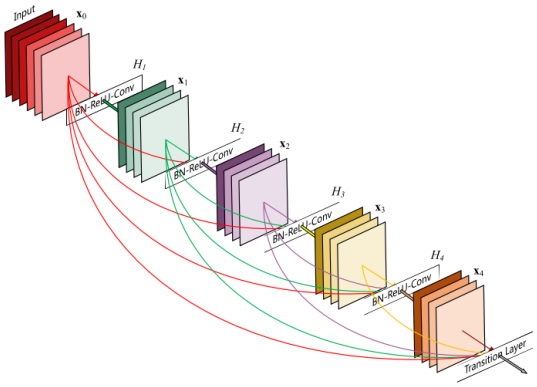


FIGURE 2.9 – **Bloc de DenseNet.** Avant chaque convolution, les cartes de features des blocs précédents sont concaténées. Figure extraite de [47]. Lien vers une animation plus explicite.

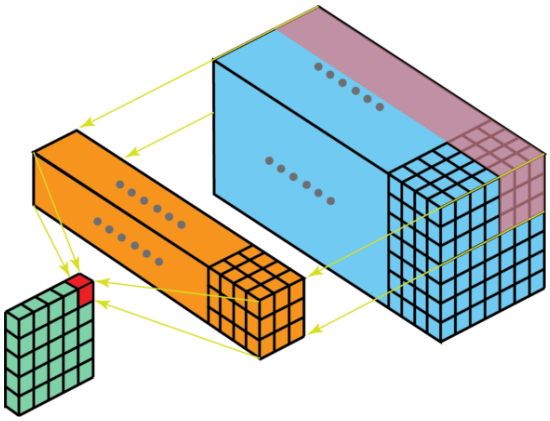


FIGURE 2.10 – **Convolution classique.** Une étape de convolution est le produit scalaire (multiplication terme à terme puis somme) du bloc orange avec le bloc rose, ce qui donne le cube rouge. Le filtre (bloc jaune) est convolué au bloc de cartes de features (bloc bleu). La carte de features (tranche verte) résulte de cette convolution. La longueur et la largeur du filtre permettent une convolution spatiale. La profondeur permet la combinaison des cartes de features du bloc bleu. Figure extraite de [48].

Bloc résiduel par concaténation et DenseNet

DenseNet [47] introduit un nouveau type de bloc résiduel. Au lieu d'additionner les cartes de features comme dans ResNet, le DenseNet les concatène. Le nombre de filtres de convolution par couche est beaucoup plus petit, de même que le nombre de cartes de features qui en résulte. À ce petit nombre de cartes sont concaténées les cartes obtenues par les couches précédentes : si ces cartes sont nécessaires, il n'y a plus besoin de les réapprendre. La convolution suivante aura lieu sur toutes les cartes de features concaténées.
Ajouter toutes les cartes de features précédentes fait augmenter le nombre de cartes de façon exponentielle. Par conséquent, ce mécanisme est restreint à des séries de quelques couches et, entre les séries, les cartes sont combinées pour diminuer leur nombre puis sous-échantillonnées.
Du fait de la réutilisation des cartes apprises précédemment, les DenseNets auraient besoin de moins de paramètres que les architectures précédentes, en atteignant de meilleures performances que les réseaux ResNets. Ces connections par saut permettent également un entraînement facilité.
En comparaison des Resnets, les DenseNets ont apparemment moins tendance à l'overfitting sur des petits datasets [46, 47].

Factoriser les convolutions

Sur un bloc de cartes de features (figure 2.3), les filtres de convolutions classiques réalisent deux fonctions dans le même temps. Ils sont premièrement chargés de filtrer chaque carte de features d'un bloc, à la recherche d'un certain motif. Par la suite, ils réalisent une combinaison linéaire entre ces filtrages. Les convolutions séparables permettent de séparer ces deux étapes [49]. Cela permet un grand allègement du nombre de paramètres dans le réseau puisque les filtres spatiaux peuvent

être réutilisés dans des combinaisons linéaires différentes. Les réseaux Xception [50], MobileNet [51] et EfficientNet [52] utilisent ce type de convolution.

Normalisation par batch

La normalisation par batch est une technique qui accélère et stabilise l'entraînement des réseaux. Elle améliore parfois leur performance [53]. L'idée est de normaliser chaque carte de features pour que la moyenne à travers le batch soit de zéro et la variance égale à 1. Lors de l'étape d'optimisation, les paramètres des couches menant à une carte de features évoluent. Il est probable que les statistiques de la carte de features évoluent aussi. Si ces statistiques sont modifiées, la couche de convolution suivante doit s'adapter à ce déplacement – tout en apprenant des convolutions utiles – et cela ralentit le processus d'apprentissage. Ce processus s'appelle le déplacement interne par covariance ou *internal covariance shift*. Les avantages que procure la normalisation par batch ont d'abord été attribués à la limitation du décalage de ces statistiques lors de l'apprentissage. Santukar et al.. [54] ont récemment remis en question cette explication. D'après les expériences menées en l'absence de normalisation par batch, l'évolution des statistiques des cartes n'est pas plus erratique. Par contre, leur étude montre que l'efficacité de la normalisation par batch serait plutôt due au lissage de fonction de coût. En allant dans la direction de modification indiquée par le gradient, l'erreur décroît de manière beaucoup plus continue dans le cas de l'utilisation de la normalisation. La normalisation des cartes de features transforme donc le paysage de l'optimisation pour le rendre plus régulier et donc prévisible. On peut aller plus vite dans une direction (en augmentant le taux d'apprentissage ou learning rate).

FIGURE 2.11 – **Visualisations de 6 cartes de features.** Les visualisations sont obtenues par la même méthode que figure 2.4, mais en maximisant une carte de features plutôt que la valeur d'un neurone. Une technique de régulation de l'optimisation permet d'obtenir des images moins bruitées. Chaque colonne correspond à une couche. Visualisations extraites de [55].

2.2.6 Explication de la classification

Les recherches exposées précédemment concernaient l'amélioration des capacités de discernement des réseaux de neurones. Néanmoins, si une très bonne performance d'un réseau augmente la confiance que l'on a dans ses décisions, cela ne remplace pas la confiance que l'on peut avoir dans un système que l'on comprend. Par conséquent, en parallèle, l'interprétabilité de cette "boîte noire" qui apprend par elle-même peut être étudiée, pour comprendre ce qui mène le réseau à sa prédiction.

Visualisation de features

Les visualisations des motifs auxquels réagissent chaque neurone/carte de features/groupe de neurones, peuvent être obtenues en optimisant l'image d'entrée pour maximiser les valeurs prises par ces groupes de neurones

(rappel de la figure 2.4). On peut voir quelques exemples figure 2.11 de maximisation de cartes de features. L'image optimisée en entrée peut être une image du dataset ou bien une image de bruit. Dans le premier cas, on verra ce qu'il est nécessaire de modifier dans l'image pour une activation maximale.

Une autre façon de visualiser les motifs qui activent un groupe de neurone ou un simple neurone est de chercher dans les exemples du dataset ceux qui déclenchent le neurone. Le problème potentiel est qu'on peut observer des images qui corrèlent avec les causes et extrapoler à tort, par exemple, si des balles de baseball font réagir un neurone, il peut s'agir d'un neurone sensible aux balles de baseball ou simplement aux rayures.

Attribution et cartes d'influences

L'idée derrière l'attribution est de découvrir quelles sont les relations entre neurones. La technique la plus répandue est celle des cartes de saillance. Elle correspond à une carte de chaleur qui souligne la pertinence de chaque pixel pour la classification du réseau. On fait passer une image dans le réseau, puis en rétro-propageant le gradient de l'erreur pour une classe jusqu'aux pixels, on peut obtenir cette carte de chaleur [57]. On trouve alors comment chaque pixel influence le résultat.

Au contraire des cartes de saillances, les techniques de carte d'activation spécifique à une classe (CAM [56] et Grad-CAM [58]) ne retro-propagent pas le gradient jusqu'aux pixels mais jusqu'à la dernière couche convolutionnelle du réseau (voir figure 2.12). Plus particulièrement, pour obtenir la carte de chaleur à partir de la couche considérée, on combine les cartes de features de la couche, pondérées par le gradient moyen spécifique à chaque carte et à la classe considérée. La carte de chaleur possède les dimensions spatiales de la couche de convolution : elle est bien plus petite que l'image de départ. On va donc sur-échantillonner cette carte pour qu'elle ait la taille de l'image de départ. Ces cartes permettent de visualiser l'influence de chaque position spatiale au niveau d'une des couches hautes du réseau.

Interfaces utilisateur

Olah et al. [59] présentent différentes interfaces qui utilisent les visualisations de features et les cartes d'activation en synergie pour comprendre le traitement de l'information par le réseau relativement à chaque image (lien vers les interfaces). Les interfaces jouent avec des visualisations de neurones seuls, de groupes de neurones, des cartes d'activation projetées non seulement sur l'image de départ, mais aussi sur les couches de neurones intermédiaires, afin de mieux cerner ce qui influence la valeur d'une activation et le concept qu'elle renferme. La figure 2.13 présente une de ces interfaces. Afin d'obtenir cette visualisation, une couche de neurones a été factorisée pour réduire le grand nombre de neurones en 6 groupes. Chaque vignette est une visualisation d'un groupe de neurones, obtenu par factorisation de matrice, et on observe sur la carte les couleurs correspondant à chaque groupe.

2.2.7 Applications en imagerie biologique

Une des premières applications du deep learning à l'imagerie biologique a été

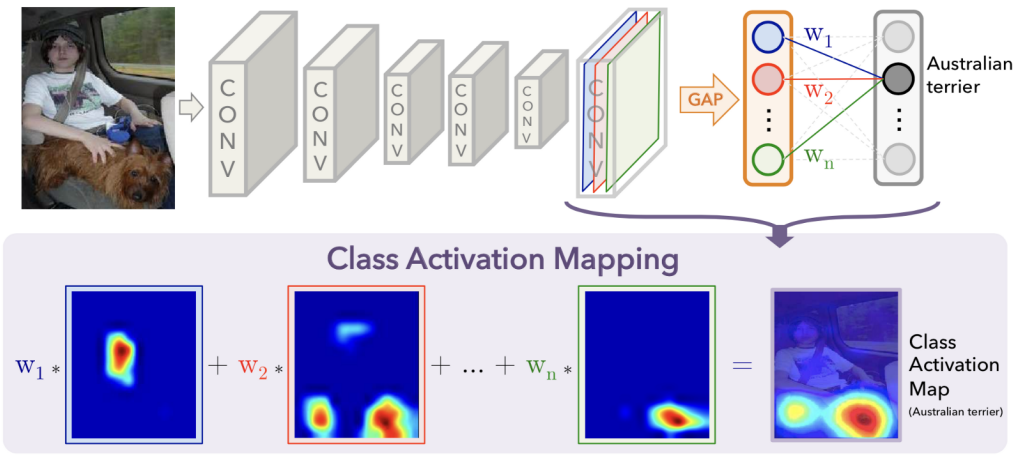


FIGURE 2.12 – **Carte d'activation par classe.** Figure extraite de [56].

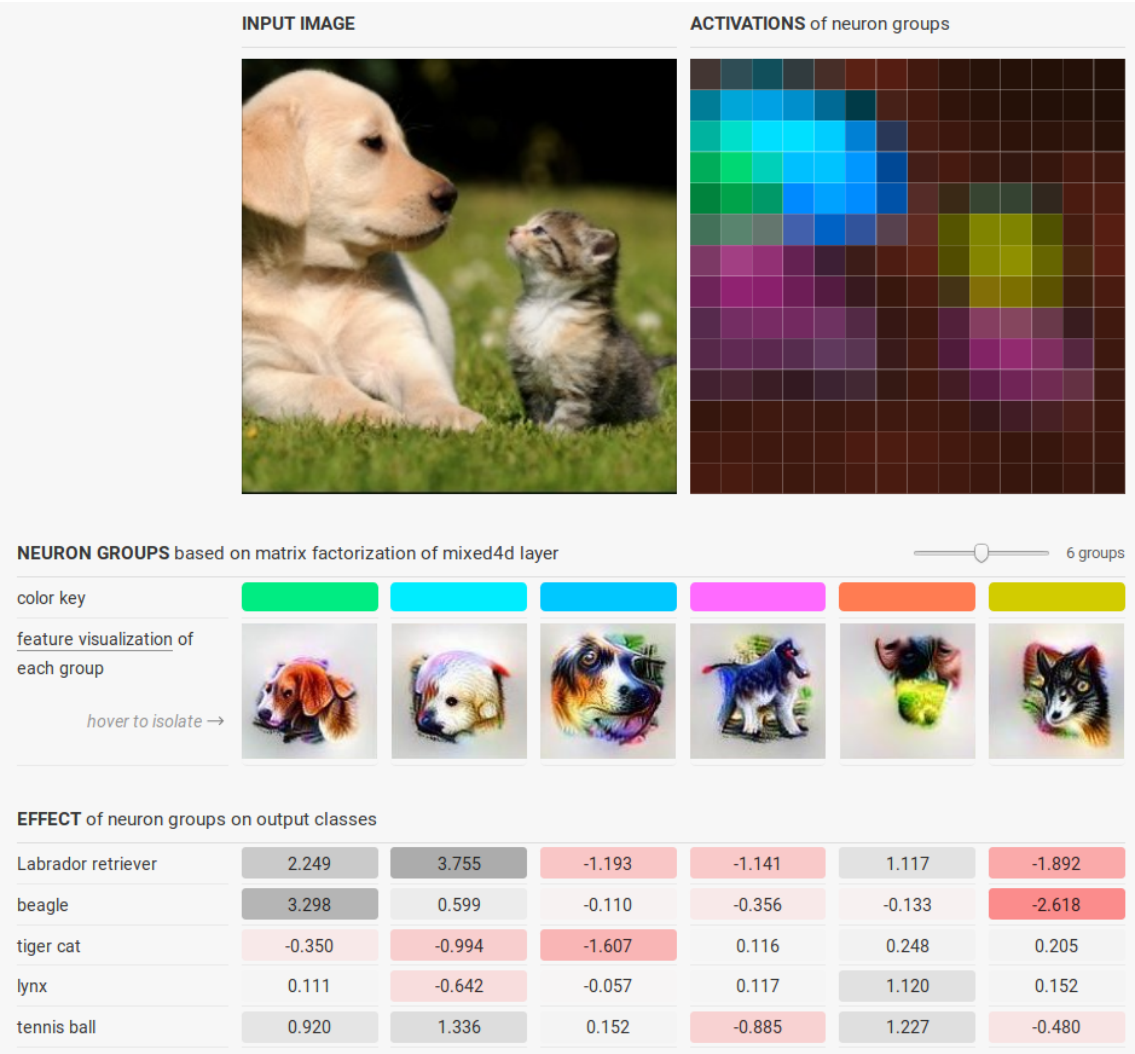


FIGURE 2.13 – **Interface visualisation de groupe de features et carte d'activation spécifique à ces groupes.** Les groupes correspondent plus ou moins à l'oreille, le front, le museau du chien, au corps des animaux, à l'herbe, et à la tête du chat. Un tableau permet en plus de montrer l'influence de chaque groupe pour la classification dans différentes classes. Enfin, l'effet de chaque groupe sur la classe de classification est visible dans le tableau. Figure extraite de [59].

celle, en 2005, de la classification des structures sub-cellulaires de l'embryon du ver C. Elegans [60]. Dans leurs travaux, un entraînement sur des petits patchs d'image de 40x40 devaient conduire un réseau à prédire quel était le type du pixel central : noyau, cytoplasme ou membrane. En concordance avec l'approfondissement des réseaux pour la classification d'images naturelles, Cireşan et al. utilisent un réseau plus profond que le précédent pour classer des pixels comme appartenant à une cellule en mitose ou non, dans des images de tissus cellulaires. On verra par la suite, section 2.4.1, que d'autres d'autres types de structures de réseau améliorent la classification de pixels, ainsi que la rapidité de la prédiction.

Pour conclure ce chapitre, remarquons qu'une autre application du deep learning est la classification d'images entières, qui nécessite moins d'effort de labellisation que les applications pixel à pixel.
Kraus et al. [61] utilisent ainsi un réseau, non seulement pour classifier les images mais ensuite pour segmenter les cellules à l'aide de cartes de saillance.
Godinez et al. [62], eux, présentent un réseau multi-échelle, avec 7 branches qui examine une image à des résolutions différentes. Les auteurs cherchent à classifier des images de cellules de cancer du sein traitées avec des molécules anticancéreuses (dataset BBBC021 [63]). Les molécules sont regroupées par mécanisme d'action, et l'objectif pour le réseau est de prédire le mécanisme d'action d'une molécule qui a été retirée du dataset d'entrainement. Grâce à la dernière couche du réseau qui renvoie une probabilité, ils reconstruisent avec succès les courbes doses-réponses des molécules, à partir de la probabilité d'être différent du contrôle négatif.
Kraus et al. [64] créent un réseau pour classifier des images de levures selon la localisation de protéines marquées par GFP.
En 2019, Yang et al. [65] utilisent un réseau pré-entraîné, sur un dataset de lignées cellulaires de fibroblastes issus de patients atteints d'amyotrophie spinale (SMA) et de personnes saines, dans le but de déterminer si les 2 populations sont différenciables.

2.3 Réseaux adverses génératifs

Les Generative Adversarial Networks (GANs) sont une innovation récente passionnante du machine learning. Comme leur nom l'indique, les GANs sont des réseaux générateurs : ils fabriquent de nouvelles données qui se confondent avec les données d'entraînement. Aujourd'hui, ils atteignent des capacités de réalismes stupéfiantes : les visages présentés en figure 2.15 sont tous synthétiques.
Les GANs ont été introduits pour la première fois en 2014, par Ian Goodfellow et al. [67]. Les GANs sont constitués de deux réseaux : un générateur et un discriminateur, avec des objectifs en compétition (voir figure 2.14). Le but du générateur est de générer des images provenant de la même distribution que les images du dataset d'entraînement. Le but du discriminateur est de distinguer les fausses images, générées, des vraies images du dataset. Le discriminateur doit maximiser l'accuracy associée à la prédiction des vraies ou fausses images. Le générateur est optimisé en vue de la minimiser, c'est-à-dire pour tromper le discriminateur. Par conséquent, on appelle ces réseaux "adversaires".

Dans cette partie, on présentera différentes architectures et quelques stratégies d'optimisation des GANs

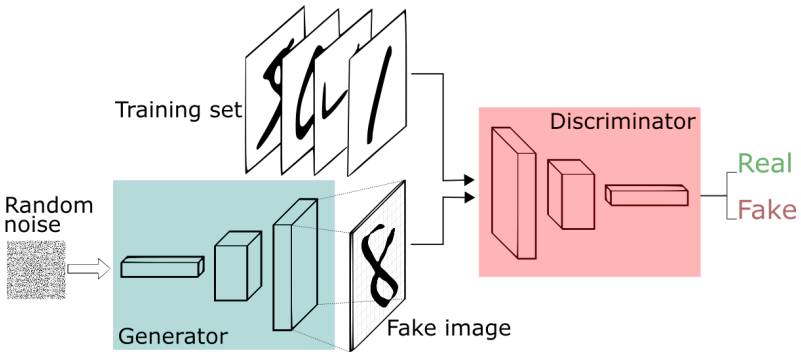


FIGURE 2.14 – **Schéma d'un GAN.** Image reprise de [66].

FIGURE 2.15 – **Évolution de la qualité des visages générés à partir de GAN.** De droite à gauche : Vanilla GAN [67], DCGAN [68], CoGAN [69], GAN progressif [70], StyleGAN [71]. Pour aller plus loin dans l'émerveillement, vous pouvez regarder cette vidéo sur le réseau StyleGAN.

qui ont mené à l'évolution illustrée figure 2.15.

2.3.1 Vanilla GAN, du bruit à l'image

La base de données Mnist [72] est constituée d'un ensemble de chiffres écrits à la main. Les images sont en noir et blanc, normalisées et centrées dans des images de 28 pixels de côté. Le générateur du GAN introduit par Goodfellow et al. [67] est entraîné à synthétiser des images de chiffre similaires à partir de vecteurs de bruit. Chacune des composantes d'un vecteur est tiré aléatoirement dans une distribution normale, et à chaque vecteur correspondra une image. Toutes les couches qui suivent sont denses ou fully connected, il n'y a aucune convolution, comme dans le cas du réseau présenté figure 2.2. La dernière couche est redimensionnée pour obtenir une grille 2D : l'image générée. Après entraînement, le réseau est capable de produire des images de chiffres ainsi que des visages ressemblant à des images réelles sans pour autant être de très bonne qualité.

2.3.2 Deep Convolutional GAN ou DCGAN

Le DCGAN [68] améliore énormément les capacités du premier GAN (voir figure 2.16). Le vecteur de bruit au début du réseau est projeté dans un espace beaucoup plus grand (1024 cartes de features de taille 4x4). S'en suit une série de convolutions transposées de strides 2, permettant le sur-échantillonnage des images, jusqu'à arriver à des images de 64 par 64. Après chaque étape de convolution transposée, on trouve une activation ReLU (voir section 2.2.5) puis une normalisation par batch (voir section 2.2.5). La dernière activation du générateur est une fonction tangente hyperbolique. Radford et al. justifient ce choix par l'affirmation qu'une fonction bornée aide le générateur à découvrir la saturation, et à couvrir tout l'espace des couleurs. En conséquence, les images des datasets doivent être normalisées entre -1 et 1, pour avoir les mêmes valeurs que les images générées. Le discriminateur utilise des convolutions avec stride

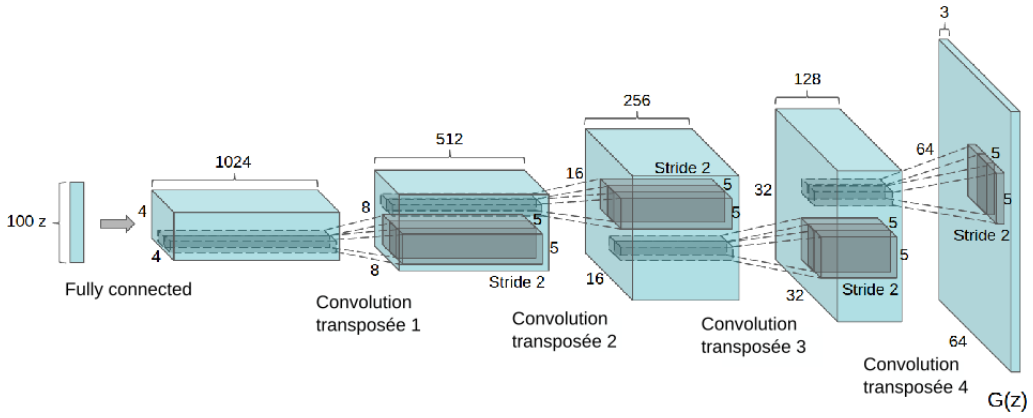


FIGURE 2.16 – **Schéma du DCGAN.** Figure adaptée de [68].

2 pour sous-échantillonner les cartes de features, ainsi que des activations Leaky RelUs.

Les avantages de ce modèle comparé aux générateurs précédents sont :

1. L'entraînement est plus stable.
2. Le discriminateur apprend une vraie représentation des images, puisqu'en extrayant les features de la dernière couche, on peut entraîner un classifieur, non plus à reconnaître les vraies images des fausses mais des classes. Le discriminateur a appris les représentations générales des différentes classes.
3. Dans l'espace latent, on peut réaliser des opérations arithmétiques. Par exemple, pour les visages, si l'on soustrait la moyenne de vecteurs générant des hommes à la moyenne de vecteurs générant des hommes à lunette, et qu'on ajoute la moyenne de vecteurs générant des femmes, on obtient une image de femme portant des lunettes. De plus, la pose (orientation de la tête) est modélisée linéairement. Ainsi, il existe une direction de l'espace dans lequel est défini z, le vecteur de bruit, pour laquelle les têtes tournent de droite à gauche.

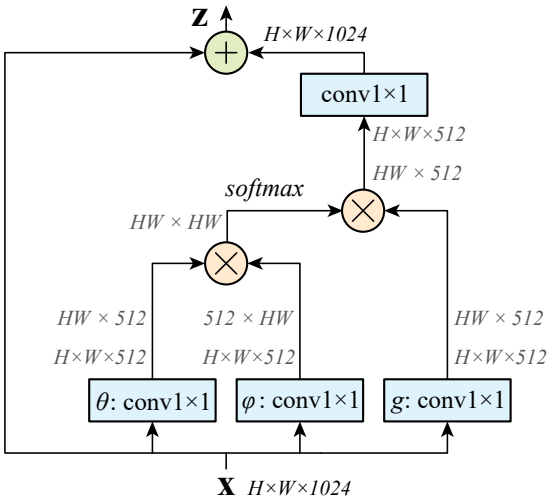


FIGURE 2.17 – **Bloc non-local.** H x W x 1024 sont les dimensions du bloc de features, H la hauteur, W la largeur (*width*) et 1024 le nombre de cartes de features. Figure adaptée de [73]

2.3.3 Self-Attention GAN ou SaGAN

Avant l'arrivée des couches de self-attention, les GANs étaient mis en difficulté pour représenter des classes d'images comportant des contraintes structurelles. Par exemple, tandis que les GANs génèrent avec performance des images où la texture est prédominante (paysage, ciel, océan), ils ont plus de difficultés à représenter les images à fortes composantes géométriques, comme par

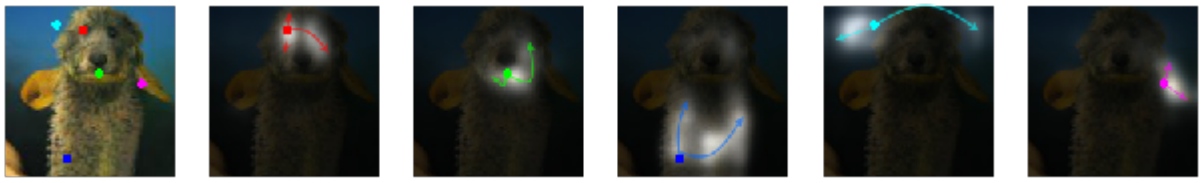


FIGURE 2.18 – **Cartes de correspondances.** Des positions sont choisies (points colorés sur l'image de gauche). Pour chaque position, est exposée la carte de correspondance avec toutes les autres positions de l'image. Figure extraite de [74]

exemple les images d'animaux, pour lesquelles il faut compter le nombre de pattes [74, 75]. Les formes géométriques nécessitent notamment des corrélations longues distances entre les pixels.

Pour palier ce manque, Wang et al. [73] introduisent un modèle de classification de vidéos capable de capturer les dépendances longue-distance entre les features. Ils utilisent un système de bloc non-local schématisé figure 2.17. Après combinaison des cartes de features par les convolutions 1x1 (taille du kernel de 1), on aplatit les dimensions spatiales (HW x 512) et on réalise la matrice des correspondances spatiales (HW x HW). Chaque ligne, de taille HW, correspond à une position h_i, w_j. Pour chaque position, on a donc sa correspondance avec toutes les positions de l'image (voir figure 2.18). Lorsqu'on filtre les cartes de features (voie de droite sur le schéma), avec ces correspondances, on a pour chaque position et pour chaque carte de feature, une information qui provient de multiples zones de l'image.

Le SaGAN [74] reprend dans l'architecture de son générateur, ce bloc de correspondances spatiales, en plus de couches convolutionnelles plus traditionnelles.

2.3.4 GAN progressif ou ProGAN

Le GAN progressif [70] se différencie des autres par le fait qu'il s'entraîne sur des images de plus en plus grande résolution, en ajoutant des couches au générateur et au discriminateur à chaque augmentation de résolution. Il donne notamment des résultats époustouflants pour générer des images de célébrités en 1024x1024 pixels (figure 2.15). Cet entraînement progressif apporte de la stabilité dans les grandes résolutions et accélère le processus d'apprentissage.

FIGURE 2.19 – **Exemple de transfert de style.** Figure adaptée de [76]

2.3.5 Un GAN basé sur les styles ou StyleGAN

Avant de parler du StyleGAN, il faut mentionner le domaine de recherche de transfert de style d'une image à une autre comme présenté en figure 2.19 [77-79]. Huang et al. [76] font l'hypothèse que le style d'une image est encrypté dans les statistiques des cartes de features. Ils introduisent l'AdaIN ou Adaptative Instance Normalization. Ils sélectionnent une couche d'un réseau de classification pré-entraîné. Ils extraient les cartes de features des deux images de style et de contenu pour cette couche là. La carte

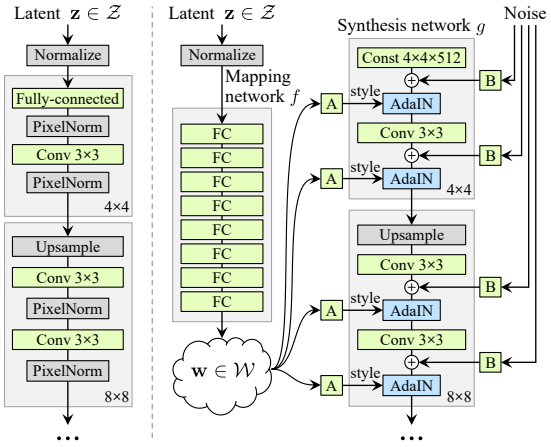


FIGURE 2.20 – **Générateur de GAN normal à gauche. Générateur de StyleGAN à droite.** Comme le DCGAN et les autres réseaux générateurs, le réseau de synthèse va générer des cartes de features de plus en plus grandes, commençant par des cartes de 4x4, puis 8x8 jusqu'à atteindre la taille de l'image. Seules les deux premiers blocs sont schématisés ici pour chaque réseau, arrivant donc à une taille de carte de 8x8. Figure extraite de [71].

de l'image de contenu est normalisée puis dé-normalisée avec la moyenne et la variance de la carte correspondante de l'image de style. Après décodage (on ne détaillera pas ici comment le décodeur est obtenu), on obtient l'image de droite de la figure 2.19, et ce pour n'importe quelle image de style et de contenu.

Le StyleGAN est l'un des générateurs de visages le plus convaincant en 2019 (voir figure 2.15). L'architecture du StyleGAN [71] est très particulière et s'inspire de ces couches AdaIN [76]. Elle a été conçue pour permettre un contrôle du processus de synthèse d'image. En effet, dans un GAN traditionnel (à gauche figure 2.20), le vecteur latent z est à la base du vecteur. Il subit convolutions et autres transformations. Dans le StyleGAN, le vecteur latent z sert à paramétrer les transformations. Précisément, le vecteur latent z est d'abord transformé en un vecteur d'espace latent intermédiaire w, par le réseau mapping, qui sera lui même transformé via des transformations affines A en style, qui sont les statistiques des couches d'AdaIN du réseau de synthèse. Ces normalisations de cartes de features sont les seules entrées dont dispose, après chaque étape de convolution, le réseau synthétiseur pour générer des visages. Le réseau de synthèse génère le visage que le réseau de mapping lui décrit. Afin de pouvoir générer les détails aléatoires (grain de peau, cheveux, etc...), du bruit est également ajouté avant chaque étape AdaIN.

Les auteurs vont plus loin dans leur innovation afin de permettre un meilleur désenchevêtrement de l'espace latent intermédiaire. Un espace latent désenchevêtré est composé de sous-espaces linéaires contrôlant chacun un facteur de variation de l'image. Le désenchevêtrement permet notamment une plus grande interprétabilité des axes de variation de l'image. Ils améliorent donc ce désenchevêtrement en mélangeant les vecteurs latents intermédiaires w_1 et w_2 provenant de deux vecteurs latents z_1 et z_2 puis en générant l'image associée. Cette astuce d'entraînement induit la décorrélation des informations contenues dans chacune des variables du vecteur w.

2.3.6 BigGAN

Le GAN présenté par Brock et al [80] est l'un des meilleurs générateurs du moment. Les modèles deviennent imposants avec 4 fois plus de paramètres que les réseaux précédents et des tailles de batch qui atteignent 2000 images. L'entraînement d'un tel réseau nécessite des capacités de calcul gigantesques. Des blocs de self-attention sont utilisés, comme dans le réseau SaGAN, pour permettre des dépendances longues dis-

tances entre les objets dessinés.

2.3.7 Les mesures de performance

Lorsque l'on souhaite entraîner un GAN, on doit faire face à un certain nombre de difficultés. Un problème des GANs est leur évaluation [81]. En effet, il n'existe pas de fonction universelle permettant de quantifier la performance d'un GAN. Les réseaux de classification se basent sur l'accuracy sur le dataset de validation pour arrêter l'entraînement, et l'accuracy sur le dataset de test pour évaluer les performances générales du réseau. Pour les GANs, il n'y a pas vraiment de mesure qui validerait la qualité des images ainsi que leur diversité. Le discriminateur donne sa perception de la qualité des images produites via son accuracy. Néanmoins, sa perception varie au cours de l'entraînement et une mauvaise accuracy n'est pas gage de bonne génération d'images, elle peut simplement être due à un mauvais entraînement du discriminateur.

Le score Inception

Pour résoudre ce problème, Salimans et al. introduisent le score Inception [82].Inception est un réseau, qui a été introduit pour classifier les images du dataset d'Imagenet. Supposons que l'on cherche à créer un générateur d'images d'animaux. En premier lieu, on génère des images x, puis on utilise le classifieur Inception [83] pour obtenir y, les prédictions des classes du modèle pour chacune de ces images. Si chacune des images x contient un objet d'une classe (animal) de bonne qualité, alors la probabilité associée à cette classe sera proche de 1 tandis que les autres seront proches de 0. En d'autres termes, la distribution conditionnelle $p(y|x)$ ne sera pas du tout uniforme (faible entropie). Par ailleurs, un bon générateur génère des images variées, donc la distribution marginale des classes $p(y)$, la moyenne des prédictions pour chaque classe en parcourant les images, est uniforme (de forte entropie). Pour un bon générateur, les deux distributions $p(y|x)$ et $p(y)$ seront donc très différentes. C'est justement cette différence que mesure l'*Inception score* via la divergence de Kullback-Leibler $E_x[KL(p(y|x)||p(y))]$. Plus les distributions sont différentes, plus le générateur est bon. Cette mesure a été largement utilisée [70, 84, 85]. Cependant, ce score peut être élevé si, pour chaque classe, le générateur produit la même image et ne reflète donc pas la variété intra-classe. L'utilisation du score Inception pour évaluer des GANs est critiqué lorsque le dataset d'entraînement n'est pas un sous dataset d'Imagenet. En effet, le score utilise les prédictions pour les classes d'Imagenet, ces prédictions sont de moindre intérêt pour des datasets complètement différents.

La distance Inception de Fréchet ou FID

La mesure de qualité la plus généralement utilisée, introduite après le score Inception, est la distance Inception de Fréchet ou FID [86]. Elle utilise également le modèle Inception pré-entraîné sur le dataset d'Imagenet. À l'inverse du score Inception, une petite distance est signe de plus grande ressemblance du dataset synthétique avec le vrai dataset. L'Inception score prédit les classes d'Imagenet pour les images du dataset synthétique. La FID se contente de calculer les 2048 activations de la dernière couche du modèle pour chaque image. Puis il compare les statistiques de ces activations avec celles calculées sur le datatset d'images

réelles. Pour des raisons de simplicité de calcul, le FID fait l'approximation que les valeurs prises par les activations peuvent être résumées par une distribution Gaussienne multivariée. Chaque activation correspond à une variable de la gaussienne. Les moyennes pour chaque variable – ainsi que les matrices de covariances de ces variables – sont calculées pour les deux jeux d'images. Puis la distance entre les deux Gaussiennes est calculée suivant la formule de la distance de Fréchet (2.1), avec x un jeu d'images réelles, g un jeu d'images générées, $\mu_{\{x,g\}}$ les vecteurs de moyennes, et $\Sigma_{\{x,g\}}$ la matrice de covariance.

$$FID(x,g) = \|\mu_x - \mu_g\|_2^2 + Tr(\Sigma_x + \Sigma g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}) \quad (2.1)$$

À la différence du score Inception, le FID permet de mesurer si l'on génère les différents modes du dataset. En effet, Brock et al. [80], auteurs du BigGAN, introduisent une façon simple de trouver un compromis entre qualité et variété de leur dataset de synthèse. On choisit, après l'entraînement, selon que l'on souhaite des images de grande variété ou de grande qualité. Lorsque l'on se dirige vers la qualité, le score Inception reste élevé tandis que le FID s'effondre : la distance est grande entre un pic de Dirac et une distribution multimodale ou une gaussienne de grande variance.

Quoiqu'il en soit, le grand intérêt des deux mesures présentées ici est leur corrélation avec la perception humaine de la qualité des images [82, 86].

Pour terminer, notons que l'entraînement des GANs peut être instable. Sans rentrer dans les détails, différentes techniques existent notamment pour contrer la disparition du gradient de l'erreur ou son explosivité [87].

Des techniques existent également pour empêcher l'effondrement de mode. L'effondrement de mode arrive, par exemple, lorsqu'un générateur entraîné sur un dataset de chiffre (MNIST [72]) ne parvient pas à générer tous les chiffres.

2.4 La transformation d'image à image

La transformation d'image à image est un problème fondamental pour la vision par ordinateur. Le but est d'apprendre une fonction de transformation d'une image d'entrée vers une image d'arrivée. Divisons ce problème en deux catégories. D'une part, les cas supervisés, où il existe dans le dataset des paires d'images associée. On apprend alors la transformation de l'une à l'autre. D'autre part, les cas non-supervisés, pour lesquels on dispose de datasets composés de deux types d'images pour lesquels il n'existe pas d'appariement, et où l'on cherche une transformation d'un domaine à l'autre.

2.4.1 Images appariées, apprentissage supervisé

Une des tâches d'analyse d'image de cette catégorie est la segmentation sémantique d'image. Le but de la segmentation est de déterminer à quelle classe ou label appartiennent chaque pixel (voir figure 2.21). Pendant l'apprentissage, un réseau va être chargé de produire une carte de segmentation. À la suite de cela, on compare la carte produite avec la vérité terrain et l'erreur faite pour chaque pixel sera rétropropagée dans le réseau pour optimiser ses paramètres.

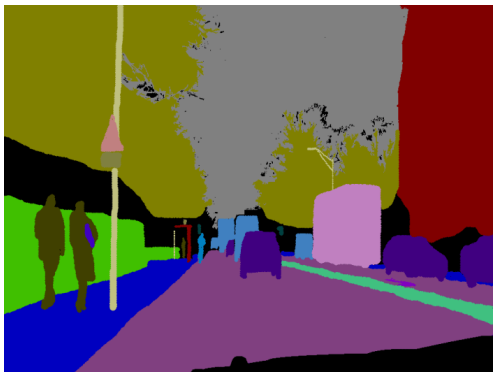


FIGURE 2.21 – **Vérité terrain pour la segmentation sémantique d'une image de conduite.** Cette figure appartient au dataset de segmentation de vidéo de conduite "CamVid" [88]. Le dataset contient 32 différents labels. Cette image correspond à la carte de segmentation pour une image prise dans la rue. Chaque couleur sur l'image correspond à un label. Cette carte est donc une image de la même taille que la photo, mais pour chaque pixel, on ne dispose pas des valeurs RGB, mais d'un label qui correspond à l'objet qu'il compose. Ex : un piéton correspondra au numéro 3, une voiture, au numéro 5, etc...

Réseau fully convolutional et U-Net

Long et al. [90] ont proposé un réseau totalement convolutionnel. Cela permet notamment de s'affranchir du redimensionnement des images car des images de toutes les tailles peuvent être générées. Ce réseau permet de réaliser une segmentation sémantique à partir des différentes couches de cartes de features d'un réseau de classification (voir figure 2.22). Chacune des couches de cartes contient encore une information spatiale, mais plus on avance dans le réseau plus l'information est grossière : une image de 256x256 peut, par exemple, être représentée par des cartes de features de 16x16. Cependant, plus on avance dans le réseau, plus les cartes intègrent des informations d'une plus grande zone de l'image, et par conséquent, sont riches en informations sémantiques, i.e la nature des objets dans la zone regardée. Leur idée, reprise par Ronneberger et al. [89] dans leur U-net (voir figure 2.23), est de combiner des informations locales (de textures, de couleurs par exemples) avec des informations de plus en plus globales sur les objets représentés pour réaliser une segmentation fine mais ayant un sens global. Plus concrètement, le fully convolutionnal network prend les blocs de cartes de features avant chaque phase de pooling (voir section 2.2.5) et les combine pour obtenir une carte de prédiction. Cette carte a la même taille que les cartes de features qui ont été combinées. Elle sera donc sur-échantillonnée postérieurement afin d'atteindre la taille de l'image totale. Plutôt que de réaliser un sur-échantillonnage bi-linéaire, le sur-échantillonnage sera une convolution transposée, qui peut être optimisée lors de l'entraînement du réseau. Ce réseau a été un grand pas en avant pour la segmentation, en devenant l'état de l'art de la segmentation en 2015.

Ronneberger et al. [89] s'appuient donc sur ce réseau pour construire le U-Net (voir figure 2.23) afin de segmenter des images de cellules. Leur réseau permet un entraînement à partir d'un moins grand nombre d'images et une plus grande précision. En effet, le grand nombre d'opérations de convolution dans la partie droite du réseau permet de conserver l'information de contexte sémantique à des résolutions plus grandes.

Pix2Pix

Pix2pix est une méthode qui utilise un GAN pour réaliser la transformation d'une image vers l'autre [91], les deux images étant appariées, par exemple des segmentations d'images, des images

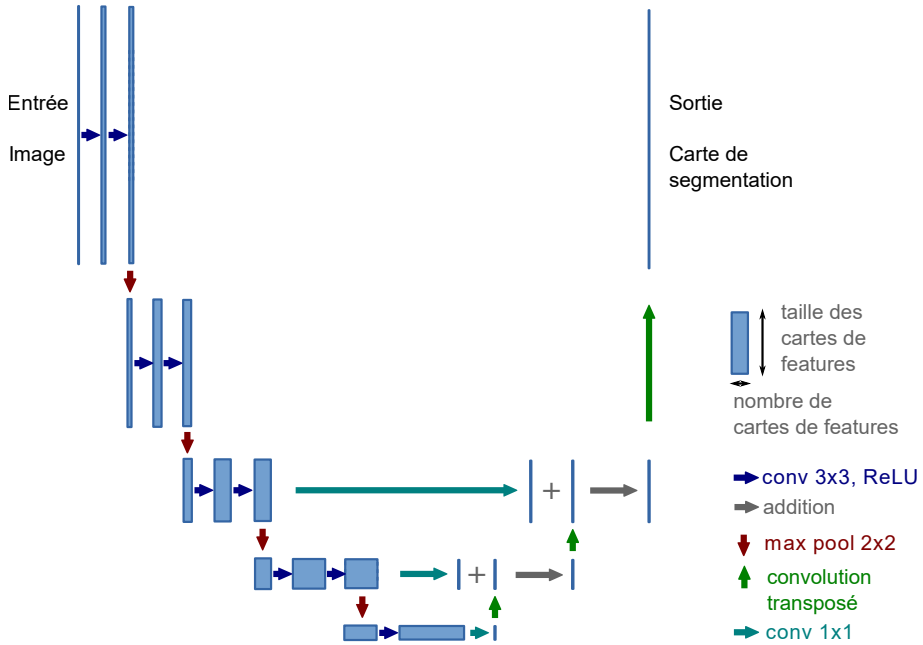


FIGURE 2.22 – **Fully convolutional network.** Représentation du premier réseau totalement convolutionnel. La partie gauche du U représente le réseau classifieur utilisé. Les convolutions 1x1 permettent de combiner les cartes de features dans le bloc dont elles partent pour obtenir une carte de segmentation. Cette carte de segmentation est grossière puisque de résolution inférieure à celle de l'image d'entrée. On appelle ces connections des connections de saut, car elles sautent le reste du réseau. Chaque carte de prédiction est additionnée à celle de l'étage d'en dessous, sur-échantillonnée par la convolution transposée. On obtient en fin de compte la grande carte de segmentation, de même taille que l'image d'entrée. Figure adaptée de [89].

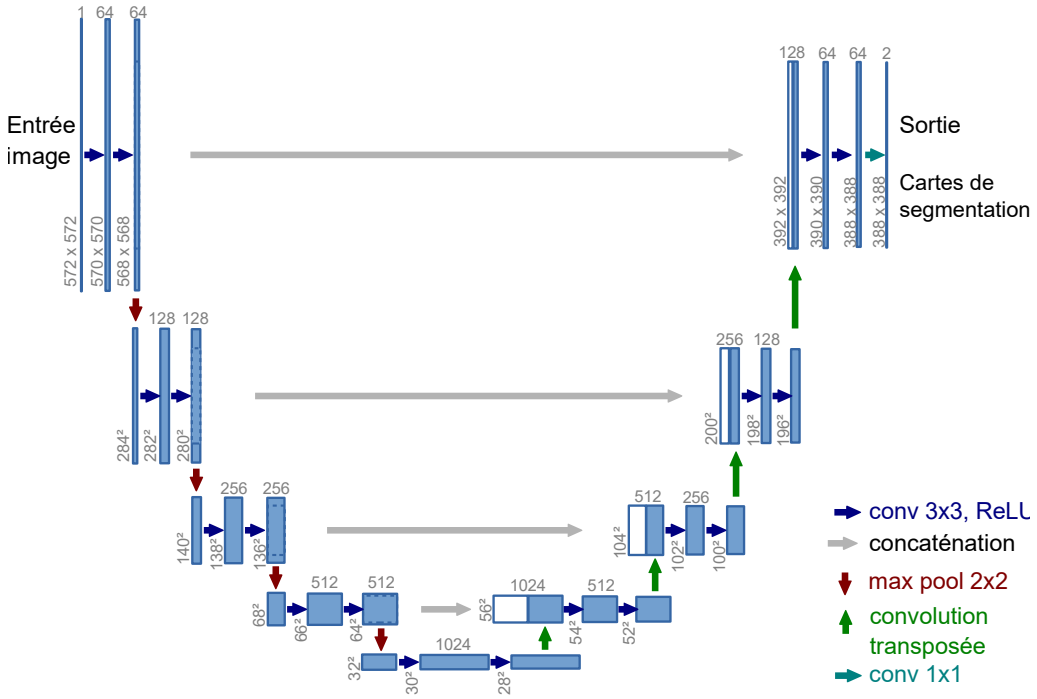


FIGURE 2.23 – **U-net.** Comme le fully convolutionnal network, la partie gauche du U correspond à l'architecture typique d'un réseau de classification. La différence principale avec le précédent réseau est l'ajout dans la partie droite d'un grand nombre de cartes de features. Ces cartes ne correspondent plus simplement à des cartes de segmentation de plus petite résolution, mais vont aider à mieux reconstruire la carte finale. On concatène les cartes de features du bloc de la partie gauche avec celui provenant de la convolution transposée de l'étage du dessous. Figure extraite de [89].

RGB traduites en noir et blanc, des objets et leurs contours, etc.

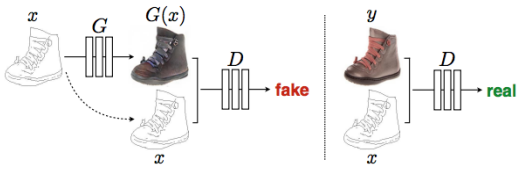


FIGURE 2.24 – **Schéma du réseau pix2pix.** Figure extraite de [91]

Comme le montre la figure 2.24, le but du discriminateur est de savoir reconnaître une vraie paire d'images d'une fausse. Le générateur a pour but de générer l'image-paire de celle qu'on lui donne en entrée, ou, du moins, une image qui trompera le discriminateur. Le discriminateur apprend les statistiques d'appariement des images et les transmet au générateur pour qu'il optimise ses poids. La structure du générateur est similaire à celle du DCGAN [68] mais complétée pour prendre la forme d'un U-net, avec les connections de saut. Le discriminateur est identique à celui du PatchGAN [92]. Ce dernier est un discriminateur qui ne voit que des patchs d'image de 70 par 70 se chevauchant. Il prédit, pour chacun des patchs s'il provient d'une vraie image. Le discriminateur (D sur la figure 2.24) prend en entrée l'image générée et l'image dont elle provient, ce qui permet de forcer la corrélation entre les deux. La force de l'utilisation du GAN ici, est de permettre l'obtention d'images ayant l'air réelles, voir figure 2.25. En effet, avant pix2pix, les fonctions de coût pour

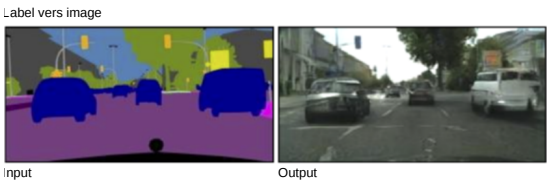


FIGURE 2.25 – **Exemple de transformation d'image à image.** Figure extraite de [91]

entraîner le générateur d'images était conçue manuellement. Par exemple, une possibilité est de minimiser pour chaque paire d'images la distance euclidienne, pixel à pixel, entre l'image synthétisée et l'image cible. Cela donne des résultats flous [93]. Le discriminateur permet de résoudre des problèmes de haut niveau, sans nécessiter leur traduction en fonction de coût. Si la problématique se formule : "je souhaite obtenir des images qui ont l'air réelles", le réseau va apprendre seul à traduire cela en termes techniques pour le générateur. Ce type de réseau est aussi appelé conditionnel, dans le sens où l'image d'arrivée est conditionnée par l'image de départ.

2.4.2 Images non-appariées, apprentissage non-supervisé

On s'est intéressé précédemment à des cas où il existe des paires alignées dans les datasets d'images sources et cibles. Or, pour un grand nombre de tâches, il n'existe pas d'images appariées, seulement une correspondance entre les datasets source et cible. Des exemples de ce type de problèmes sont le transfert de style (voir figure 2.19), le transfert de saison, l'amélioration de la qualité de photographies, etc...

Cycle-consistency GAN ou CycleGAN

Le CycleGAN [94], schématisé figure 2.27, permet de trouver une transformation contrôlée entre deux datasets. Il a notamment permis de transformer des chevaux en zèbres, des photographies en peintures de Monet (figure 2.26), des vues aériennes en cartes, des pommes en oranges... Il est en fait particulièrement adapté à toutes les transformations requérant des changements de texture, mais beaucoup

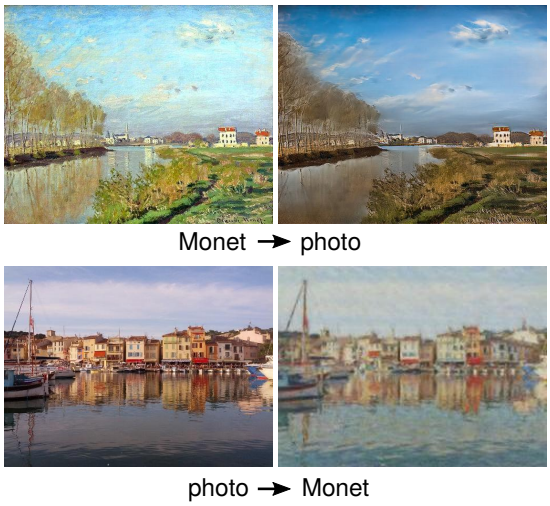


FIGURE 2.26 – **D'un tableau à une photo et inversement.** Exemple de traduction de tableau de Monet en photographie et inversement, par 2 générateurs entraînés simultanément. Figure extraite de [94]

moins pour des tâches nécessitant des transformations géométriques comme transformer des chats en chiens.

L'entraînement des réseaux se fait à l'aide de trois fonctions de coût différentes. La première est celle du discriminateur, qui pénalise une image qui ne ressemble pas à une image du domaine cible. La seconde est celle de la cohérence de cycle, après un passage dans un générateur puis l'autre, on doit retomber sur la même image. Enfin, la troisième est la fonction de coût d'identité : une image x qui passe dans le générateur qui a pour cible de domaine X doit rester elle-même. Les structures des réseaux génératifs sont reprises de [79]. Ils sont composés de deux couches de convolution sous-échantillonnante (stride de 2), de plusieurs blocs résiduels, puis de 2 couches de convolutions transposées, permettant le sur-échantillonnage. Comme pour Pix2pix 2.4.1, le discriminateur est un PatchGAN [92].

Il a été montré [95] que les réseaux entraînés de cette façon pouvaient réussir à cacher de l'information sous forme de détails à des fréquences très élevées, qui ne sont pas visibles par l'oeil humain. Ainsi, lors de la reconstruction de photographies aériennes depuis des cartes de type Google Maps, des détails de photographie sont reconstruits alors qu'ils seraient a priori impossibles à inférer à partir d'une carte. Le réseau cache de l'information dans la carte obtenue à partir de la photographie, pour permettre une meilleure reconstruction. La fonction de coût associée à la cohérence de cycle peut donc avoir des effets pervers.

StarGAN, un CycleGAN multi-domaines

Le CycleGAN ne peut prendre en compte que deux domaines, chacun étant le domaine cible de l'autre. Mais si l'on veut, à partir de pommes, générer des oranges, des pèches, des poires, comment fait-on ? StarGAN permet cette projection vers un nombre arbitraire de domaines, les uns sur les autres [96]. Il est possible de résoudre ce problème avec des CycleGANs mais cela aurait nécessité un entraînement par couple de domaines. Choi et al. [96] proposent d'utiliser un générateur et un discriminateur conditionnels pour l'entraînement (voir figure 2.28). À chaque domaine correspond un label, une classe. Pour chaque image, le discriminateur va apprendre s'il s'agit d'une vraie image ou bien d'une image de synthèse, ainsi que le label de l'image quand il s'agit d'une vraie : deux termes dans la fonction de coût. Dans le même temps, le générateur apprend à générer des images de qualité en empêchant le discriminateur de les distinguer des vraies images (fonction de coût opposée à celle du discriminateur). Mais, il apprend aussi à générer des images associées à cha-

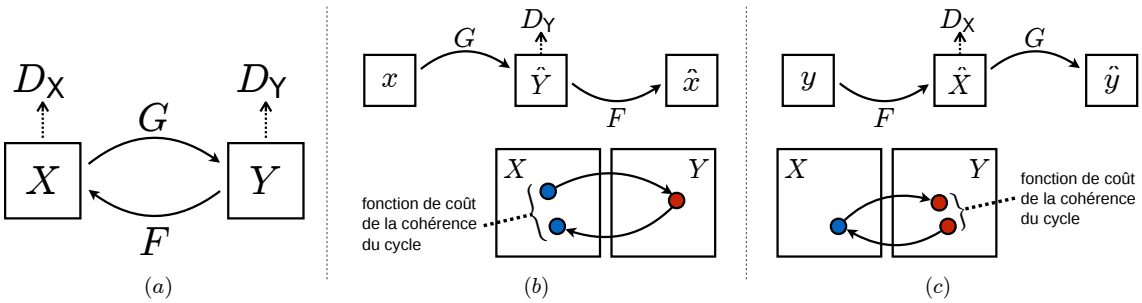


FIGURE 2.27 – **Schéma du réseau cycleGAN.** (a) Le réseau G transforme les images du dataset X vers le dataset Y, le réseau F de Y vers X. (b) Le discriminateur D_X est chargé de différencier les vraies images x des fausses $\hat{x}$ ou $F(x)$, générées à partir de $\hat{Y}$. Il en est de même pour le discriminateur D_Y avec les images du dataset Y. Afin de contraindre encore les deux transformations, deux fonctions de coût supplémentaires sont introduites : la première s'assure que lorsqu'une image de X est transformée vers Y puis, de là, vers X, l'image que l'on obtient est proche de l'image de départ au sens de la norme L_1 ; (c) la seconde est l'inverse de la première. Figure extraite de [94].

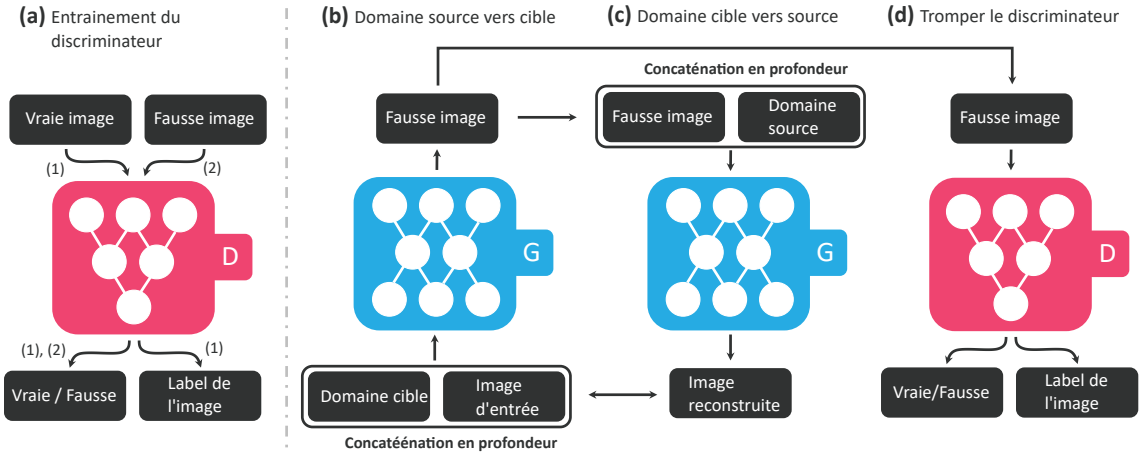


FIGURE 2.28 – **Schéma de l'entraînement de StarGAN.** (a) Le discriminateur D discrimine les vraies des fausses images et apprend à retrouver le label des vraies images. (b) Le générateur G transforme l'image d'entrée vers le domaine cible. (c) Le même générateur G re-transforme l'image générée vers le domaine de départ. L'image ainsi reconstruite doit être proche de l'image d'entrée. (d) La fausse image doit être classée dans le bon domaine (label) par le discriminateur et être classée comme étant vraie. Figure adaptée de [96].

cun des labels. Pour cela, un vecteur de label (des zéros pour toutes les positions sauf celle correspondant au label que l'on souhaite générer) est concaténé à chaque position de l'image d'entrée. Le générateur tente de produire une image de ce label. Pour s'assurer que la transformation génère une image du bon domaine, le discriminateur la catégorise et rétropropage l'erreur commise si c'est le cas. L'entraînement conserve la cohérence de cycle du cycleGAN : après avoir généré une image d'un autre label, on essaie de revenir à l'image de départ.
Récemment, une version de starGAN reprenant le principe de bloc de self-attention a été publiée [97]. Sur le dataset de transformation de visages vers divers domaines (femme, homme, couleurs de cheveux), les auteurs rapportent une amélioration des fonds des images, qui nécessitent une cohérence à longue distance.

2.4.3 Application en biologie et imagerie médicale

Les GANs en général et les générateurs d'image vers image en particulier, commencent à être utilisés de plus en plus massivement dans le domaine de l'imagerie bio-médicale [98].
La reconstruction d'images (avec pix2pix par exemple) a pu être utilisée comme moyen de post-traitement des images de scanner, afin de les débruiter [99], et ainsi de diminuer la quantité de radiations reçue par les patients et de désengorger les services d'imagerie médicale, tout en améliorant la résolution des clichés obtenus [100]. CycleGAN a aussi été utilisé pour débruiter les structures fines des microtubules dans des images de criblage à haut-contenu, en augmentant ainsi le débit de la technique d'imagerie [101].
Les GANs permettent l'anonymisation des données. En effet, la génération de données par les GANs peut compenser le manque de données disponibles pour les cas pathologiques, plus rares que les cas contrôles. Ainsi, Frid-Adar et al. [102] ont amélioré la sensibilité et la spécificité de leur classifieur de lésions du foie en équilibrant les différentes classes dans leur dataset grâce à un DCGAN. Le même procédé est appliqué par Bailo et al. [103] pour augmenter des datasets de globules rouges à des fins de segmentation et de détection.
Les modèles CycleGAN et pix2pix rendent également possibles les transformations entre types d'imagerie, avec pix2pix quand des exemples appariés d'images existent pour un même patient et avec cycleGAN sinon. Wolterink et al. montrent que pour obtenir des images de scanner à partir d'images d'IRM, le modèle CycleGAN est préférable au pix2pix, car moins sensible aux déformations non-rigides des tissus mous (bouche, gorge, etc).
En criblage à haut-contenu, starGAN a été utilisé pour égaliser les batchs, afin de gommer les variations nuisibles à l'interprétation biologique [104].

2.5 Sous-populations d'images : espaces latents et clustering

Lorsqu'on parle d'hétérogénéité cellulaire, on pense à celle qui est présente dans les tissus, les organes, mais pas forcément à celle présente dans les cultures cellulaires pour lesquelles sont plus souvent utilisées des cellules clones. L'hétérogénéité entre cellules ne prend pas seulement son origine dans la génétique, mais aussi dans l'abondance relative de ses ARNs, de ses protéines [105], des phases cellulaires par exemple.

En criblage à haut contenu, l'hétérogénéité cellulaire a souvent été ignorée ou perçue comme empêchant la compréhension des perturbations imposées aux cultures cellulaires [106]. Une revue de 2014 [107] pointe ainsi du doigt le fait que 60 à 80% des études de criblage à haut-contenu n'utilisent qu'une ou deux features extraites de l'image, réduisant ainsi la richesse du criblage. Certains criblages ne nécessitent cependant qu'un petit nombre de features pour obtenir de très bons résultats. Une autre pression à la réduction de dimension est celle héritée de l'organisation de la recherche de criblage à haut débit et des contrôles qualité des essais, qui poussent à obtenir un facteur Z' (critère univarié) permettant de séparer les contrôles.
Pourtant des solutions existent pour mettre à profit cette complexité. L'une d'entre elles est la recherche de sous-populations cellulaires [106]. En effet, on sait que des sous-populations cellulaires différentes peuvent répondre de différentes manières à une perturbation, un traitement. On peut envisager de modéliser toutes les sous-populations des expériences et, par la suite, de définir les traitements par les fréquences relatives de ces sous-populations. Cela donne un profil pour chacun des traitements que l'on peut alors comparer aux traitements-contrôles et entre eux.

Ce workflow revient à obtenir des features représentant chaque image d'un dataset, puis à réaliser un clustering ou regroupement des images. La plupart du temps, les images sont représentées par des vecteurs de features de haute dimension. Il sera nécessaire d'adapter l'outil de clustering, ou bien de réduire la dimension des données.

2.5.1 Représentation d'image par des features

La qualité du clustering dépend principalement de la qualité de la représentation des cellules ou des images par le vecteur de features qui les résume. Ainsi, obtenir de bonnes représentations est essentiel. Est présentée ci-dessous, une liste non exhaustive de méthodes de création de représentations.

"Features off-the-shelf"

Comme nous l'avons vu précédemment 2.2.4, il est possible d'utiliser un réseau entraîné pour de classification comme extracteur de features (voir section 2.2.4). Le réseau peut être entraîné sur le dataset considéré ou à défaut sur un dataset différent, comme Imagenet [38].

Auto-encodeurs

Un auto-encoder est un réseau qui apprend de façon non supervisée la représentation d'image. En effet, il est composé de deux parties (voir figure 2.29), une partie "encodeur" qui compresse l'image en l'encodant dans un espace de dimension bien plus petite [109]. À cette fin, la couche centrale du réseau est réduite en nombre de neurones. La deuxième partie du réseau est le "décodeur". Celui-ci, à partir de la représentation compressée, va essayer de reconstruire l'image de départ. L'entraînement est réalisé en minimisant l'erreur de reconstruction de l'image.

Une architecture d'auto-encodeur intéressante est l'auto-encoder variationnel ou VAE [110]. Les neurones

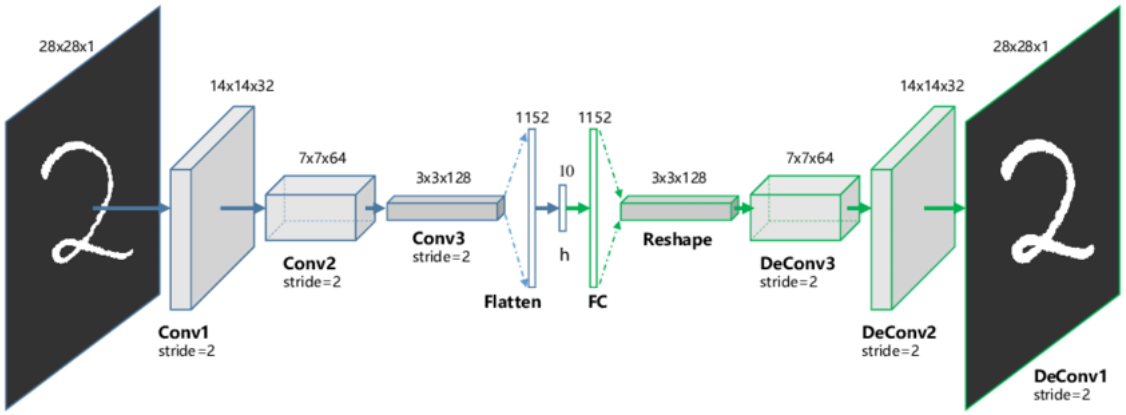


FIGURE 2.29 – **Autoencoder.** Figure extraite de [108].

de la couche centrale, correspondant à un goulot d'étranglement, ne sont pas directement utilisés par la couche suivante (i.e la première couche du décodeur). À la place, ces neurones correspondent aux statistiques de distributions gaussiennes desquelles seront tirées aléatoirement les valeurs des neurones qui seront utilisées par le décodeur. Cette architecture permet d'imposer une certaine continuité à l'espace dans lequel se trouvent les représentations.

Réseau entraîné par fonction triplet

Introduite par Weinberger et al. [111], la fonction d'erreur par triplet permet d'apprendre une représentation des images par comparaison de distances. Afin d'entraîner le réseau, trois images sont nécessaires pour calculer l'erreur : une image ancre, une image positive et une image négative. En termes de représentation, l'image positive doit être plus proche de l'image ancre que l'image négative (voir figure 2.30). La fonction de coût est calculée en fonction des distances des deux paires $distance_-$ et $distance_+$ à l'aide de la formule $\mathcal{L} = max(d_+ - d_- + marge, 0)$. La fonction doit être minimisée pour que l'instance positive soit plus proche de l'ancre d'une

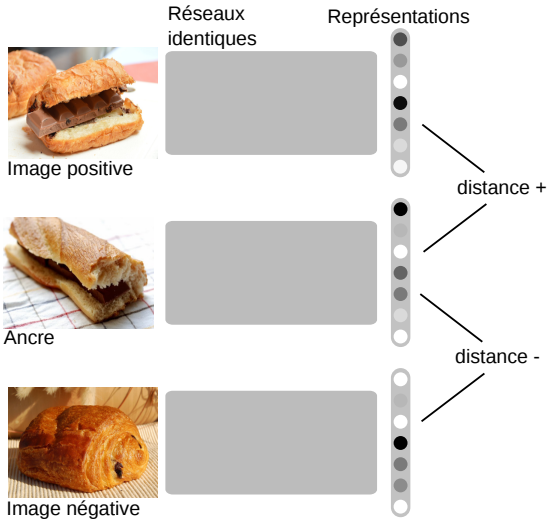


FIGURE 2.30 – **Triplet loss.** Sur le schéma, les blocs gris correspondent au même réseau, réalisant les inférences de features pour trois images différentes. La distance entre les représentations des deux pains au chocolat ($distance_+$) doit être plus petite que celle entre le pain au chocolat ancre et la chocolatine ($distance_-$).

certaine marge par rapport à l'instance négative. On n'apprend donc plus une classification mais une métrique.
La difficulté pour entraîner des réseaux avec cette fonction de coût est de trouver des triplets d'images difficiles pour le réseau, c'est-à-dire des triplets pour lesquels l'image négative est à la même distance ou plus proche de l'image ancre que l'image positive. Le nombre de triplets d'images de la sorte va diminuer au cours de l'entraînement, à mesure que le réseau devient plus performant. De nombreuses études s'intéressent à l'optimisation de cette recherche de triplets [112, 113].

Une autre problématique pour ce type d'entraînement est sur quel dataset entraîner le réseau. Ando et al. [114] montrent qu'il est possible de transférer les capacités d'un réseau entraîné à apprendre une métrique sur un dataset d'images naturelles pour une application avec des images biologiques. Après une recherche par texte sur ordinateur, les images sélectionnées pour le même texte sont considérées similaires. Cela a permis à des chercheurs de Google de constituer un dataset de similarité sur lequel un réseau est entraîné. Ando et al.[114] utilisent ce même réseau pour classifier des images cellules, traitées avec des molécules différentes. Ils parviennent à identifier les traitements ayant le même mécanisme d'action cellulaire (dataset BBBC021 [63]). Ils réalisent en plus une réduction de dimension par analyse en composantes principales (PCA), une normalisation des axes et un alignement des batchs par alignement des matrices de covariances des contrôles négatifs.

Générateur d'images et retour au vecteur latent

Une autre façon d'obtenir des features à partir d'images peut être d'utiliser des GANs. Les GANs apprennent à générer des images à partir d'un vecteur latent. Cependant, retrouver le vecteur latent, pour une image donnée, n'est pas aisé.
Dumoulin et al.[115] entraînent conjointement un réseau à générer des images et à inférer le vecteur latent d'une image.
Le styleGAN (cf paragraphe 2.3.5) est utilisé pour générer des images crédibles à partir de vecteurs latents. Très récemment, la version 2 de l'architecture et de l'entraînement [116] permettent d'inverser le processus et de retrouver le vecteur latent correspondant à une image.

2.5.2 Réduction de dimension

Une fois des features obtenues pour chaque image, il est possible de réaliser le clustering. Cependant, la plupart du temps, le nombre de features est grand : les données sont représentées dans un espace de grande dimension (150 à 1000). Or, en grande dimension, les distances que l'on a l'habitude de manipuler, n'ont plus le même sens, c'est ce que l'on appelle communément le fléau de la dimensionnalité. Les distances sont de plus en plus concentrées [117]. Cette concentration des distances affecte le bon fonctionnement des algorithmes de clustering qui s'appuient sur des mesures de distance pour déterminer les zones de grandes densités et la similarité entre les points [118].
Heureusement, les données sont rarement réparties uniformément dans cet espace de grande dimension et sont le plus souvent concentrées dans un repliement de dimension, comme une feuille repliée vit dans un espace à trois dimen-

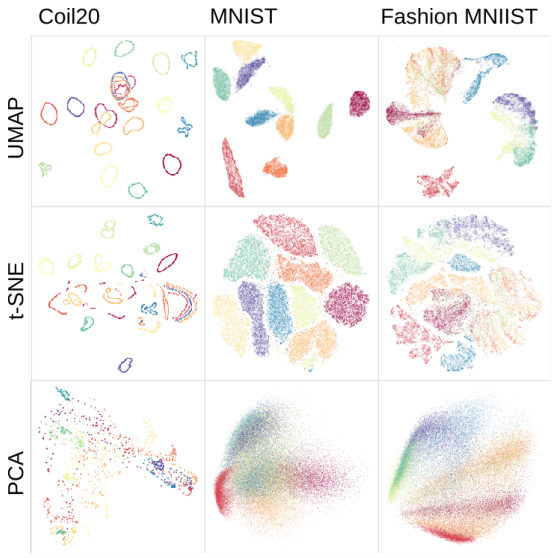


FIGURE 2.31 – **Comparaison d'algorithme de réduction de dimension sur 3 différents datasets.** Figure adaptée de [119].

sions alors que les points qui la composent sont en deux dimensions. Il est donc commun de réaliser une étape de réduction de dimension avant d'utiliser un clustering. Les trois techniques présentées par la suite sont comparées figure 2.31.

Analyse en composantes principales (PCA)

L'analyse en composantes principales ou PCA permet de passer d'un espace de départ, où les features peuvent être très corrélées, à un espace où les features sont linéairement indépendantes. Les dimensions dans l'espace d'arrivée sont des combinaisons linéaires des dimensions de départ. Ces combinaisons permettent de définir l'axe de plus grande variabilité, puis le second, orthogonal au premier, de la plus grande variance possible, etc... L'avantage principal de la PCA est de permettre l'interprétabilité des axes de projection si les features de départ ont un sens.

tSNE

La méthode de visualisation de données par t-SNE [120] est non linéaire, elle ne préserve pas la structure de distance des données, contrairement à la PCA. Elle permet de s'appuyer sur les similarités locales des points pour transformer les données dans un espace de faible dimension. La méthode t-SNE essaye, pour chaque point, de faire correspondre la similarité avec ses N plus proches voisins en grande dimension avec leur similarité en faible dimension. Les voisins sont choisis de façon stochastique avec un poids plus élevé pour les voisins les plus proches, ce qui permet de conserver une certaine cohérence à moyenne distance.

Uniform Manifold Approximation and Projection ou UMAP

Le Uniform Manifold Approximation and Projection (UMAP) apprend à déplier un espace de faible dimension dans lequel les points sont regroupés afin d'obtenir un espace de représentation de dimension inférieure à celle de l'espace global [119]. Si l'objectif du t-SNE et du UMAP est le même, leurs fondements mathématiques diffèrent fortement. Cependant, à la différence de t-SNE, UMAP est un algorithme de projection : un point qui n'a pas été utilisé pour la réduction de dimension peut être projeté dans le nouvel espace. En termes de visualisation de dataset, UMAP donne des résultats de qualité comparable à l'algorithme de tSNE mais préserve davantage la conformation globale du dataset. Le tSNE ne pénalise pas des points proches dans l'espace de représentation alors qu'ils sont lointains dans l'espace de départ. En termes de rapidité, UMAP est beaucoup plus performant et permet de traiter des data-

sets plus larges et en plus grandes dimensions.

Ces deux méthodes non-linéaires présentent l'inconvénient de ne pas (ou peu) conserver la notion de densité et de distance, puisque ces notions varient localement pour l'algorithme. Cela peut poser problème pour obtenir des clusters à partir de méthodes basées sur ces concepts.

2.5.3 Méthodes de Clustering

L'apprentissage de regroupement d'objets, dit clustering s'effectue de manière non-supervisé : il ne nécessite pas de label pour les données d'entrée. Le but de l'algorithme est de s'adapter à la structure des données.
Comme illustré figure 2.32, les performances en termes de rapidité d'exécution sont très variables selon les méthodes adoptées et dépendent, pour certaines énormément, du nombre d'objets à regrouper en clusters.
Certaines méthodes nécessitent de notifier explicitement le nombre de clusters que l'on attend.
Nous présentons par la suite trois techniques de clustering, disponibles dans la librairie python *scikit-learn* [122].

K-means

Le K-means ou K-moyenne est un des algorithmes les plus simples, les plus rapides et les plus utilisés. Il procède itérativement en commençant par K centres. Chaque point du dataset va être assigné au cluster le plus proche. Puis, les centres des clusters sont recalculés. Ces deux étapes sont répétées jusqu'à convergence de l'algorithme, i.e. lorsque les centres ne bougent plus. Plus qu'un algorithme de clustering, il s'agit d'un algorithme de partitionnement : il partitionne le dataset en cellules de Voronoï, en minimisant les distances intra-partitions.

Propagation par affinité

La propagation par affinité [123] regroupe les points en identifiant des exemples représentatifs des clusters. L'algorithme est basé sur la recherche itérative de la pertinence d'un point pour représenter un autre point. La pertinence est fonction de la similarité du point avec son potentiel représentant mais aussi des choix des points qui lui sont similaires. L'algorithme se base donc sur une mesure de similarité entre chacun des points. Le clustering par propagation d'affinité ressemble également à un partitionnement de l'espace.

HDBSCAN

Le clustering spatial basé sur la densité (DBSCAN [124]) regroupe les points qui sont densément regroupés et séparés par des zones de densité plus faible, reléguant les autres points au statut de bruit. Un point considéré comme étant du bruit peut être labellisé de façon "douce", en lui assignant des probabilités d'appartenir à l'un ou à l'autre des clusters. Cette méthode ne requiert pas de nombre de clusters en paramètre d'entrée. En revanche, elle se base uniquement sur la densité des points, et nécessite deux paramètres pour évaluer ce que l'on appelle dense.
Le DBSCAN hiérarchique ou HDBSCAN [125] permet de clusteriser correctement des clusters n'ayant pas la même densité. Le seul pré-requis est qu'une zone de moindre densité les sépare. Pour arriver à cela, le HDBSCAN fait varier la notion de densité et sépare les zones

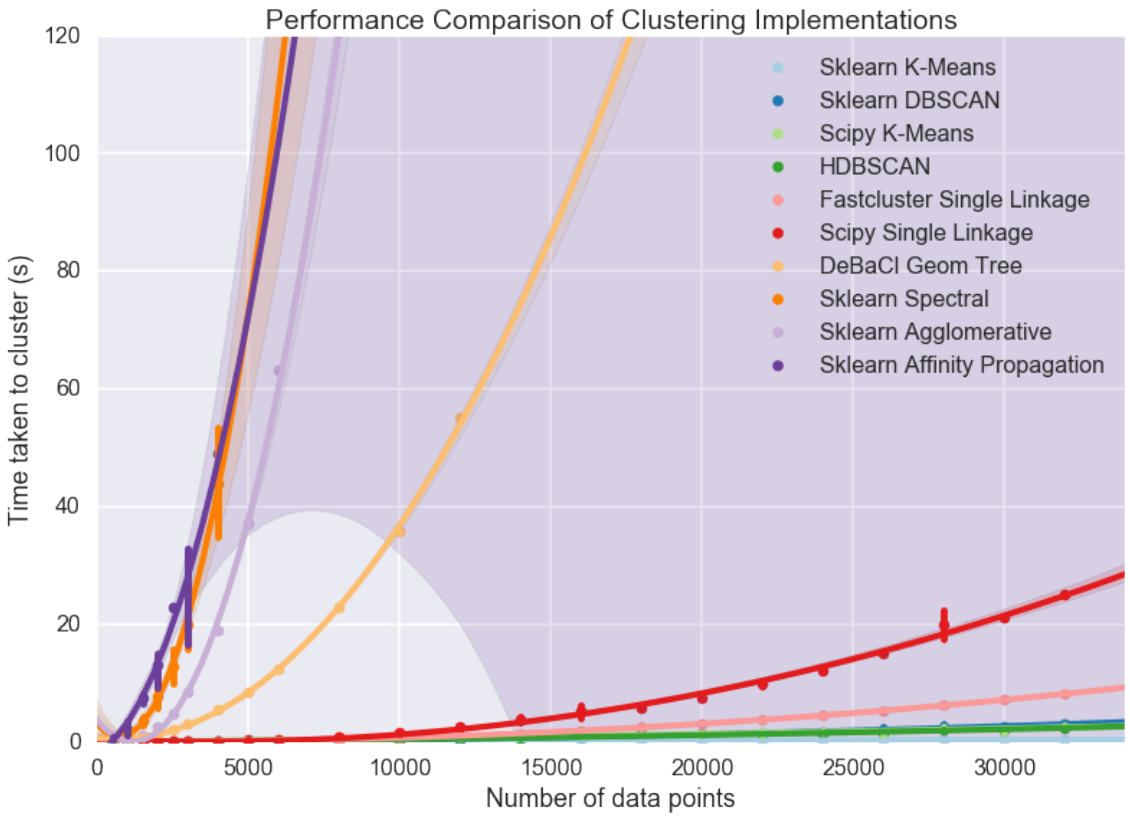


FIGURE 2.32 – **Performance de diverses implémentations de méthodes de clustering.** Figure extraite de [121]

de regroupement en augmentant la valeur de la densité seuil, créant ainsi un arbre de clusters.
Cette méthode de clustering est censée bien fonctionner en grande dimension (jusque environ 50 d'après les auteurs).

2.6 Conclusion

Ce chapitre vise à donner un aperçu des différents mécanismes moléculaires à l'œuvre dans la maladie de Parkinson, de l'extrême diversité des outils développés en deep learning ces dix dernières années, ainsi que de la variété des applications que l'on retrouve dans le sous domaine de l'analyse d'images. Nous avons en particulier insisté sur la possibilité d'obtenir une représentation d'un dataset d'images sous forme de sous-populations, à l'aide de technique de clustering.
La suite de ce manuscrit tentera d'articuler ces différents concepts et techniques afin d'élucider les différences phénotypiques entre deux cultures de neurones isogéniques à une mutation près, identifiant un risque accru de développer la maladie de Parkinson chez les personnes qui en sont porteuses.

Chapitre 3

Classification de phénotypes

3.1 Introduction

Dans ce chapitre, il est premièrement question de savoir s'il est possible de distinguer les deux contrôles cellulaires du crible de Ksilink, d'abord en utilisant l'extracteur de features WndCharm, puis grâce à des réseaux convolutionnels. Cette section de résultats est donc dédiée à la classification des images, et ce que l'on peut tirer comme connaissance de l'étude des réseaux classieurs que l'on entraîne.

3.2 Datasets

Les deux datasets utilisés dans la section proviennent de Ksilink. Ils sont tous les deux composés de plaques de culture multi-puits dans lesquels les neurones dopaminergiques (voir la section 1.3) sont cultivées.
Le premier dataset est composé de 12 plaques, avec un total de 148 puits dans lesquels une culture cellulaire est réalisée. Chaque image de puits correspond à un pavage de 20 différents champs de vision. Chaque champ comporte 3 canaux de fluorescence, un pour chaque marqueur moléculaire. La taille de chaque image est de 2560×2160 pixels. Sur chacune des plaques, on trouve autant de puits contenant des cellules WT (contrôle négatif) que de puits contenant des cellules mutantes G2019S issues de patient (contrôle positif).
Le second dataset est composé d'images ne provenant que d'une seule plaque, de 60 puits.

Les datasets sont compatibles, les 12 premières plaques du premier dataset ayant simplement été générées avant celles du second. L'avantage de travailler avec le deuxième dataset est de ne pas avoir à se préoccuper de la normalisation inter-plaques pour contrer les effets de batch. Le label d'une image est le label du puits dont on l'a extraite ("WT" ou "mutant : :LRRK2-G2019S").

3.3 WndCharm

Organisation du dataset et création d'imagette. Publié en 2008 et testé sur des datasets de biologie et de reconnaissance faciale, WndCharm [126] est une méthode de classification qui effectue une première étape d'extraction de features à partir des images. Le classifieur apprend par la suite à ranger les images dans des classes prédéfinies, en donnant des poids plus ou moins grands à chaque feature. Comparé à des classifieurs conçus spécifiquement pour des datasets de biologie, le classifieur de WndCharm est plus performant. Il possède l'avantage d'extraire des features généralisables à n'importe quel dataset.

L'algorithme WndCharm est singulier dans sa stratégie d'extraction des features. Il effectue des calculs identiques sur chacun des canaux de l'image, ainsi que sur leurs transformées de Fourier, Wavelet, et de Chebyshev. Les features calculées sont des décompositions polynomiales des features de texture, des features de contraste, et des features de distribution d'intensité de pixels.

Dans le but de vérifier si les images comportaient suffisamment d'information pour être classifiées, j'ai utilisé l'extracteur de features de WndCharm sur le premier dataset comportant 12 plaques. L'extraction renvoie environ 2500 features par image, en prenant en compte leurs trois canaux de fluorescence.

Les features ont des gammes de valeurs

Traitement des données	nombre de composantes nécessaires	nombre d'outliers sur les 3040 images
Données brutes	3	0
Mise à l'échelle, pas de seuil	9	0
Mise à l'échelle, seuil 1000	22	129
Mise à l'échelle, seuil 5	278	530

TABLE 3.1 – **PCA : Nombre de composantes nécessaires pour expliquer 99% de la variance.**

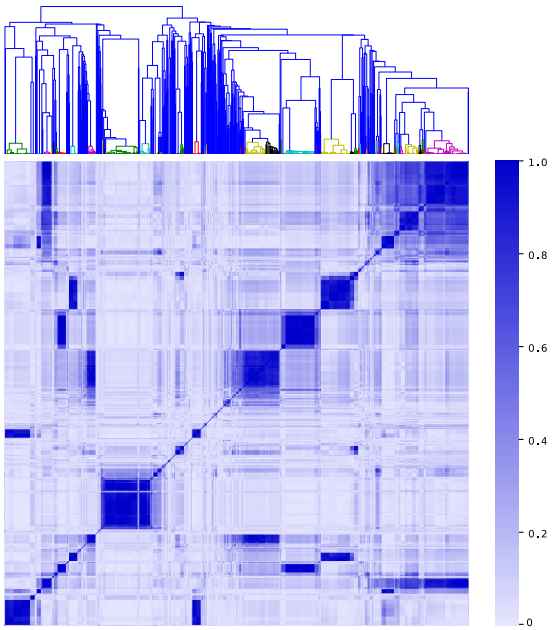


FIGURE 3.1 – **Matrice de corrélation des features, ordonnées par clusters hierarchiques.** La couleur bleue correspond à des features très corrélées.

très variables. De plus, des effets de plaques ou de batch sont susceptibles d'exister. Pour toutes ces raisons, les données sont traitées après une mise à l'échelle des features. Pour chaque feature et pour chaque plaque, la mise à l'échelle consiste à retirer la médiane des valeurs prises par les contrôles négatifs, et à diviser par l'écart entre deux centiles (ici 1% et 99%) du contrôle négatif. Ainsi, les contrôles négatifs de toutes les plaques sont alignés, et les gammes de valeurs prises par les features sont similaires entre elles.

On se demande à quel point les features sont corrélées, et dans combien de dimensions les données reposent. Pour cela, on peut tracer la matrice de corrélation des features 3.1. On remarque que de nombreuses features sont corrélées. On peut par ailleurs regarder combien de composantes sont nécessaires pour expliquer 99% de la variance (voir tableau 3.1). Pour chaque puits, si une feature normalisée dépasse une valeur seuil fixée, le puits est considéré comme outlier. La culture cellulaire est potentiellement trop différente des autres pour être inclue dans l'analyse. On remarque que, lorsqu'on enlève des puits outliers, de plus en plus de features sont nécessaires pour expliquer la variance. En effet, les outliers contribuent énormément à la variance, et donnent un poids démesuré à quelques features. Quel que soit le seuil de détection des outliers, il est clair que le nombre de composantes expliquant la variance est bien plus petit que le nombre total de features (2500). Les données évoluent dans un espace d'au maximum 280 dimensions.

J'ai ensuite entraîné un classifieur simple (méthode des k plus proches voisins) à reconnaître le contrôle négatif et le contrôle positif. Pour estimer l'accuracy, on réalise une validation croisée. On définit le dataset d'entraînement comme l'ensemble des plaques du dataset entier moins une. Cette plaque sert alors de dataset de validation sur lequel l'accuracy est calculée. Afin d'évaluer

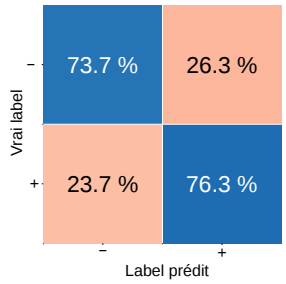


FIGURE 3.2 – **Matrice de confusion pour une classification par la méthode des k plus proches voisins.** k = 5. Classification réalisée avant normalisation des features par plaque.

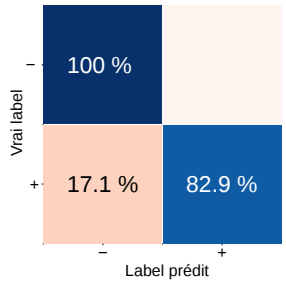


FIGURE 3.3 – **Matrice de confusion de la même classification après mise à l'échelle robuste de chaque plaque.**

au mieux l'accuracy, chaque plaque est utilisée tour à tour pour la validation. L'accuracy finale est la moyenne des accuracies calculées lors de ces différents entraînements. Figure 3.2, on trouve la matrice de confusion de la validation croisée, que l'on remplit grâce au label réel (WT et LRRK2-G2019S) de chaque image et au label prédit par le classifieur. L'accuracy est de 75%. Si l'on met à l'échelle les features (voir figure 3.3), on obtient une bien meilleure accuracy de 91.4%. Cette différence s'explique par le fait que la méthode de classification par les plus proches voisins repose sur la distance euclidienne, qui est fortement sensible aux gammes de valeurs des features.

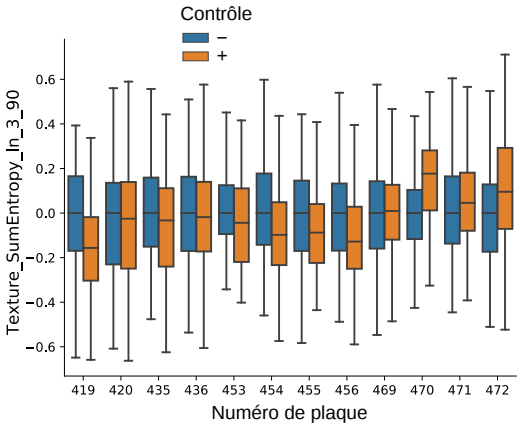
On peut également choisir de classifier à l'aide de l'analyse discriminante linéaire. Cette dernière permet de trouver l'axe de l'espace qui maximise la variance interclasse et minimise la variance intraclasse. Cette technique permet de donner un poids plus important aux features qui mènent à la meilleure classification. Ces features n'ont pas de noms explicites, et il est difficile de les interpréter. Autre fait notoire, les features avec les poids les plus importants ne varient pas toujours dans le même sens entre deux contôles sur chacune des plaques (voir figure 3.4).

Poids	Nom de feature
9521	Texture Entropy In FourierWavelet 3 90 ch1
8739	Texture SumEntropy In 3 0 ch0
6365	Texture SumEntropy In 3 90 ch0
6294	Texture Variance In 3 0 ch0
6240	Texture SumEntropy In 3 135 ch0
6186	Texture SumEntropy In Fourier-Cheby 4 0 ch0
6079	Texture SumEntropy In 4 90 ch1

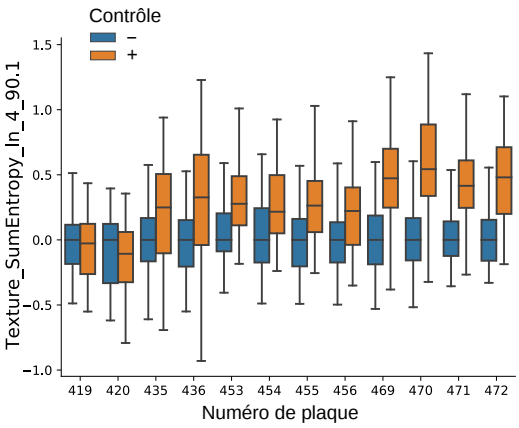
TABLE 3.2 – **7 features les plus discriminantes entre les deux classes.** ch0, ch1 et ch2 correspondent aux 3 canaux des images sur lequel la feature est calculée

Cette analyse a permis de montrer qu'il existe, entre les deux conditions contrôle, des différences suffisamment importantes pour pouvoir apprendre à les classifier avec une bonne précision. En revanche, les features étant difficilement interprétables, les raisons de cette bonne classification restent impénétrables. Est-il possible d'obtenir des résultats similaires de classification à l'aide d'outils de deep learning ? Et, si c'est le cas, comment permettre l'interprétation de la classification ?

3.4 Deep learning pour classifier



(a) 3ème feature la plus discriminante.



(b) 7e feature la plus discriminante.

FIGURE 3.4 – **Variation des valeurs prises par deux features pour chaque plaque et chaque condition.**

3.4.1 Réseau multi-échelles

Parmi nos interrogations sur les différences de phénotypes, se pose la question de l'échelle à laquelle elles se trouvent. En effet, les variations de phénotype pourraient se situer à l'échelle cellulaire, au niveau du corps cellulaire et de ses protubérances, ou bien encore à l'échelle du réseau cellulaire. Afin de prendre toutes les informations en considération pour réaliser la classification, j'ai utilisé un réseau multi-échelles. Dans un premier temps, j'ai souhaité réutiliser le réseau multi-échelles de Godinez & Al. [62]. Il est constitué de 7 branches réalisant des convolutions sur la même image, mais à des résolutions divisées par deux de branche en branche. Ce premier réseau est testé sur le premier dataset, composé d'images 2560 × 2160 pixels.

La figure 3.5 illustre un entraînement avec le premier dataset dont on a enlevé une plaque. La classification est globalement moins précise qu'avec WndCharm, les réseaux n'arrivant que rarement à 90% d'accuracy. L'entraînement est bruité, on verra que l'on aurait pu l'améliorer avec une planification de

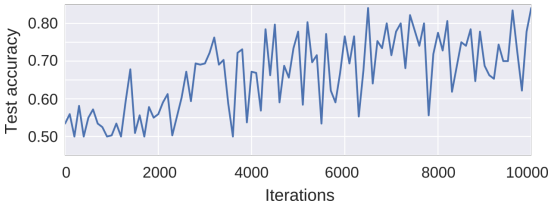


FIGURE 3.5 – **Exemple d'évolution de l'accuracy sur le dataset de validation.** Au fil des itérations successives, l'accuracy obtenue sur le dataset de validation est mesurée.

l'évolution du taux d'apprentissage (voir section 3.4.4).

L'observation des filtres de la première couche du réseau montre des motifs de bruit, très diférents des filtres bien définis du réseau AlexNet entraîné sur le dataset d'Imagenet (voir figure 2.5 de l'état de l'art). Des filtres réguliers ressemblant à des filtres de Gabor ou de contraste de couleurs rassurent quant à la convergence et à la généralisabilité des features apprises [26, 36].

Plusieurs raisons poussent à s'orienter vers d'autres réseaux. D'une part, le modèle est très lourd à entraîner sur des images aussi grandes (batch maximal de 1 image) : le réseau possède

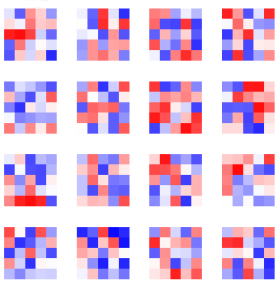


FIGURE 3.6 – **Filtres de convolution de la première couche du réseau.** Ces filtres correspondent à ceux de la première branche du réseau multi-échelles, et au canal de couleur rouge. Le rouge correspond à une valeur positive du poids de l'élément du filtre convolutionnel, le bleu à une valeur négative.

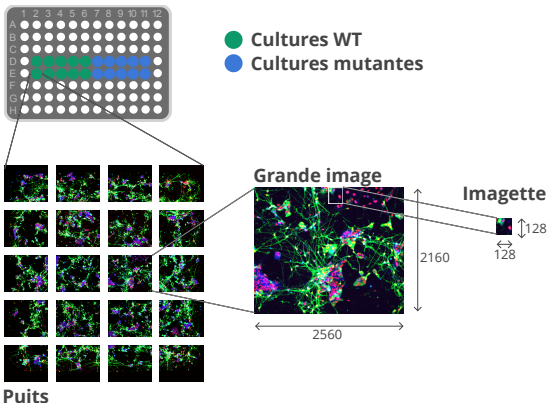


FIGURE 3.7 – **Organisation du dataset et création d'imagettes.** On attribue à chaque imagette la classe du puits dont elle provient.

42 millions de coefficients, et notre dataset ne compte que 2000 images, ce qui parait trop peu pour permettre la convergence du réseau sans risquer l'overfitting. D'autre part, chacune des branches n'est pas très profonde, et l'on verra, dans la section 3.4.5, que cela limite ses capacités.

Notons pour terminer qu'aucune étape de normalisation n'a été réalisée sur ce dataset avant l'entraînement du réseau. Il est fort à parier qu'un alignement des intensités des plaques permettrait sans doute d'obtenir de meilleurs résultats.

3.4.2 Dataset

Par la suite, on travaille avec des images plus petites, découpées dans les champs de vue larges (grandes images, voir figure 3.7) : on découpe chacun de ces champs en une multitude d'imagettes de 256×256 ou 128×128, décalées de 50px les unes par rapport aux autres. Les tailles et la valeur du décalage sont arbitraires.

Les essais de classification suivants sont réalisés sur le second dataset. Celui-ci permet de s'affranchir des variations plaque à plaque, et de se concentrer sur les différences essentielles entre les deux phénotypes.

Sur ce second dataset, avec la technique de découpage, on obtient un dataset d'environ 1 million d'images. Ce dataset comporte beaucoup de redondances. Si les imagettes pavent le champ de vue de façon disjointe, le dataset revient à 55 mille imagettes.

3.4.3 DenseNet

La méthode de transfert de connaissance en deep learning est très populaire. Elle consiste à utiliser un réseau pré-entraîné sur un grand dataset – le plus souvent Imagenet – et d'affiner les poids du réseau sur son propre dataset (voir section 2.2.4). Le transfert de connaissance est souvent utilisé lorsque l'on travaille sur de petits datasets.

On utilise cette méthode avec le réseau DenseNet (voir section 2.2.5). On obtient une accuracy de 81% sur les imagettes de 256×256px (voir figure 3.8). On décide que, lorsque plus de la moitié des imagettes d'un puits sont classées

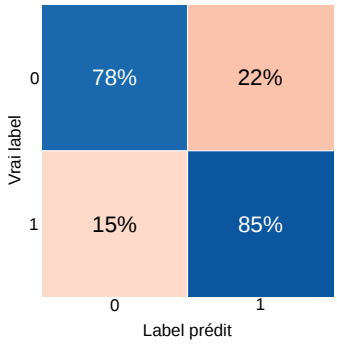


FIGURE 3.8 – **Matrice de confusion pour le réseau DenseNet appliqué à des images de** 256×256**px.**

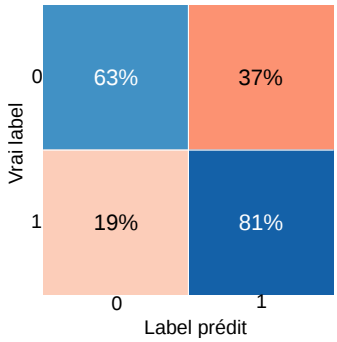


FIGURE 3.9 – **Matrice de confusion pour le réseau DenseNet sur des images de** 128×128**px.**

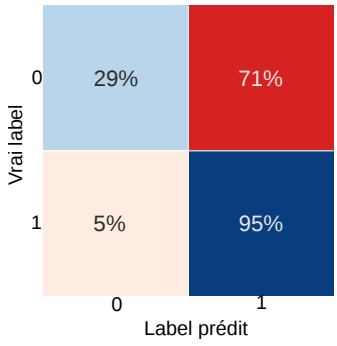


FIGURE 3.10 – **Matrice de confusion pour le réseau DenseNet sur des images de** 256×256**px, en égalisant le background des images.**

comme appartenant à l'un des deux labels, le puits est labellisé comme tel. Lorsque l'on agrège ainsi les résultats au niveau des puits, l'accuracy est alors de 100%.

Les inférences peuvent être faites sur des images de n'importe quelle taille. En effet, les dimensions de l'image vont induire les dimensions spatiales des blocs de features, obtenus après chaque couche de convolution. Après la dernière couche de ce type, les nouveaux réseaux présentent une couche sous-échantillonnage obtenu par moyennage global de chaque carte de features (voir figure 2.3 de l'état de l'art). Ainsi chaque carte est représentée par une seule valeur, sa moyenne. La classification se fait sur cette moyenne.

Dans le cas de nos images biologiques, notre hypothèse est que les phénotypes de chaque condition ne sont pas obligatoirement visibles sur chaque imagette. Plus grande est l'image prise en compte, plus certain sera le réseau de sa décision, puisqu'il intègrera plus d'information, et donc, probablement, de l'information relative au phénotype observé. Lorsque la taille de l'image d'entrée est réduite à 128×128, les inférences sont effectivement moins valides, avec une accuracy de 72% (voir figure 3.9). On notera que le réseau n'est pas ré-entraîné avec ces tailles d'images, on effectue les inférences avec le réseau précédent.

Lors de récents examens des images, on a pu remarquer que le background contenait beaucoup de bruit, sans doute des débris cellulaires. Afin de gommer ces informations, on décide, pour chaque canal de l'image, d'un seuil bas en dessous duquel il n'y aura pas de valeur de pixel. Tous les pixels situés au-dessous de cette valeur prendront la valeur-seuil. Ainsi, l'information contenue dans les débris – qui sont plus

fluorescents que le "vrai" background – sera effacée. Lorsque l'on réalise les inférences sur ces images débruitées, l'accuracy chute drastiquement de 81% à 62%. Deux hypothèses peuvent expliquer ce changement. (i) Le background des images contient de l'information différente entre la classe WT et la classe LRRK2-G2019S. L'égalisation du background retire de l'information, il en résulte une mauvaise accuracy pourla classe WT. Il est donc probable que cette classe contienne plus d'information encryptée dans le background comparativement à la classe LRRK2-G2019S. (ii) Le contraste entre le reste de l'image et le background est important pour la classification, et l'égalisation du background tend à diminuer ce contraste.
Il faut ré-entraîner les réseaux pour en avoir le cœur net : si c'est un problème de contraste qui a changé, alors le réseau s'adaptera lors du nouvel entraînement. Si c'est le contenu en lui-même du background de l'image qui impacte la classification, l'accuracy devrait rester affectée.

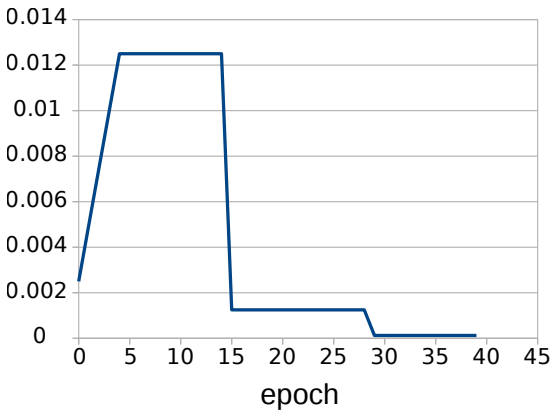


FIGURE 3.11 – **Évolution planifiée du taux d'apprentissage à travers les epochs.**

3.4.4 AlexNet et CBR-Small

On choisit d'entraîner un réseau plus ancien, AlexNet [26]. Par rapport à d'autre réseaux populaires, ce réseau est censé être moins biaisé vers la reconnaissance de la texture que de la forme [33] (voir section 2.2.3).

Le réseau est entraîné par transfert d'apprentissage, comme DenseNet l'était précédemment. De plus, on met ici en place un entraînement avec planification du taux d'apprentissage (voir figure 3.11). Le taux d'apprentissage permet de déterminer la vitesse de modification des poids du réseau. Un fort taux d'apprentissage induira une grande modification des paramètres en moyenne. L'entraînement commence

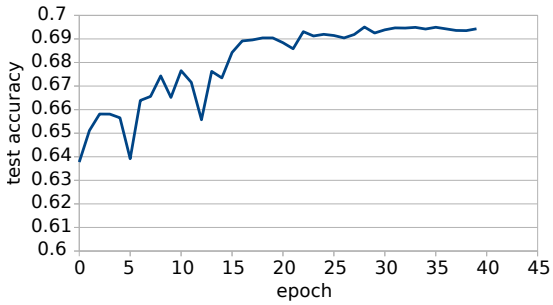


FIGURE 3.12 – **L'accuracy sur les données de validation à travers les epochs.**

avec un échauffement du taux d'apprentissage [127] qui évite la déstabilisation de l'entrainement par les couches profondes [128]. Puis on décroît le taux d'apprentissage afin d'affiner les paramètres. Sur la figure 3.12, on remarque que l'entraînement est beaucoup plus stable quand il touche à sa fin, i.e. le taux d'apprentissage est plus faible, les poids évoluent moins, et l'accuracy aussi.

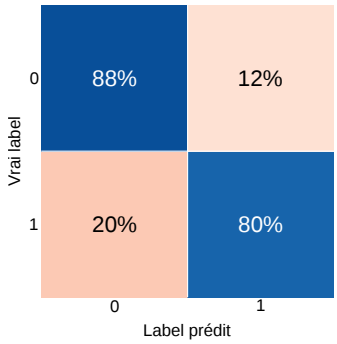


FIGURE 3.13 – **Matrice de confusion pour le réseau AlexNet sur des images de** 256×256**px avec background originel.**

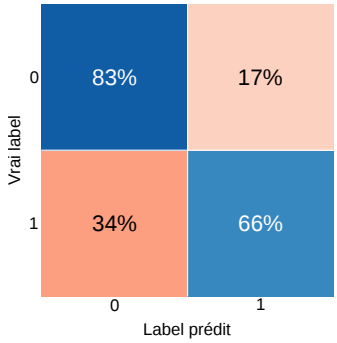


FIGURE 3.14 – **Matrice de confusion pour le réseau Alexnet sur des images de** 256×256**px avec égalisation du background.**

Sur les petites imagettes, pour le dataset avec le background, AlexNet atteint une accuracy de 84%. Cette accuracy descend à 74.5% lorsque l'on égalise le background. Cette diminution de l'accuracy tend à confirmer que le background contient une information différente dans chacun des deux types de culture de neurones WT et LRRK2-G2019S.

On entraîne également un réseau beaucoup plus petit que les autres, le CBR-Small de l'article de Raghu et al. [36]. L'entraînement a lieu sans transfert d'apprentissage. Les images mesurent 256×256px, et sont sous-échantillonnées pour que leur taille soit divisée par deux. Malgré l'égalisation du background des images, censée pénaliser l'apprentissage, le réseau parvient à une accuracy de 71%.

3.4.5 Champs réceptifs et nombres de paramètres des réseaux

Les valeurs des champs réceptifs de chaque réseau, la stride effective, ainsi que le nombre de poids qu'ils contiennent sont résumés dans le tableau 3.3. Le champ réceptif du réseau correspond ici à la taille de la zone de pixels d'entrée, qui influence un neurone tout au bout de la chaîne de convolution (voir section 2.2.3). Ce neurone appartient à une carte de features composée d'autres neurones. Les zones d'influence de deux neurones situés côte-à-côte sont décalées d'un certain nombre de pixels. Ce nombre est la stride effective.

On remarque que le champ réceptif du réseau multi-échelle de Godinez et al. [62] est très faible, mais il se calcule à différentes résolutions d'image. Toutefois, on peut mettre en doute sa capacité à observer des formes complexes.

Le réseau DeepLoc [64] qui n'a pas donné de résultat concluant sur notre dataset, contient 192 millions de paramètres dont 190 millions pour une série de trois couches denses (voir figure 2.2). Contrairement aux couches de convolution, le nombre de paramètres d'une couche dense (utilisée sans sous-échantillonnage par moyennage global) croît linéairement avec le nombre de

Réseau	Champ réceptif	Stride effective	Nombre de paramètres (en millions)
Multiscale Godinez	14*	1	42**
DeepLoc	50	8	2 + 190***
DenseNet	2071	32	12
AlexNet	195	32	57
CBR-Small	121	16	2

TABLE 3.3 – **Tableau récapitulatif des champs réceptif et nombres de paramètres pour les réseaux utilisés dans cette étude.** *pour chacune des résolutions. **pour classifier les grandes images. ***2 millions pour les convolutions, 190 pour les couches denses.

pixels de l'image d'entrée.
Le DenseNet, qui est de loin le réseau le plus profond du panel, possède un champ réceptif très grand.

3.5 Spatialité du phénotype

Les cultures des deux lignées neuronales étant très similaires, une hypothèse peut être émise : le phénotype des deux types de neurones se retrouvent dans chaque puits, quelle que soit sa classe. C'est ce que nous confirme la faible accuracy au niveau des imagettes. Afin de comprendre où se retrouvent ces différents phénotypes, cette partie s'articule autour de deux techniques découvrant la spatialité du phénotype sur l'image.

3.5.1 Spatialité du phénotype au niveau du puits

L'impossibilité de segmenter les cellules, et la volonté d'intégrer de l'information sur une portion d'image la plus grande possible, ne nous permettent pas de descendre à une résolution fine de la spatialité du phénotype. Une première tentative a été de mettre à profit le découpage de chaque puits en imagettes de 256 × 256 pixels décalées les unes par rapport aux autres de 50 par 50. En effet, ce décalage signifie que, 36 images (6x6) contiennent chacun de ces patchs de 50 par 50 (voir figure 3.15).

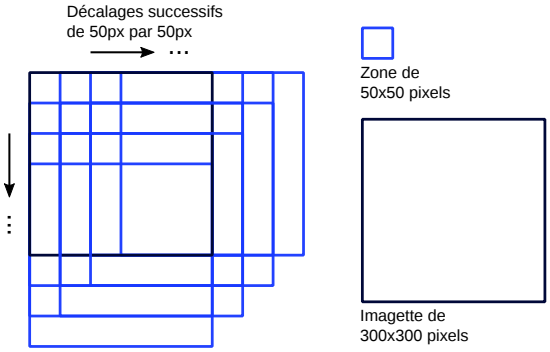


FIGURE 3.15 – **Découpage de la grande image en imagettes.** Les imagettes sont schématisées par les grands rectangles noirs et bleus. Le pavage par décalage crée artificiellement des carrés de 50x50 qui sont contenus dans plusieurs imagettes.

Le réseau DenseNet précédemment étudié permet de calculer, pour chaque patch, sa probabilité d'exposer un phénotype sain ou malade. On infère les probabilités pour chacune des images qui le contiennent, puis on réalise la moyenne de ces probabilités. Par cette moyenne, on approxime la probabilité de la contribution du patch à la classification en l'une ou l'autre des conditions.
La figure 3.16 illustre ce pavage sur un puits de la condition WT. La figure 3.17 sélectionne le détail encadré dans la figure précédente. Les différences de classification observées entre les images (b) et (c) sont dues à l'effondrement des performances du DenseNet lorsque les inférences sont réalisées sur des images à background égalisé. On note que les zones d'intense foisonnement d'excroissances neuronales restent classifiées par

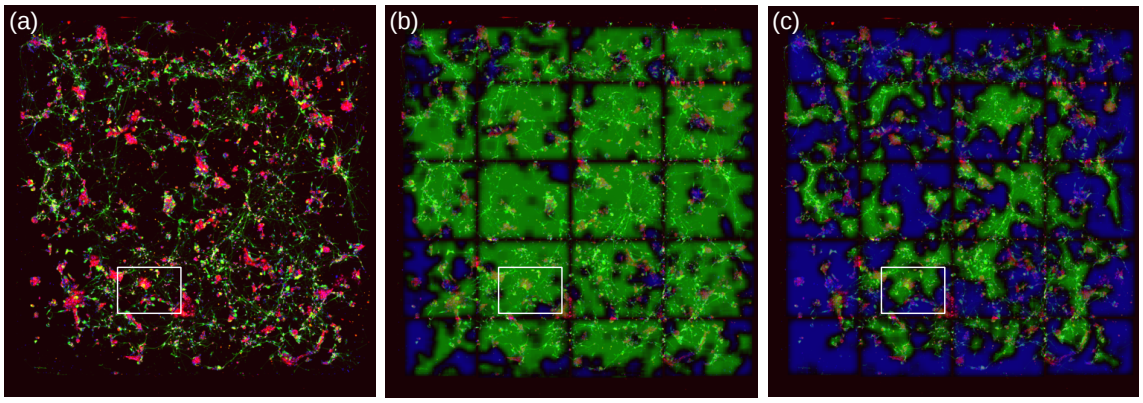


FIGURE 3.16 – **Pavage par patchs de** 50×50 **du puits.** Pour chaque patch, sa probabilité d'appartenir à une classe est représentée par l'intensité de sa couleur. Les inférences sont réalisées par DenseNet. Pour l'image de fluorescence : en vert le marqueur des neurones dopaminergiques (TH), en bleu le marqueur des noyaux (DAPI), en rouge le marqueur de l'alpha-synucléine. Pour les probabilités de patchs : en bleu la condition LRRK2-G2019S, en vert la condition WT. (a) Image du puits. (b) Probabilités inférées sur les images sans modification de background. (c) Probabilités inférées sur les images avec égalisation de background.

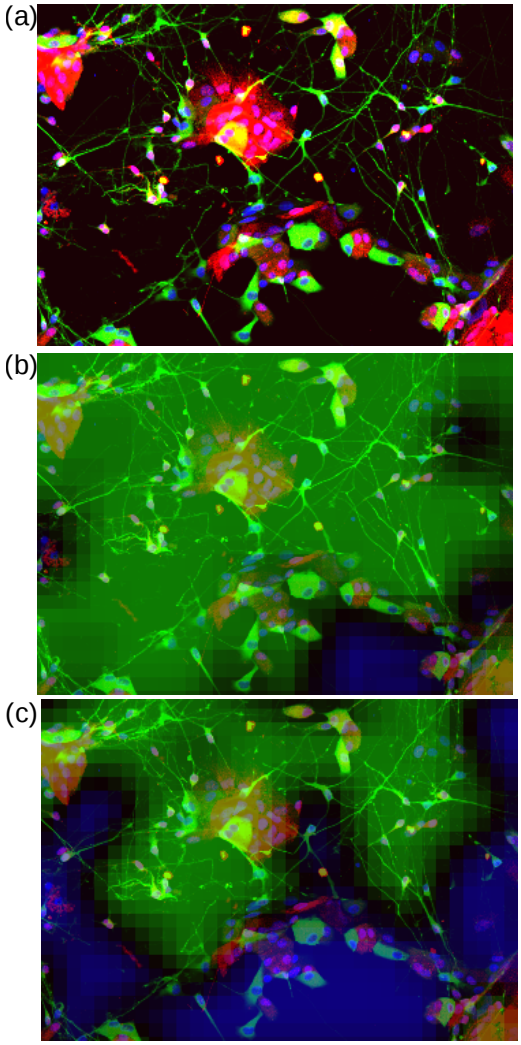


FIGURE 3.17 – **Détail du pavage.**

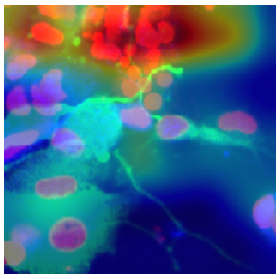


FIGURE 3.18 – **Superposition d'une image de fluorescence et de la carte d'activation correspondante pour la classe LRRK2-G2019S.** Les noyaux sont (exceptionnellement) représentés en rouge. Pour la carte de chaleur, le rouge correspond à une forte importance de la région, le bleu à une importance moindre.

le label WT, tandis que les zones dans le bas de l'image, et notamment les cellules contenant moins d'excroissances, voient modifiée leur classification vers le label LRRK2-G2019S.

3.5.2 Spatialité du phénotype au niveau de l'imagette

La méthode des patchs pour inférer le phénotype sur des plus petites zones est approximative. En effet, la mesure de probabilité est très indirecte, et il est facile de trouver un contre-exemple à son efficacité : ainsi, un patch présentant des caractéristiques d'une classe, entourés de nombreux patchs de l'autre, présenterait une haute probabilité d'appartenir à la mauvaise classe.
Comme on l'a vu section 2.2.6, il est possible d'utiliser l'information spatiale contenue dans les blocs de cartes de features pour tenter de localiser les zones de l'imagette qui influencent le plus la classification. Ces zones sont représentées par une carte de chaleur. Pour l'obtenir, on pondère les cartes de features de la dernière couche de convolution par les poids associés à chacune dans la couche dense de classification. On obtient deux cartes correspondant aux zones influençant la classification vers une classe ou une autre. On normalise les valeurs obtenues entre 0 et 1, et on sur-échantillonne pour obtenir une carte de chaleur de la même taille que l'image d'entrée. On peut visualiser sur la figure 3.18 la superposition de l'image et de la carte de features associée à sa classe. Encore une fois, on utilise pour cette visualisation le réseau DenseNet.

Afin de mieux visualiser les phénotypes sous les zones de haute influence, on préfère cacher les régions ayant une valeur de chaleur inférieure à 0.5, et souligner celles de valeur supérieure à 0.8 à l'intérieur d'une courbe iso en blanc (voir figures 3.19 et 3.20). Ces cartes d'activation à *cut-off* de chaleur sont ainsi qualifiées d'aérées.

On remarque que, de la même façon que pour les images de puits, les axones et dendrites sont plutôt une marque du phénotype WT (e, f figure 3.19) tandis que les cellules à moindre intensité de marqueur TH et à signal diffus d'alpha-synucléine sont plutôt marqueurs de phénotype LRRK2-G2019S (d, e, f figure 3.20).
La découverte récente d'un biais de bruit de fond, lors de l'entraînement du réseau DenseNet éclaire d'une lumière nouvelle le résultat obtenu sur la carte (d) de la condition WT, figure 3.19. Après retrait des variations du background des images, la classification pour le domaine WT s'est effondrée, et cette carte laisse à penser que le réseau s'appuie justement sur le background de l'image pour classer celle-ci.

3.6 Discussion et perspectives

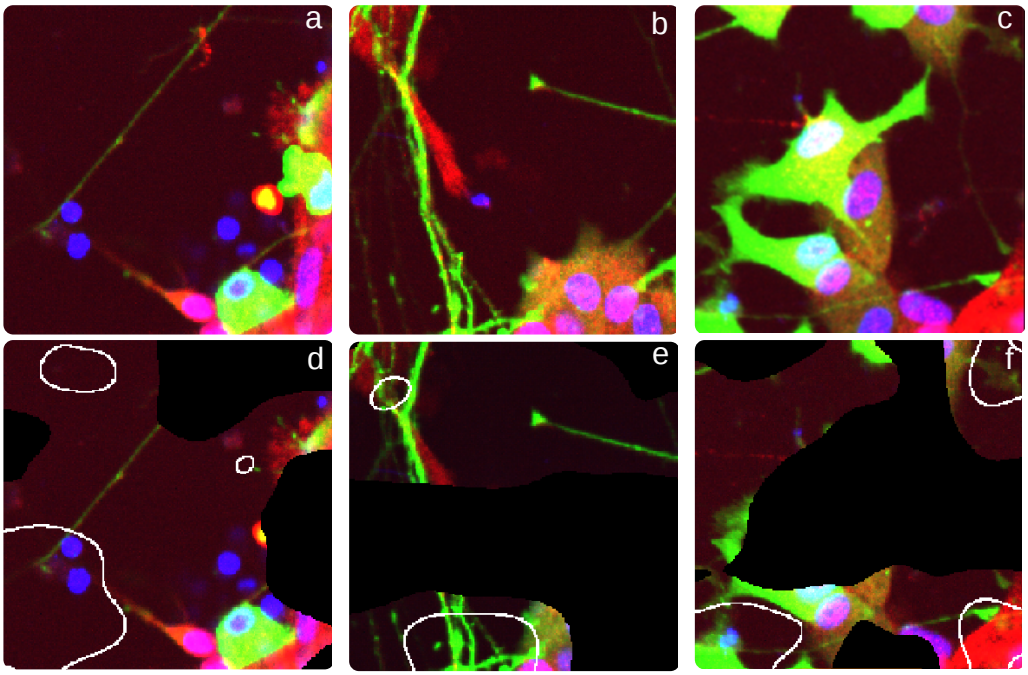


FIGURE 3.19 – **Carte d'activation correspondant à la condition WT.** a,b,c sont des images de cette classe. d,e,f sont la superposition des cartes d'activation aérées et des images.

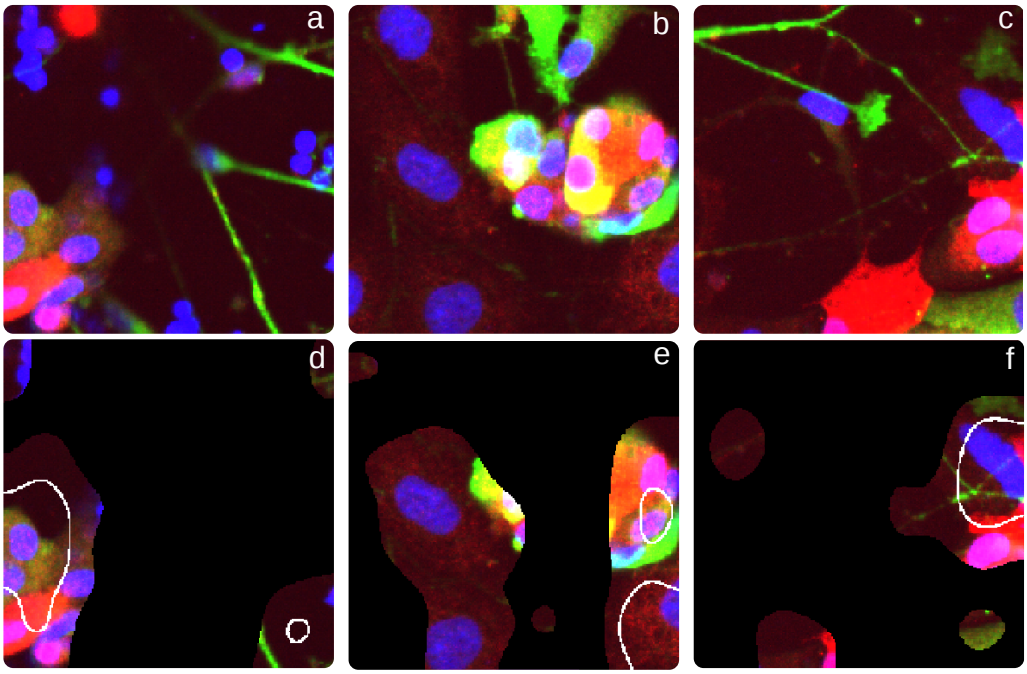


FIGURE 3.20 – **Carte d'activation correspondant à la condition LRRK2-G2019S.** a,b,c sont des images de cette classe. d,e,f sont la superposition des cartes d'activation aérées et des images.

3.6.1 Apprentissage par transfert de connaissance

Grâce à des réseaux pré-entraînés sur le dataset d'Imagenet [35], on a pu classer nos deux catégories d'images, avec une classification par puits de 100%, et une classification par imagette de 75%.
Raghu et al. [36] montrent les limitations de l'utilisation de cette technique de pré-entrainement sur des réseaux apprenant à classifier des images (voir section 2.2.4). Elle n'apporte pas d'avantage significatif aux réseaux l'utilisant. Leur apprentissage se voit simplement accéléré, et les features apprises sont différentes de celles apprises sans cette technique. Ils mettent également en évidence que les réseaux qui contiennent beaucoup de paramètres ont une plus grande inertie au changement, et conservent une trace de ce pré-entraînement à la fin de leur entraînement. Ce n'est pas le cas des petits réseaux, tel que le CBR-Small réutilisé dans nos travaux. Ngiam et al. [129] confirment que le pré-entraînement n'est pas toujours pertinent. Il le devient lorsqu'on choisit un dataset dont la distribution d'images ressemble à celle du dataset d'intérêt. Relativement à notre propre dataset, assez bruité, je ne suis pas convaincue que l'utilisation d'un grand réseau pré-entraîné ne soit pas pertinente. En effet, l'inertie qui caractérise ce type de réseau pourrait lui permettre de ne pas trop être affecté par les erreurs multiples de labellisation des imagettes.

Pour ce qui est de la pertinence du dataset, il existe aujourd'hui de grands jeux de données biologiques qui pourraient permettre un pré-entraînement des réseaux comme le dataset publique de criblage de siRNA (RxRx1). Il est consitué de 1000 classes, de 6 canaux de fluorescence, et de différents types cellulaires. Les données et les canaux sur lesquels préentraîner le réseau peuvent être adaptés, relativement aux cellules et marqueurs utilisés sur notre propre dataset. Cependant, le criblage ne prend pas en compte des cellules neuronales. Ce type de dataset, contenant des formes plus spécifiques à celles observées en biologie, pourrait s'avérer plus intéressant pour les étapes de pré-entraînement que le dataset Imagenet.

3.6.2 Cartes d'activations et visualisation de features

On a montré l'intérêt de cette méthode, avec la découverte de phénotypes caractéristiques de chacune des conditions. Cette technique est plus précise –et sans doute plus fiable– que celle utilisée dans le découpage de puits par patch.
Seulement, pour des images de taille 224×224px, les cartes de features de la dernière couche font 7×7 neurones. La résolution spatiale n'est pas fine. On pourrait se référer à un bloc de cartes de features en amont de celui considéré ici. La résolution de la carte serait plus fine (par exemple, la couche précédente renvoie des cartes de 14×14).
Par ailleurs, le lien entre la localisation dans les cartes de features des dernières couches et la localisation sur l'image n'est pas directe. En reliant ainsi directement la spatialité de la carte de sortie à celle de l'image d'entrée, on ne prend pas en compte le fait que l'information est passée par l'ensemble des couches du réseau. Il serait intéressant d'observer les contributions des localisations entre chacune des couches ; ce sont ces contributions qu'Olah et al. [59] sont parvenus à afficher de façon visuelle et interactive sur le lien suivant : lien vers les interfaces).

La normalisation des cartes peut cacher

l'intensité des phénotypes observés, si tant est que les phénotypes cellulaires pèsent différemment pour le choix du réseau. En effet, une probabilité forte pour un phénotype WT peut être due à un phénotype local moyen, et à un phénotype LRRK2-G2019S très faible. Le phénotype moyen peut potentiellement être mis en valeur autant qu'un phénotype fort. Dans la gradation de l'intensité de la carte de chaleur, on pourrait prendre en compte les valeurs des neurones juste avant la couche softmax (voir figure 2.3). Ces neurones sont plus représentatifs de l'intensité des phénotypes que les probabilités d'appartenance à chacune des classes, qui sont plus comparatives qu'absolues.

On pourrait aussi envisager d'affiner les cartes de puits grâce aux cartes d'activation. On intègrerait, pour chaque position, de l'information venant de toutes les directions grâce aux imagettes contenant cette position.

Le souci de l'interprétation de la classification par la localisation des phénotypes est qu'il suffit qu'une information de bruit corrèle avec la position pointée par le réseau pour que l'humain puisse mal analyser la signalisation. Les visualisations de features (voir section 2.13) peuvent être un outil intéressant pour décorréler les informations pertinentes ou non à une position donnée. Les interfaces interactives que l'on a citées précédemment, associent les outils de localisation que sont les cartes d'activation avec ces visualisations de features pour une meilleure interprétation. Il est à noter que les visualisations de features peuvent se révéler assez psychédéliques (voir figure 2.11), et qu'il faut parfois une certaine dose d'inspiration pour les interpréter.

3.7 Conclusion

La classification d'images nous a permis de constater que, malgré des génotypes proches, les cultures de cellules neuronales de type WT et LRRK2-G2019S peuvent être discriminées. On a aussi constaté que les images de type WT contiennent peut-être une information cachée dans background de l'image, sans doute des débris cellulaires. Les cartes d'activation et le pavage du puits par patchs de probabilité nous ont permis d'obtenir une certaine intuition des phénotypes des deux conditions, phénotypes dont on poursuivra l'étude par la suite grâce à des techniques de transformation d'images et de clustering.

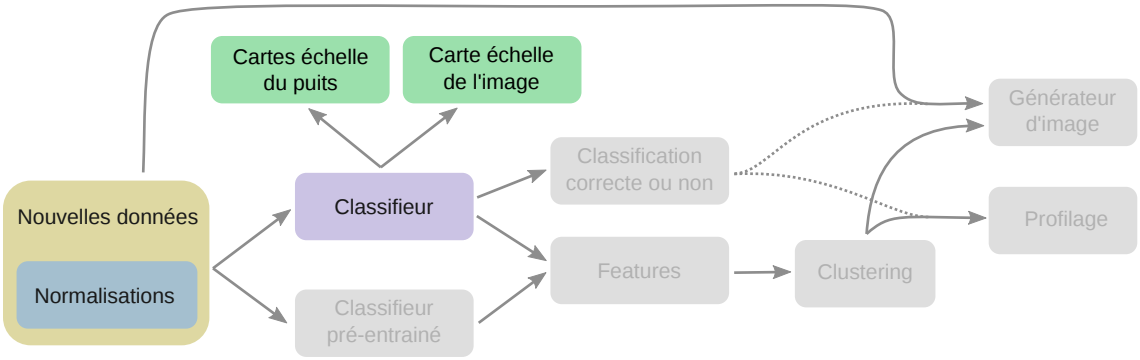


FIGURE 3.21 – **Schéma récapitulatif des techniques associées à la classification que l'on a exploitées.**
À partir d'un classifieur entraîné sur les données, des cartes phénotypiques spatiales ont été construites, à l'échelle du puits et de l'imagette.

Chapitre 4

Génération de phénotypes comme outil explicatif

4.1 Introdution

4.1.1 Motivation

Les cartes d'activation nous donnent des informations sur la localisation du phénotype. En revanche, elles n'expliquent pas ce qui, dans la zone mise en évidence, déclenche la prédiction. Par ailleurs, que ce soit dans les cultures de cellules mutées G2019S ou dans les cultures de cellules *wild-type*, la variabilité du phénotype est grande. Et, dans cette grande complexité, il est délicat de percevoir des décalages fins entre les images.

4.1.2 CycleGAN

Aussi, avons-nous pensé à détourner l'utilisation du CycleGAN (voir section 2.4.2). Utilisé pour les tâches artistiques – modifier une photographie en tableau – ou pour de l'augmentation de taille de dataset, nous avons voulu le mettre à profit pour expliquer des différences de phénotypes cellulaires. En effet, la génération d'image à image peut nous permettre de transformer une image de cellule saine en image de cellule malade et d'observer les différences sur deux images proches. En d'autres termes, on peut s'affranchir de la variabilité de phénotype intra-classe. Par ailleurs, la structure en cycle du CycleGAN nous permet d'espérer que la variation que l'on observe sera minimale, et que l'on percevra toujours la structure précédente dans l'image d'arrivée.
Dans ce chapitre, on démontre que la méthode donne des résultats convaincants sur des datasets "simples", où le phénotype est mesurable. Dans un second temps, on essaie d'appliquer la méthode au dataset de LRRK2, pour lequel une mesure directe du phénotype n'est pas possible.

4.2 Matériels et méthodes générales

Le CycleGAN est composé de 4 réseaux, 2 générateurs et 2 discriminateurs. L'objectif de cet outil est de transformer des images d'un domaine A en image de domaine B. Par conséquent, un réseau générateur est responsable de transformer les images A en B et inversement pour le second. Quant aux discriminateurs, ils sont chargés de déterminer, pour l'un si les images du domaine A sont réelles ou générées, et de même, pour l'autre, si les images du domaine B sont authentiques. Le premier objectif des générateurs est de tromper leur discriminateur respectif.
Le second objectif des générateurs est de ne pas trop s'éloigner de l'image d'origine lors de la transformation, pour permettre au générateur inverse de reconstruire l'image de départ avec précision. Les générateur sont donc censés apprendre des transformations inverses ($G_{A\to B}(G_{A\to B}) = Identité$).

Dans l'implémentation utilisée [130], les générateurs adoptent une architecture de U-net [89] ; les discriminateurs, une architecture de type PatchGAN [92]. Le champ réceptif des discriminateurs est de 94x94 pixels : la sortie du discriminateur est une carte de $n \times n$ neurones, chaque neurone couvre un patch d'images de 94x94 pixels et prédit si le patch provient d'une véritable image ou bien d'une image générée.

L'entraînement se fait alternativement en optimisant les discriminateurs à l'aide de vraies images et d'images générées, puis en optimisant les générateurs à l'aide des erreurs et réussites des discriminateurs ainsi que

de l'erreur faite lors de la reconstruction (erreur de cycle).
On entraîne les réseaux sur plusieurs epochs, chaque epoch correspondant à un nombre d'itérations. Chaque itération est une optimisation sur un batch d'images. Afin de pouvoir comparer les générateurs au cours de l'entraînement, toutes les n itérations, on enregistre le résultat des transformations $A \to B$ et $B \to A$ sur le même set d'images du domaine A et du domaine B.
Un entraînement dure entre 4 et 9 jours sur une carte graphique de type NVIDIA Tesla V100 PCIe 32 GB.

Dans un premier temps, on enregistre les réseaux à la fin de chaque epoch.

4.3 Les datasets "preuve de concept"

Nous avons voulu vérifier quantitativement, sur des datasets plus simples que celui de LRRK2, que les transformations phénotypiques observées sont fiables. Le laboratoire BioPhenics, plateforme de screening de l'institut Curie, nous a mis à disposition deux datasets pour lesquels le phénotype des deux contrôles est connu et mesurable (dataset de translocation de protéine et de désagrégation de l'appareil de Golgi). On peut donc obtenir une mesure directe de la réussite ou non des générateurs d'image.

4.3.1 Le dataset de translocation

Biologie et images

Le dataset de translocation est composé de cellules issues de cancer du sein, qui ont été cultivées avec ou sans ajout de TNF (Tumor Necrosis Factor).

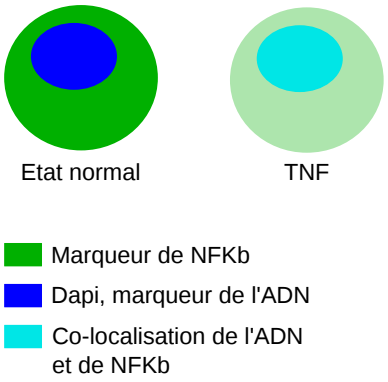


FIGURE 4.1 – **Schéma des deux conditions du dataset de translocation de NF-κB.** À l'état normal la protéine est située dans le cytosol. Activée par le TNF, elle s'accumule dans le noyau.

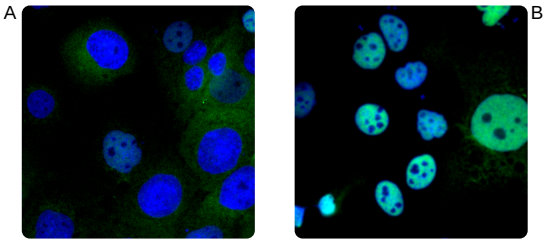


FIGURE 4.2 – **Exemple des deux contrôles du dataset de translocation.** (A) état nomal. (B) activation par le Tumor Necrosis Factor.

La protéine TNF est connue pour induire l'activation de la protéine NF-κB. Lorsque NF-κB est inactivée, elle est située dans le cytosol (voir figure 4.1). Après son activation, elle se déplace jusque dans le noyau, où elle s'accroche à des séquences d'ADN. On appelle ce processus de déplacement, la "translocation". Le dataset contient 1 plaque de 32 puits pour chaque condition (voir figure 4.2). Pour chaque puits, on a 4 champs de vue de 1400x1400 pixels.

Génération d'images

On entraîne les générateurs du CycleGAN à générer des images de l'état NF-kappaB activée (domaine B) depuis des images à l'état normal (domaine A) et

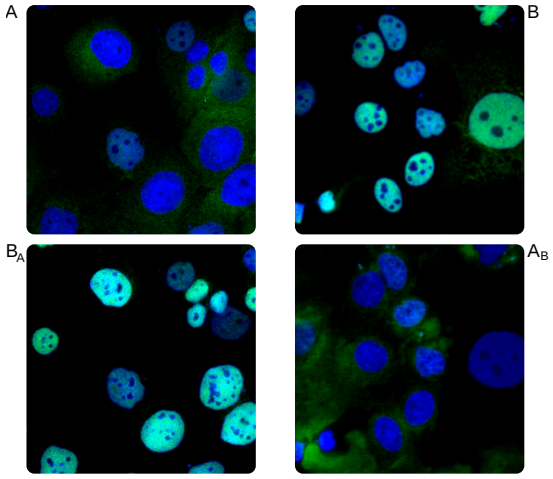


FIGURE 4.3 – **Exemple de génération d'images.** A et B sont les vraies images du dataset. B_A et A_B sont des images générées respectivement à partir de A et B, vers la condition opposée.

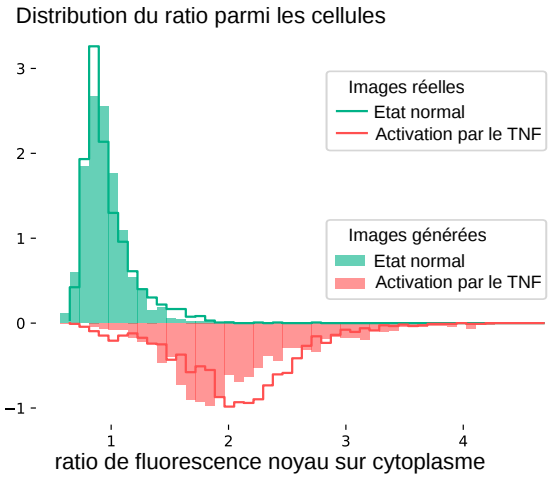


FIGURE 4.4 – **Distributions du ratio d'intensité de fluorescence dans le noyau et dans le cytoplasme dans les 4 types d'images.** Les lignes représentent les distributions des vraies images, tandis que les histogrammes pleins représentent les distributions des images générées. Le vert correspond à l'état normal, le rouge à l'état du NF-kappaB activée par le TNF.

inversement. On utilise des portions des grandes images, de 128x128 pixels.
Les images générées dans les itérations de la fin de l'entraînement sont stables (on n'observe par de différences entre chaque itération) et convaincantes (voir figure 4.3). On cherche maintenant à vérifier par une mesure que le phénotype opposé est bien atteint.

Mesure du phénotype

Sur les datasets de translocation de protéines cytosoliques vers le noyau, une mesure du phénotype est le ratio de l'intensité de fluorescence liée à la protéine dans le noyau sur celle mesurée dans le cytoplasme :

$$\frac{I_{fluo}^{Noyau}}{I_{fluo}^{Cytop}} \tag{4.1}$$

J'ai donc réalisé la segmentation des noyaux des cellules par "watershed", mesuré la fluorescence moyenne dans le canal vert dans la zone correspondant au noyau. Puis, j'ai délimité un anneau de 4 pixels de rayon autour du noyau et mesuré la fluorescence dans cette zone-là, considérant que ce serait une mesure représentative de la fluorescence dans tout le cytoplasme.

On visualise, figure 4.4, les distributions de ce ratio d'intensité dans les différents types d'images. Comme attendu, on observe que les distributions des images générées sont plus proches de la distribution qu'elles essaient d'imiter que de la distribution dont elles proviennent. Plus exactement, la distribution de l'état normal généré (histogramme vert plein) est plus proche de la distribution du vrai état normal (histogramme vert en ligne) que de la distribution de l'état activé dont les images proviennent (histogramme rouge ligne) et inversement pour la distribution de l'état activé généré. En termes quantitatifs, pour chacune des conditions, les distances de Wasserstein entre les distributions des vraies images et des images générées sont faibles (voir tableau 4.1). À titre de comparaison, les distances de

		Vraies images	
		Etat normal	Etat activé
Images générées	Etat normal	0.04	1.1
	Etat activé	1.1	0.1

TABLE 4.1 – **Distance de Wasserstein entre les distributions de ratio d'intensité de fluorescence des vraies images et des images générées.**

Wasserstein entre deux échantillons du même domaine sont de 0.01 pour l'état normal et 0.02 pour l'état activé. La mesure confirme que la génération statistique des phénotypes est satisfaisante.

Il est aussi possible d'agréger les valeurs de fluorescence en calculant la moyenne par puits (voir figure 4.5). Ici encore, les distributions correspondent bien et ne se recouvrent pas.

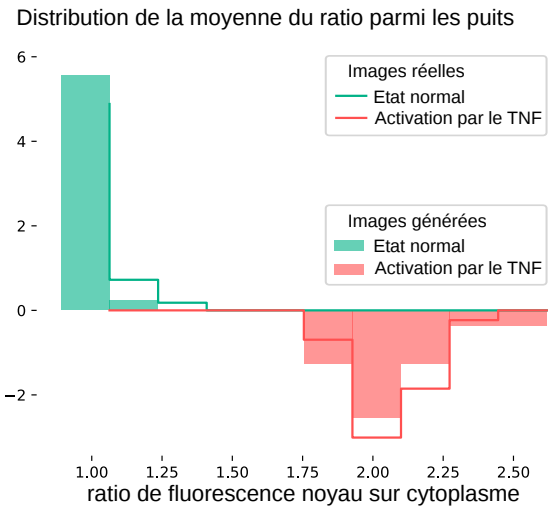


FIGURE 4.5 – **Distribution de la moyenne par puits du ratio d'intensité de fluorescence.**

4.3.2 Le dataset de l'explosion du Golgi

Biologie et images

Le nocodazole est connu pour empêcher la polymérisation des microtubules au sein des cellules eucaryotes [131]. Ce faisant, il induit la fragmentation de l'appareil de Golgi qui va se répartir aux différentes portes de sortie du réticulum endoplasmique, en prenant la forme de mini-piles [132] (voir figure 4.6). Le dataset est composé de cultures cellulaires pour lesquelles on marque une protéine du Golgi à l'aide du marqueur GM130 en vert, ainsi que le noyau en bleu (voir figure 4.7). Dans la condition normale, le Golgi est regroupé, proche du noyau. Expérimentalement, les puits de cette condition ne contiendront que du média cellulaire et du DMSO. Dans le cas où du nocodazole est ajouté, on observe une fragmentation de l'appareil de Golgi en petits blobs.

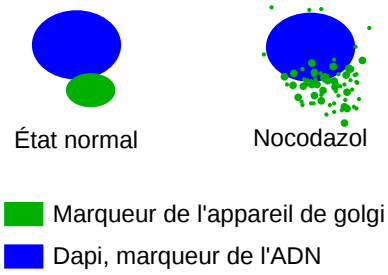


FIGURE 4.6 – **Schéma des deux conditions du dataset de fragmentation du Golgi.** À l'état normal le Golgi est situé près du noyau, agrégé. Après ajout du nocodazole, le Golgi explose en blobs.

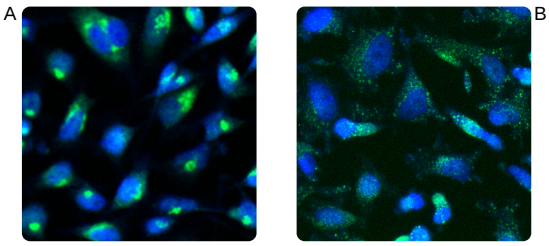


FIGURE 4.7 – **Exemple des deux contrôles du dataset de translocation.** (A) : état normal, agrégé. (B) : état fragmenté.

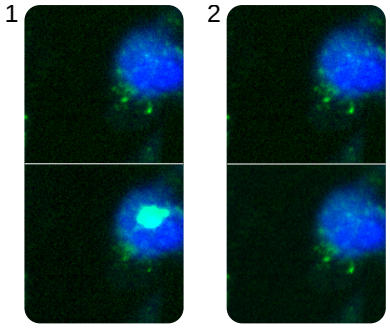


FIGURE 4.8 – **Performances inégales des générateurs.** Exemple de génération d'images de la condition désagrégée (images du haut) vers la condition normale, agrégée (images du bas). 1 et 2 correspondent à des générateurs différents. 1 est le meilleur générateur de l'entraînement considéré, c'est le générateur sauvegardé à l'epoch 13, tandis que 2, sauvegardé à l'epoch 45, le générateur le moins performant.

Le dataset contient 3 plaques de 84 puits pour chaque condition (normal/désagrégé). Pour chaque puits, on a 4 champs de vue de 1392x1040 pixels.

Choisir un générateur

On entraîne les générateurs du CycleGAN à générer des images de cellules à Golgi fragmenté depuis des images à l'état normal et inversement. Encore une fois, on utilise des portions des grandes images, de 128x128 pixels.

Pour le dataset du golgi, l'entraînement du CycleGAN est assez erratique, de sorte que, à quelques itérations d'optimisation d'écart, les générateurs donnent lieu à des images de qualité beaucoup plus irrégulière que pour le dataset de la translocation (voir figure 4.8).

Lorsque l'entraînement ne converge pas naturellement, il faut alors faire un choix parmi les générateurs pour trouver le bon. Au cours de l'entraînement, on ne peut pas enregistrer tous les générateurs faute de mémoire suffisante. Par contre, on peut essayer de sauvegarder un générateur de façon plus intelligente que l'aléatoire (méthode précédente). On a tenté pour ce dataset de conserver, tout au long de l'entraînement, les meilleurs discriminateurs (D_A et D_B) et générateurs ($G_{A\to B}$ et $G_{B\to A}$). On met à jour chaque réseau conservé lorsqu'on en trouve un meilleur. On décrète qu'un générateur est le meilleur lorsqu'il réalise des meilleurs scores, en terme de la fonction de loss de l'entraînement, que le générateur sauvegardé, face au meilleur discriminateur et au discriminateur actuel. De même, on considère qu'un discriminateur devient meilleur discriminateur lorsqu'il obtient des meilleurs scores que le discriminateur sauvegardé face au meilleur générateur et au générateur actuel. Un exemplaire de chaque réseau est sauvegardé pour chaque epoch.

Des entraînements utilisant cette méthode de sauvegarde et la méthode aléatoire sont réalisés. On obtient ainsi un grand nombre de générateurs, parmi lesquels il faut sélectionner le plus performant.

Pour choisir parmi les différents générateurs enregistrés, j'ai tenté d'utiliser un réseau DenseNet [47] entraîné à reconnaître les labels des images : état normal ou ajout de nocodazole.
Chaque générateur est évalué de la même façon. Pour un ensemble d'images du domaine source, les probabilités d'appartenance à chaque label sont prédites par le DenseNet. Les prédictions sont ensuite réalisées pour les mêmes images mais transformées vers le domaine cible par le générateur. La qualité du générateur est mesurée par la moyenne de la distance entre les probabilités prédites pour chaque paire d'images. Plus la transformation

est réussie, plus la distance entre les prédictions devrait augmenter.
Lorsque l'on applique cette technique aux générateurs sauvegardés pour tous les entraînements, le choix du DenseNet sélectionne un générateur sauvegardé grâce à la méthode de sauvegarde aléatoire. Cependant, on ne peut pas conclure à une meilleure efficacité de l'aléatoire, un entraînement pouvant se révéler mauvais de bout en bout, et le nombre d'entraînements réalisés n'est pas suffisant pour pouvoir assurer une différence significative.

Des exemples d'images que l'on obtient sont présentées figure 4.9. L'image générée dans le domaine B_A présente des mini-piles caractéristiques de l'ajout de nocodazole. La transformation dans le domaine A_B reconstitue l'appareil de Golgi de forme plus compacte.

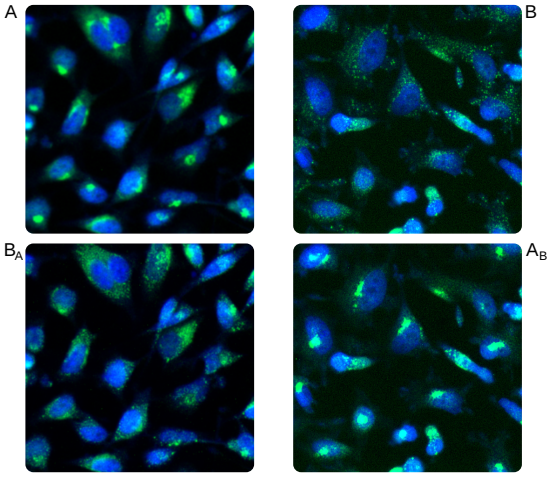


FIGURE 4.9 – **Exemple de génération d'images.** A et B sont les vraies images du dataset. B_A et A_B sont des images générées respectivement à partir de A et B, vers la condition opposée.

Mesure du phénotype

On cherche maintenant à vérifier par une mesure que le phénotype opposé est bien atteint.

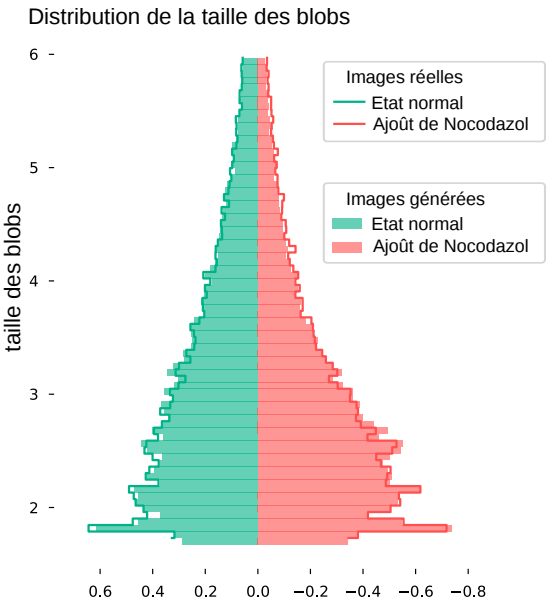


FIGURE 4.10 – **Distributions de la taille des blobs.** Un même nombre de blobs a été sélectionné pour chaque image.

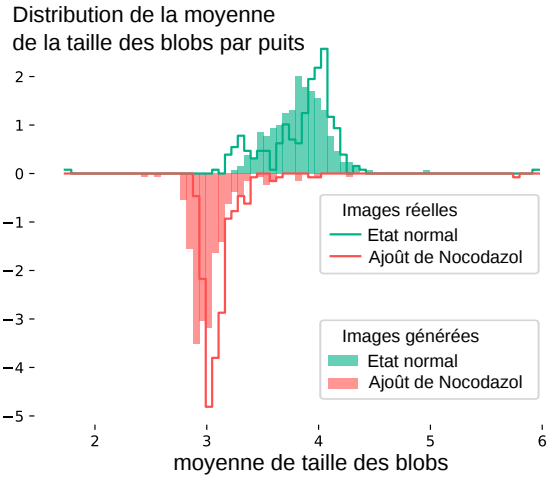


FIGURE 4.11 – **Distributions des moyennes de taille de blobs par puits, pour les images réelles et générées, à l'état normal et après ajout de Nocodazol.**

La mesure type sur les datasets de dislocation de Golgi est le nombre et la taille des blobs de Golgi. J'ai donc réalisé la détection de ces blobs sur les images agrandies par sur-échantillonnement bilinéaire, à l'aide de la bibliothèque python OpenCV [133]. Le sur-échantillonnage facilite la détection des blobs de très faible rayon. La mesure est loin d'être parfaite, avec des

		Vraies images	
		Etat normal	Etat désagrégé
Images générées	Etat normal	0.06	0.72
	Etat désagrégé	0.82	0.09

TABLE 4.2 – **Distance de Wasserstein entre les distributions de taille de blob des vraies images et des images générées.** À titre de comparaison, lorsqu'on prend deux échantillons différents de vraies images de l'état normal, la distance entre les distributions de mesure est de 0.02. Elle est de 0.09 pour des échantillons pris dans les vraies images à l'état désagrégé. Et la distance entre les distributions des vraies images des deux conditions est de 0.74

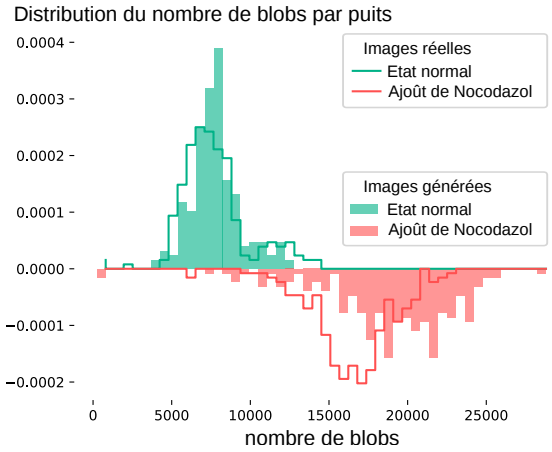


FIGURE 4.12 – **Distributions du nombre de blobs par puits, pour les images réelles et générées, à l'état normal et après ajout de Nocodazol.**

fausses détections ou des dédoublements de blob, mais elle est réalisée de façon identique pour toutes les images. On considère, par conséquent, que les différences observées sont dues à une évolution du phénotype.

La figure 4.10 présente les distributions de taille des blobs, et la figure 4.11 la distribution des moyennes de ces tailles par puits. Dans chaque cas, le décalage observé entre les distributions des domaines sources et cibles est notable. De même que précédemment, pour le dataset de translocation, pour chacune des conditions, les distances de Wasserstein entre les distributions des vraies images et des images générées sont faibles (voir tableau 4.1), et de l'ordre de grandeur de la distance entre deux échantillons aléatoires piochés dans les vraies images (voir tableau 4.2). Les images générées sont plus proches du domaine d'arrivée que du domaine de départ au sens de notre mesure.

On peut aussi regarder les distributions de nombre de blobs par puits. Là encore, les distributions se décalent pour se rapprocher de celle du domaine cible. On note même que les distributions des mesures pour les images générées sont mieux séparées que pour les images réelles. Selon la distance de Wasserstein, l'écart entre les distributions de taille, de l'état normal et de l'état désagrégé, est de 0.74 pour les images réelles, et de 0.79 pour les images générées. Cet effet est sans doute dû à la méthode de sélection des générateurs, qui privilégie les générateurs de phénotypes extrêmes. Le DenseNet, préentraîné sur les images du dataset de Golgi, se trompe moins lorsque les images sont plus caricaturales.

Sur des critères mesurables objectivement, nous pouvons constater que l'algorithme CycleGAN permet de transformer des images d'un domaine biologique à un autre.

4.4 LRRK2

4.4.1 Le dataset d'entraînement

Pour entraîner le CycleGAN sur les phénotypes neuronaux, le dataset utilisé est celui présenté et employé dans les sections 3.4.2, 3.4.3, et 3.4.4 du Chapitre traitant de classification. La taille des images est de 128×128. Afin de produire des images exposant les différences relevées par les classifieurs, le dataset est filtré pour ne contenir que les images correctement catégorisées par le discriminateur DenseNet.
Le biais concernant le background de l'image décrit dans les sections citées ci-dessus n'ayant pas été remarqué au moment des expériences décrites dans le présent chapitre, les images comportent leur background originel.

4.4.2 Vers une autre mesure de qualité de générateur : la distance Inception de Fréchet (FID)

Les échecs de la méthode Densenet

Précédemment, à la section 4.3.2, on a vu que l'on pouvait utiliser un classifieur DenseNet pré-entraîné sur le dataset pour choisir le meilleur générateur parmi tous les entraînements effectués.
Dans cette section, deux inconvénients de cette méthode sont illustrés.

Les GANs souffrent du risque d'effondrement de mode ou *mode collapse*. Cela se produit lorsque plusieurs entrées sont transformées en la même sortie [134]. Il existe également un cas d'effondrement partiel, où la même caractéristique, de texture, de couleur ou autre, se retrouve sur toutes les images ou un grand nombre d'entre elles.
Il est important que la mesure

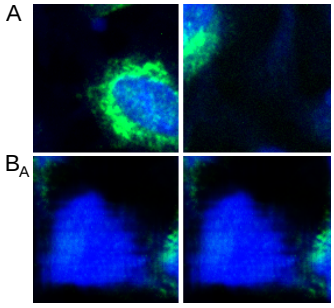


FIGURE 4.13 – **Meilleur générateur sélectionné par DenseNet pour la transformation état normal → nocodazole ou état désagrégé.**

d'évaluation soit sensible à ce défaut de diversité. Or, comme on le vérifie expérimentalement figure 4.13, ce n'est pas le cas de notre mesure avec DensNet. Le meilleur générateur au sens de DenseNet, génère toujours la même image, c'est un effondrement de mode complet. Théoriquement, il suffit que l'image générée induise une prédiction maximale de la classe cible par le DenseNet, et le générateur sera considéré comme un générateur parfait.

Un autre défaut de la méthode est qu'elle ne prend pas en compte la fidélité des images générés aux vraies images. On a deux exemples figures 4.14 et 4.15. Dans la première figure, la transformation apprise par le réseau est d'inverser les canaux rouge et vert. Cela suffit apparemment à tromper le DenseNet, qui n'a pas appris à distinguer les images générées des vraies mais à discriminer les images du domaine A de celle du domaine B.
La figure 4.15 décrit la transformation d'une image par le meilleur générateur, au sens de l'évaluation par DenseNet, pour le dataset LRRK2. Le générateur a appris à cacher tous les axones par des taches bleues. Le canal bleu est celui des noyaux, et la morphologie de ces tâches ne fait absolument pas penser à celle des noyaux. Le DenseNet a pri-

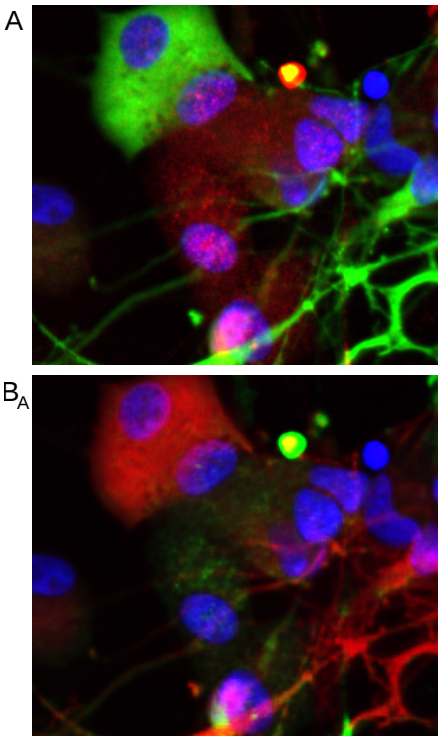


FIGURE 4.14 – **Un des meilleurs générateurs sélectionnés par DenseNet pour la transformation WT → G2019S.**

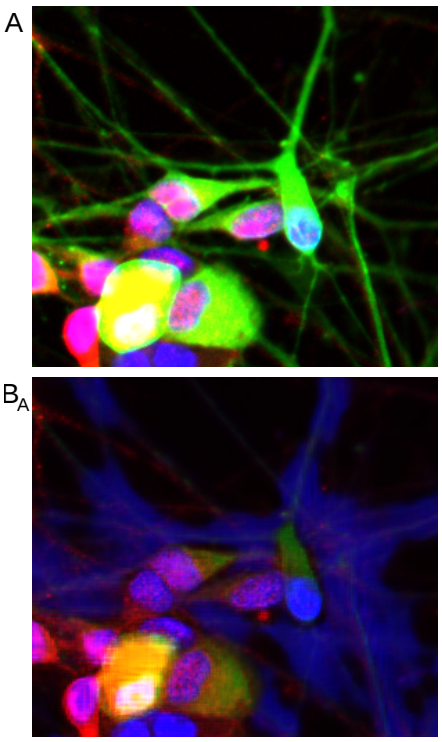


FIGURE 4.15 – **Meilleur générateur sélectionné par DenseNet pour la transformation WT → G2019S.**

vilégié un générateur de transformation qui privilégie l'éloignement maximal du domaine source plutôt que le rapprochement au domaine cible. Dans la partie consacrée à la transformation du Golgi (4.3.2), la meilleure séparabilité des distributions des images générées par rapport aux distributions réelles est sans doute due à la même cause.

La FID

Bien que le CycleGAN originel n'utilise pas de mesure d'évaluation des générateurs, il existe un certain nombre de mesures de qualité, dont certaines ont été présentées dans le Chapitre 2.3.7 d'état de l'art. Notre attention s'est portée sur l'une des plus utilisées [135], la distance Inception de Fréchet [86]. Pour rappel, elle permet d'évaluer la distance qui sépare deux ensembles d'images d'un point de vue sémantique. La méthode utilise un réseau Inception [83] pré-entraîné sur le dataset Imagenet [35] (voir section 2.2.4). Des vecteurs de features sont extraits pour chaque image des deux échantillons. Puis, les distributions des vecteurs de chaque échantillon sont assimilées à deux gaussiennes multivariées. On calcule la distance entre ces gaussiennes par la formule de Fréchet.

La FID est censée remédier aux deux problèmes rencontrés par la méthode d'évaluation précédemment utilisée avec le DenseNet. Tout d'abord, dans le cas de l'effondrement de mode, si une seule image est générée, cela va fortement impacter la matrice de covariance de la gaussienne multivariée, et la distance sera très grande. Cette distance compare les statistiques des images générées avec cellles des vraies images, ce qui correspond à la notion intuitive de "ressemblance".
Un dernier avantage de la FID est

qu'elle serait corrélée à l'appréciation de l'oeil humain quant à la qualité des images [86].

	FID B vs B	FID B vs A
Toutes images	185	240
Images bien classées	192	318

TABLE 4.3 – **Valeur de la FID entre les domaines A et B pour le dataset complet et le dataset composé uniquement des images bien classées.**

Le tableau 4.3 rassure quant à la pertinence de cet outil d'évaluation de distance pour notre dataset biologique. La distance entre les deux domaines A et B est calculée pour deux datasets : d'une part le dataset complet, et d'autre part le dataset uniquement composé des images correctement classées dans leurs classes respectives par un réseau classifieur. Le dataset filtré ne contient pas toutes les images dont le phénotype est plutôt associé à la classe opposée à la sienne. Dans le dataset des images bien classées, les distributions d'images sont donc mieux séparées que dans le dataset complet, pour lequel il existe un overlap entre les deux classes A et B. On vérifie alors que la distance entre les deux classes dans le cas du dataset épuré (FID de 318) est plus grande que dans le cas du dataset complet (FID de 240).

La FID a été introduite pour mesurer la qualité des GANs classiques. Dans un GAN classique, le générateur génère une image à partir d'un vecteur de bruit (voir section 2.3). Dans ce cas là, la distance mesurée est celle entre le dataset d'entraînement composé d'images réelles, et les images générées. Le CycleGAN, lui, appartient à la catégorie des générateurs images vers images. Par conséquent, on peut calculer non seulement la distance entre les images

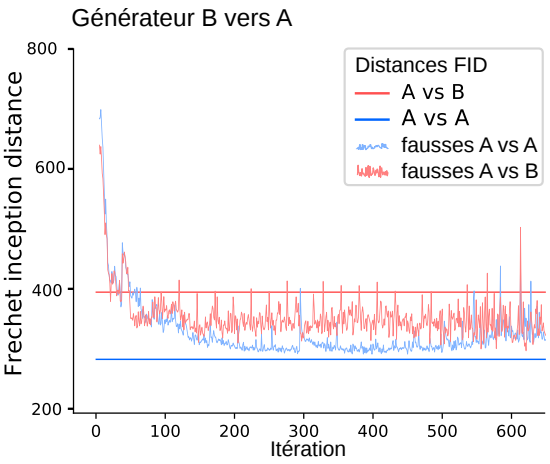


FIGURE 4.16 – **Mesures de la FID au cours de l'entraînement.**

générées et les vraies images du domaine cible, mais aussi la distance entre les images générées et le domaine source.

Lors de l'entraînement, les mesures de FID sont calculées toutes les n itérations[1]. Figure 4.16, en bleu, sont représentées les valeurs de la FID entre les fausses images du domaine A et les vraies. Le but de l'entraînement est de se rapprocher de la ligne bleue qui correspond à la distance entre deux échantillons aléatoires du domaine A. En rouge est représentée la distance entre les images générées (domaine A) et des images du domaine dont elles proviennent. Le second but de l'entraînement est que la distance entre A générées et vraies B atteigne la ligne rouge qui correspond à la distance entre les deux domaines A et B.

Le calcul se fait sur des échantillons de 5000 images. Pour calculer la distance entre les images générées dans le domaine A et les images du domaine B, on prend soin de ne pas sélectionner des images appariées, c'est à dire une B et la fausse A qui en découle.

Le calcul de la FID tout au long

1. Le calcul de la FID est coûteux en temps et ne peut pas être effectué à chaque itération.

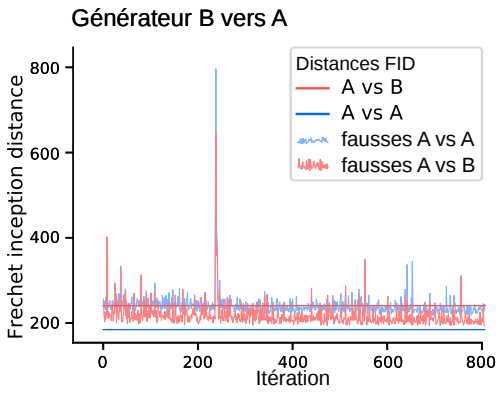


FIGURE 4.17 – **Mesures de la FID lors d'un mauvais entraînement.**

de l'entraînement rajoute beaucoup de temps de calcul, mais permet d'avoir un aperçu global de la qualité de l'entraînement. Figure 4.16, l'entraînement est bruité avec des oscillations pour la distance entre les images générées et les images du domaine source. Figure 4.17, l'entraînement est mauvais pour le générateur : la courbe rouge est en dessous de la courbe bleue, ce qui signifie que les images générées sont plus proches du domaine source que du domaine cible, et ce, tout au long de l'entraînement. Il est très fréquent d'obtenir des entraînements moyens, voire mauvais, sur le dataset complexe de LRRK2.

Le calcul de FID tout au long de l'entraînement permet de ne sauvegarder que les générateurs obtenant de bons résultats. Pour le domaine cible, on sauvegarde les 20 générateurs qui présentent la distance images réelles/images générées la plus petite. Parmi les générateurs obtenus de cette façon, certains obtiennent des scores similaires. Cependant, il existe des différences entre les transformations qu'ils génèrent. Le tableau 4.4 présente des valeurs de FID obtenues lors de l'évaluation des deux meilleurs générateurs $G_{B\to A}$ sauvegardés lors d'un entraînement, et la figure 4.18 illustre les différences de transformation. Le FID A vs A correspond à la distance fixe entre deux échantillons d'images A, réelles. C'est la distance que l'on souhaite atteindre dans la colonne FID A générée vs A. Pour ces deux générateurs, cette valeur est la même.
Les images A générées proviennent du domaine B et l'on souhaite qu'elles s'écartent de ce domaine source. La distance fixe, FID A vs B, correspond à la distance entre les deux domaines. Cette distance peut être un objectif pour la valeur FID A générées vs B. Ces valeurs sont d'ailleurs très différentes pour les deux générateurs et cette différence se reflète sur les images générées par ces deux réseaux (voir figure 4.18). Le générateur le plus éloigné du domaine source (itération 14-4800) complexifie davantage le signal vert émis autour des noyaux (première colonne de (A)), et fait parfois disparaître ces noyaux (bloc d'images A sur la figure). On n'observe pas de différence dans le cas des images de la colonne B. Dans la colonne C, c'est le générateur de l'itération 14-0900 qui augmente l'intensité du signal vert.
Pour observer les différences phénotypiques entre les deux types de cultures, on sélectionne des générateurs pour lesquels les deux mesures de FID sont les plus proches de leurs objectifs respectifs. Dans ce cas là, le générateur 14-4800 est sélectionné.

4.4.3 Normalisation

L'activation de la dernière couche du réseau U-net est une fonction tangente Hyperbolique (tanh). L'ensemble-image de la tangente hyperbolique est l'intervalle $[-1, 1]$. Le générateur ne peut donc apprendre à reconstruire que des images avec des valeurs de pixels comprises dans cet intervalle là. Par conséquent, les images doivent être normalisées entre -1 et 1. Les normalisations essayées sur

Itération	FID A vs A	FID A générées vs A	FID A générées vs B	FID A vs B
14-0900	282	295	365	394
14-4800	282	295	405	394

TABLE 4.4 – **Valeur de la FID pour deux itérations.**

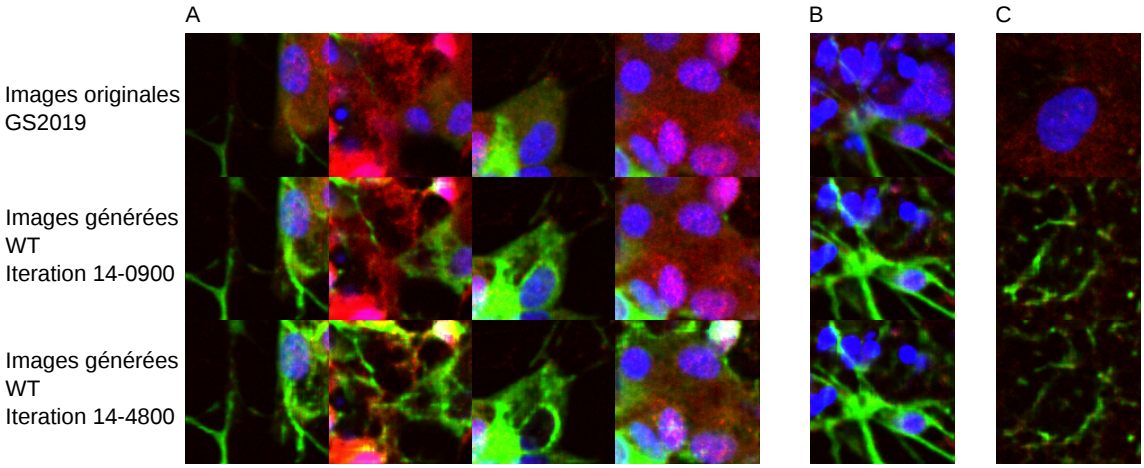


FIGURE 4.18 – **Comparaison de générations d'images pour deux générateurs.** Les images du domaine B (G2019S) sont transformées dans le domaine A par deux générateurs $G_{B\to A}$, correspondant aux deux itérations du tableau 4.4. La rangée supérieure présente les images réelles. Les deux rangées inférieures présentent les images transformées dans le domaine A (WT).

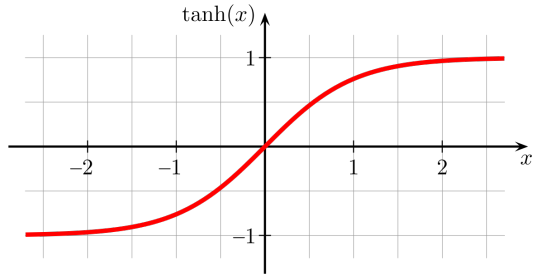


FIGURE 4.19 – **Tangente hyperbolique.**

ces datasets sont linéaires.
Pour chaque canal de fluorescence, la normalisation min-max met à l'échelle les valeurs des pixels de façon à ce que le maximum soit à 1 et le minimum à -1. Le minimum et le maximum sont définis à l'échelle de la plaque entière – et non image par image.
La normalisation avec centile *percentile normalization* n'étire pas entre le min et le max, mais entre des valeurs centiles. Par exemple, pour chaque canal, on peut mettre à l'échelle de façon à ce que la valeur du 1er centile soit à -1 et celle du 99e centile à 1. Les valeurs qui dépassent sont plafonnées à 1 et -1. À nouveau, ces centiles sont calculés sur la plaque entière, et pour chacun des canaux de fluorescence. Cette dernière normalisation permet de s'affranchir des valeurs extrêmes (outliers), qui réduisent la taille de la plage dynamique des valeurs de pixel.

La figure 4.20 illustre les effets de ces normalisations sur les capacités des générateurs d'images. Les intensités de chaque canal ont été re-normalisées entre 0 et 255 (8bits) de différentes façons. Les différences d'intensité ne sont pas un effet de normalisation. L'effet se retrouve plutôt dans la capacité du générateur à synthétiser un signal rouge granuleux. En effet, les centiles ont été choisis différemment pour chaque canal. Si un centile de 99.9% convient

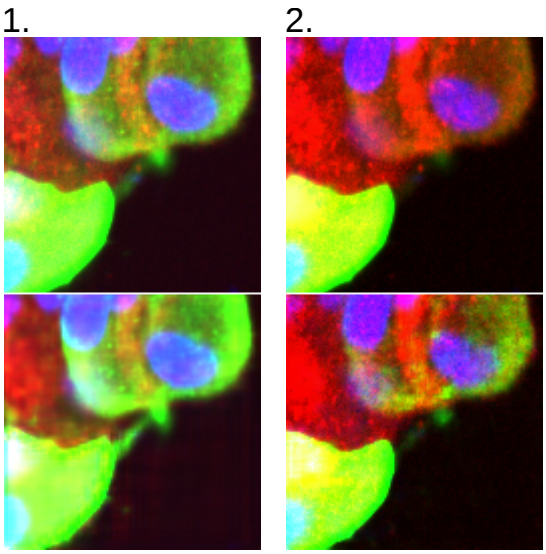


FIGURE 4.20 – **Effet de la normalisation sur la capacité de reconstruction des générateurs.** Ligne du haut, vraies images. Ligne du bas, images synthétiques. 1. Avec une normalisation min-max. 2. Avec une normalisation par percentiles.

pour la valeur maximum du canal vert, il faut descendre à 97% pour le canal rouge pour obtenir une bonne gamme dynamique des pixels et permettre des synthèses de signal rouge comme observées colonne 2.

4.4.4 Les phénotypes observés

Les changements de phénotype sont plus impressionnants lorsque l'on peut passer d'une image à l'autre en superposé, comme dans un GIF par exemple. Cependant, le format ne permettant pas cette forme de présentation, des exemples de transformations d'images sont exposés côte à côte figure 4.21.
Malgré le fait que l'entraînement a été réalisé sur des images de 128x128 pixels, il possible de générer des images de taille arbitraire, puisque les réseaux sont complètement convolutionnels. On notera cependant que le champ réceptif au sein des U-nets utilisés pour la génération est de 62x62 au maximum. La cohérence spatiale ne peut donc pas excéder 124x124. Pour ordre de grandeur, les carrés blancs dessinés sur les images mesurent 85x85 pixels.

Les figures 4.22, 4.23, 4.24, 4.25, 4.26 présentent des agrandissements des régions d'intérêt encadrées sur la figure principale 4.21. Pour chacune des caractéristiques phénotypiques que l'on abordera, on observera comment elle est générée et gommée selon que l'on observe la transformation $A \to B$ ou $B \to A$.
Pour rappel, les noyaux des cellules sont représentés en bleu, la tyrosine hydroxylase (TH) en vert, et l'alpha-synucléine, en rouge. Figure 4.22, on observe que l'intensité du marqueur de tyrosine hydroxilase (marqueur des neurones dopaminergiques) est plus forte dans les axones des neurones de culture WT.
Figure 4.23, aux alentours des corps cellulaires des neurones, la complexité et l'intensité des structures vertes est plus grande.
Figure 4.24, les cellules à gros noyaux et corps cellulaires, contenant de l'alpha-nucléine sont plus nombreux dans les cultures G2019S.
Figure 4.25, on observe un plus grand nombre de petits noyaux dans les images de cultures de neurones G2019S.
Figure 4.26, dans les agrégats de cellules, on observe une plus grande présence et intensité d'un signal diffus d'alpha-synucléine.

4.5 Discussion et perpectives

4.5.1 Interprétation biologique

La méthode expérimentale pour obtenir des cultures de neurones est décrite section 1.3. Des fibroblastes de patients malades de Parkinson sont dédifférenciés

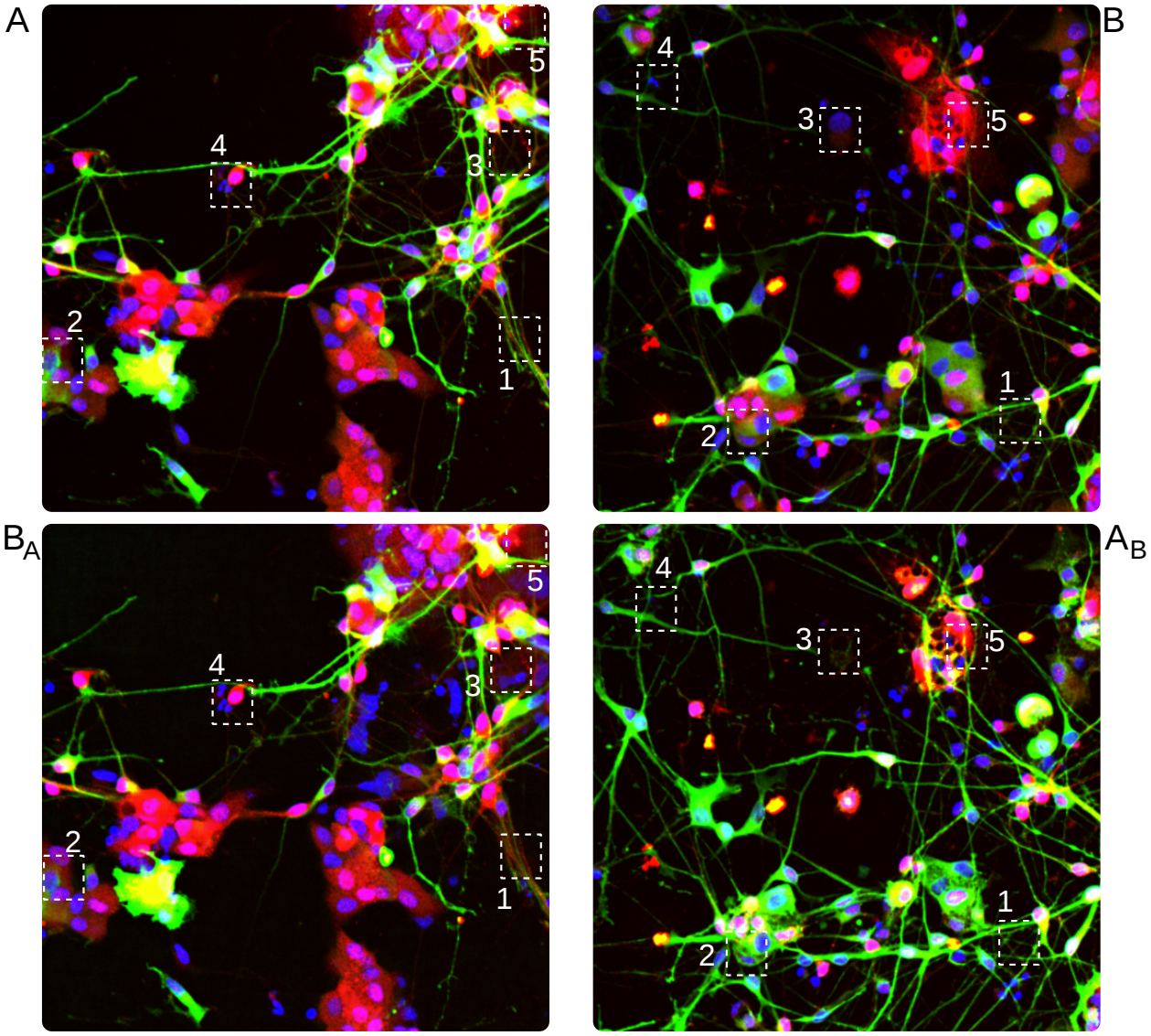


FIGURE 4.21 – **Exemple d'images générées.** Colonne de gauche, transformation d'images de cultures de neurones WT en neurones porteurs de la mutation G2019S. Colonne de droite, transformation de cultures de neurones porteur de la mutation G2019S en neurones WT. Les régions encadrées en blanc, dans chacune des paires d'images, illustrent les changements phénotypiques que l'on observe. Les figures suivantes en présentent des agrandissements.

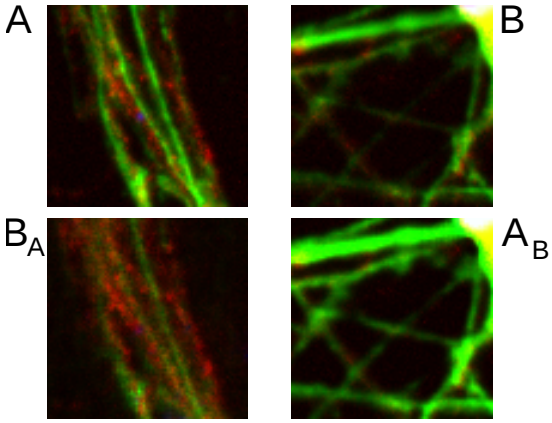


FIGURE 4.22 – **Détail 1.** L'intensité du marqueur vert augmente dans les prolongements des corps cellulaires des neurones WT générés. Colonne de gauche : WT vers G2019S. Colonne de droite : G2019S vers WT.

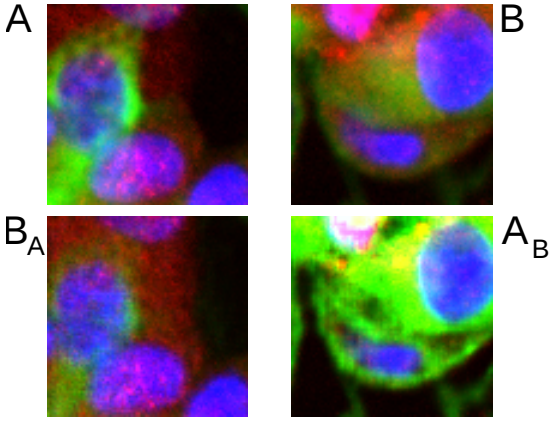


FIGURE 4.23 – **Détail 2.** Le marqueur vert possède une plus grande complexité structurelle autour des noyaux (en bleu) des neurones WT que des neurones G2019S. Colonne de gauche : WT vers G2019S. Colonne de droite : G2019S vers WT.

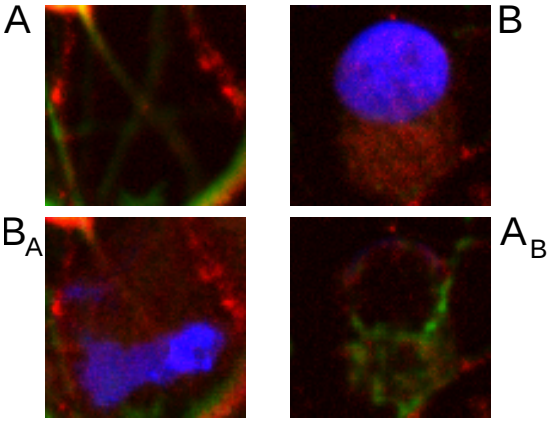


FIGURE 4.24 – **Détail 3.** Apparition ou disparition de cellules à gros noyaux et corps cellulaire. Colonne de gauche : WT vers G2019S. Colonne de droite : G2019S vers WT.

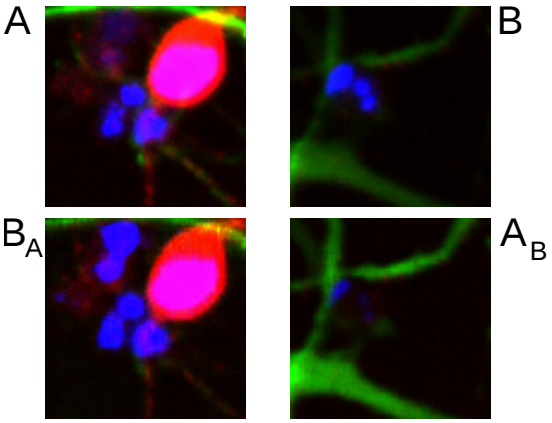


FIGURE 4.25 – **Détail 4.** Apparition et disparition de petits noyaux. Colonne de gauche : WT vers G2019S. Colonne de droite : G2019S vers WT.

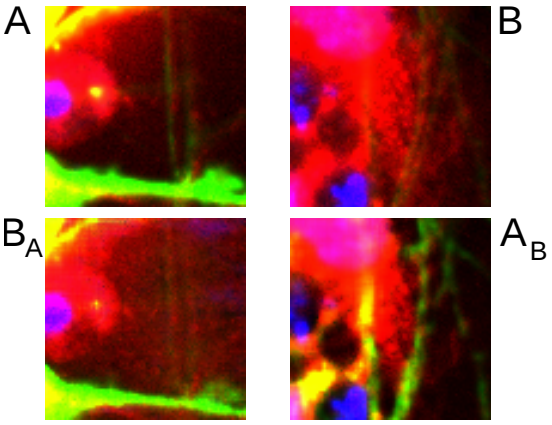


FIGURE 4.26 – **Détail 5.** Augmentation ou diminution d'un signal diffus d'alpha-synucléine. Colonne de gauche : WT vers G2019S. Colonne de droite : G2019S vers WT.

en iPSCs. Le génome de la moitié des iPSCs est modifié pour corriger la mutation G2019S du gène de LRRK2 grâce à l'outil CRISPR-cas9, afin que le génome s'aligne avec le génotype WT. Les deux lignées ainsi obtenues sont transformées en cellules progénitrices de neurones puis on induit la transformation de celles-ci en neurones dopaminergiques. Cette différenciation n'est jamais de 100% des cellules : si déjà 50% des cellules sont différenciées en neurones dopaminergiques, on considère la différenciation un succès. On notera que la seule différence entre les deux types cellulaires devrait être la mutation de LRRK2. Néanmoins, il est possible que la modification du génome par CrisprCas9 ait engendré d'autres modifications que celle rétablissant le génotype WT sur le gène LRRK2.

La figure 4.24 nous montre qu'il existe un déséquilibre de présence de ces cellules rouges à gros noyau sans signal de marqueur de neurone dopaminergique (fluorescence verte) entre les populations de cellules WT et porteuses de la mutation. Il se pourrait que ces cellules soient des cellules progénitrices (NPCs) qui ne se seraient pas différenciées en neurone. En effet, en culture cellulaire, le passage est le procédé par lequel sont sélectionnées des cellules d'une plaque de culture arrivant à saturation, pour perpétuer la lignée cellulaire. Liu et al. [136] montrent que des cellules progénitrices de neurones porteuses de la mutation LRRK2-G2019S perdent leur potentiel de différenciation au cours des passages successifs.
Borgs et al. [137] réalisent des cultures de neurones WT et LRRK2-G2019S. Ils quantifient cette persistance du nombre de NPCs dans les cultures mutantes. Le pendant de cette observation est la moins grande quantité de neurones dopaminergiques. Ils montrent également que le pourcentage de neurones dopaminergiques parmi l'ensemble des neurones générés est beaucoup plus élevé dans les cultures WT. Cela est cohérent avec l'observation de l'augmentation générale de l'intensité de la tyrosine hydroxylase (fluorescence verte) lors de la transformation d'image LRRK2-G2019S vers le domaine WT par le CycleGAN.

Dans les cultures de cellules LRRK2-G2019S, on observe le surnombre de petits noyaux, correspondant à des noyaux morts (voir figure 4.25). Avec l'absence d'une observation concordante dans la bibliographie, nous ne pouvons avancer avec certitude une hypothèse : s'agit-il d'une conséquence de la mutation ou d'un artefact dû aux conditions de culture ou d'obtention des lignées cellulaires ?

Dans leur revue, Civiero et al. [138] recensent une vingtaine d'études rapportant la réduction des excroissances dendritiques dans différents modèles expérimentaux porteurs de la mutation LRRK2-G2019S. La mutation pourrait donc empêcher une maturation normale des neurones. Sur la figure 4.21 dans les images de la colonne de droite, la transformation de la région en haut à droite de la vignette 1 semble décorer les corps cellulaires d'une complexité de signal de thyrosine hydroxylase, faisant penser à des dendrites. Cette observation contribue à valider notre modèle comme un modèle biologique en accord avec les connaissances sur la biologie de la maladie de Parkinson.

4.5.2 Taille du champ réceptif des réseaux

L'application de la technique du CycleGAN à la génération d'images de neurones, possède une limitation impor-

tante : son incapacité à générer des images avec une cohérence spatiale de plus de 120 pixels. En effet, à cette résolution d'image, le neurone dépasse du champ de cohérence.
Afin d'augmenter la taille du champ réceptif des neurones, il est possible d'augmenter la profondeur des réseaux. Concrètement, cela correspond à ajouter des couches de convolution dans les réseaux. Cette technique a été utilisée sur le dataset LRRK2 mais avec des images mal normalisées et sans évaluer les générateurs à l'aide du FID.
Augmenter la profondeur des réseaux permet d'augmenter la cohérence maximale théorique entre les pixels. Cependant, comme on a pu le voir dans l'état de l'art (voir sections 2.4.2 et 2.3.3), les CycleGANs – et les générateurs d'images en général – sont plus adaptés pour modifier ou générer des textures plutôt que des formes. Wang & Al. [73] ont introduit les couches de self-attention qui permettent d'obtenir des corrélations longue distance entre pixels (voir section 2.3.3). Ces couches ont été reprises avec succès dans les générateur SaGAN et BigGAN [74, 80]. Introduire des couches de self-attention dans un réseau de transformation d'images pourrait augmenter sa capacité à construire des dentrites ou des axones, qui nécessite de la cohérence sur une longue distance.

4.5.3 Transfert de style

La FID permet de sélectionner un générateur produisant des images de distributions proches de celles d'un autre groupe d'images. Ces distributions sont calculées à l'aide d'un réseau pré-entraîné. Cette méthode ressemble aux techniques de transfert de style [77-79]. En effet, Johnson et al. associent au générateur d'images stylisées une fonction de coût qui cherche à aligner les activations de deux images, activations calculées dans une ou plusieurs couches d'un réseau pré-entraîné. Cependant, le transfert de style n'aligne pas les distributions de deux groupes d'images mais les représentations de style de deux images. Pour notre application, il est essentiel que ce soit les statistiques de groupes qui soient alignées car l'on souhaite observer des différences phénotypiques statistiques.
Nazki & Al. [139] combinent cependant le CycleGAN et la fonction de coût de perception utilisée par Johnson & Al. pour du transfert de style [79]. Dans cette étude de 2020, c'est le style de l'image source et de l'image transformée qui sont alignés par la fonction de coût. Cela permettrait de renforcer le caractère minimaliste de la transformation, et d'assurer à l'image d'arrivée des qualités visuelles identiques à l'image de départ.

4.5.4 FID

Borji [140] liste un nombre de caractéristiques que la mesure d'évaluation d'un GAN devrait posséder. La FID est la bonne élève du panel des mesures existantes. Elle excelle dans toutes les catégories, que ce soit pour le pouvoir discriminateur entre les vraies images et les images générées, la détection de la diversité des échantillons, l'invariance aux distorsions d'images, la corrélation avec la perception humaine, et son efficacité de calcul.

La FID se mesure à partir de features extraites d'un réseau pré-entraîné sur ImageNet. Cela permet une implémentation simplifiée, car ces réseaux se téléchargent facilement dans toutes les librairie de deep learning. Cependant on pourrait songer à utiliser des réseaux permettant d'extraire des

features plus spécifiques à ces images de fluorescence.

Jiu & Al. [141] introduisent une mesure similaire au FID, mais qui utilise des features extraites d'un réseau autoencodeur (voir section 2.5.1). Ce type de réseau est conçu pour extraire le minimum de features lalentes à l'image qui permettent sa reconstruction. Dans notre cas, on pourrait également penser à utiliser un réseau entraîné à classifier les deux types de culture de neurones.

Par ailleurs, la transférabilité de cette mesure à des images de fluorescence pourrait se mesurer à partir des transformations sur les datasets de Golgi et de translocation. On pourrait corréler la FID avec la distance de Wasserstein entre les distributions des mesures effectuées sur les images générées et les images réelles.

4.6 Conclusion

Les GANs ont été majoritairement utilisés à des fins artistiques, et pour générer plus de données d'entraînement lorsque les datasets sont trop petits. Je montre ici qu'il est possible d'utiliser le CycleGAN pour aider à l'interprétabilité d'une classification d'images, et plus généralement pour découvrir les différences phénotypiques entre deux types de cultures cellulaires proches. Les capacités de cette architecture ont été démontrées sur des datasets dont on connaît a priori le phénotype, phénotype que l'on peut aisément mesurer. Puis, le CycleGAN a été appliqué au dataset de LRRK2. La transformation d'une image permet d'un seul regard de mettre en relief les différences statistiques entre les populations. Cette stratégie permet de se passer du biais de l'observateur qui compile lui-même différences en observant une série d'images. Une transformation effectuée par le réseau correspond à une différence significative entre populations. Les différences morphologiques et de types cellulaires observées dans nos images ont pu être associées à des études de la bibliographie, par exemple, la sur-représentation des neurones dopaminergiques dans les cultures WT et au contraire, la sur-représentation des cellules progénitrices dans les cultures LRRK2-G2019S.

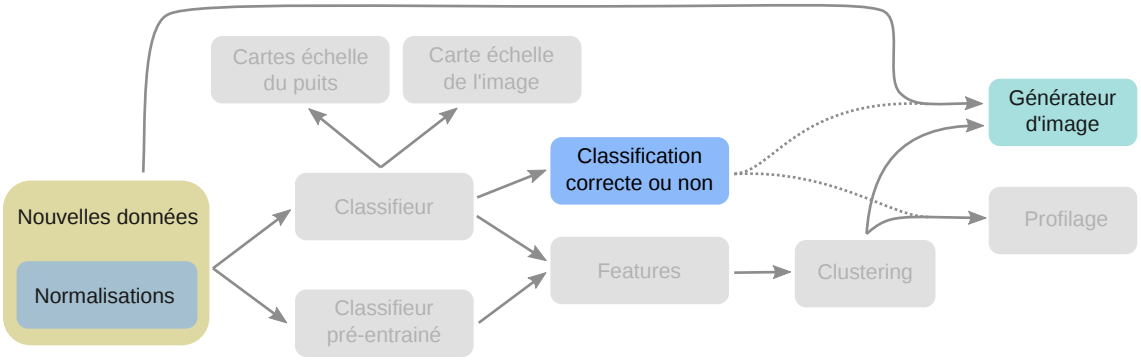


FIGURE 4.27 – **Schéma récapitulatif de la génération de phénotype des conditions WT et LRRK2-G2019S.**
On réalise des transformations entre conditions à partir de données du dataset LRRK2, filtrées par la justesse de la prédiction d'un classifieur.

Chapitre 5

Sous-populations pour la compréhension de phénotypes hétérogènes

5.1 Motivation

Les chapitres précédents essayaient de mettre en valeur les caractéristiques surreprésentées dans les conditions WT ou LRRK2-G2019S (mutant). Néanmoins, la technique de la génération d'image ne peut pas quantifier cette surreprésentativité. Rien ne nous donne la fréquence d'une caractéristique – par exemple la présence de cellule NPCs – dans l'une et l'autre condition. Certaines caractéristiques sont peut-être même spécifiques d'une des deux conditions, on ne peut le vérifier pour le moment, qu'en parcourant l'ensemble des images à la recherche de cette caractéristique.

L'utilisation de sous-populations pour étudier les différences phénotypiques a été utilisée par Slack et al. [106] pour étudier la dynamique de la réponse cellulaire à des perturbations chimiques. En effet, les auteurs font l'hypothèse de l'hétérogénéité de la réponse cellulaire à ces perturbations. Ils modélisent cette hétérogénéité via ces sous-populations. Ils utilisent également la représentation des images par la fréquence des sous-populations pour étudier la réponse cellulaire à l'augmentation de la dose de perturbateur chimique. Ces doses-réponses sont essentielles pour caractériser les molécules à potentiel pharmaceutique dans le processus de recherche et de développement de médicament. Enfin, ils se servent de cet outil pour mesurer la similarité de deux traitements, et de façon sous-jacente, de leurs mécanismes d'action.
L'utilisation de sous-population est plus généralement une technique utilisée pour étudier les questions biologiques où l'hétérogénéité cellulaire est essentielle pour comprendre les dynamiques ou simplement représenter des cultures différentes (voir section 2.5).

L'analyse en sous-populations permet de voir des différences entre cellules dans les différentes populations, elle rend plus lisible les résultats en instaurant un nombre réduit de catégories, et permet de comparer les cultures en fonction des fréquences des sous-populations.

Dans ce chapitre, j'ai étudié quels sont les différents phénotypes présents dans les images et leurs fréquences respectives dans chacune des conditions (WT et mutant) afin de pouvoir quantifier les phénotypes de chaque puits. Dans notre cas, ne pouvant pas descendre à l'échelle cellulaire, les sous-populations sont réalisée à partir d'images.

5.2 Clustering sur features extraites de DenseNet

5.2.1 Extraction de features

Pour la première tentative de clustering, j'ai utilisé le réseau DenseNet et des features extraites de la dernière couche de ce réseau (voir section 3.4.3 pour l'entraînement du DenseNet). On utilise les images avec un background non modifié. L'accuracy du réseau sur ces images est de 81%.
La dernière couche de convolution du réseau contient 1024 cartes de features, ce qui fait, après sous-échantillonnage par moyenne globale, un vecteur de 1024 featues. Il est notoire que le clustering est difficile en très haute dimension. J'ai voulu intégrer une réduction de dimension automatique dans le réseau lui-même. Pour cela, j'ai ajouté une couche au réseau avant de l'entraîner. Cette couche contient 20 convolutions 1×1 qui combinent les 1024 cartes du dernier bloc pour en sortir 10. Après l'étape de sous-échantillonnage par moyennage

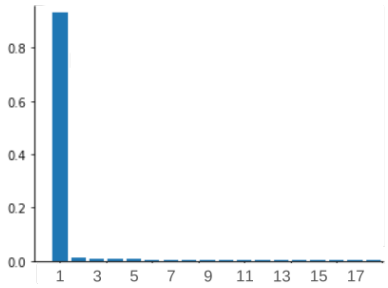


FIGURE 5.1 – **Pourcentage de la variance expliquée pour chaque composante de PCA à partir des 20 features tirées de DenseNet.**

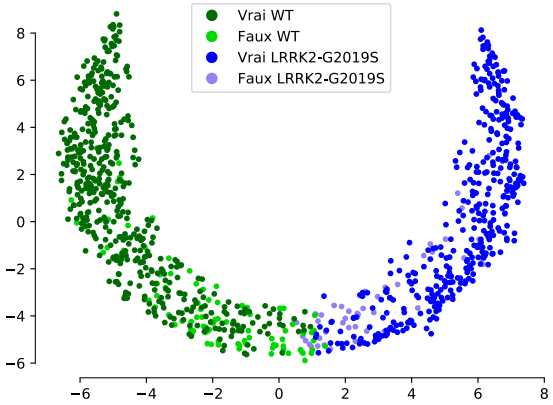


FIGURE 5.2 – **Visualisation des représentations des images dans l'espace des features de Densenet réduit par UMAP.** Chaque point correspond à une image. La couleur des points fait référence à la classification des images par le réseau DenseNet.

global, on obtient des vecteurs de 20 features pour chaque image.

5.2.2 Étude de la représentation des données

Lorsque l'on réalise une analyse en composantes principales sur les features des images (voir figure 5.1), on remarque que plus de 90% de la variance des données peuvent être expliqués par une seule composante. Nos données semblent évoluer dans un espace à une dimension.

On cherche à avoir une idée de la disposition des images les unes par rapport aux autres dans l'espace latent appris par le réseau. L'algorithme UMAP 2.5.2 nous permet de réduire à deux le nombre de dimensions tout en conservant le voisinage des points et la structure globales des données. La technique mentionnée nous permet de visualiser une représentation de nos données présentée figure 5.2. On remarque une séparation nette entre les points verts et les bleus. Cette délimitation correspond au plan qui sépare les images classifiées par DenseNet comme des images de neurones WT de celles classifiées comme contenant des cellules porteuses de la mutation. Les images sont très étirées de part et d'autre de cette délimitation.

Les features extraites de cette façon, s'étirent donc en une dimension, le long d'une courbe séparée par le plan de classification. On remarque d'ailleurs les faux positifs de part et d'autre de cette limite. Il est probable que la couche précédant celle des 20 cartes de features permette une représentation suffisante pour classifier les images et que cette couche ajoutée ne fasse qu'exacerber cette représentation en l'étirant suivant l'axe normal au plan de séparation des classes.

5.2.3 Création des clusters

Après l'étape d'extraction des features, nous réalisons une étape de clustering. Pour réaliser l'étape de clustering des images, on choisit d'utiliser l'algorithme de propagation d'affinité. Ce choix n'est pas vraiment justifiable, cet algorithme peut être intéressant dans les cas où les distances entre les points ne sont pas symétriques (relation entre points unidirectionnelle par exemple). Pour notre problème à distance symétrique et à grande quantité de données, il s'agit simplement d'un algorithme extrêmement

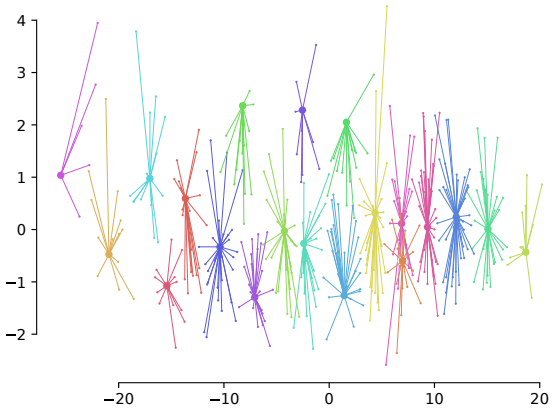


FIGURE 5.3 – **Visualisation des 20 clusters dans un espace réduit par PCA.**

lent et ne performant pas mieux que le très efficace K-moyennes. Cependant, si l'algorithme n'est pas performant, il donne des résultats souventsimilaires à ceux obtenus par l'algorithme de K-mmoyennes.
L'algorithme est exécuté sur les données en 20 dimensions.
Le pourcentage d'images contenues dans chaque cluster varie entre 1 et 8%.

Pour visualiser les clusters, nous utilisons ici une représentation classique de résultats de la propagation : chaque cluster est résumé par un representant (le point plus gros que les autres au centre des groupes) auquel sont rattachés les points appartenant au même cluster. Les données de départ étant en 20 dimensions, nous utilisons ici la projection des points par la PCA calculée plus haut (voir figure 5.1). Les clusters s'étalent selon la direction de plus grande variance trouvée précédemment par la PCA.

Sur la figure 5.4, sont représentés des exemples d'images pour chaque cluster. On remarque qu'il y a une densité d'images riches en fluorescence verte plus grande du côté des clusters à grande majorité d'images de cultures WT (en vert). De même, on observe une plus grande quantité de cellules présentant un fort signal d'alpha synucléine du coté mutant (fluorescence bleue).

Néanmoins, les clusters ne sont pas très homogènes. Le 17e cluster figure 5.4 contient des images d'axones verts, de cellules agrégées les unes sur les autres (roses et vertes), et des cellules à gros noyaux, rouges. Il semblerait que les phénotypes au sein des clusters ne corrèlent pas entre eux mais corrèlent avec leur pouvoir discriminant. Ce qui veut dire que deux images de phénotypes très différents mais très spécifiques d'une condition peuvent se retrouver dans le même cluster. On regroupe les images selon leur probabilité d'appartenir à une classe ou l'autre, pas sur des caractéristiques communes morphologiques ou d'intensité de marqueur. En effet, les cluster sont alignés sur l'axe normal au plan de classification, plus un cluster est loin, plus il est composé d'images que le classifieur catégorise avec certitude. Cette éloignement au plan de classification est représenté sur l'image par la valeur de la moyenne de la première composante PCA sur le cluster (voir colonne de gauche sur l'image de visualisation des clusters 5.4).
Puisque les clusters sont inhomogènes, dans chacun d'entre eux, peuvent être retrouvées des sous-populations très différentes.

On notera aussi que les représentations des images prennent en compte le bruit présent dans le background des images puisque ces représentations sont fortement liées à la classification par DenseNet dont la qualité dépend de la présence des informations contenues dans le background (voir section 3.4.3). Cela pourrait impacter l'homogénéité des clusters si cette information prédomine lors du choix de regroupement des images.

Pour améliorer l'homogénéité des clus-

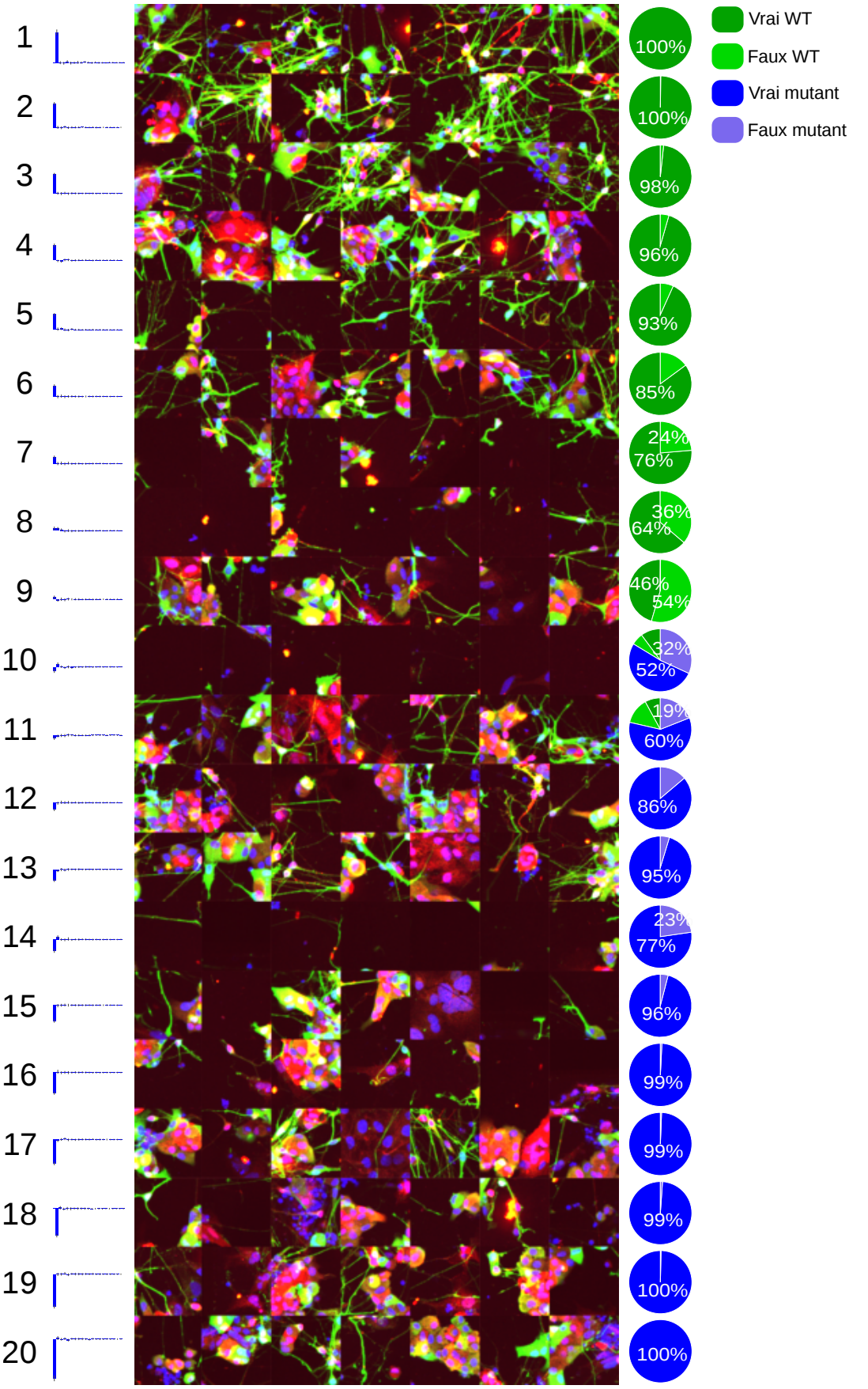


FIGURE 5.4 – **Visualisation des clusters.** À chaque ligne correspond un cluster. La colonne de gauche montre la moyenne des valeurs des composantes de PCA pour chaque cluster. Les camemberts sur la droite représente la composition de chaque cluster en rapport avec la classification par DenseNet. Les images sont piochée aléatoirement dans chaque cluster.

ters, on a également tenter d'augmenter le nombre de clusters et d'utiliser des features blanchies par PCA [114]. Le blanchiment par PCA correspond à utiliser les composantes de PCA normalisées pour représenter les images. Cela procure un plus grand poids à des features comportant possiblement une information non-essentielle pour la classification, mais intéressante pour un clustering. L'information mise en valeur peut tout aussi bien être du bruit. Il a semblé que l'homogénéité des groupes d'images étaient meilleure, toutefois, les clusters étaitent très redondants, de l'hétéréogénéité persistait et la visualisation de 100 clusters est plus pénible.

Pour terminer, il semble que les features apprises ne sont pas les plus à-même de fournir un clustering de qualité.

5.3 Clustering sur features extraites de CBR-Small

5.3.1 Extraction de features

Pour cette partie suivante, j'ai voulu utiliser des features extraites d'un réseau simple ayant appris sur le dataset avec background égalisé (voir section 3.4.3). Pour cela, nous avons choisi le réseau CBR-Small dont on a parlé brièvement dans le chapitre classification, section 3.4.4. Les images de 256×256 sont sous-échantillonnées d'un facteur deux. L'accuracy de ce réseau sur le dataset avec fond d'image égalisé est de 71%. La dernière couche du réseau contient 256 cartes de features, ce qui résulte en une représentation de chaque image par un vecteur de 256 features.

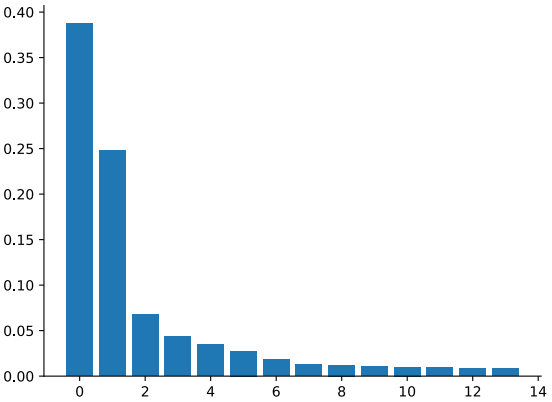


FIGURE 5.5 – **Pourcentage de la variance expliquée pour chaque composante de PCA à parir des vecteurs de features du CBR-Small.**

5.3.2 Étude de la représentation des données

Une PCA (analyse en composantes principales) est réalisée sur les vecteurs associés aux images et il semble que les données évoluent cette fois en plus grande dimension.

Comme dans la partie précédente, on réduit la dimension les données à deux features grâce à l'outil UMAP. On visualise les points dans ce nouvel espace figure 5.6. Comme dans la section ci-dessus, les points sont disposés de façon très continue dans l'espace. Il n'y a pas de clusters de plus grande densité visible. Toutefois, à la différence de la projection des images précédentes, les points verts sont beaucoup moins séparés des bleus dans cette représentation-ci. Grâce à une interface interactive qui nous permet d'observer les images correspondant à chaque point, on note que qualitativement, la distribution des phénotypes à travers cette représentation est assez continue. La pointe en bas à gauche correspond à des images très noires, en haut, ce sont les images à forte teneur en axone, en bas à droite, à haut signal d'alpha-synucléine. Malheureusement, il est dif-

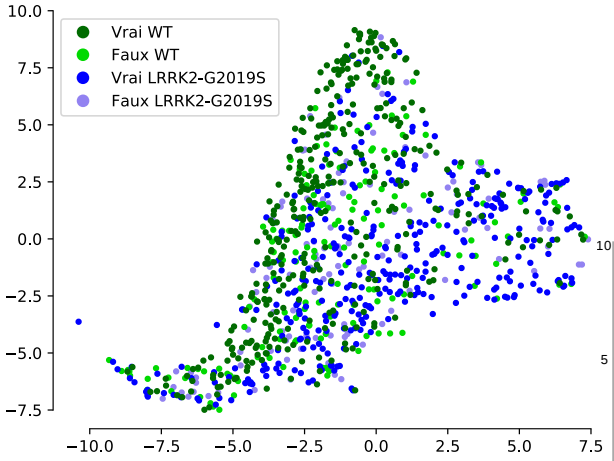


FIGURE 5.6 – **Visualisation des représentations des images dans l'espace des features du CBR-Small réduit par UMAP.** Chaque point correspond à une image. La couleur des points fait référence à la classification des images qu'ils représentent par le réseau CBR-Small.

ficile de visualiser cette continuité via ce support.

5.3.3 Création des clusters

À la vue de l'uniformité de la distribution des points dans cet espace à deux dimensions, les algorithmes s'appuyant sur des hétérogénéités de densité serait inefficaces pour regrouper les données. On choisit donc d'utiliser un algorithme de partitionnement des données : la méthode des K-moyennes ou K-means (voir section 2.5.3). Pour cet algorithme, il est nécessaire de donner un nombre de clusters souhaité en paramètre d'entrée. Ce choix a été fait de façon très arbitraire. Les figures 5.7 et 5.8 nous montre le résultat du clustering pour 15 et 23 clusters. La couleur de chaque cluster reflète la proportion d'images WT au sein du cluster. On remarque que les clusters verts et bleus sont assez séparés, mais l'augmentation du nombre de clusters fait apparaître un groupement vert en bas à droite dans la partie plutôt

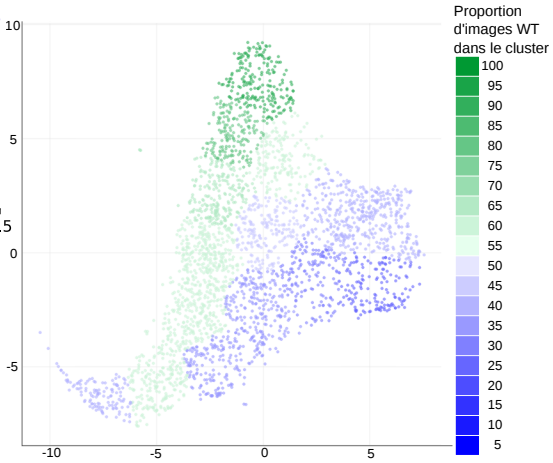


FIGURE 5.7 – **Visualisation de 15 clusters obtenus par l'algorithme K-moyennes, dans un espace réduit par UMAP.** La couleur des clusters représente la proportion d'images de chaque type dans le cluster.

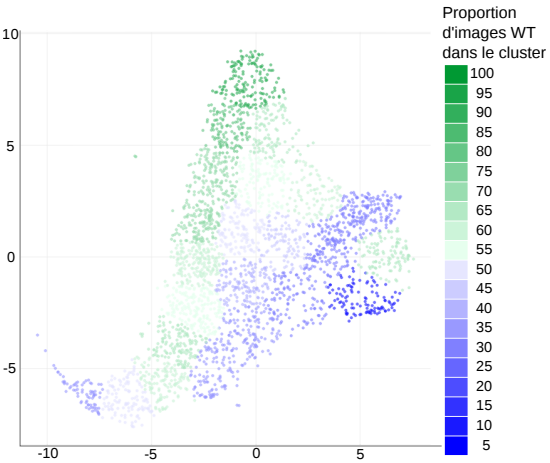


FIGURE 5.8 – **Visualisation de 23 clusters obtenus par l'algorithme K-moyennes, dans un espace réduit par UMAP.** La couleur des clusters représente la proportion d'images de chaque type dans le cluster.

bleue. On conservera ce nombre de clusters pour la suite.

On peut visualiser la fréquence des clusters dans chaque puits figure 5.9. On remarque une grande dissimilarité des profils entre les puits WT, à gauche et les puits LRRK2-G2019S, à droite, et au contraire, une grande similarité des profils intra-condition. Cela soutient l'utilisation des clusters comme description de l'hétérogénéité cellulaire.

Sur la figure 5.10, sont représentés des exemples d'images pour chaque cluster. Même remarque que prédemment, les images à plus grande intensité du marqueur des neurones dopaminergiques (fluorescence verte) proviennent de clusters à grande proportion d'images de cultures WT (en vert dans les camemberts). Et l'on observe une plus grande quantité de cellules présentant un fort signal d'alpha synucléine du coté mutant (en bleu dans les camemberts). Le cluster numéro 6, cependant, présente des images à très fort signal d'alpha-synucleine. Il correspond au cluster vert au milieu des clusters bleus, dans la figure 5.8. L'intensité du signal d'alpha-synucléine n'est pas représentative de l'état muté. Qualitativement, les clusters nous semblent beaucoup plus homogènes que précédemment.

Les clusters obtenus avec les features extraites de DenseNet contiennent des images provenant le plus souvent des mêmes conditions (WT ou mutant). Les images sont uniformément classée dans une classe pour chaque cluster (revoir figure 5.4). À l'opposé, les clusters obtenus grâce aux features extraites par le réseau CBR-Small sont plus mixte en terme de conditions. Les clusters sont aussi mélangés en label de classification : les clusters contiennent tous des images WT et mutant, bien et mal classées. Les features extraites du DenseNet permettent de réaliser une très bonne classification. Cependant, ces features perdent de vue des notions plus spécifiques qui seraient utiles dans un clustering. Au contraire, le petit réseau CBR-Small semble moins à même de trouver une représentation convenable pour une classification, mais conserve des features plus générales.
Le fait que les deux classes sont assez bien identifiées dans chaque cluster (accuracy supérieure à 50% pour chacune des classes dans chaque cluster) signifie qu'il existe un degré d'information supplémentaire dans chaque cluster qui permet de différencier les images de cultures WT des images de cultures LRRK2-G2019S.

5.4 Génération conditionnelle d'un cluster à l'autre

Les premières parties de ce chapitre nous permettent d'obtenir une composition des puits en fréquence de cluster. La problématique qui se pose alors est de savoir de quoi sont composés les clusters. On retombe sur la problématique initiale qui est de comprendre les différences statistiques entre deux groupes d'images. Cette fois-ci, la question se pose avec 23 groupes d'images.

Le StarGAN a justement été conçu pour ce type de problématique (voir section 2.4.2). Il généralise l'utilisation du CycleGAN à un nombre illimité de domaines en n'utilisant qu'un seul générateur pour toutes les transformations. Pour ce faire, le label du domaine cible est donné en entrée du réseau en même temps que l'image à transformer. De même, le StarGAN ne contient qu'un seul discriminateur. Il est chargé non seulement de déterminer si les images sont réelles ou ont été générées, mais

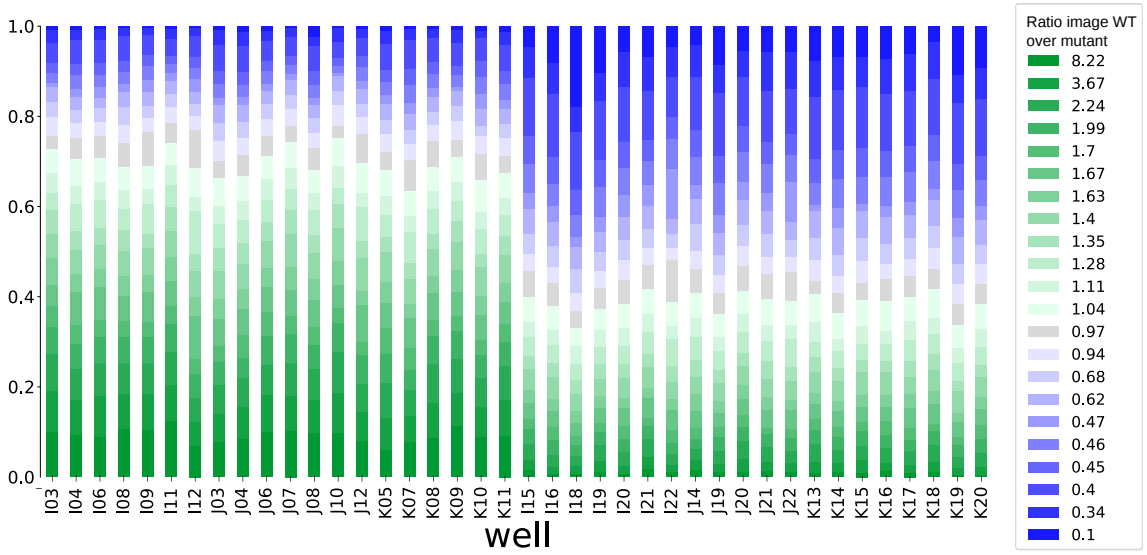


FIGURE 5.9 – **Fréquence des clusters dans chaque puits.** En abscisse les différentes colonnes correspondent aux puits sur la plaque de criblage. À gauche, de I03 à K11 se trouvent les puits correspondant au WT. À droite, de I15 à K20 se trouvent les puits LRRK2-G2019S. En ordonnée, la proportion d'images. Chaque barre correspond à l'empilement des 23 clusters trouvés ci-dessus par l'algorithme des K-moyennes. La couleur pour ces clusters est déterminée par leur ratio d'images WT sur les images mutantes et explicitée à droite de la figure.

aussi d'apprendre à retrouver leur label. Le générateur est ainsi pénalisé si le label retrouvé par le discriminateur n'est pas le bon. Dans notre cas, le label est le numéro du cluster. On notera que, comme le CycleGAN, le générateur apprend à réaliser la transformation minimale vers le cluster cible. Le générateur est en effet optimisé de façon à ce que la transformation inverse permette de retomber sur l'image d'entrée.
Le générateur de l'implémentation StarGAN utilisée possède une architecture contenant des blocs résiduels (voir section 2.2.5). Isola et al. [91] montre que cette structure est plus simple à entraîner.

La figure 5.12 présente les transformations de 10 images, centroïdes de 10 clusters, vers les 23 domaines que représentent chaque cluster. Le fait que ce soit la même image transformée dans des clusters différents permet de se rendre compte de certaines différences statistique entre les clusters.
Les axones et les dendrites des neurones sont plus complexes et la tyrosine hydroxylase (fluorescence verte) plus concentrée dans le cluster 1 comparé au cluster 2. Mais le cluster 1 semble contenir plus de noyaux morts. En effet, la transformation de l'image 4 présente des différences de quantité de ronds de fluorescence bleue.
Entre les clusters 2 et 14, la différence ne semble pas tant être l'intensité du marqueur des neurones dopaminergiques (fluorescence verte) mais la complexité des protrusions neuronales autour du soma. La densité des noyaux est plus grande pour des images du cluster 22 comparée à celle du cluster 23.
L'intensité du signal d'alpha-nucléine n'est pas plus grande pour les images du cluster 3 en comparaison du cluster 2. À la vue des exemples d'images piochées dans ces clusters figure 5.10, cette interprétation aurait pu être faite.

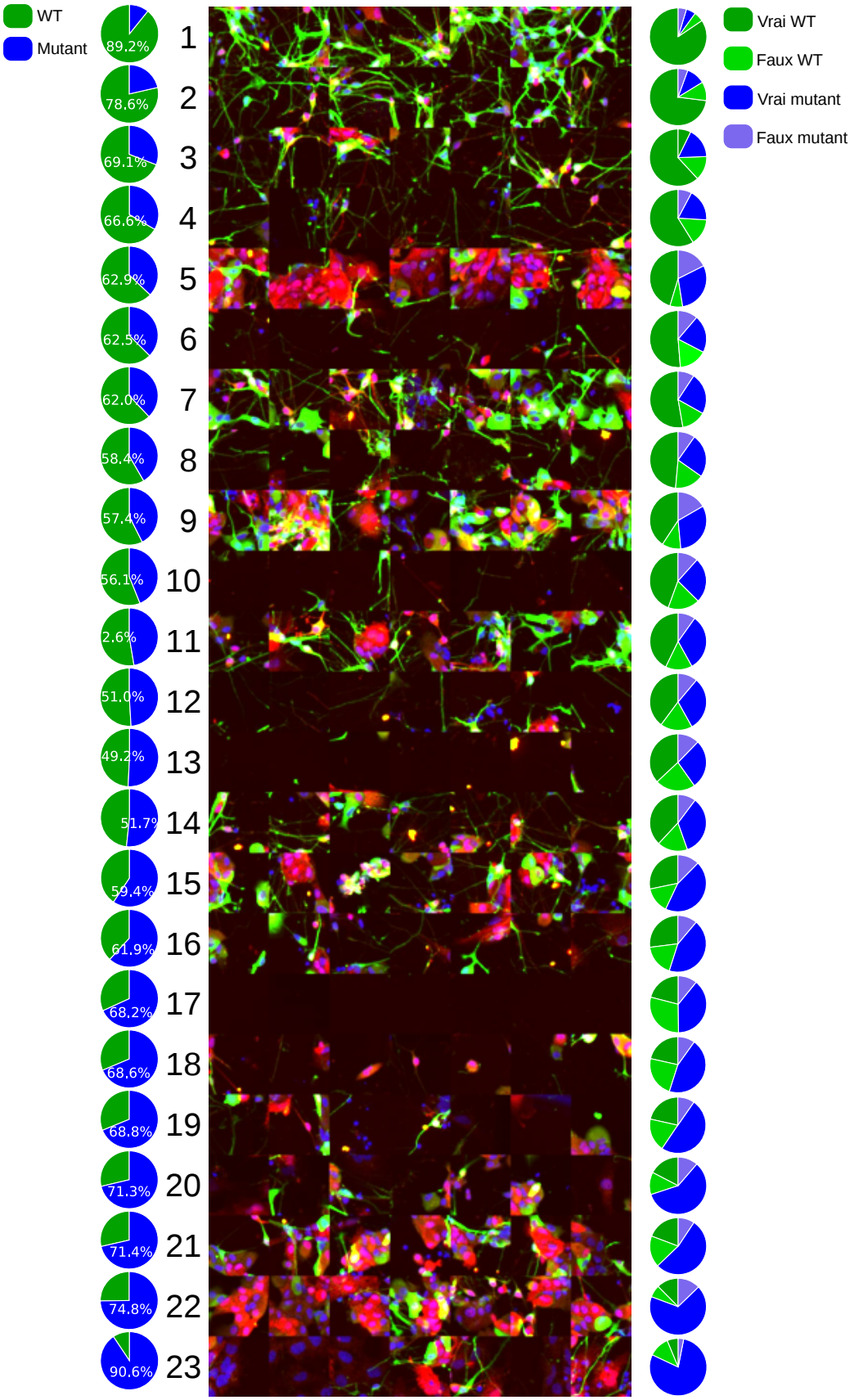


FIGURE 5.10 – **Visualisation des clusters.** À chaque ligne correspond un cluster. Les camemberts de la colonne de gauche montrent la composition en conditions d'image. Les camemberts de la colonne de droite représente la composition de chaque cluster en rapport avec la classification par CBR-Small. Les images sont piochées aléatoirement dans chaque cluster.

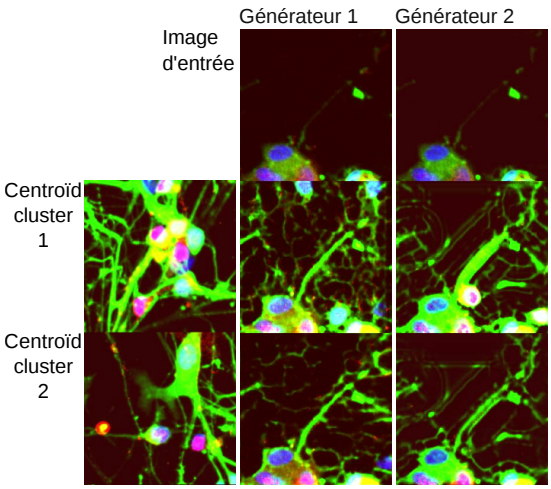


FIGURE 5.11 – **Génération avec et sans égalisation de fond.** Le générateur 1 travaille avec des images contenant un fond, le générateur 2 avec des images à fond égalisé.

Des entraînements de StarGAN ont été réalisés sur le dataset contenant le fond de l'image et sur le dataset à fond égalisé. La figure 5.11 présente deux générateurs issus de ces deux entraînements. L'image de départ est en grande partie noire. Le générateur doit donc inventer de la complexité verte pour transformer l'image vers les domaines des clusters 1 et 2. La figure illustre le fait que la complexité inventée par le générateur n'est pas du même type lorsque l'on conserve l'information contenue dans le fond. En effet, le générateur 2 ne possède pas de bruit de fond sur lequel s'appuyer pour nucléer une génération de neurone. Il s'appuie alors sur l'information la plus proche pour créer des liserés à intervalles réguliers. Le générateur 1 utilise l'information contenue dans le fond pour générer cette complexité. Les images obtenues ainsi ressemblent peu à la réalité biologique.

5.5 Discussion et perspectives

5.5.1 Pertinence des sous-populations que l'on définit

Les deux premières parties de ce chapitre démontrent l'importance d'une bonne représentation des images, afin d'obtenir des clusters les plus homogènes possible. Pour évaluer l'homogénéité des données d'un cluster et sa distance avec les autres clusters, on pourrait adapter la mesure de la FID. Des sous-populations bien définies présenteraient une distance entre elles bien plus grande que les distances à elles-mêmes.

D'autres techniques permettent l'extraction de features pour des images. Elles présentent la caractéristique de ne pas être dépendante d'une tâche de classification qui peut biaiser la représentation des données apprises, comme on l'a vu dans la première partie de ce chapitre (section 5.2). Ces techniques ont été présentées dans l'état de l'art, section 2.5.1. Les techniques d'auto-encodage, permettre d'apprendre les vecteurs latents aux images, à partir desquels il est possible de les reconstruire. Le StyleGAN version 2 permet aussi, de réaliser la fonction inverse de la génération d'image et de retrouver le vecteur dans l'espace latent, permettant la génération de l'image d'intérêt. Relativement aux tâches de clustering et de classification, il n'y a pas de système d'extraction de features qui soit meilleur dans l'absolu, tout dépend du contexte de la tâche à réaliser.

Ensuite, se pose la pertinence des sous-populations d'images au niveau biologique. En effet, les clusters obtenus à l'aide de l'algorithme de K-moyennes sont très arbitraires. Les délimitations entre les clusters par exemple, séparent des imagettes qui se ressemblent beaucoup. Dans notre cas, les représentations

des données sont réparties de façon très continue, et l'on ne distingue pas a priori de cluster de points de densité plus grande (voir figure 5.10). Il est donc difficile d'envisager une meilleure méthode sur ces données (dans cet espace de représentation à deux dimensions). De plus, le clustering au niveau des images et non au niveau cellulaire ne facilite pas l'interprétation de ces sous-populations et compromet parfois leur pertinence. En effet, comment classer une image de précurseur neuronal par dessus lequel passe un axone de neurone dopaminergique ? La densité cellulaire et l'hétérogénéité phénotypique des cellules agrégées ou superposées les unes sur les autres complexifient la tâche de clustering. On pourrait envisager de réaliser du clustering de plus faible granularité en descendant à l'échelle des pixels et permettre de regrouper des groupes de cellules, des cellules uniques ou structures cellulaires sous un même label. Larsson et al. [142] apprennent des cartes de segmentations fines de façon non-supervisée à partir d'une tâche de localisation visuelle. prédiction de position caméra dans une scène.

5.5.2 Génération d'images

Amélioration de la qualité d'images

Lors de l'étude de génération d'images appartenant à un cluster, je n'ai pas utilisé de métrique permettant d'évaluer les générateurs. Liu et al. [141] proposent une version de la FID permettant de s'adapter aux données multi-classes en adaptant une mixture de Gaussiennes aux données plutôt qu'une simple Gaussienne comme c'est le cas pour la FID classique. On pourrait ainsi déterminer le meilleur générateur d'un entraînement (pour l'intant, je choisis à l'oeil celui dont les intensités de fluorescence corrèlent le plus avec la visualisation de cluster, figure 5.6).

Les images générées souffrent d'un manque de vraisemblance lorsqu'il s'agit de créer de l'information à longue distance et dans des zones complètement noires. Afin de palier à ces manques, il pourrait être intéressant d'utiliser des blocs de self-attention (voir section 2.3.3) pour améliorer la cohérence de longue portée. De plus, afin de permettre au réseau de générer des protrusions cellulaires crédibles, il serait envisageable d'ajouter du bruit en entrée du réseau comme c'est le cas dans le générateur StyleGAN (voir section 2.3.5).

Accroître l'intérêt de la génération d'image

La figure 5.10 de visualisation des groupes d'images nous présente des clusters assez homogènes en composition. Mais comme je l'ai remarqué, au sein d'un cluster, la classification de la condition présente en plus petite proportion – WT ou LRRK2-G2019S – n'est pas mauvaise. Il existe donc de l'information supplémentaire dans chaque cluster que l'on pourrait exploiter pour mieux définir les phénotypes des deux cultures. Cela consisterait à utiliser le StarGAN, avec en entrée, l'image, le label du cluster mais aussi la classe (WT ou LRRK2-G2019S) correspondant à l'image. Le discriminateur aurait un objectif supplémentaire, celui de trouver la classe de l'image.

Cela ressemble en fait à l'utilisation du CycleGAN, mais le fait d'utiliser un découpage en clusters facilite le travail du générateur. En effet, d'un point de vue de distribution de données, utiliser des clusters pour générer des images, revient à décomposer la distribution

en modes. On s'assure ainsi de ne pas passer à côté d'un certain type d'images lors de la génération. Ce type de problème dans les GAN s'appelle le *mode drop* ou abandon de mode. Pour un générateur entraîné sur un dataset de chiffres, ce phénomène s'illustre par l'échec à la synthèse de certains chiffres alors que les autres sont très bien générés.
Dans le chapitre précédent, l'entraînement du CycleGAN était réalisé sur un dataset duquel on avait retiré les images mal catégorisées par un classifieur. On pourrait interpréter cela comme cacher des modes au générateur et donc induire volontairement un abandon de mode. En effet, en ne conservant que les images bien catégorisée, on abandonne par exemple pour le dataset WT toutes les images de cellules progénitrices de neurones.

5.6 Conclusion

Pour conclure, j'ai montré dans ce chapitre qu'il était possible d'utiliser les représentations apprises par les réseaux de classification pour regrouper les imagettes issues de culture de neurones. Il est possible de représenter les puits par des vecteurs de fréquence de sous-populations apprises de cette façon. À travers deux exemples d'extraction de features à partir de réseaux différents, je montre l'importance d'une représentation adéquate des imagettes. La génération d'images peut par la suite aider à la compréhension de ce qui fait l'identité de chaque cluster.

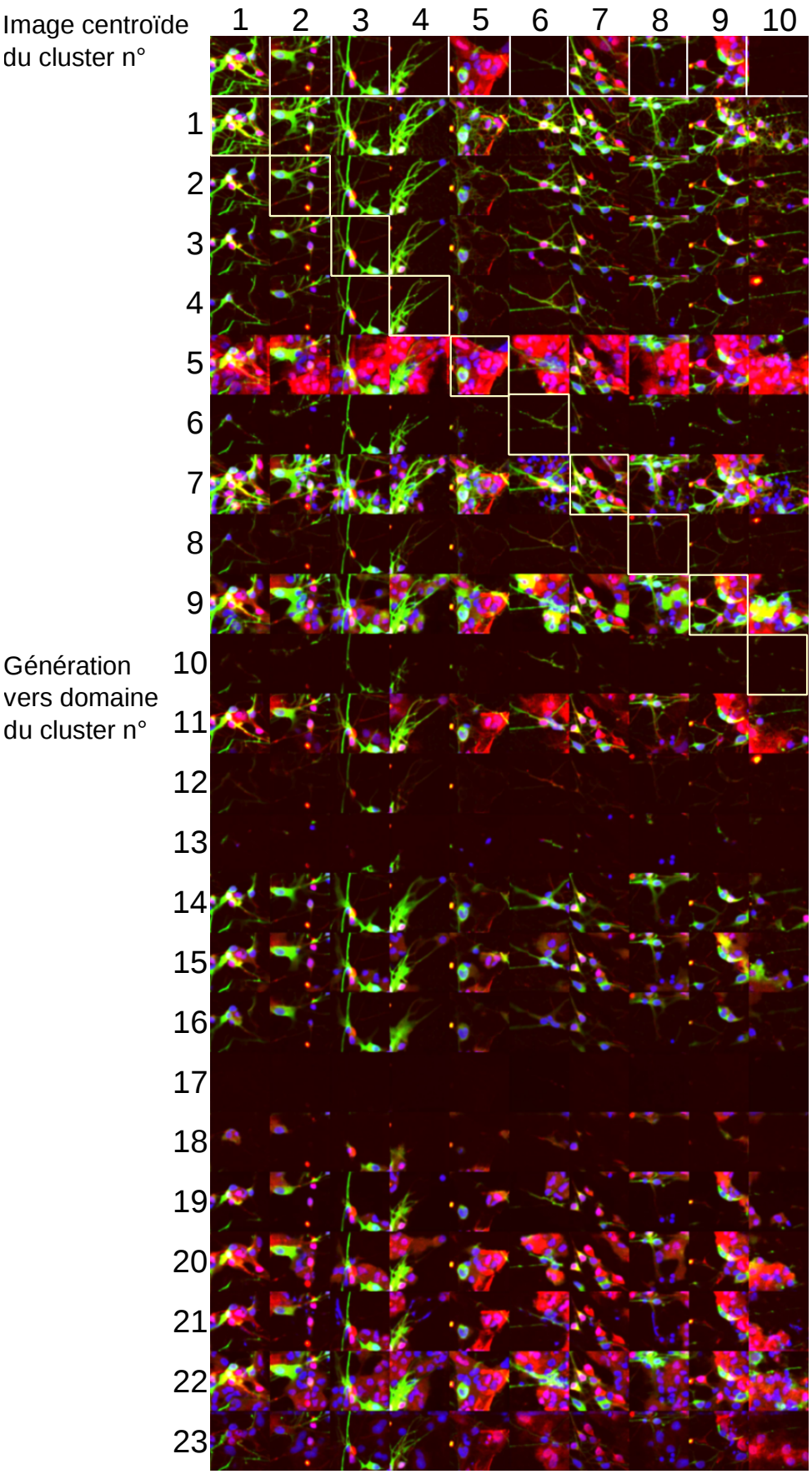


FIGURE 5.12 – **Transformation de centroïde des premier clusters dans les domaines de chaque autre cluster.** La première ligne d'image correspond aux centroïdes des 10 premiers clusters. Ce sont des images réelles. Chaque autre ligne est la transposition de ces images dans le domaine d'un cluster dont le label est indiqué sur la gauche.

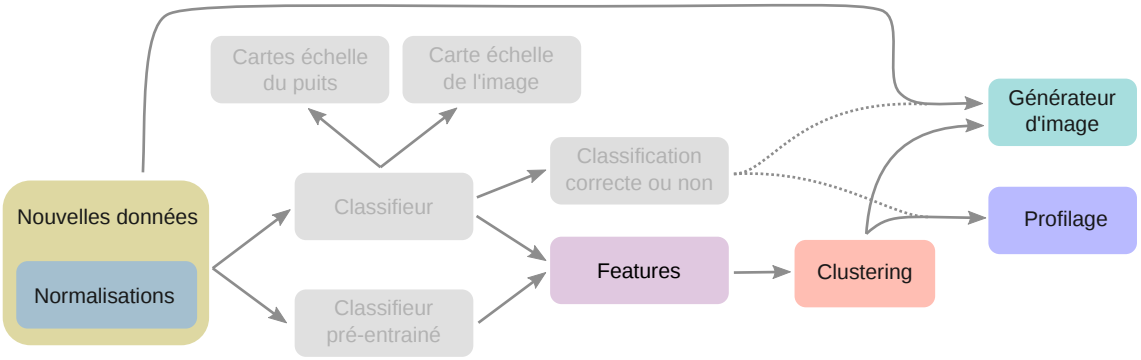


FIGURE 5.13 – **Schéma récapitulatif de la génération de profils de sous-population et de générations de clusters.** À l'aide de features apprises par des réseaux classifieurs, on clusterise les images. À partir des clusters, ou sous-populations, on réalise du profilage des puits. Pour mieux comprendre la composition des clusters, on génère des images de clusters à partir de la même image d'entrée.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion

Dans cette thèse, je me suis focalisée sur l'étude des différences phénotypiques visuelles subtiles entre des populations neuronales isogéniques – à l'exception de la mutation LRRK2-G2019S. Cette mutation est censée induire un risque accru de développer la maladie de Parkinson pour les personnes qui en sont porteuses. Je me suis appuyée pour cela sur des images de microscopie de fluorescence fournies par l'entreprise Ksilink. Ces images sont obtenues lors d'expériences préliminaires à des campagnes de criblage à haut-contenu. Les expériences visent à s'assurer que l'on peut mesurer une différence phénotypique entre ces deux cultures de neurones-contrôles, afin de pouvoir par la suite cribler des molécules ayant un potentiel thérapeutique.

Les techniques de deep learning permettent d'apprendre des représentations d'images de façon automatisée. Cet apprentissage est non-biaisé par des a priori sur la façon dont il faut construire les représentations – c'est-à-dire les descripteurs – des images. J'ai pu mettre à profit des outils de classification d'images par apprentissage profond pour m'assurer de la possibilité de distinguer les deux cultures de neurones (WT et LRRK2-G2019S). Les réseaux de neurones fonctionnent en boîte noire. Les modèles apprennent par eux-mêmes une représentation des données, à partir d'une grande quantité d'images labellisées.

J'ai créé et utilisé des outils permettant l'explication spatiale de la classification au niveau du puits de culture et de l'image d'entrée (voir Chapitre 3). Seulement, la localisation d'un phénotype de suffit pas à l'expliquer dans des images où de nombreux types cellulaires co-existent et s'enchevêtrent.

La génération d'images par la technique de deep learning a été mise à profit en biologie pour augmenter les quantités de données disponibles pour des petits datasets. J'ai dévié l'utilisation de ces outils afin de mettre en évidence les différences de phénotype entre les deux lignées cellulaires neuronales. À partir d'une image de phénotype WT, le générateur d'images modifie les caractéristiques minimales nécessaires pour obtenir une image de phénotype LRRK2-G2019S. Le générateur permet de rendre compte des statistiques de tout le jeu de données lors de la transformation d'une image simple. La recherche des différences visuelles entre les cultures en est grandement facilitée. Cette génération d'images a permis d'observer des différences morphologiques et des types cellulaires que l'on a pu associer à des études de la littérature (voir Chapitre 4).

Les cultures de neurones différenciés à partir de cellules pluripotentes présentent une forte hétérogénéité. Le processus de différenciation est optimisé pour obtenir le plus grand rendement de neurones dopaminergiques. Toutefois, les cultures contiennent aussi des cellules progénitrices de neurones, des neurones immatures, des neurones non-dopaminergiques. Une façon de quantifier cette hétérogénéité est de l'étudier à l'aide de sous-populations cellulaires. J'ai montré que ces sous-populations ou clusters d'images peuvent être obtenus grâce à des représentations apprises par des classifieurs. La qualité de la représentation est essentielle pour assurer l'homogénéité visuelle de ces clusters. À partir des sous-populations, des profils de fréquences peuvent être construits pour représenter chaque puits. Ces profils permettent une nette distinction entre puits WT et LRRK2-

G2019S. La génération d'images a encore permis de révéler les statistiques sous-jacentes aux différentes sous-populations.

6.2 Perspectives

6.2.1 Utiliser des outils de deep learning pour le criblage à haut-contenu

Explication de phénotypes

Les nouvelles technologies d'édition et de reprogrammation cellulaires offre une grande souplesse pour la génération de modèles cellulaires complexes pour des maladies à origine génétique. Les différences phénotypiques entre contrôles négatif et positif ne sont pas toujours connues a priori. Les connaissances sur la biologie associée sont parfois incomplètes. L'explication de différences phénotypiques prend alors tout son sens. L'intérêt du deep learning réside dans la facilité de son utilisation lorsque les conditions matérielles sont réunies. C'est aussi un outil non-biaisé qui ne nécessite pas d'expertise biologique, ou d'ingénierie de features. La méthode est générale, elle peut s'appliquer à tout type de jeu de données, pourvu que les images soient en nombre suffisant.

Pour le criblage en lui même

Dans ce manuscrit, nous nous sommes intéressés à l'étude d'un modèle cellulaire de la maladie de Parkinson. Cependant la partie criblage n'a pas encore été réalisée. Pour l'analyse de criblage par deep learning, deux problèmes apparaissent. Le premier concerne la génération d'une représentation capable de représenter la diversité des phénotypes qui seront observés lors des test moléculaires. Il s'agit aussi de s'assurer que cette représentation permet de rassembler les phénotypes qui se ressemblent. Le second est l'alignement des plaques de culture.
Ando et al. [114] proposent une façon originale de s'atteler à ces deux tâches. Ils utilisent un réseau entraîné à apprendre à représenter des images de requêtes internet. Ce réseau est entraîné grâce à une fonction de coût triplet (voir section 2.5.1). Cette fonction de coût permet d'apprendre une métrique plutôt qu'une classification en rapprochant les images qui se ressemblent. Ainsi, la représentation des images est réalisée dans un espace continu. Sans réentraînement, ils appliquent ce réseau à des données de criblage à haut-contenu (dataset BBBC01 [63]). Un traitement par normalisation de PCA et alignement de corrélation [143] réalisé sur les contrôles négatifs de toutes les plaques, permet de les aligner, et de minimiser les variations inter-batch. Ils dépassent les scores des études précédentes.

6.2.2 Les outils de deep learning, des algorithmes boîte noire ?

Une intelligence artificielle en quête de sens

Les prouesses réalisées par les techniques de deep learning sont étourdissantes. Les traductions réalisées par l'algorithme de DeepL sont impressionnantes de justesse, les voitures autonomes présentes moins de danger que celles conduites par les humains, les réseaux de neurones permettent des classifications d'images au delà des capacités humaines. Cependant, une certaine réticence à leur utilisation peut naître du fait de leur apprentissage et de leur utilisation sous forme d'algorithme boîte-noire. Un réseau clas-

sifieur ne donne pas d'explication lorsqu'il réalise une catégorisation. Pour certaines applications, la performance impressionnante du réseau peut suffire à gagner la confiance de ses utilisateurs, mais d'autres applications plus sensibles, nécessitent de fournir les indications de la façon dont le choix a été réalisé. Par exemple, ce ne peut être le cas pour la classification d'une demande de prêt ou des outils d'aide à la décision médicale.

Les criblages pharmaceutiques sont extrêmement coûteux. Pour obtenir d'une entreprise pharmaceutique qu'elle dépense 1 millions d'euros pour mettre à disposition sa bibliothèque de molécules pour un criblage, les arguments exposés doivent être de poids. Les deux sous-parties suivantes présentent deux méthodes dont le potentiel m'est apparu très important.

Interfaces interactives pour la visualisation de classification

Des outils de visualisation de features et de cartes d'activation permettent d'obtenir des intuitions sur la classification réalisée (voir sections 2.2.6 et 2.2.6). Cependant les interfaces interactives introduites par Olah et al. [59] sont d'un autre niveau d'explicativité (voir lien vers les interfaces précédemment cité dans le chapitre 3). On y découvre les carte d'activations reliant les couches entre-elles (voir figure 6.1). Chaque carte d'activation correspond en effet à un bloc de cartes de features. Les cartes de chaleur en jaune et bleu correspondent approximativement aux deux classes "Labrador Retriever" et "Tiger cat". Si l'on survole une des cases, on peut observer comment cette case est influencée spatialement dans les couches précédentes.
Une autre visualisation presentée dans l'état de l'art, figure 2.13, permet de décomposer un bloc de features en composantes principales, permettant de visualiser spatialement ces groupes sémantiques.
Les interfaces interactives permettent de démoduler l'information contenue localement par une carte de features dans les couches très hautes afin d'allier la compréhension du type de features (grâce à des visualisations de features) et leur localisation, dans des couches plus basses et donc plus résolues.

Vers des représentation ayant un sens

Afin de donner du sens à la représentation, quoi de mieux que de de donner du sens aux features qui la composent ? C'est ce que permettent de réaliser les architectures de type StyleGAN [71, 116].
StyleGAN a été introduit pour réaliser de la génération de visage. C'est le meilleur générateur à ce jour. Sa puissance réside dans le fait que les features latentes des images sont orthogonales entre-elles (non-corrélées). Cela permet d'avoir un grand contrôle sur les images générées. Il se trouve que les features sont aussi interprétables. La seconde version de l'algorithme [116] permet également de retrouver les features latentes correspondant à une image d'entrée. Avec un dataset suffisamment grand pour déployer la technique, il est donc théoriquement possible d'obtenir des descripteurs d'image interprétables. Non seulement le deep learning serait capable d'apprendre seul des descripteurs, mais ces derniers pourraient être aussi informatifs que ceux définis par des humains.

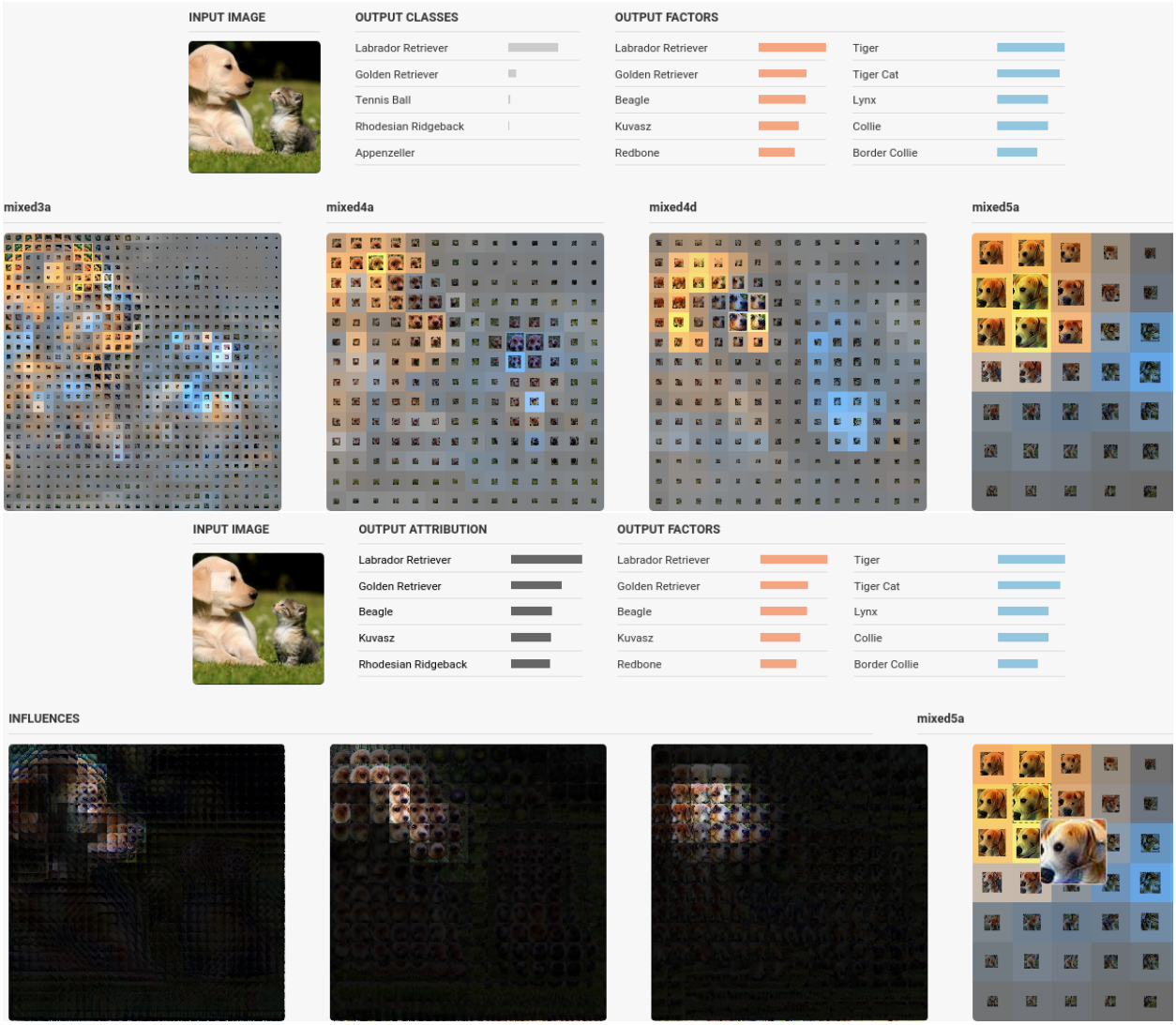


FIGURE 6.1 – **Cartes d'attributions intéractives.**

Chapitre 7

Glossaire

Glossaire

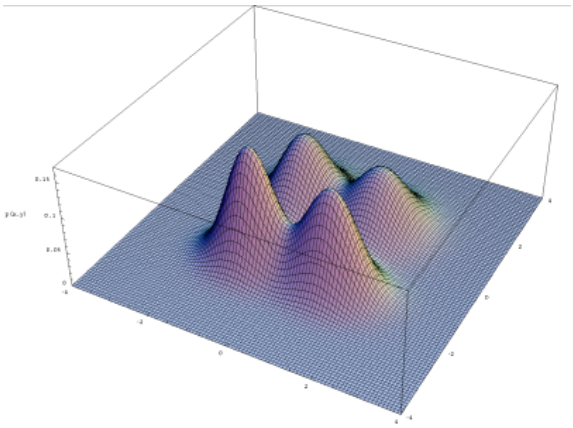
accuracy L'accuracy pourrait se traduire par précision mais ce serait une traduction non exacte. Elle correspond au pourcentage d'échantillons bien classifiés. La précision est le nombre d'échantillons positifs classifiés comme positifs divisé par le nombre total d'échantillons positifs.. 33, 64, 116, 117

batch 1) En biologie, un batch, ou lot en français, est un ensemble de plaques qui sont cultivées ensemble, dans les mêmes conditions. L'effet de batch fait référence aux différences qu'il peut y avoir entre batchs du fait de variations expérimentales non contrôlées, par exemple le taux d'humidité de l'air, des petites variations de température, des qualités cellulaires ou de réactifs qui sont différentes. Lorsqu'on clusterise les échantillons contenus sur les plaques, des conditions différentes peuvent se retrouver regroupées ensemble parce qu'elles se trouvent sur la même plaque ou dans le même batch. Pour éviter cela on peut aligner les données des plaques artificiellement en traitant les données. 2)En deep learning, l'algorithme apprend par itération, en essayant petit à petit de s'approcher d'une solution optimale qui minimise l'erreur de prédiction du réseau. On pourrait imaginer donner toutes les images à chaque itération pour que les corrections apportées aux poids du réseau fassent le plus grand consensus parmi toutes les images. Or les datasets en deep learning sont beaucoup trop gros comparés à la place disponible dans les mémoires des cartes graphiques. Pour cette raison, on procède par batch d'images : on ne donne à chaque itération qu'une petite fraction des images.. 35, 51, 54, 60, 61, 63, 77, 116, 117

clustering Le clustering est la tâche de regrouper les objets de façon à ce que les objets dans un même groupe soient plus semblables que dans des groupes distincts.. 52, 54–57, 117

epoch En deep learning, une epoch est un passage de tout le dataset d'entraînement pour l'apprentissage du ou des réseaux. Par passage, on entend réaliser la prédiction pour une image ou un batch d'images puis la rétropropagation de l'erreur pour optimiser les poids. En deep learning, le nombre de paramètres à optimiser dans les modèles impose un long apprentissage et des milliers d'epochs sont parfois nécessaires pour faire converger l'algorithme et obtenir une bonne accuracy.. 66, 77, 80, 117, 119

feature Descripteur ou caractéristique d'un objet appartenant à un dataset. Le nombre de features définit la dimension de l'espace de représentation des données.. 117



mode On appelle mode, dans une distribution d'objets, ces "bosses" où s'accumulent un grand nombre d'objets. La figure ci-dessus présente un exemple de distribution bi-variée (la distribution est en deux dimensions), et multimodale (il y a plusieurs bosses). Dans un espace à haute dimension telles des images d'un dataset, on imagine les modes comme des zones de cet espace, très densément peuplées.. 44, 107, 117

non-supervisé Le but d'un entraînement non-supervisé va plutôt être de trouver une structure dans le jeu de données que l'on va pouvoir exploiter. Par exemple, les méthodes de clustering ou regroupement, vont chercher à trouver un certain nombre de groupes ou classes parmi les données.. 56, 106, 117

overfitting Se dit pour un modèle qui s'est sur-ajusté à ses données d'apprentissage et qui généralise mal au dataset de validation (mauvaise accuracy).. 26, 30, 32, 34, 64, 117

représentation Les représentations d'objets correspondent à des vecteurs de features qui font référence de façon plus ou moins explicite à des caractéristiques de ces objets.. 117

stride Pour réaliser une convolution en 2 dimensions sur une image par exemple, on glisse un filtre ou kernel d'une certaine taille, le long des 2 dimensions spatiales de l'image. À chaque position pour laquelle le filtre s'arrête, on calcule la somme des pixels sous le filtre, pondérée par les poids des filtres. La stride, ou foulée en français, est le paramètre qui définit de combien de pixels le filtre va se décaler avant de s'arrêter. Lien vers un site avec des images animées pédagogiques.. 29, 33, 39, 49, 117

supervisé un entraînement supervisé est un entraînement à l'aide d'un jeu de données pour lesquelles on a un label ou valeur que l'on veut apprendre à prédire. On utilise alors cette vérité terrain pour l'entraînement. Voir non-supervisé pour le contraire.. 25, 117

WT Wild-Type. Se dit en biologie, de la forme naturelle, ou de référence, d'un organisme, d'un génome, d'un gène ou encore d'une protéine. Dans notre cas, le génotype WT est celui qui ne contient pas la mutation LRRK2-G2019S.. 17, 117

Liste des Figures

Liste des Tableaux

Bibliographie

1. Durand, A.-A. *Alzheimer : 900 000 malades, 2 millions d'aidants, 19 milliards d'euros de coûts… et 0 traitement* https://www.lemonde.fr/les-decodeurs/article/2016/09/21/alzheimer-900-000-malades-2-millions-d-aidants-19-milliards-d-euros-de-couts-et-0-traitement_5001325_4355770.html (page 13).
2. *Le déficit du régime de retraite est imaginaire, selon un collectif* Capital. https://www.capital.fr/votre-retraite/il-ny-a-pas-de-probleme-structurel-de-financement-des-retraites-en-2025-selon-leconomiste-henri-sterdyniak-1355703 (page 13).
3. *Maladies neurodégénératives* Santé Publique France. https://www.santepubliquefrance.fr/maladies-et-traumatismes/maladies-neurodegeneratives (page 13).
4. Inserm (pages 13, 24).
5. *L'innovation pharmaceutique à l'aune des brevets* Les Echos. https://www.lesechos.fr/2008/11/linnovation-pharmaceutique-a-laune-des-brevets-502349 (page 14).
6. Swinney, D. C. & Anthony, J. How were new medicines discovered ? *Nature Reviews Drug Discovery* **10,** 507-519. https://doi.org/10.1038/nrd3480 (juin 2011) (pages 14-16).
7. *Attrition in the Pharmaceutical Industry* (éd. Alex, A., Harris, C. J. & Smith, D. A.) https://doi.org/10.1002/9781118819586 (John Wiley & Sons, Inc, déc. 2015) (page 14).
8. Pammolli, F., Magazzini, L. & Riccaboni, M. The productivity crisis in pharmaceutical R&D. *Nature Reviews Drug Discovery* **10,** 428-438. https://doi.org/10.1038/nrd3405 (juin 2011) (page 15).
9. Poewe, W. *et al.* Parkinson disease. *Nature reviews Disease primers* **3,** 17013 (2017) (pages 15, 22, 23).
10. Moffat, J. G., Vincent, F., Lee, J. A., Eder, J. & Prunotto, M. Opportunities and challenges in phenotypic drug discovery : an industry perspective. *Nature Reviews Drug Discovery* **16,** 531-543. https://doi.org/10.1038/nrd.2017.111 (juil. 2017) (page 16).
11. Prior, M. *et al.* Back to the Future with Phenotypic Screening. *ACS Chemical Neuroscience* **5,** 503-513. https://doi.org/10.1021/cn500051h (juin 2014) (page 16).
12. Takahashi, K. *et al.* Induction of Pluripotent Stem Cells from Adult Human Fibroblasts by Defined Factors. *Cell* **131,** 861-872. https://doi.org/10.1016/j.cell.2007.11.019 (nov. 2007) (page 17).
13. Jinek, M. *et al.* A Programmable Dual-RNA-Guided DNA Endonuclease in Adaptive Bacterial Immunity. *Science* **337,** 816-821. https://doi.org/10.1126/science.1225829 (juin 2012) (page 17).

14. Nguyen, H. N. *et al.* LRRK2 Mutant iPSC-Derived DA Neurons Demonstrate Increased Susceptibility to Oxidative Stress. *Cell Stem Cell* **8,** 267-280. https://doi.org/10.1016/j.stem.2011.01.013 (mar. 2011) (page 17).

15. F, M. *et al. Frèquence de la maladie de Parkinson en France. Données nationales et régionales 2010-2015* (Santé publique France, avr. 2018). https://solidarites-sante.gouv.fr/IMG/pdf/rapport-frequence-maladie-parkinson-france.pdf (page 22).

16. *Parkinson's disease* Institut du cerveau et de la moelle épinière. https://icm-institute.org/en/parkinson-2/ (page 22).

17. Rubinsztein, D. C. The roles of intracellular protein-degradation pathways in neurodegeneration. *Nature* **443,** 780-786. https://doi.org/10.1038/nature05291 (oct. 2006) (page 23).

18. Giguère, N. & Trudeau, L.-É. La taille de l'axone est un facteur majeur influençant la dépense énergétique et la vulnérabilité des neurones dopaminergiques dans la maladie de Parkinson. *médecine/sciences* **32,** 342-344. https://doi.org/10.1051/medsci/20163204010 (avr. 2016) (page 24).

19. Deng, H., Wang, P. & Jankovic, J. The genetics of Parkinson disease. *Ageing Research Reviews* **42,** 72-85. https://doi.org/10.1016/j.arr.2017.12.007 (mar. 2018) (page 24).

20. Volpicelli-Daley, L. A. *et al.* G2019S-LRRK2 Expression Augments -Synuclein Sequestration into Inclusions in Neurons. *Journal of Neuroscience* **36,** 7415-7427. https://doi.org/10.1523/jneurosci.3642-15.2016 (juil. 2016) (page 24).

21. Bose, A. & Beal, M. F. Mitochondrial dysfunction in Parkinson's disease. *Journal of Neurochemistry* **139,** 216-231. https://doi.org/10.1111/jnc.13731 (août 2016) (page 24).

22. LeCun, Y. *et al.* Backpropagation applied to handwritten zip code recognition. *Neural computation* **1,** 541-551 (1989) (pages 26, 33).

23. Lin, M., Chen, Q. & Yan, S. *Network In Network* 2013. arXiv : 1312.4400 [cs.NE] (page 29).

24. Bridle. in () (page 29).

25. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016) (page 31).

26. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *Imagenet classification with deep convolutional neural networks* in *Advances in neural information processing systems* (2012), 1097-1105 (pages 31, 33, 63, 66).

27. Fukushima, K. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* **36,** 193-202. https://doi.org/10.1007/bf00344251 (avr. 1980) (page 30).

28. Hubel, D. H. & Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology* **195,** 215-243. https://doi.org/10.1113/jphysiol.1968.sp008455 (mar. 1968) (page 30).

29. Hien, D. H. T. *A guide to receptive field arithmetic for Convolutional Neural Networks* https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807 (page 30).

30. Araujo, A., Norris, W. & Sim, J. Computing Receptive Fields of Convolutional Neural Networks. *Distill* **4.** https://doi.org/10.23915/distill.00021 (nov. 2019) (page 30).

31. Livingstone, M. S. & Hubel, D. H. Anatomy and physiology of a color system in the primate visual cortex. *Journal of Neuroscience* **4,** 309-356 (1984) (page 30).

32. Hubel, D. H. & Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology* **160,** 106-154. https://doi.org/10.1113/jphysiol.1962.sp006837 (jan. 1962) (page 30).

33. Geirhos, R. *et al.* ImageNet-trained CNNs are biased towards texture ; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv :1811.12231* (2018) (pages 30, 66).

34. Sinha, S., Garg, A. & Larochelle, H. Curriculum By Texture. *arXiv preprint arXiv :2003.01367* (2020) (page 30).

35. Deng, J. *et al. Imagenet : A large-scale hierarchical image database* in *2009 IEEE conference on computer vision and pattern recognition* (2009), 248-255 (pages 30, 72, 84).

36. Raghu, M., Zhang, C., Kleinberg, J. & Bengio, S. *Transfusion : Understanding Transfer Learning for Medical Imaging* 2019. arXiv : 1902.07208 [cs.CV] (pages 30, 32, 63, 67, 72).

37. Koch, G., Zemel, R. & Salakhutdinov, R. *Siamese Neural Networks for One-shot Image Recognition* in (2015) (page 30).

38. Razavian, A. S., Azizpour, H., Sullivan, J. & Carlsson, S. CNN Features Off-the-Shelf : An Astounding Baseline for Recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops.* http://dx.doi.org/10.1109/CVPRW.2014.131 (2014) (pages 31, 52).

39. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. *How transferable are features in deep neural networks ?* in *Advances in neural information processing systems* (2014), 3320-3328 (page 32).

40. Raghu, M., Gilmer, J., Yosinski, J. & Sohl-Dickstein, J. *Svcca : Singular vector canonical correlation analysis for deep learning dynamics and interpretability* in *Advances in Neural Information Processing Systems* (2017), 6076-6085 (page 32).

41. Morcos, A., Raghu, M. & Bengio, S. *Insights on representational similarity in neural networks with canonical correlation* in *Advances in Neural Information Processing Systems* (2018), 5727-5736 (page 32).

42. Ciregan, D., Meier, U. & Schmidhuber, J. *Multi-column deep neural networks for image classification* in *2012 IEEE conference on computer vision and pattern recognition* (2012), 3642-3649 (page 32).

43. Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. *Nature Neuroscience* **2,** 1019-1025. https://doi.org/10.1038/14819 (nov. 1999) (page 33).

44. Glorot, X., Bordes, A. & Bengio, Y. *Deep Sparse Rectifier Neural Networks* in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (éd. Gordon, G., Dunson, D. & Dudík, M.) **15** (PMLR, 2011), 315-323. http://proceedings.mlr.press/v15/glorot11a.html (page 33).

45. Maas, A. L., Hannun, A. Y. & Ng, A. Y. *Rectifier nonlinearities improve neural network acoustic models* in () (page 33).

46. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* http://dx.doi.org/10.1109/CVPR.2016.90 (2016) (pages 33, 34).

47. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. *Densely connected convolutional networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4700-4708 (pages 34, 80).

48. Bai, K. *A Comprehensive Introduction to Different Types of Convolutions in Deep Learning* https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215 (page 34).

49. Sifre, L. & Mallat, S. Rigid-motion scattering for image classification. *Ph. D. thesis* (2014) (page 34).

50. Chollet, F. *Xception : Deep learning with depthwise separable convolutions* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 1251-1258 (page 35).

51. Howard, A. G. *et al.* Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861* (2017) (page 35).

52. Tan, M. & Le, Q. V. *EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks* 2019. arXiv : 1905.11946 [cs.LG] (page 35).

53. Ioffe, S. & Szegedy, C. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167* (2015) (page 35).

54. Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. *How does batch normalization help optimization ?* in *Advances in Neural Information Processing Systems* (2018), 2483-2493 (page 35).

55. Olah, C., Mordvintsev, A. & Schubert, L. Feature Visualization. *Distill.* https ://distill.pub/2017/feature-visualization (2017) (page 35).

56. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. *Learning deep features for discriminative localization* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 2921-2929 (pages 36, 37).

57. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks : Visualising image classification models and saliency maps. *arXiv preprint arXiv :1312.6034* (2013) (page 36).

58. Selvaraju, R. R. *et al. Grad-cam : Visual explanations from deep networks via gradient-based localization* in *Proceedings of the IEEE international conference on computer vision* (2017), 618-626 (page 36).

59. Olah, C. *et al.* The building blocks of interpretability. *Distill* **3,** e10 (2018) (pages 36, 37, 72, 113).

60. Ning, F. *et al.* Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing* **14,** 1360-1371 (2005) (page 38).

61. Kraus, O. Z., Ba, J. L. & Frey, B. J. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics* **32,** i52-i59 (2016) (page 38).

62. Godinez, W. J., Hossain, I., Lazic, S. E., Davies, J. W. & Zhang, X. A multi-scale convolutional neural network for phenotyping high-content cellular images. *Bioinformatics* **33,** 2010-2019 (2017) (pages 38, 63, 67).

63. Ljosa, V., Sokolnicki, K. L. & Carpenter, A. E. Annotated high-throughput microscopy image sets for validation. *Nature methods* **9,** 637-637 (2012) (pages 38, 54, 112).

64. Kraus, O. Z. *et al.* Automated analysis of high-content microscopy data with deep learning. *Molecular Systems Biology* **13,** 924. https://doi.org/10.15252/msb.20177551 (avr. 2017) (pages 38, 67).

65. Yang, S. J. *et al.* Applying Deep Neural Network Analysis to High-Content Image-Based Assays. *SLAS DISCOVERY : Advancing Life Sciences R&D* **24,** 829-841 (2019) (page 38).

66. Silva, T. *A Short Introduction to Generative Adversarial Networks* https://sthalles.github.io/intro-to-gans/ (page 39).

67. Goodfellow, I. J. *et al. Generative Adversarial Networks* 2014. arXiv : 1406.2661 [stat.ML] (pages 38, 39).

68. Radford, A., Metz, L. & Chintala, S. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks* 2015. arXiv : 1511.06434 [cs.LG] (pages 39, 40, 48).

69. Liu, M.-Y. & Tuzel, O. *Coupled Generative Adversarial Networks* 2016. arXiv : 1606.07536 [cs.CV] (page 39).

70. Karras, T., Aila, T., Laine, S. & Lehtinen, J. *Progressive Growing of GANs for Improved Quality, Stability, and Variation* 2017. arXiv : 1710.10196 [cs.NE] (pages 39, 41, 43).

71. Karras, T., Laine, S. & Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* http://dx.doi.org/10.1109/CVPR.2019.00453 (2019) (pages 39, 42, 113).

72. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278-2324 (1998) (pages 39, 44).

73. Wang, X., Girshick, R., Gupta, A. & He, K. *Non-local neural networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 7794-7803 (pages 40, 41, 92).

74. Zhang, H., Goodfellow, I., Metaxas, D. & Odena, A. Self-attention generative adversarial networks. *arXiv preprint arXiv :1805.08318* (2018) (pages 41, 92).

75. Miyato, T. & Koyama, M. cGANs with projection discriminator. *arXiv preprint arXiv :1802.05637* (2018) (page 41).

76. Huang, X. & Belongie, S. *Arbitrary style transfer in real-time with adaptive instance normalization* in *Proceedings of the IEEE International Conference on Computer Vision* (2017), 1501-1510 (pages 41, 42).

77. Huang, X. & Belongie, S. *Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization* in *The IEEE International Conference on Computer Vision (ICCV)* (2017) (pages 41, 92).

78. Gatys, L. A., Ecker, A. S. & Bethge, M. *Image Style Transfer Using Convolutional Neural Networks* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) (pages 41, 92).

79. Johnson, J., Alahi, A. & Fei-Fei, L. *Perceptual losses for real-time style transfer and super-resolution* in *European conference on computer vision* (2016), 694-711 (pages 41, 49, 92).

80. Brock, A., Donahue, J. & Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv :1809.11096* (2018) (pages 42, 44, 92).

81. Shmelkov, K., Schmid, C. & Alahari, K. *How good is my GAN ?* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 213-229 (page 43).

82. Salimans, T. *et al. Improved techniques for training gans* in *Advances in neural information processing systems* (2016), 2234-2242 (pages 43, 44).

83. Szegedy, C. *et al.* Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* http://dx.doi.org/10.1109/CVPR.2015.7298594 (2015) (pages 43, 84).

84. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. *Improved training of wasserstein gans* in *Advances in neural information processing systems* (2017), 5767-5777 (page 43).

85. Zhang, H. *et al. StackGAN : Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks* in *The IEEE International Conference on Computer Vision (ICCV)* (2017) (page 43).

86. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. *Gans trained by a two time-scale update rule converge to a local nash equilibrium* in *Advances in neural information processing systems* (2017), 6626-6637 (pages 43, 44, 84, 85).

87. Mescheder, L., Geiger, A. & Nowozin, S. Which training methods for GANs do actually converge ? *arXiv preprint arXiv :1801.04406* (2018) (page 44).

88. CamVid - The Cambridge-driving Labeled Video Database. http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/ (page 45).

89. Ronneberger, O., Fischer, P. & Brox, T. *U-net : Convolutional networks for biomedical image segmentation* in *International Conference on Medical image computing and computer-assisted intervention* (2015), 234-241 (pages 45-47, 76).

90. Long, J., Shelhamer, E. & Darrell, T. *Fully convolutional networks for semantic segmentation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 3431-3440 (page 45).

91. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* http://dx.doi.org/10.1109/CVPR.2017.632 (2017) (pages 45, 48, 103).

92. Li, C. & Wand, M. *Precomputed real-time texture synthesis with markovian generative adversarial networks* in *European conference on computer vision* (2016), 702-716 (pages 48, 49, 76).

93. Zhang, R., Isola, P. & Efros, A. A. *Colorful image colorization* in *European conference on computer vision* (2016), 649-666 (page 48).

94. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. *Unpaired image-to-image translation using cycle-consistent adversarial networks* in *Proceedings of the IEEE international conference on computer vision* (2017), 2223-2232 (pages 48-50).

95. Chu, C., Zhmoginov, A. & Sandler, M. Cyclegan, a master of steganography. *arXiv preprint arXiv :1712.02950* (2017) (page 49).

96. Choi, Y. *et al. Stargan : Unified generative adversarial networks for multi-domain image-to-image translation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 8789-8797 (pages 49, 50).

97. He, Z. *et al. Self-attention StarGAN for Multi-domain Image-to-Image Translation* in *International Conference on Artificial Neural Networks* (2019), 537-549 (page 51).

98. Yi, X., Walia, E. & Babyn, P. Generative adversarial network in medical imaging : A review. *Medical image analysis,* 101552 (2019) (page 51).

99. Wolterink, J. M., Leiner, T., Viergever, M. A. & Išgum, I. Generative adversarial networks for noise reduction in low-dose CT. *IEEE transactions on medical imaging* **36,** 2536-2545 (2017) (page 51).

100. Chen, Y. *et al. Efficient and accurate mri super-resolution using a generative adversarial network and 3d multi-level densely connected network* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018), 91-99 (page 51).

101. Lee, H.-C., Cherng, S. T., Miotto, R. & Dudley, J. T. Enhancing high-content imaging for studying microtubule networks at large-scale. *arXiv preprint arXiv :1910.00662* (2019) (page 51).

102. Frid-Adar, M. *et al.* GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* **321,** 321-331 (2018) (page 51).

103. Bailo, O., Ham, D. & Shin, Y. M. *Red blood cell image generation for data augmentation using Conditional Generative Adversarial Networks* 2019. arXiv : 1901.06219 [cs.CV] (page 51).

104. Qian, W. W. *et al.* Batch Equalization with a Generative Adversarial Network. *bioRxiv.* eprint : https://www.biorxiv.org/content/early/2020/02/09/2020.02.07.939215.full.pdf. https://www.biorxiv.org/content/early/2020/02/09/2020.02.07.939215 (2020) (page 51).

105. Li, B. & You, L. Predictive power of cell-to-cell variability. *Quantitative Biology* **1,** 131-139. https://doi.org/10.1007/s40484-013-0013-3 (avr. 2013) (page 51).

106. Slack, M. D., Martinez, E. D., Wu, L. F. & Altschuler, S. J. Characterizing heterogeneous cellular responses to perturbations. *Proceedings of the National Academy of Sciences* **105,** 19306-19311 (2008) (pages 52, 96).

107. Singh, S., Carpenter, A. E. & Genovesio, A. Increasing the content of high-content screening : an overview. *Journal of biomolecular screening* **19,** 640-650 (2014) (page 52).

108. Guo, X., Liu, X., Zhu, E. & Yin, J. *Deep Clustering with Convolutional Autoencoders* in (oct. 2017), 373-382. isbn : 978-3-319-70095-3 (page 53).

109. Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* **37,** 233-243. https://doi.org/10.1002/aic.690370209 (fév. 1991) (page 52).

110. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114* (2013) (page 52).

111. Weinberger, K. Q. & Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* **10,** 207-244 (2009) (page 53).

112. Schroff, F., Kalenichenko, D. & Philbin, J. *Facenet : A unified embedding for face recognition and clustering* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 815-823 (page 54).

113. Hermans, A., Beyer, L. & Leibe, B. *In Defense of the Triplet Loss for Person Re-Identification* 2017. arXiv : 1703.07737 [cs.CV] (page 54).

114. Ando, D. M., McLean, C. Y. & Berndl, M. Improving Phenotypic Measurements in High-Content Imaging Screens. *bioRxiv.* eprint : https://www.biorxiv.org/content/early/2017/07/10/161422.full.pdf. https://www.biorxiv.org/content/early/2017/07/10/161422 (2017) (pages 54, 100, 112).

115. Dumoulin, V. *et al.* Adversarially learned inference. *arXiv preprint arXiv :1606.00704* (2016) (page 54).

116. Karras, T. *et al. Analyzing and Improving the Image Quality of StyleGAN* 2019. arXiv : 1912.04958 [cs.CV] (pages 54, 113).

117. Radovanović, M., Nanopoulos, A. & Ivanović, M. Hubs in space : Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research* **11,** 2487-2531 (2010) (page 54).

118. Domingos, P. A Few Useful Things to Know about Machine Learning. *Commun. ACM* **55,** 78–87. issn : 0001-0782. https://doi.org/10.1145/2347736.2347755 (oct. 2012) (page 54).

119. McInnes, L., Healy, J. & Melville, J. Umap : Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv :1802.03426* (2018) (page 55).

120. Maaten, L. v. d. & Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **9,** 2579-2605 (2008) (page 55).

121. Hdbscan's authors. *Benchmarking Performance and Scaling of Python Clustering Algorithms* https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html (page 57).

122. Pedregosa, F. *et al.* Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825-2830 (2011) (page 56).

123. Frey, B. J. & Dueck, D. Clustering by Passing Messages Between Data Points. *Science* **315,** 972-976. https://doi.org/10.1126/science.1136800 (fév. 2007) (page 56).

124. Ester, M., Kriegel, H.-P., Sander, J., Xu, X. *et al. A density-based algorithm for discovering clusters in large spatial databases with noise.* in *Kdd* **96** (1996), 226-231 (page 56).

125. McInnes, L., Healy, J. & Astels, S. hdbscan : Hierarchical density based clustering. *The Journal of Open Source Software* **2.** https://doi.org/10.21105%2Fjoss.00205 (2017) (page 56).

126. Orlov, N. *et al.* WND-CHARM : Multi-purpose image classification using compound image transforms. *Pattern Recognition Letters* **29,** 1684-1693. https://doi.org/10.1016/j.patrec.2008.04.013 (août 2008) (page 60).

127. Goyal, P. *et al. Accurate, Large Minibatch SGD : Training ImageNet in 1 Hour* 2017. arXiv : 1706.02677 [cs.CV] (page 67).

128. Gotmare, A., Keskar, N. S., Xiong, C. & Socher, R. *A Closer Look at Deep Learning Heuristics : Learning rate restarts, Warmup and Distillation* 2018. arXiv : 1810.13243 [cs.LG] (page 67).

129. Ngiam, J. *et al.* Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv :1811.07056* (2018) (page 72).

130. Linder-Norén, E. *Keras-GAN* https://github.com/eriklindernoren/Keras-GAN (page 76).

131. Vasquez, R. J., Howell, B, Yvon, A. M., Wadsworth, P & Cassimeris, L. Nanomolar concentrations of nocodazole alter microtubule dynamic instability in vivo and in vitro. *Molecular Biology of the Cell* **8,** 973-985. https://doi.org/10.1091/mbc.8.6.973 (juin 1997) (page 79).

132. Cole, N. B., Sciaky, N, Marotta, A., Song, J. & Lippincott-Schwartz, J. Golgi dispersal during microtubule disruption : regeneration of Golgi stacks at peripheral endoplasmic reticulum exit sites. *Molecular biology of the cell* **7,** 631-650 (1996) (page 79).

133. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000) (page 81).

134. Goodfellow, I. NIPS 2016 tutorial : Generative adversarial networks. *arXiv preprint arXiv :1701.00160* (2016) (page 83).

135. Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O. & Gelly, S. *Assessing Generative Models via Precision and Recall* 2018. arXiv : 1806.00035 [stat.ML] (page 84).

136. Liu, G.-H. *et al.* Progressive degeneration of human neural stem cells caused by pathogenic LRRK2. *Nature* **491,** 603-607. https://doi.org/10.1038/nature11557 (oct. 2012) (page 91).

137. Borgs, L. *et al.* Dopaminergic neurons differentiating from LRRK2 G2019S induced pluripotent stem cells show early neuritic branching defects. *Scientific Reports* **6.** https://doi.org/10.1038/srep33377 (sept. 2016) (page 91).

138. Civiero, L., Cogo, S., Biosa, A. & Greggio, E. The role of LRRK2 in cytoskeletal dynamics. *Biochemical Society Transactions* **46,** 1653-1663. https://doi.org/10.1042/bst20180469 (nov. 2018) (page 91).

139. Nazki, H., Yoon, S., Fuentes, A. & Park, D. S. Unsupervised image translation using adversarial networks for improved plant disease recognition. *Computers and Electronics in Agriculture* **168,** 105117. https://doi.org/10.1016/j.compag.2019.105117 (jan. 2020) (page 92).

140. Borji, A. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding* **179,** 41–65. issn : 1077-3142. http://dx.doi.org/10.1016/j.cviu.2018.10.009 (2019) (page 92).

141. Liu, S., Wei, Y., Lu, J. & Zhou, J. An improved evaluation framework for generative adversarial networks. *arXiv preprint arXiv :1803.07474* (2018) (pages 93, 106).

142. Larsson, M. *et al. Fine-Grained Segmentation Networks : Self-Supervised Segmentation for Improved Long-Term Visual Localization* in *The IEEE International Conference on Computer Vision (ICCV)* (oct. 2019) (page 106).

143. Sun, B. & Saenko, K. Deep CORAL : Correlation Alignment for Deep Domain Adaptation. *Computer Vision – ECCV 2016 Workshops,* 443–450. issn : 1611-3349. http://dx.doi.org/10.1007/978-3-319-49409-8_35 (2016) (page 112).

RÉSUMÉ

Le criblage à haut contenu connaît un essor important depuis le milieu des années 2000. Cette technologie est d'un intérêt primordial pour l'industrie pharmaceutique car elle permet en principe la découverte de molécules à visée thérapeutique pour des maladies dont les voies moléculaires sont mal identifiées. Jusqu'à présent un phénotype cellulaire mesurable doit au préalable être identifié afin d'évaluer l'effet d'une banque de composés sur celui ci. Du point de vue de l'analyse d'image, les cellules traitées sont détectée automatiquement sur des centaines de milliers d'images et des mesures de descripteurs permettent de différencier finement les traitements effectifs par rapport à un contrôle négatif. En collaboration avec les entreprises Ksilink et Sanofi, nous avons été confrontés à un nouveau type de crible à haut contenu pour identifier des composés efficaces contre la maladie de Parkinson. Celui ci, présentant des images de neurones, rendait caduque l'analyse dépendant d'une segmentation robuste des cellules. Par ailleurs, grâce à des techniques de différentiation cellulaire et d'édition de génome, les contrôles du modèle cellulaire sont deux cultures de neurones, isogéniques à une mutation près : la mutation GS2019 induisant la maladie. Néanmoins, l'hétérogénéité propre à ces cellules de forme complexe, et les différences fines entre les deux lignées isogéniques ne permettent pas à l'oeil humain l'identification de deux phénotypes distincts. Dans le but de permettre la mise en évidence automatique des différences entre phénotypes, nous avons proposé d'employer des approches de deep learning. Ce travail s'est décomposé principalement en deux étapes. La première étape a consisté à identifier une architecture de réseau capable de classifier des images de neurones. Nous avons appris que les cultures de neurones montraient des différences de phénotype de manière très hétérogène et donc pas systématique. La deuxième étape a consisté à proposer des méthodes pour expliquer et interpréter les différences subtiles de phénotype, ceci afin de s'assurer que le crible soit effectué sur la base d'une différence avérée entre phénotypes et non sur la base d'un biais technique. Partant du principe que les différences entre phénotypes neuronaux sont difficiles à appréhender visuellement entre deux images différentes du à la grande variabilité naturelle des neurones, nous proposons l'idée que la transformation d'une même image d'un phénotype à un autre peut représenter une approche intéressante. En effet, nous montrons qu'il est possible d'entraîner des réseaux antagonistes à transformer une image de neurone non porteur de la mutation en neurone porteur et inversement. De cette façon, nous avons pu mettre en évidence des potentiels biais de l'assay mais également ce que nous pensons être de véritables différences morphologiques liés à la mutation pathologique auparavant invisibles.

MOTS CLÉS

Deep learning, Traitement d'images, GAN, Criblage phénotypique

ABSTRACT

High-content screening has experienced a significant growth since the mid-2000s. This technology is of primary interest to the pharmaceutical industry as it allows in principle the discovery of therapeutic molecules for diseases whose molecular pathways are poorly identified. Until now, a measurable cell phenotype must first be identified in order to evaluate the effect of a compound library on it. From an image analysis point of view, treated cells are automatically detected in hundreds of thousands of images and measurements of descriptors allow to finely differentiate effective treatments from a negative control. In collaboration with the companies Ksilink and Sanofi, we have been confronted with a new type of high-content screen to identify compounds effective against Parkinson's disease. This one, presenting images of neurons, made analysis dependent on robust cell segmentation obsolete. In addition, using cell differentiation and genome editing techniques, the controls of the cell model are two neuron cultures, isogenic but for one mutation : the GS2019 mutation that induces the disease. Nevertheless, the heterogeneity of these complex cells and the fine differences between the two isogenic lines do not allow the human eye to identify two distinct phenotypes. In order to allow the automatic detection of differences between phenotypes, we have proposed to use deep learning approaches. This work was mainly divided into two steps. The first step consisted in identifying a network architecture capable of classifying neural images. We learned that neuron cultures show phenotype differences in a very heterogeneous and therefore not systematic way. The second step consisted in proposing methods to explain and interpret subtle differences in phenotype, to ensure that the screening is performed on the basis of a proven difference between phenotypes and not on the basis of a technical bias. Based on the premise that differences between neuronal phenotypes are difficult to visually apprehend between two different images due to the high natural variability of neurons, we propose the idea that transforming the same image from one phenotype to another may represent an interesting approach. Indeed, we show that it is possible to train antagonistic networks to transform an image of a neuron that does not carry the mutation into a neuron that carries it and vice versa. In this way, we have been able to highlight potential assay biases but also what we believe to be true morphological differences related to the pathological mutation that were previously invisible.

KEYWORDS

Deep learning, Image analysis, GAN, Phenotypic screening