Some contributions to deep learning for metagenomics

Thanh Hai Nguyen

DOCTORAL THESIS
SORBONNE UNIVERSITY

Spécialité : Computer Science

École doctorale n°130: Informatics, Telecommunication and Electronic

organized

at UMMISCO, IRD, Sorbonne Université, Bondy and Integromics, Institute of Cardiometabolism and Nutrition, Paris

under the direction of Jean-Daniel ZUCKER, Nataliya SOKOLOVSKA and Edi PRIFTI

presented by

NGUYEN Thanh Hai

for obtaining the degree of:

DOCTOR SORBONNE UNIVERSITY

Thesis Title :

Some Contributions to Deep Learning for Metagenomics

Defended 26 th September, 2018

with the following juries:

Pr.	Tu-Bao HO	Reviewer
Pr.	Mohamed ELATI	Reviewer
Pr.	Yann CHEVALEYRE	Examinator
Pr.	Blaise HANCZAR	Examinator
Pr.	Jean-Pierre BRIOT	Examinator
Pr.	Jean-Daniel ZUCKER	Advisor
Dr.	Nataliya SOKOLOVSKA	Co-Advisor
Dr.	Edi PRIFTI	Co-Advisor

Version Tuesday 9th October, 2018, 15:27

Contents

Acknowledgements

First and foremost, I would like to express my deepest gratitude and appreciation to my advisors, Prof. Jean-Daniel ZUCKER, Assist. Prof. Nataliya SOKOLOVSKA, and Dr. Edi PRIFTI who have supported, guided, and encouraged me during over three years and who are great mentors in my study as well in various aspects of my personal life. I will never forget all your kindness and supportiveness. Also, I would like to especially thank Prof. Jean-Daniel who not only created my PhD candidate position, but also helped me to find the scholarship for PhD. Thank you very much for all!

I am very grateful to the reviewers and examiners in my jury, Prof. Tu-Bao HO, Prof. Mohamed ELATI, Prof. Jean-Pierre BRIOT, Prof. Yann CHEVALEYRE, and Prof. Blaise HANCZAR for their insightful comments and constructive suggestions.

In particular, I would like to thank Dr. Nguyen Truong Hai and Mrs. Nguyen Cam Thao who supported my financial for the period of high school, university, and who influenced my life choices, transmitted me the passion and brought me to computer science when I was a high school student. I would like to thank Assoc. Prof. Huynh Xuan Hiep who introduced me to the great advisors. Also, thank you Dr. Pham Thi Xuan Loc for giving me useful advice for my life in France. In addition, a big thank to Prof. Jean Hare who contributed a great thesis template to compose the thesis manuscript.

My PhD would not have begun without financial support from the 911 Vietnamese scholarship. I acknowledge the Vietnamese Government and Campus France for the quality support. In addition, thank you Can Tho University, my workplace in Vietnam, for facilitating me to complete my research.

Furthermore, I would like to thank all Integromics team members, and my friends for interesting discussions and the time spent together, thank you so much for supporting me throughout my studies in France. I would like to thank Dr. Chloé Vigliotti, Dr. Dang Quoc Viet, Nguyen Van Kha, Dr. Nguyen Hoai Tuong, Dr. Nguyen Phuong Nga, Dr. Le Thi Phuong, Dr. Ho The Nhan, Pham Ngoc Quyen, Dao Quang Minh, Pham Nguyen Hoang, and Solia Adriouch for their necessary supports for my life in France. Also, thank you Kathy Baumont, secretary at l'UMI 209 UMMISCO, for completing my administrative procedures.

Last but not least, I thank my family members, my parents, Vo Thi Ngoc Lan and Nguyen Van E. A big thank to my mother, Ngoc Lan, for motivating me to never stop trying. Thank you, my uncles, Thanh Hong, Phuong Lan, Thanh Van and my cousin, Phuong Truc for supporting the financial and providing me precious advices.

Abstract

Metagenomic data from human microbiome is a novel source of data for improving diagnosis and prognosis in human diseases. However, to do a prediction based on individual bacteria abundance is a challenge, since the number of features is much bigger than the number of samples. Therefore, we face the difficulties related to high dimensional data processing, as well as to the high complexity of heterogeneous data. Machine Learning (ML) in general, and Deep Learning (DL) in particular, has obtained great achievements on important metagenomics problems linked to OTU-clustering, binning, taxonomic assignment, comparative metagenomics, and gene prediction. ML offers powerful frameworks to integrate a vast amount of data from heterogeneous sources, to design new models, and to test multiple hypotheses and therapeutic products.

The contribution of this PhD thesis is multi-fold: 1) we introduce a feature selection framework for efficient heterogeneous biomedical signature extraction, and 2) a novel DL approach for predicting diseases using artificial image representations.

The first contribution is an efficient feature selection approach based on visualization capabilities of Self-Organising Maps (SOM) for heterogeneous data fusion. We reported that the framework is efficient on a real and heterogeneous dataset called MicrObese, containing metadata, genes of adipose tissue, and gut flora metagenomic data with a reasonable classification accuracy compared to the state-of-the-art methods.

The second approach developed in the context of this PhD project, is a method to visualize metagenomic data using a simple fill-up method, and also various state-of-the-art dimensional reduction learning approaches. The new metagenomic data representation can be considered as synthetic images, and used as a novel data set for an efficient deep learning method such as Convolutional Neural Networks. We also explore applying Local Interpretable Model-agnostic explanations (LIME), Saliency Maps and Gradient-weighted Class Activation (Grad-CAM) to identify important regions in the newly constructed artificial images which might help to explain the predictive models.

We show by our experimental results that the proposed methods either achieve the state-of-the-art predictive performance, or outperform it on public rich metagenomic benchmarks.

Résumé

Les technologies à haut débit telles que le séquençage du génome entier ont révolutionné la recherche biologique. Cet apport technologique a permis d'augmenter remarquablement la quantité de données biologiques disponibles. En effet,l'acquisition de données devient moins coûteuse, et la quantité croissante de données omiques fournit des vues générales sans précédent sur les organismes vivants et les systèmes biologiques. D'autre part, l'apprentissage par machine statistique est un domaine en plein essor, à l'intersection des mathématiques, de l'informatique et des statistiques. Depuis déjà quelques décennies, l'apprentissage automatique a été appliqué à un certain nombre de défis biologiques: la modélisation prédictive (classification), la modélisation descriptive (clustering) et la réduction de la dimensionnalité. Par exemple, Edoardo et al. dans [18] utilisent des méthodes d'apprentissage automatique sur des profils quantitatifs de microbiome (abdondances relatives) pour prédire des maladies spécifiques.

En outre, l'apprentissage automatique fournit des machines dédiées au traitement des données biologiques. Cet apport matériel permet de répondre aux problématiques classiques du traitement de ces nouvelles données biologiques. Un exemple de ces problématiques pourrait être :comment traiter des tâches où le nombre d'instances est trop petit par rapport au nombre de dimensions, ou encore comment gérer les données structurées (arbres, graphes, hyper graphes).

Ainsi, aujourd'hui, le défi consiste donc à traiter, analyser et interpréter cette grande quantité de données disponibles afin d'en obtenir des connaissances biologiques fondamentales et pratiques.

Les maladies cardiométaboliques (CMD) sont des troubles métaboliques progressifs conduisant à des stades chroniques de maladies cardiovasculaires et d'insuffisance cardiaque. Pendant longtemps, la diversité génétique du microbiome a été ignorée et le même traitement a été appliqué à tous les patients ayant un diagnostic similaire. Cependant, il n'était pas clair comment les patients individuels réagissent à ce traitement. Donc, Les progrès dans le traitement des données provenant de grandes études épidémiologiques et génomiques devrait contribuer à la résolution des épidémies cardiométaboliques mondiales. L'identification des patients réagissant aux thérapies est cruciale pour fournir le traitement le plus approprié et éviter les médicaments inutiles. L'apprentissage automatique possède des outils puissants pour intégrer une grande quantité de données provenant de sources hétérogènes, concevoir de nouveaux modèles, tester de multiples hypothèses et des produits thérapeutiques.

La première contribution de cette thèse correspond au développement de méthodes d'intégration de données (phénotypes biocliniques et données environnementales) avec des omiques personnalisées (métagénomique, métabolomique, transcriptomique) dans le but

de développer de nouvelles stratégies pour la médecine personnalisée.

La métagénomique est un domaine de recherche qui étudie de nombreux génomes provenant de divers microorganismes prélevés dans un environnement. Dans une étude de métagénomique, les données sont obtenues à partir d'un environnement, par exemple, un gramme de sol ou une partie d'un organisme vivant (par exemple, l'intestin humain). La recherche de réponses sur l'origine et la composition des données permet de déterminer l'identité, l'abondance et la fonctionnalité des gènes présents dans ces organismes [78]. Un échantillon métagénomique est traditionnellement décrit par sa composition taxonomique microbienne. Cette composition est souvent décrite à l'aide de l'abondance relative des taxons microbiens de l'une des sept catégories taxonomiques majeures: domaine, royaume, classe, ordre, famille, genre et espèce. Déterminer l'abondance relative d'une bactérie et l'aligner aux maladies de l'hôte nous permet d'avoir une idée d'un diagnostic à un stade précoce. Ce type d'études peut également fournir une compréhension plus profonde du mécanisme de la maladie [83]. Cependant, l'association de microbes individuels à un type particulier de maladie a révélé des résultats incohérents [83] en raison de problèmes différents tels que la complexité des maladies et la quantité limitée de données observées. De plus, les données biologiques en général, et la métagénomique en particulier, sont des données complexes, car les données de grande dimension sont très difficiles à interpréter par des êtres humains.

Les méthodes d'apprentissage d'ensemble telles que Random Forest donnent souvent des résultats très raisonnables sur les données métagénomiques [18], mais ces algorithmes fonctionnent toujours comme une "boîte noire". La détection de biomarqueurs de signaux associés à des facteurs de risque pour la santé et la visualisation de résultats pouvant être facilement interprétés par des experts humains sont d'une grande importance dans les domaines biologiques et médical.

Nous avons accès à une importante quantité de données expérimentales biologiques de grande dimension, et formalisons le problème du traitement des données en tant que tâche d'apprentissage supervisé. *La seconde et principale contribution de la thèse est l'introduction d'une nouvelle approche visuelle pour la métagénomique basée sur des incorporationsplongements utilisant une variété d'algorithmes différents. Ces méthodes visuelles fournissent non seulement des images 2D révélant des distributions d'unités taxonomiques opérationnelles (OTU), mais nous permettent également de tirer parti des techniques d'apprentissage en profondeur pour produire des résultats exceptionnels par rapport aux données 1D.*

Les algorithmes d'apprentissage automatique ont récemment révélé des résultats impressionnants dans divers domaines de la biologie et de la médecine. Les applications de l'apprentissage automatique en bioinformatique comprennent la prédiction des processus biologiques (par exemple, les tâches de prédiction sur la fonction génique [151], les maladies humaines [18], etc.), la prévention des maladies [150], et un traitement personnalisé [152, 52]. Au cours de la dernière décennie, l'apprentissage en profondeur a remporté un succès impressionnant sur divers problèmes tels que la reconnaissance de la parole, la classification des images et le traitement du langage naturel [5, 65, 17]. Parmi les diverses variantes méthodologiques des réseaux d'apprentissage en profondeur, les Réseaux de Neurones Convolutif (CNN) ont été largement étudiés [65], en particulier dans le domaine du traitement de l'image. Il convient de noter que les CNN sont plus performants que les humains dans certaines applications [183]. De plus, parmi une variété de réseaux

d'apprentissage en profondeur, les réseaux CNN ont le plus grand impact dans le domaine de l'informatique de la santé [149]. Cependant, en raison du manque de données d'apprentissage dans de nombreuses tâches bioinformatiques où le nombre de fonctionnalités variables est supérieur au nombre d'échantillons, il est difficile de former un CNN sans sur-apprentissage. Pour cette raison, les CNN affichent généralement des performances médiocres dans de nombreuses tâches bioinformatiques. Suite à cela,plusieurs études ont conclu que les approches d'apprentissage en profondeur peuvent ne pas convenir aux applications métagénomiques [78] . *Dans cette thèse, nous remettons en question cette conclusion, et nous montrons que l'apprentissage en profondeur est un outil efficace qui donne des résultats très raisonnables sur les données métagénomiques par rapport à l'apprentissage automatique standard. Nous proposons diverses techniques d'apprentissage en profondeur de la visualisation de données qui révèlent des résultats prometteurs sur 25 ensembles de données métagénomiques différents liés à différentes maladies.*

Sélection de caractéristiques pour des données hétérogènes à l'aide de cartes auto-organisationnelles profondes

Après la croissance rapide de la quantité de données métagénomiques et les améliorations récentes dans les unités de traitement par ordinateur, la recherche sur l'apprentissage automatique appliquée à la métagénomique a obtenu de nombreux résultats à lapointe de la technologie. Cependant, cela constitue également un défi pour le traitement de données de grande dimension et de nombreuses et diverses sources. Un cadre d'intégration de diverses sources et de sélection de caractéristiques est nécessaire dans des applications pratiques. L'intégration de données hétérogènes est une tâche potentielle et difficile avec un objectif ambitieux, celui d'augmenter la performance de l'apprentissage supervisé. En effet, diverses sources de données ont tendance à contenir différentes parties de l'information sur le problème étudié. L'apprentissage structuré et l'intégration des données permettent de mieux comprendre les propriétés et le contenu des données biologiques en général et des données "omiques" (métabolomique, métagénomique, lipidomique, etc.) en particulier. La combinaison de pièces complémentaires issues de différentes sources de données est susceptible de fournir plus de connaissances, car des types distincts de données fournissent des vues distinctes de la machinerie moléculaire des cellules. Les structures hiérarchiques et les méthodes d'intégration de données révèlent les dépendances qui existent entre les composants cellulaires et aident à comprendre la structure du réseau biologique. Les modèles graphiques suivent une organisation naturelle et une représentation des données, et constituent une méthode prometteuse de traitement simultané hétérogène des données. Les variables cachées dans un modèle hiérarchique graphique peuvent efficacement agglomérer les informations des instances observées via la réduction de la dimensionnalité, puisque moins de variables latentes sont capables de résumer plusieurs entités. Cependant, l'intégration des variables latentes est une étape cruciale de la modélisation. L'apprentissage multimodal, la fusion de données hétérogènes ou l'intégration de données impliquent la mise en relation d'informations de nature différente. Dans les applications biologiques et médicales, les données provenant d'une source sont déjà de haute dimension. Par conséquent, l'intégration de données augmente encore plus la dimensionnalité d'un problème, et une procédure de sélection de caractéristiques ou de réduction de di-

mension est absolument nécessaire à la fois pour rendre les calculs traitables et pour obtenir un modèle compact et facilement interprétable. Notre objectif est de développer une approche efficace de sélection de caractéristiques qui concevra un modèle compact. La méthode doit être évolutive, pour fusionner des données hétérogènes, et être capable d'atteindre une meilleure performance généralisante par rapport à un modèle complet et à des méthodes de pointe. Une autre question importante est de savoir si l'introduction de données de nature différente a un effet positif et fournit des connaissances supplémentaires. Un aspect important de la sélection de caractéristiques est de savoir si un modèle est facilement interprétable et s'il est possible de visualiser les résultats afin d'étudier les dépendances dans le modèle.

La carte auto-organisatrice (SOM) pour sélection de fonctionnalité

La carte auto-organisatrice (SOM) est un réseau artificiel associé au paradigme d'apprentissage non supervisé [69]. Il est célèbre pour sa manière efficace de cartographier à partir d'un espace d'entrée de haute dimension dans un espace plus compact, généralement à un espace de sortie en deux dimensions. La représentation bidimensionnelle est pratique pour une visualisation, puisque la cartographie préserve les relations topologiques entre les éléments de la grille. De plus, l'espace d'entrée continu peut être mappé dans un espace de sortie discret. Le SOM appartient à des méthodes d'apprentissage compétitives, puisque les neurones sont en compétition pour être activés, et, par conséquent, un seul est activé à la fois. Le neurone gagnant est appelé le gagnant. Lorsque le gagnant est fixé, tous les autres neurones doivent se réorganiser. Fait intéressant, la MOS peut être considérée comme une généralisation non linéaire de l'analyse en composantes principales.

Cartes profondes auto-organisées non supervisées

Dans un environnement non supervisé, la procédure de sélection des entités est complètement non supervisée et l'algorithme n'effectue que la première étape, une "forward pass". Dans cette "forward pass", nous construisons une structure profonde en couches, où chaque couche est constituée des représentants des clusters du niveau précédent. Une question naturelle qui se pose est de savoir si une telle sélection de fonctionnalités non supervisée peut être bénéfique pour une tâche de prédiction. Bien qu'il soit actuellement impossible de fournir une base théorique pour cela, il y a une intuition pour laquelle une sélection profonde de fonctionnalités non supervisées devrait être performante dans la pratique. Les données réelles sont toujours bruyantes, et une "bonne" réduction de cluster ou de dimensionnalité peut réduire considérablement le bruit. Si les fonctionnalités sont liées à des grappes de «haute qualité», il est plus facile de détecter un signal à partir des données et la performance de la classification généralisée est plus élevée. La sélection de fonction hiérarchique joue ici un rôle de filtre et un filtre à plusieurs couches semble mieux performer qu'un filtre à une couche.

Cartes profondes auto-organisées supervisées

La sélection de la fonctionnalité SOM profonde supervisée est basée principalement sur l'idée avant-arrière. Les algorithmes avancés de sélection de caractéristiques gloutonnes sont basés sur la sélection d'une fonctionnalité à chaque étape pour réduire de manière

significative une fonction de coût. L'idée est de progresser de façon agressive à chaque itération, et d'obtenir un modèle peu dense. Le problème majeur de cette heuristique est qu'une fois qu'une caractéristique a été ajoutée, elle ne peut pas être supprimée, c'est-à-dire que la "forward pass" ne peut pas corriger les erreurs faites dans les itérations précédentes. Une solution à ce problème serait une passe en arrière, qui entraînerait un modèle complet, et non pas un modèle clairsemé, et supprimerait les caractéristiques avares ayant le plus petit impact sur une fonction de coût. L'algorithme arrière est à lui seul très coûteux en calcul, puisqu'il commence par un modèle complet [43]. Nous proposons un schéma de sélection d'entités hiérarchique avec SOM. Les fonctionnalités de l'étape arrière sont dessinées au hasard.

L'intégration de données est un défi, en particulier dans les applications où les données sont de grande dimension, par exemple, la métagénomique et les espèces métagénomiques, où le nombre d'observations (patients) est faible. Nous avons proposé de réduire la dimensionnalité par une approche profonde basée sur SOM, et qui apprend de nouvelles couches de données compactes, de manière hiérarchique. Nous avons considéré des outils de sélection de fonctionnalités supervisés et non supervisés, ainsi que nous avons considéré un réel défi d'intégration de données. Nous montrons que l'approche SOM profonde considérée est efficace sur un ensemble de données médicales complexes, et il est avantageux de la combiner avec les approches lasso et élastique. La sélection de fonctionnalités non supervisées diminue la charge de calcul des méthodes standard et conduit également à des performances de pointe. Bien que la discussion biomédicale détaillée sur le regroupement des caractéristiques et la qualité des signatures obtenues soit hors de portée de cet article, et qu'elle soit faite par des biologistes effectuant des recherches pré-cliniques, nous nous attendons à ce que notre cadre puisse aider à mieux stratifier les patients, et développer des méthodes de médecine personnalisée.

Approches de visualisation pour la métagénomique

La visualisation des données métagénomiques est toujours un problème difficile en biologie computationnelle en raison de sa très grande dimension, ainsi que des interactions complexes entre les microbes. En outre, les données métagénomiques montrent également des corrélations compliquées avec des facteurs environnementaux confondants [165] (par exemple, le carbone organique total, l'azote total et le pH [166]). Comme l'illustrent de nombreuses études, la visualisation de données est considérée comme une technique indispensable pour l'analyse exploratoire des données et devient une clé pour les découvertes [153]. Une bonne visualisation devrait discriminer des groupes spécifiques pour extraire les caractéristiques de ces groupes. De plus, une méthode de visualisation idéale nous permet d'analyser efficacement de telles données à grande échelle.

Dans [153], les auteurs ont déclaré que la visualisation en métagénomique est devenue un domaine attrayant avec de nombreuses publications introduisant de nombreuses approches nouvelles chaque année, et présentant de nouvelles techniques basées sur la visualisation pour générer et vérifier de nouvelles hypothèses biologiques. Ils ont présenté un aperçu des approches existantes pour visualiser les données métagénomiques. En outre, l'étude a également souligné que la visualisation la plus connue des données de composition est un graphique à secteurs qui a la forme d'un graphique circulaire séparé en morceaux.

Chacune de ces pièces représente un groupe de données correspondantes en pourcentage. Le camembert est disponible, implémenté populairement à une variété de logiciels et de plates-formes tels que Python, R [154], Excel, et ainsi de suite. Krona [155] est l'un de ces outils populaires couramment utilisés dans la communauté de la recherche. Le logiciel présente un métagénome sous la forme d'anneaux concentriques imbriqués formant un cercle ensemble. Chaque anneau correspond à un rang taxonomique. Cette visualisation révèle une vue à plusieurs niveaux des structures des données métagénomiqus. MG-RAST [157] est un serveur en ligne qui permet l'analyse et la visualisation de données métagénomiques. , représente la métagénomique dans la hiérarchie indépendamment de l'ampleur. MEGAN est un logiciel qui nous permet d'analyser et d'explorer le contenu taxonomique de grandes données métagénomiques. Une comparaison entre 3 méthodes communes est présentée dans [155].

Un grand nombre d'outils fournis incluent *AmphoraVizu* [160], le paquet de *metricsgraphics* [162] et *gplots* dans R [161]. *Phinch* [163] est aussi un logiciel utile pour montrer la composition taxonomique des communautés microbiennes.

Une méthode standard pour représenter la structure de la communauté déduite d'un ensemle de données métagénomique est la table d'abondance [153]. Cette table contient des lignes représentant les échantillons et des colonnes correspondant aux espèces microbiennes (ou fonction de gènes).

Dans ce tableau, chaque cellule contient la valeur de l'abondance relative des taxons correspondants dans l'échantillon. Une table de cartes de chaleur ("heatmap") est une version étendue de la table d'abondance. Chaque cellule de cette table est remplie d'une couleur. L'abondance différente entre 2 cellules est identifiée par des couleurs distinctes. Le paquet R *d3heatmap* [164] fournit une variété d'options pour construire un grand nombre de types de heatmaps. De plus, *Anvi'o* est également capable de représenter une carte thermique des positions des nucléotides. Le tableau des cartes thermiques est une idée clé que nous utilisons pour l'approche de remplissage.

Trouver la structure globale de la communauté microbienne en utilisant des données métagénomiques est vraiment un défi important. En plus des diagrammes et des tableaux, les chercheurs ont aussi récemment tenté des algorithmes de réduction de dimension comme Isomap, Principe Component Analysis (PCA), t-SNE dans de nombreuses études métagénomiques [170, 167, 168, 169, 170]. Chaque échantillon caractérisé par des centaines de caractéristiques (abondance relative des espèces individuelles ou genre) est appliqué à la réduction de la dimension et présenté sous la forme d'un point (ou point, pixel) sur le nuage de points de deux (2D) ou trois (3D).

Notre approche comprend les étapes suivantes: Tout d'abord, un ensemble de couleurs est choisi et appliqué à différentes approches de binning. Le binning peut être effectué sur une échelle logarithmique ou une transformation. Ensuite, les caractéristiques sont visualisées en images par l'une des deux approches différentes, à savoir ***Fill-up*** (en utilisant un tri phylogénétique ou un ordre aléatoire) ou visualisées sur la base de méthodes d'apprentissage variées telles que le stochastique t Intégration de voisin (t-SNE) [3]. La technique t-SNE est utile pour trouver des représentations fidèles pour des points de grande dimension visualisés dans un espace plus compact. Pour le remplissage phylogénétique, les caractéristiques qui sont des espèces bactériennes sont classées en fonction de leur annotation taxonomique ordonnée alphabétiquement en concaténant les chaînes de leur taxonomie (c'est-à-dire phylum, classe, ordre, famille, genre et espèce). Cet ordon-

nancement des variables intègre dans l'image des connaissances biologiques externes, ce qui reflète la relation évolutive entre ces espèces. Toutes les expériences de la section III.4, nous utilisons Fill-up avec tri phylogénétique.

Chaque méthode de visualisation est utilisée pour représenter les données d'abondance ou de présence. La dernière représentation, qui sert de contrôle, est la 1D des données brutes (avec les espèces également triées phylogénétiquement). Pour la représentation basée sur des approches d'apprentissage multiples, nous utilisons uniquement des ensembles d'apprentissage pour générer des cartes globales, des images de formation et un ensemble de tests sont créés à partir de ces cartes globales.

"Bins" d'abondance pour les images synthétiques métagénomiques

Afin de discrétiser les abondances en tant que couleurs dans les images, nous utilisons différentes méthodes de binning. Chaque corbeille est illustrée par une couleur distincte extraite de la bande de couleurs des cartes de couleurs de la carte thermique dans la bibliothèque Python, comme *jet*, *viridis* et ainsi de suite. Dans [38], les auteurs ont déclaré que *viridis* a montré une bonne performance en termes de temps et d'erreur. La méthode de binning que nous avons utilisée dans le projet est Binning non supervisé qui n'utilise pas les informations de la classe cible. Dans cette partie, nous utilisons **EQual Width binning** (EQW) avec allant [**Min, Max**]. Nous testons avec $k = 10$ bins (pour les images distinctes en couleur, et les images grises), la largeur des intervalles est $w = 0.1$, si Min = 0 et Max = 1, par exemple.

Binning basé sur la distribution de l'abondance

Typiquement, les données métagénomiques sont très clairsemées, donc sa distribution ressemble à une distribution Zéro-Gonflée, tandis que la distribution log-transformée des données est calculée par Logarithme (base 4) qui est plus normalement distribuée. Dans l'échelle logarithmique, la largeur de chaque saut est de 1 équivalent à une augmentation de 4 fois de la précédente. D'après nos observations, nous proposons une hypothèse selon laquelle les modèles fonctionneront mieux avec de telles cassures possédant des valeurs de cassures de $0, 10^{-7}, 4 \times 10^{-7}, 1.6 \times 10^{-6}, ..., 0.0065536, 1$. La première pause est de 0 à 10^{-7} qui est la valeur minimale de l'abondance des espèces connue dans 6 jeux de données du groupe A, chacune multiplie quatre fois la précédente. Nous avons appelé ce binning "SPecies Bins" **SPB** dans nos expériences.

Binning basé sur la transformation de quantile (QTF)

Nous proposons une autre approche pour classer les données, basée sur un facteur d'échelle qui est appris dans l'ensemble d'apprentissage, puis appliqué à l'ensemble de test. Avec différentes distributions de données, la normalisation est une technique couramment utilisée pour de nombreux algorithmes ML. Quantile TransFormation (QTF), une transformation non-linéaire, est considérée comme une technique de pré-traitement forte en raison de la réduction de l'effet aberrant. Les valeurs dans les données nouvelles/invisibles (par exemple, ensemble de test/validation) qui sont inférieures ou supérieures à la plage ajustée seront définies aux limites de la distribution de sortie. Dans les expériences, nous util-

isons cette transformation pour adapter le signal des caractéristiques à une distribution uniforme.

Binary Bins

Outre les deux méthodes précédentes décrites ci-dessus, nous utilisons également les cases binaires pour indiquer PResence/absence (PR), correspondant respectivement aux valeurs noir / blanc (BW) dans les images suivantes. Notez que cette méthode est équivalente à l'encodage à un seul niveau.

Génération d'images métagénomiques artificielles

Fill-up

Les images sont créées en plaçant les valeurs d'abondance / présence dans une matrice dans un ordre de droite à gauche par rangée de haut en bas. L'image est carrée et la partie inférieure gauche de l'image est vide (blanc). L'ordre est d'organiser les espèces peuvent être soit phylogénétique ou aléatoire. A titre d'exemple pour un ensemble de données contenant 542 caractéristiques (espèces bactériennes) dans l'ensemble de données sur la cirrhose, nous avons besoin d'une matrice de 24×24 pour remplir 542 valeurs d'espèces dans ce carré. La première rangée de pixels est disposée de la première espèce à la 24ème espèce, la deuxième rangée comprend de la 25ème à la 48ème et ainsi de suite jusqu'à la fin. Nous utilisons des couleurs distinctes dans l'échelle de binning avec SPB, QTF et PR pour illustrer les valeurs d'abondance des espèces et du noir et blanc pour la présence / absence, où le blanc représente des valeurs absentes.

Visualisation basée sur des algorithmes de réduction de dimensionnalité

Outre le remplissage, nous utilisons également des algorithmes de réduction de la dimensionnalité pour visualiser les fonctionnalités. La visualisation des données est un bon moyen de voir les structures des données de sorte que la forme visualisée des données améliore probablement l'apprentissage pour obtenir une meilleure performance. Les ensembles de données de grande dimension tels que la métagénomique rencontrent généralement des difficultés à interpréter, alors que nous sommes en mesure de tracer facilement des données en deux ou trois dimensions. Une idée clé de cette approche est que nous pouvons trouver les structures de données de grande dimension façonnées en images 2D où des techniques d'apprentissage en profondeur pour les images peuvent être appliquées pour améliorer les prédictions plus précises. Outre la réduction de dimensionnalité non supervisée comme ci-dessus, nous appliquons également une version supervisée avec l'algorithme LDA (Linear Discriminant Analysis), qui permet d'ajuster la densité gaussienne à chaque classe et de supposer que toutes les classes révèlent la même matrice de covariance. Chaque groupe peut être un niveau relatif supérieur d'un groupe d'organismes dans une hiérarchie taxonomique telle que le genre, la famille, l'ordre et ainsi de suite. Afin d'appliquer des méthodes de réduction de dimensionnalité pour visualiser des données de grande dimension, les caractéristiques de tous les échantillons en formation à partir de données brutes sont visualisées à l'aide d'algorithmes de réduction de dimensionnalité tels que PCA, NMF, Random Projection, t-SNE, MDS , Isomap, LLE,

LDA et Spectral Embedding (SE) dans une carte globale. La carte est ensuite utilisée pour générer des images pour les ensembles d'entraînement et de test. Chaque espèce est considérée comme un point sur la carte et seules les espèces présentes sont représentées en abondance ou en présence en utilisant les mêmes schémas de couleurs que ci-dessus.

Comme les résultats le montrent, Fill-up surpasse les approches basées sur la réduction de la dimensionnalité car tous les points utilisant Fill-up sont visibles tandis que les autres techniques subissent le problème de chevauchement des points. Deuxièmement, l'approche Fill-up intègre les connaissances antérieures sur la classification phylogénétique des caractéristiques. De plus, les images basées sur l'apprentissage des variétés sont plus complexes que les images de remplissage. Cependant, les résultats encourageants des méthodes basées sur la réduction de la dimensionnalité du T2D et OBE qui sont supérieurs au Fill-up montrent une puissance potentielle de ces approches.

Nous avons également présenté cinq méthodes de binning qui sont menées à partir de trois approches: la distribution de l'abondance, la transformation et le "one-hot encoding". L'approche des "bins" et des images est construite et produite à l'aide de données d'apprentissage seulement, évitant ainsi des problèmes trop complexes. Pour les méthodes basées sur la distribution de l'abondance, nous comparons deux méthodes, y compris l'utilisation de l'abondance originale (EQW) et l'utilisation du logarithme (SPB). Les algorithmes de transformation sont également étudiés avec des approches linéaires (MMS) et non-linéaires (QTF). Comme démontré dans les résultats, QTF et SPB sont des méthodes de binning prometteuses pour les données métagénomiques.

Une gamme variée de cartes de couleurs est étudiée avec des performances prometteuses. *Jet, rainbow, viridis* sont de bons choix, mais CNN sous-exécute souvent FC pour les images couleur, tandis que *grays* donne généralement de meilleures performances dans CNN.

Apprentissage profond pour la métagénomique

L'acquisition de données à haut débit dans le domaine biomédical a révolutionné la recherche et les applications en médecine et en biotechnologie. Aussi connues sous le nom de données "omiques", elles reflètent différents aspects de la biologie des systèmes (génomique, transcriptomique, métabolomique, protéomique, etc.) mais aussi des écosystèmes biologiques entiers acquis par la métagénomique. Il y a un nombre croissant d'ensembles de données qui sont accessibles au public. Différentes méthodes statistiques ont été appliquées pour classer les patients de contrôles [57] et certains ont également effectué des méta-analyses sur plusieurs ensembles de données [18]. Cependant, l'exploration des données omiques est difficile, car le nombre de caractéristiques d est très important et le nombre d'observations N est faible. Jusqu'à présent, les techniques les plus efficaces appliquées aux ensembles de données omiques ont été principalement la forêt aléatoire (RF) et la régression clairsemée.

Nous avons évalué toutes les représentations proposées sur six jeux de données métagénomiques dans le groupe A, qui reflètent l'abondance des espèces bactériennes et la présence dans l'intestin des patients malades et des témoins sains. Ensuite, nous appliquons notre méthode à d'autres ensembles de données sur le cancer colorectal (groupe B), des ensembles de données supplémentaires avec l'abondance de *genre* (groupe C) et

des ensembles de données sur l'abondance des familles de gènes (groupe D). Puisque DL fonctionne particulièrement bien pour la classification d'images, nous nous sommes concentrés dans ce travail sur l'utilisation des CNN appliqués aux images.

Les benchmarks de données métagenomiques

Nous avons évalué notre méthode sur 25 ensembles de données différents divisés en **quatre** groupes (A, B, C, D et E).

Le groupe A correspond à des ensembles de données comprenant des espèces bactériennes liées à diverses maladies, notamment: cirrhose du foie (CIR), cancer colorectal (COL), obésité (OBE), maladie intestinale inflammatoire (IBD) et diabète de type 2 (T2D) [18, 19, 20, 36, 2, 10, 32, 141, 142], avec CIR (n = 232 échantillons avec 118 patients), COL (n = 48 patients et n = 73 individus en bonne santé), OBE (n = 89 non obèses et n = 164 obèses), IBD (n = 110 échantillons dont 25 étaient atteints de la maladie) et T2D (n = 344 individus dont n = 170 sont des patients T2D). En outre, un ensemble de données, à savoir WT2, qui comprend 96 femmes européennes avec n = 53 patients DT2 et n = 43 individus en bonne santé est également considéré. Les ensembles de données d'abondance sont transformés pour obtenir une autre représentation basée sur la présence de caractéristiques lorsque l'abondance est supérieure à zéro (0). Ces données ont été obtenues en utilisant les paramètres par défaut de MetaPhlAn2 [30] comme détaillé dans Pasolli et al. [18].

Le groupe B contient 526 échantillons métagénomiques de la maladie du cancer colorectal (COL) provenant de cohortes chinoise, autrichienne, américaine, allemande et française, respectivement C1, C2, C3, C4. Ces cohortes ont été analysées dans l'article cité [37]. Un ensemble de données supplémentaire (C5) a été créé en fusionnant C1, C2, C3 et C4. Ces ensembles de données comprennent des séquences de séquençage métagénomique (utilisant la plateforme de séquençage Illumina Hiseq 2000/2500271 avec des profondeurs de séquençage similaires (longueur de lecture 100 pb et profondeur de séquençage cible 5 Go)) avec 271 contrôles et 255 cas de COL. Les séquences de faible qualité ont été éliminées en utilisant Trimmomatic v _0.36.

Le groupe C comprend les données de laboratoire de Sokol [8] consistant en des informations sur le microbiome de 38 sujets sains (HS) et de 222 patients atteints de MICI. L'abondance comprend 306 UTO avec une abondance de genre. Les patients dans ces données sont classés en deux catégories selon le phénotype de la maladie colite ulcéreuse (UC) et la maladie de Crohn (CD). Chaque catégorie est divisée en deux conditions (flare (f), si les symptômes des patients s'aggravent ou réapparaissent et la condition remission(r), si les symptômes des patients diminuent ou disparaissent). L'ensemble de données a été divisé en sous-ensemble avec la maladie de Crohn iliaque (iCD) et la maladie de Crohn du côlon (cCD). La description détaillée des données a été présentée dans [11].

Le groupe D a les mêmes échantillons de CIR [10], COL [32], IBD [141], OBE [19], T2D [142], WT2D [2] que le groupe A, mais les données incluent l'abondance des familles de gènes générées par le réseau d'analyses métaboliques unifiées HMP (HUMAnN2) [140] avec une très grande dimension allant jusqu'à plus d'un million de caractéristiques. Les données sont téléchargées depuis le paquet ***curatedMetagenomicData*** dans R.

Pour chaque échantillon, l'abondance des espèces / genres / gènes est une proportion relative et est représentée par un nombre réel - l'abondance totale de toutes les espèces /

genres / sommes de gènes à 1.

Nous examinons également notre approche sur deux ensembles de données (groupe E) en analyse autonome (Crohn et HIV) [194] avec le nombre de comptages de taxa microbiens au niveau du genre. Le HIV contient 155 échantillons, tandis que Crohn comprend 662 patients atteints de la maladie de Crohn et 313 témoins. L'ensemble de données sur le HIV comprend 62 éléments, le dernier indiquant un facteur de risque du HIV, MSM: "Hommes ayant des rapports sexuels avec des hommes". Nous utilisons un codage à chaud avec 1 si MSM est vrai et 0 si MSM est faux.

Architectures de réseau neuronal convolutif et modèles utilisés dans les expériences

Cas unidimensionnel

Afin de prédire la maladie en utilisant les données 1D, nous utilisons un réseau neuronal (FC) entièrement connecté et un réseau neuronal convolutif 1D (CNN1D). Le modèle FC comprend une couche entièrement connectée et donne une sortie. C'est un modèle très simple mais la performance est relativement élevée. La structure de ce réseau contient une couche entièrement connectée avec une fonction sigmoïde. CNN1D inclut une couche convolutionnelle 1D avec 64 filtres et un pool maximum de 2. Nous adaptons les données aux algorithmes d'apprentissage classiques tels que RF [118] (avec 50 arbres) et SVM [119] (noyaux de Sigmoid, Radial , Linear) pour les données 1D.

Cas bidimensionnel

Les images dans Fill-up varient de 16×16 à 24×24 (pour les ensembles de données dans le groupe A) en fonction du nombre de fonctionnalités tandis que, dans t-SNE, nous utilisons des images 24×24 pour tous les datasets.

L'architecture est conduite à partir des résultats étudiés sur une variété d'architectures de CNN appliquées aux représentations basées sur Fill-up, à l'aide de SPB, d'images grises. Nous comparons également les réseaux convolutifs de type VGG (convnet) proposés dans Keras (une API de réseaux neuronaux de haut niveau, écrite en Python [109]) document https://keras.io/getting-started/sequential-model-guide/ avec une petite modification comme base. Les résultats montrent que les performances augmentent en fonction de la largeur des CNN. Cependant, la performance diminue lorsque nous ajoutons plus de couches en raison d'un sur-ajustement. CNN avec une couche convolutionnelle et un grand nombre de filtres surpassent FC pour CIR, COL, IBD tandis que WT2, OBE semblent rencontrer remplissages avec CNN. Dans la plupart des cas, les architectures de CNN avec une couche convolutive obtiennent de meilleures performances que les architectures à deux convolutions et à VGG. Cependant, pour WT2, les architectures à deux couches semblent apporter une légère amélioration. Pour les CNN peu profonds, l'abandon dans FC semble ne pas améliorer la performance mais il fonctionne sur des réseaux profonds tels que VGG. VGG avec un taux de décrochage de 0,2 révèle la meilleure performance parmi les variations de VGG dans nos expériences. Dans certains cas, la combinaison entre l'abandon dans les couches Convolutional et les couches FC améliore les performances par rapport à l'application aux seules couches FC (dans CNN-l132, CNN-l2f16). Pour CIR, CNN avec une couche convolutionnelle surclassent FC. CIR et IBD atteignent le

pic à CNN-l1f64 tandis que OBE et T2D effectuent le meilleur à CNN1f16 et CNN1l20. D'un autre côté, WT2 est confronté à un problème de sur-adaptation sans meilleurs résultats dans les CNN par rapport à FC. L'utilisation de l'abandon n'est apparemment pas efficace.

Comme les résultats le montrent, une architecture simple telle qu'une couche à convolution unique présente de meilleures performances que des architectures plus complexes et plus profondes telles que VGG.

Lorsque les résultats sont exposés, l'approche Fill-up surpasse les méthodes de visualisation basées sur l'apprentissage de la réduction de la dimensionnalité. Cela peut être dû à plusieurs facteurs. Tout d'abord, les entités du remplissage sont toutes visibles alors que les fonctions des autres méthodes se chevauchent souvent. Deuxièmement, l'approche Fill-up intègre les connaissances antérieures sur la classification phylogénétique des caractéristiques. De plus, les images basées sur l'apprentissage des variétés sont plus complexes que les images de remplissage. Il est à noter qu'avec le Fill-up, nous montrons des améliorations significatives de quatre ensembles de données, tandis que le t-SNE révèle une amélioration significative sur un ensemble de données, le T2D. Le modèle FC surpasse le modèle CNN en images couleur tandis que le modèle CNN obtient de meilleures performances que le FC pour les images en gris et en noir et blanc. En outre, les représentations basées sur des images 2D donnent de meilleurs résultats par rapport aux données 1D. En général, la méthode Met2Img proposée surpasse l'état de l'art à la fois sur les espèces et les données sur l'abondance du genre.

Notre approche est également évaluée sur un vaste ensemble de données différentes, y compris l'abondance des espèces, l'abondance du genre et l'abondance des familles de gènes. Ces ensembles de données sont caractérisés par différentes dimensions allant de mille à des millions de fonctionnalités. Les résultats montrent que l'augmentation du nombre d'attributs considérés est également susceptible d'améliorer la performance par rapport aux méthodes classiques d'apprentissage automatique.

Le temps de calcul des GPU pour les CNN est considérablement amélioré par rapport au CPU, mais il n'y a pas de différence considérable dans le modèle FC. Cela indique que le GPU ne semble fonctionner efficacement qu'avec des modèles complexes.

Actuellement, nous étudions différentes architectures d'apprentissage en profondeur, et explorons également l'intégration d'autres données omiques hétérogènes.

Principales contributions

La contribution principale de cette thèse est un framework appelé **Met2Img** qui fournit une nouvelle approche pour la visualisation des données métagénomiques, et l'architecture correspondante des CNN appliqués aux visualisations (Chapter ref chap: deeplearning). Nous avons également présenté un cadre de sélection de caractéristiques pour intégrer des sources hétérogènes de grande dimension basées sur les capacités de visualisation puissantes des cartes auto-organisatrices (SOM) et des machines vectorielles de support (SVM).

Nous avons proposé un cadre pour réduire la dimensionnalité par une approche profonde basée sur la SOM associée à un SVM. La structure profonde vise à visualiser les clusters en 2D sur la base de la capacité robuste de SOM. La méthode permet d'obtenir

des résultats raisonnables par rapport à la performance de l'état de l'art. Le cadre devrait aider à mieux stratifier les patients et à développer des approches de la médecine personnalisée.

Nous avons également présenté un cadre Met2Img qui aide à visualiser les caractéristiques en tant qu'images synthétiques et qui permet les puissantes capacités d'apprentissage en profondeur dans la classification des images. Nous avons exploré une variété de méthodes de visualisation, y compris les approches basées sur le remplissage et l'apprentissage multiple, en utilisant divers types de classes pour réduire les effets des erreurs d'observation mineures afin d'éliminer le bruit dans les données.

Comme résultats sur les ensembles de données du groupe A comprenant 6 ensembles de données d'abondance d'espèces liées à cinq maladies avec le nombre de caractéristiques allant de 381 à 572, Fill-up réalise des améliorations significatives sur 4 ensembles de données sur 6 tels que CIR, IBD, OBE, WT2 basé sur des méthodes d'apprentissage de réduction de la dimensionnalité telles que t-SNE, et NMF surpasser sur l'ensemble de données T2D par rapport à l'état de l'art [18]. Bien que nous n'obtenions pas de résultats significatifs sur les données COL du groupe A par rapport aux RF, les performances sur les jeux de données sur le cancer colorectal dans le groupe B sont améliorées avec 4 ensembles de données sur 5 obtenant des résultats significatifs comparés aux RF. des OTU considérées est proche de 2000). Remarquables, les "bins" SBP montrent des améliorations significatives des modèles FC et CNN par rapport à l'état de l'art qui n'utilisait que sept espèces enrichies trouvées dans [37]. Si l'on compare avec un autre Ph-CNN de pointe, sur l'abondance au niveau du genre (avec un plus petit nombre de caractéristiques allant de 237 à 257 caractéristiques), notre cadre obtient également des résultats encourageants avec des résultats significatifs. De plus, le framework fonctionne bien sur un ensemble de validation externe. En comparant les résultats de l'abondance des familles de gènes (groupe D) avec le nombre de caractéristiques allant jusqu'à plus d'un million, nous obtenons également des performances supérieures à celles des RF, même si seules de très petites images sont réalisées dans cette analyse. Les résultats sur l'abondance des familles de gènes montrent également des améliorations considérables pour le modèle FC en termes de temps d'exécution et de précision. Les modèles d'information de commande phylogénétique intégrée de Fill-up surpassent le tri aléatoire sur les ensembles de données COL, IBD. Cela montre que les informations phylogénétiques intégrées dans les images peuvent améliorer les performances. En outre, le tri des espèces qui ont une relation sur l'ordre phylogénétique côte à côte peut être une information utile pour la prédiction.

Nous avons proposé trois types de classes efficaces, à savoir **SBP**, **QTF** et **PR**. SBP est plus efficace que les autres pour l'abondance des espèces dans la plupart des situations, mais QTF aide à améliorer remarquablement la performance de l'Isomap et présente des résultats substantiels pour l'abondance du genre. Pour les classes QTF, pour chaque pli, les données sont transformées en une distribution uniforme, puis les images sont créées en fonction des données transformées. Par conséquent, QTF a tendance à consommer plus de temps dans l'exécution par rapport à SPB où l'ensemble des images créées une fois au début. Bien que le PR se révèle être le pire, sa performance atteint aussi l'art de l'art. Remarquablement, PR surperforme SPB lorsque nous employons des images très fortement compressées pour l'abondance des familles de gènes.

Une variété de dix cartes de couleurs quantitatives est explorée et évaluée dans cette thèse. Pour les données en forme de distribution uniforme, viridis fonctionne bien dans le

terme de précision tandis que le jet, arc-en-ciel semble approprié pour des distributions en forme de cloche. En outre, l'échelle de gris est un choix approprié, il est à noter que les images grises présentent des résultats substantiels à la fois FC et CNN. *Grays* donne aussi souvent de meilleures performances dans CNN comparé à FC. Cependant, *grays* a besoin de plus d'époques pour converger contrairement aux images couleur.

Explorations futures

Nos études préliminaires sur la visualisation des données sur le microbiome pourraient mener à plusieurs directions pour les travaux futurs. Le cadre proposé explore l'utilisation potentielle des données métagénomiques. Cependant, bien que le Met2Img ne soit étudié que pour la métagénomique, la méthode peut être appliquée directement à toute autre donnée " omics ", afin d'effectuer une analyse basée sur la fonction plus avancée.

La quantité croissante de données provenant de sources multiples entraîne davantage d'exigences pour l'intégration de données hétérogènes. La prédiction de diverses maladies peut être plus précise si les classificateurs combinent ces données hétérogènes plus efficacement. Une combinaison de données métagénomiques et d'autres données telles que les facteurs diététiques, environnementaux, etc. permet une meilleure précision diagnostique. De plus, l'apprentissage en profondeur en général, et les réseaux de neurones convolutionnels en particulier, est un domaine de recherche actif où des résultats de performance excitants sont régulièrement rapportés. De plus en plus d'architectures de CNN sont proposées pour améliorer les performances et performer mieux que les humains dans de nombreuses applications. Si la performance des méthodes d'apprentissage en profondeur appliquées au traitement de l'image va encore s'améliorer, les idées pour présenter les données comme des images pour une analyse efficace ont un bon terrain pour devenir réalité.

De nos jours, la vitesse de traitement des données métagénomiques (abondance des familles de gènes) est plutôt lente, et les progrès récents dans les algorithmes distribués et les calculs parallèles sont grandement nécessaires pour réduire le temps d'exécution. Actuellement, la génération d'images synthétiques prend beaucoup de temps et de mémoire pour les données de grande dimension telles que les matrices d'abondance de gènes, en particulier pour le QTF. Dans [37], les auteurs ont calculé l'importance des changements d'abondance bactérienne disponibles dans les ensembles de données sur le cancer colorectal pour extraire sept marqueurs bactériens pour le diagnostic du cancer colorectal à un stade précoce. Cette méthode d'extraction de caractéristiques qui identifie sept espèces bactériennes révélant une abondance différentielle dans le CRC par rapport aux témoins dans l'ensemble des quatre cohortes, est un grand potentiel pour réduire la dimensionnalité des données d'abondance des familles de gènes conduisant à des images de plus petite résolution. la procédure de formation sera beaucoup moins complexe. De tels résultats nous incitent à approfondir la visualisation des fonctionnalités significatives dans les environnements où le nombre de fonctionnalités est extrêmement élevé. Cependant, ces méthodes peuvent demander beaucoup plus de temps pour le calcul de la signification statistique des UTO. Par conséquent, il est nécessaire de trouver un compromis entre le temps d'inférence et la précision.

Les CNN avec des architectures assez profondes font toujours face à un sur-ajustement

alors que l'augmentation de la largeur de l'architecture peut améliorer les performances. Dans cette thèse, nous avons seulement considéré les petites images allant de 16×16 à 48×48, donc les exigences sur les CNN ne sont pas très complexes. Cependant, des données plus importantes telles que l'abondance des gènes avec le nombre de caractéristiques allant jusqu'à des millions, des architectures plus profondes devraient être étudiées avec précision. L'optimisation des hyper-paramètres pour les réseaux CNN constitue également une limitation et consomme beaucoup de temps. Notre étude n'a pas évalué l'efficacité de l'application du réseau pré-formé dans des images naturelles telles que VGG16, VGG19, ResNet50, etc. Une des raisons est que la plupart des analyses de cette thèse se concentrent sur de petites images. des images telles que 224×224. En outre, il existe certaines limites des ressources de calcul, de sorte que de tels réseaux lourds n'ont pas encore été complètement examinés. Par conséquent, ce problème devrait explorer des stratégies plus avancées pour améliorer encore la performance de la classification.

Le t-SNE est un outil prometteur pour la visualisation. Cependant, il souffre de ne pas être généralisable. Un problème important pour le t-SNE est de savoir comment gérer de nouvelles données. Lorsque nous avons effectué t-SNE sur l'ensemble de formation, nous devons également réexécuter l'algorithme entier sur l'ensemble de données ajouté de nouvelles données. Certaines recherches visant à résoudre ce problème sont apparues mais l'implémentation n'est pas terminée en Python. Un autre problème pour t-SNE est que le t-SNE consomme une énorme quantité de mémoire et de temps d'exécution pour l'abondance des gènes. Cela conduit au fait que d'autres études devraient être étudiées pour améliorer l'algorithme en termes de consommation de mémoire et de temps d'inférence.

Chapter I

Introduction

Contents

I.1 Motivation

High-throughput technologies such as whole-genome sequencing revolutionized biological research and transformed it from a relatively data-poor discipline into a domain which is rich in data. Today the challenge is to process, analyze, and to interpret the available data, and to derive fundamental and practical biological knowledge. The data acquisition becomes cheaper, and the increasing amount of omics data provides with unprecedented broad views of living organisms and biological systems. Statistical machine learning is a relatively young but actively developing field on the intersection of mathematics, computer science, and statistics. For already a couple of decades machine learning has been applied to a number of biological challenging: predictive modeling (classification), descriptive modeling (clustering), and dimensionality reduction. Besides, machine learning provides machineries for data processing, typical for biological applications: how to deal with tasks where the number of instances is too small but the dimensionality it too high, and how to cope with structured (sequences, trees, graphs, hyper graphs) data.

The Cardiometabolic diseases (CMD) are progressive metabolic disorders leading to chronic stages of cardiovascular diseases and heart insufficiency. For a long time the genetic diversity has been ignored and the same treatment has been applied to all patients with a similar diagnosis. However, it was not clear how individual patients respond to it. Advances in data processing from large epidemiological and genome-wide studies are expected to contribute to the resolution of the worldwide Cardiometabolic epidemics. The identification of responders to therapies is crucial to provide the most appropriate treatment and avoid unnecessary medications. Machine learning possesses powerful

frameworks to integrate a vast amount of data from heterogeneous sources, design new models, and test multiple hypotheses and therapeutic products.

After the rapid growth in the amount of the metagenomic data and the recent improvements in computer processor units, the research on machine learning applying to metagenomics has achieved numerous state-of-the-art results. However, this is also a challenge for processing high-dimensional data and from numerous various sources. A framework for integrating a variety of sources and selecting features is needed in practical applications. Heterogeneous data integration is a potential and challenging task with an ambitious goal to increase performance of supervised learning, since various sources of data tend to contain various parts of information about the problem. Structure learning and data integration allow to better understand the properties and content of biological data in general and of "omics" data (metabolomics, metagenomics, lipidomics, etc.) in particular. Combining complementary pieces issued from different data sources is likely to provide more knowledge, since distinct types of data provide distinct views of the molecular machinery of cells. Medical and biological knowledge can be naturally organized into hierarchies: symptoms of diseases are observed and pathological states on all levels of omics data are hidden. Hierarchical structures and data integration methods reveal dependencies that exist between cellular components and help to understand the biological network structure. Graphical models follow a natural organization and representation of data, and are a promising method of simultaneous heterogeneous data processing. Hidden variables in a graphical hierarchical model can efficiently agglomerate information of observed instances via dimensionality reduction, since fewer latent variables are able to summarize multiple features. However, integration of latent variables is a crucial step of modeling. Multi-modal learning, heterogeneous data fusion, or data integration, involves relating information of different nature. In biological and medical applications, data coming from one source are already high-dimensional. Hence, data integration increases the dimensionality of a problem even more, and some feature selection or dimensionality reduction procedure is absolutely needed both to make the computations tractable and to obtain a model which is compact and easily interpretable.

As the first contribution of this thesis, we develop methods of data integration of bioclinical and environmental phenotype together with personalized omics (metagenomics, metabolomics, transcriptomics) with the objective of developing new strategies for personalized medicine.

Metagenomics is a research field that focuses on numerous genomes from various microorganisms collected from an environment. In a metagenomic study, data is obtained from an environment, e.g., a gram of soil or a part from a living organism (for example, human gut). Finding the answers on the origin and composition of the data allows to determine the identity, abundance, and the functionality of the organisms [78, 15]. A metagenomic sample is traditionally described by its microbial taxonomic composition that can be a relative abundance of microbial taxa of one of major seven taxonomic categories including domain, kingdom, phylum, class, order, family, genus, and species. Determining relative abundance of a bacteria, and linking it to host diseases allows us to have an idea of a diagnosis at its early stage. It can also provide a deeper understanding of the disease mechanism [83]. However, association of individual microbes of a particular type of a disease has revealed inconsistent results [83] due to different problems such as the complexity of diseases, and the limited amount of observed data. Furthermore, biological

data in general, and metagenomics in particular, are complex data, since high-dimensional data are very hard to interpret by human beings.

Ensemble learning methods such as Random Forest often yield very reasonable performance on metagenomic data [18], but these algorithms still work as a "black-box". Detecting signal biomarkers associated with health risk factors, and visualization of results which can be easily interpreted by human experts, is of great importance in biological and medial domains. The visualization of metagenomic data is still a challenging issue in computational biology due to its very large dimensionality, as well as complex interactions among microbes. Metagenomic data also shows complicated correlations with confounding environmental factors [165] (for example, total organic carbon, total nitrogen and pH[166]). As illustrated in numerous studies, data visualization is considered as an indispensable technique for the exploratory data analysis and becomes a key for discoveries [153]. A good visualization should discriminate between specific groups to extract characteristics of these groups. Furthermore, an ideal visualization method enables us to analyze such large-scale data efficiently.

We have access to vast amounts of biological high-dimensional experimental data, and we formalize the problem of data processing as a supervised learning task. *The second, and the main contribution of the thesis is the introduction of a novel visual approach for metagenomics based on embeddings using a variety of different algorithms. These visualization methods not only produce 2-dimensional images revealing Operational Taxonomic Units (OTUs) distributions, but also enable us to apply deep learning techniques to reach promising classification results.*

Machine learning algorithms have recently revealed impressive results across a variety of biology and medicine domains. The applications of machine learning in bioinformatics include predicting of biological processes (for example, prediction tasks on gene function [151], human diseases [18, 42, 63, 72, 143]), prevention of diseases [150, 64], and personalized treatment [152, 52]. In the last decade, deep learning has gained an impressive success on a variety of problems such as speech recognition, image classification, and natural language processing [5, 65, 17]. Among various methodological variants of deep learning networks, the Convolutional Neural Networks (CNN) have been extensively studied [65], especially in the field of image processing. Noteworthy, CNN are able to perform better than humans in some applications [183]. Moreover, among a variety of deep learning networks, CNN have the greatest impact for the field of health informatics [149]. However, due to the lack of training data in many bioinformatics tasks where the number of features is bigger than the number of samples, it is difficult to train a CNN without overfitting. For this reason, CNN usually show poor performance in many bioinformatics tasks. This had led to several studies, see, e.g.,[78] where it was reported that "the deep learning approaches may not be suitable for metagenomic applications". *In this thesis, we challenge this important question, and we show that deep learning is an efficient tool achieving very reasonable results on metagenomic data compared to standard machine learning methods. We consider various deep learning techniques that reveal promising results on 25 different metagenomic datasets related to different diseases.*

I.2 Brief Overview of Results

The thesis includes three main contributions, and is organized as follows. In Chapter II, we introduce a feature selection framework for heterogeneous data integration. Our contributions in deep learning applied to metagenomics are presented in Chapter III with the visualization approaches. The architectures for the deep learning are discussed in Chapter IV. Concluding remarks and perspectives are provided in Chapter V.

I.2.1 Chapter II: Heterogeneous Biomedical Signatures Extraction based on Self-Organising Maps

This work aims to find a feature selection framework for heterogeneous high dimensional sources. Feature selection is needed in numerous applications, where information comes from a variety of heterogeneous high-dimensional data sources. Different sources of data include various parts of information on the problem, so integrating heterogeneous data enables probably to improve the performance of supervised learning algorithms.

In this chapter, we propose a framework based on powerful visualization capabilities of self-organizing- maps (SOM) which together with the Support Vector Machines (SVM) allow to visualize biomedical signatures as well as to reach a reasonable predictive performance compared to the-state-of-the-art. The SOM calculate the similarity of patterns to classify the units [184]. They use competitive learning as opposed to error-correction learning of other artificial neural network [185] and apply a neighborhood function to retain the topological properties of input. We investigate a simple deep feature selection framework which constructs layers of a deep structure layer-wise based on SOM (can be learned either in a supervised or unsupervised mode) to figure out clusters in 2D. The framework is evaluated on real data including meta-data, alimentary patterns of patients, gene expressions of adipose tissue, and gene abundance of gut flora. As we show below, the performance based on the features extracted by the feature selection framework is similar to the one with all available features.

This work has led to a publication in the International Joint Conference on Neural Networks (IJCNN).

- N. Sokolovska, H. T. Nguyen, K. Clément, J.-D. Zucker. *Deep Self-Organizing Maps for Efficient Heterogeneous Biomedical Signatures Extraction.* International Joint Conference on Neural Networks (IJCNN), 2016, pages 5079-5086, IEEE, Vancouver, Canada.

I.2.2 Chapter III: Visualization approaches for metagenomics

Chapter III discusses visualization methods for metagenomic data using Fill-up and supervised and unsupervised dimensionality reduction methods including such algorithms as the t-SNE, LDA, Isomap, and some other methods. The Fill-up method relies on filling a square matrix whose size depends on the number of features with abundance/presence values for each sample. The image is square where the size is a rounded square root of the number of features. For example, we use images of 24×24 for the data with 542 features, since the ceiling of the square root of 542 is 24. For the visualizations based on dimensionality reduction algorithms, a global map includes coordinates of features determined

by dimensionality reduction algorithms learned from a training set. It is expected that the features which are similar in the magnitude of abundance will be close. Based on this assumption, we build images for all samples from both training and testing sets.

In addition, we explore a number of discretization or binning methods as well a variety of colormaps to build synthetic images from metagenomic abundance data. As observed from numerical results on six public datasets, we propose *SPecies Bin* (SPB) based on species abundance distribution. Furthermore, we introduce a new binning approach based on *Quantile transformation* which is called QTF. We also deploy PResence/absence (PR) binning which only indicates whether a feature is present or not in a sample.

Furthermore, we investigate ten quantitative colormaps for images. Our results show that *viridis* and *grays* can provide a sufficient discrimination for classification on large scale data.

I.2.3 Chapter IV: Deep learning for metagenomics using embeddings

Deep learning methods were reported to be efficient techniques for practical applications [5, 65] such as image classification, text recognition, etc. However, applying deep learning to metagenomics is challenging because of high data complexity, and the small size of observed samples. In this setting, classical machine learning algorithms such as Random Forest (RF) usually show a better result than deep learning [78, 11]. This chapter investigates various architectures of deep learning methods such as Convolutional Neural Networks, and their application to the proposed visualizations (Chapter III). The proposed CNN architecture that is considered in Chapter III and Chapter IV includes ***one convolutional layer and 64 kernels, followed by a max pooling of 2×2***. Although this architecture is relatively modest, it outperforms deep architectures such as VGG-like models. Furthermore, we demonstrate encouraging results on 25 real metagenomic benchmarks related to different diseases characterized by various number of features ranging from hundreds to millions features. The proposed method is evaluated on species abundance, genus abundance and gene-families abundance and compared to the-state-of-the-art of standard machine learning algorithm such as RF, SVM [18, 37] as well CNN on metagenomics such as Ph-CNN [11].

Some our recent results on deep learning for metagenomics have led to two articles at the Workshop on Machine Learning for Health of the Conference and Workshop on Neural Information Processing Systems (NIPS) 2017, and the annual French Conference in Machine Learning, la Conférence sur l'Apprentissage automatique (CAp 2018). In the first paper, we presented preliminary ideas and illustrated them by some results using our own implementation in Torch [1]. The second paper is an advanced version of it, and it shows the results of our package written in Python. The empirical performance shows a significant improvement compared to the-state-of-the-art (MetAML [18] and Ph-CNN [11]).

- T. H. Nguyen, Y. Chevaleyre, E. Prifti, N. Sokolovska, J.-D. Zucker. *Deep Learning for Metagenomic Data: using 2D Embeddings and Convolutional Neural Networks.* NIPS 2017 Workshop on Machine Learning for Healthcare. In Proceedings of the NIPS ML4H 2017 Workshop in Long Beach, CA, USA.

- T. H. Nguyen, E. Prifti, Y. Chevaleyre, N. Sokolovska, J.-D. Zucker. *Disease Clas-*

sification in Metagenomics with 2D Embeddings and Deep Learning. In Proceedings of Conférence d'Apprentissage (CAP) 2018, Rouen, France.

It is planned to submit the results presented in the following chapters III and IV to the *Scientific Reports*.

Chapter II

Feature Selection for heterogeneous data using Deep Self-Organising Maps

Contents

Abstract: Feature selection is a challenging task, and is needed in numerous practical applications where data are supposed to be integrated from a variety of heterogeneous high dimensional data sources. In this chapter, we present a feature selection framework based on a robust visualization algorithm, Self-Organising Maps (SOM) to learn deep structures in either a supervised or unsupervised way. We propose a supervised version of the deep SOM that is implemented with a linear Support Vector Machine (SVM) and a forward-backward procedure to converge to an optimal feature set. Our numerical experiments show that our method achieves a reasonable performance in accuracy (ACC) compared to the-state-of-the-art approaches on the a large-scale biomedical data set.

II.1 Introduction

In this chapter, our goal is to develop an efficient feature selection approach which will design a compact model. The method needs to be scalable, to fusion heterogeneous data,

and be able to reach a better generalizing performance compared to a full model and to state-of-the-art methods. Another important question is whether introducing data of different nature have a positive effect, and provides additional knowledge. An important aspect of feature selection is whether a model is easily interpretable, and whether it is possible to visualize the results in order to investigate dependencies in the model.

We propose a framework which is based on SOM [69]. In this contribution, we run the learning procedure with linear support vector machines. The deep linear SVM has been considered and tested in [56] and it was reported that replacing the soft-max function in a deep structure by a linear SVM leads to a better accuracy. The learning procedure minimizes a hinge or a margin-based loss, and not the cross-entropy. Our contribution is multi-fold:

- We introduce and consider a simple deep feature selection framework which constructs layers of a deep structure layer-wise, the deep structure is based on the capability of SOM to visualize clustering in 2D, the proposed deep architecture can be learned either in a supervised or unsupervised mode,
- We illustrate that the proposed framework is efficient on a real original rich heterogeneous MicrObese [46] data set, which contains meta-data, i.e., clinical parameters and alimentary patterns of patients, gene expressions of adipose tissue, and gene abundance of gut flora. We efficiently extract compact new data representations structured into a multi-level hierarchy.
- We evaluate the prediction power of the models with the reduced dimensionality showing that the proposed approach reaches the state-of-the-art performance.

II.2 Related work

Learning features from unlabeled data is important in many applications [47], including bioinformatics and medical informatics, where the number of medical analysis or interventions is critical. Upper layers of hierarchical structures are more abstract data representations, whereas lower layers are low level features from data. [76] states that optimization in deep structures is not obvious. A possible explanation is that standard gradient-based approaches may easily get stuck near poor solutions. To learn a complex model efficiently, it is sometimes useful and beneficial to split a task into simpler models that can be estimated sequentially. The inference is extremely expensive in densely connected networks. Dimensionality reduction and feature selection is already a classical problem associated with deep structures (see [69, 46, 53] for an overview). Several heuristics have been proposed to make the problem tractable. Some of them are based on greedy layerwise inference, [50], and the inference is reported to be quite efficient. It has been shown [47] that feature selection with deep structures is sensitive to the number of hidden layers in graphs, and to the choice of an optimization algorithm. So, in [47] it was demonstrated that a simple K-means clustering can provide very efficient new features representation (for an image processing task). When extracting features from plentiful unlabeled data, the dimension of a problem becomes easily very big. Apart from numerous feature selection methods, there are approaches how to deal with manifold. [58], e.g., proposed a classifier which is insensitive to local directions changes along the manifold.

The idea to do feature selection using SOM is not new. [70] introduced a simple greedy heuristic iterative approach for feature selection which includes 4 steps: 1) learn a SOM and label map; 2) if the classes overlap, then add a new feature or replace a feature; 3) if a feature does not improve the separability of the groups, eliminate or replace this feature; 4) retrain and re-label the map. We also propose a feature selection algorithm based on a clustering, and, namely, a SOM. Note, however, that [70] clusters observations and greedily looks for features ameliorating the separation of classes. We, on the contrary, cluster features, and look for best representatives in each feature cluster. Clustering of features has been already considered by [73, 68]. The principal interest was to build classifiers in a semi-supervised manner and to help analysts in their choice of features or attributes. Another motivation of [70] was to illuminate relationships between attributes, their relative importance for classification, and to better understand structure in data. Another clustering of features was done in [59]. [59] has introduced an algorithm called FAST which consists of two simple steps: 1) features are clustered (by using graph-theoretic clustering methods); 2) the most representative features somehow strongly related to classes are selected from each cluster to form a subset of new features. This approach is close to our idea. However, we do not estimate any relations to classes while choosing best representatives from clusters. In this study, we use SOM clustering, however, it is possible to investigate the clustering with medoids for the same purpose. Partitioning around medoids (PAM) is introduced and described in details by [51], [49] . This is another quite efficient and robust clustering, which can be used for a hierarchy construction [49]. In an already classical deep architecture, in convolutional nets, the non-linearities are sigmoid functions, and new representations are learned in supervised mode using gradient descent. The advantages of the convolutional nets and SVM are combined in [66]. Deep structures learn complex mappings by transforming their inputs through multiple layers of nonlinear transformations . There are several motivations to learn such an architecture. It is, first of all, a way to combine supervised and unsupervised learning. Second, there is a number of functions that can be used to compose weakly non-linear transformations. [61] introduced a multilayer kernel machines, which are based on an iterative procedure, repeated for each layer of a structure: 1) compute principal components in the feature space induced by a nonlinear kernel, and 2) prune components that are less informative from the feature space. Our approach, in its unsupervised mode, is a convolutional net. An interesting parallel between [61] and us, apart from using SVM, is that SOM is a nonlinear generalization of the PCA. Another avenue of research is controlling structure in data by penalty terms such as lasso-like methods. So, [71] proposed recently to add some convex constraints to the logistic regression penalized by the L1 norm to produce a sparse model which involves a hierarchy restriction on feature interactions: an interaction is included if one or both features are marginally important. The disadvantage of the method is that the number of features in this approach is very big even for moderate-size applications, since the approach tests all interactions pairwise.

II.3 Deep linear support vector machines

To learn a hierarchical model, a training algorithm has access to n i.i.d. data points. We can either have labeled pairs $(X_i; Y_i)_{1 \leq i \leq n}$, or an unlabeled data set $(X_i)_{1 \leq i \leq n}$. The

input variable or covariate is $X \in \chi$, and the class variable is $Y \in \mathcal{Y}$, if the problem is supervised. The covariate variables are high-dimensional, and $X_i = (X_{i,1}, ...X_{i,d})$, where d is the dimensionality of the problem. We are interested, in particular, to reduce the number of features in the model, so that the dimensionality of our problem becomes $r \ll d$, and so that we can carry out a classification task on a much more compact, and probably less noisy, feature space. A deep structure can be learned with an SVM. A version of deep linear SVM which we exploit in our framework, has been introduced by [56] . The function in the linear case takes the following form (Eq. II.1):

$$\mathcal{L}(w) = \min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max(1 - w^T x_i yi, 0) \tag{II.1}$$

and the rule to take a decision (Eq. II.2)

$$\hat{y} = \arg\max_{y} w^T xy \tag{II.2}$$

Let h_j be hidden or latent layers in the hierarchy, then (Eq. II.3)

$$\frac{\partial \mathcal{L}(w)}{\partial h_j} = -Cwy_j \mathbf{1}_{\{w^T h_j y_j > 1\}} \tag{II.3}$$

A way of producing probabilistic outputs from a kernel method (see [67]) is to use (Eq. II.4)

$$p(y|x, w) = \frac{\exp w^T x}{1 + \exp(w^T x)} \tag{II.4}$$

The logistic function can be used as an activation function of a deep learning structure. In such a deep supervised learning based on the linear SVM, we have two alternating steps. The forward step: apply logistic regression function to provide probabilistic interpretation and to activate units (weights are fixed). The backward step: compute the gradient presented as eq. (II.3), and update the weights. The gradient is one of the linear SVM, it is convex and differentiable, and we can apply any standard gradient descent method.

II.4 Self-Organising Maps for feature selection

The Self-Organising Map (SOM) is an artificial network associated with the unsupervised learning paradigm [69]. It is famous for its efficient manner to map from a high dimensional input space into a more compact space, usually to a two-dimensional output space. The two-dimensional representation is practical for a visualization, since the mapping preserves topological relations between elements on the grid. Moreover, the continuous input space can be mapped into a discrete output space. The SOM belongs to competitive learning methods, since neurons compete to be activated, and, as a result, only one is activated at a time. The winning neuron is called the winner. When the winner is set, all the other neurons have to re-organize themselves. Interestingly, the SOM can be seen as a non-linear generalization of principal component analysis. Given high-dimensional data $x \in \mathbb{R}^d$, the connection weights between observations i and the neurons of the grid j can be presented as $wj = w_{ij} : j = 1, ..., K; i = 1; ..., n$, where K is the number of neurons on the grid. A discriminate function (Eq. II.5)which is widely used, and which we also use

in our experiments, is the squared Euclidean distance between an observation x and the weight vector w_j, for all j

$$D_j(x) = \sum_{i=1}^{n} (x_i - w_{ji})^2 \tag{II.5}$$

The structure learning can be either supervised, or unsupervised. This algorithm includes four steps detailed in Algorithm 1.

Algorithm 1 Self-Organising Maps Learning Procedure

1: Initialization: all connection weights are initialized randomly;
2: Competition: for each observation, and all features, the neurons compute their values of a discriminant function;
3: Cooperation: The winner determines the spatial location of a topological neighborhood for other neurons, what provides the basis for cooperation between neighboring neurons;
4: Adaptation: Excited neurons decrease their values through an adjustment of the connection weights;

II.4.1 Unsupervised Deep Self-Organising Maps

In an unsupervised setting, the feature selection procedure is completely unsupervised, and the algorithm performs only the first step, a forward pass. In this forward pass, we construct a deep structure layer-wise, where each layer consists of the clusters representatives from the previous level. A natural question which arises is whether such an unsupervised feature selection can be beneficial for a prediction task. Although it is currently impossible to provide a theoretical foundation for it, there is an intuition why a deep unsupervised feature selection is expected to perform and performs better in practice. Real data are always noisy, and a "good" clustering or dimensionality reduction can significantly reduce the noise. If features are tied into clusters of "high quality", then it is easier to detect a signal from data, and the generalizing classification performance is higher. The hierarchical feature selection plays here a role of a filter, and a filter with multiple layers seems to perform better than a one-layer filter.

II.4.2 Supervised Deep Self-Organising Maps

The supervised deep SOM feature selection is based mostly on the forward-backward idea. Forward greedy feature selection algorithms are based on a greedily picking a feature at every step to significantly reduce a cost function. The idea is to progress aggressively at each iteration, and to get a model which is sparse. The major problem of this heuristic is that once a feature has been added, it cannot be removed, i.e. the forward pass can not correct mistakes done in earlier iterations. A solution to this problem would be a backward pass, which trains a full, not a sparse, model, and removes greedily features with the smallest impact on a cost function. The backward algorithm on its own is computationally quite expensive, since it starts with a full model [43]. We propose a hierarchical feature selection scheme with SOM which is drafted as Algorithm 2. The features in the backward step are drawn randomly.

Algorithm 2 Feature Selection with Forward-Backward SOM

```
for each layer l ∈ L do {bottom up}
    Run a SOM
    Select representatives from each cluster to propagate them to an upper level
end for
for each layer l ∈ L do {top down}
    Estimate accuracy for level l
    Greedily update selected features
end for
```

II.5 Experiment

In this section, we describe our experiments and results on a real rich, and original biomedical data set. To construct the SOMs, we use somtoolbox1 from Matlab. We also use SOM graphics from [54], [62].

II.5.1 Signatures of Metabolic Health

The biomedical problem of our interest is a real problem which is a binary classification of obese patients. The aim is to stratify patients in order to choose an efficient appropriate personalized medical treatment. The task is motivated by a recent French study [46] of gene-environment interactions carried out to understand the development of obesity. It was reported that the gut microbial gene richness can influence the outcome of a dietary intervention. A quantitative metagenomic analysis stratified patients into two groups: group with low gene gut flora count (LGC) and high gene gut flora count (HGC) group. The LGC individuals have a higher insulin resistance and low-grade inflammation, and therefore the gene richness is strongly associated with obesity-driven diseases. The individuals from a low gene count group seemed to have an increased risk to develop obesity-related cardiometabolic risk compared to the patients from the high gene count group. It was shown [46] that a particular diet is able to increase the gene richness: an increase of genes was observed with the LGC patients after a 6-weeks energy-restricted diet. [19] conducted a similar study with Dutch individuals, and made a similar conclusion: there is a hope that a diet can be used to induce a permanent change of gut flora, and that treatment should be phenotype-specific. There is therefore a need to go deeper into these biomedical results and to identify candidate biomarkers associated with cardiometabolic disease (CMD) risk factors and with different stages of CMD evolution.

II.5.2 Dataset description

The MicrObese corpus contains meta-data, genes of adipose tissue, and gut flora metagenomic data. For each patient, we have the information to which class he or she belongs. There are two classes, high gene count (HGC) and low gene count (LGC) classes. Therefore, our problem is a binary prediction task from heterogeneous data. In general, 49 patients have been hired and examined at the Pitié-Salpêtrière Hospital hospital, Paris, France [46], but as to the genes of the adipose tissue, we faced the problem of missing data, and not for all patients their class, LGC or HGC is provided. We decided to

impute missing data by median values for the adipose tissue data. The patients who were not clearly stratified into the LGC or HGC group, were excluded from the analysis. Therefore, in our experiments we have access to 42 observations (patients). To get rid of important noise, after the discussion with pre-clinical researchers, we run a significance test (Kruskal-Wallis), and we keep those variables for which the raw (not adjusted for the multiple hypothesis testing) p-values < 0.05.

Figure II.1 is a hierarchical structure based on SOM. Each upper layer is constructed from variables which are the closest ones to the unit centers of the previous level. Here we also perform data integration. We carry out feature extraction for four data sources – metagenomic species, environmental data, host variables, and genes expressions for adipose tissue. We do feature selection separately for each data source (three layers). Then we integrate all selected variables in one analysis and obtain a mixed signature (also three layers). Taking into consideration that we would like to get a well-balanced signature, where each data type is presented by some features, the SOM of the lower levels of the hierarchy are constructed per data source, since the number of parameters are extremely different in, e.g., adipose tissue data and in the block of environmental variables. Although Figure II.1 provides a schematic overview, the maps on the figure are exactly what we get in our experiments. It is interesting to see that lower levels where the number of parameters is quite big, do not reveal specific structures in data. The highest levels, on the contrary, show well-organized clusters. Figure II.2 illustrates the quantization error associated with hierarchies on each data sources and on the mixed hierarchy. It is easy to see that in all cases the quantization error diminishes. Figure II.3A illustrates the patients separation after the feature selection, where 1 stands for high gene count patients, and 2 for the low gene count ones. Note that each cluster may contain several patients.

The framework of Figure II.1 can be applied to the whole MicroObese cohort, both to the HGC and to the LGC data points (we do the 10-folds cross validation in all our classification experiments), but we can also split the data into the HGC and LGC data sets, and extract signatures for each group. These results that can be found on Figure II.4A and B are very interesting for clinicians and researchers doing pre-clinical research, since these signatures allow them to better characterize the groups of patients.

Figure II.4C shows the result of the prediction using the HGC and LGC groups. The signature, therefore, characterizes the discrimination between two classes. It is a well-reported fact that biological and medical signatures are rather unstable. See, for instance, [75], where a comparison of more than thirty feature selection methods has been made, and where it has been shown that the stability of modern state-of-the-art approaches is rather low.

Another avenue to analyze signatures, is to construct Bayesian networks and to study the relations between the variables. We carry out feature selection with the deep SOM, and we run a Bayesian network on the selected variables. Figure II.5 reveals the signature relations of the high gene count group and the low gene count group with the Bayesian network. The highest level of the deep SOM structure and the Bayesian networks provide complementary results. If we compare the relations for the HGC group, (Figure II.4A and II.5A), we will see that the SOM clusters and the Bayesian networks quite often provide similar results, however, in some cases they reveal different relations between variables of interest. It is interesting, that the number of selected features for the LGC is bigger

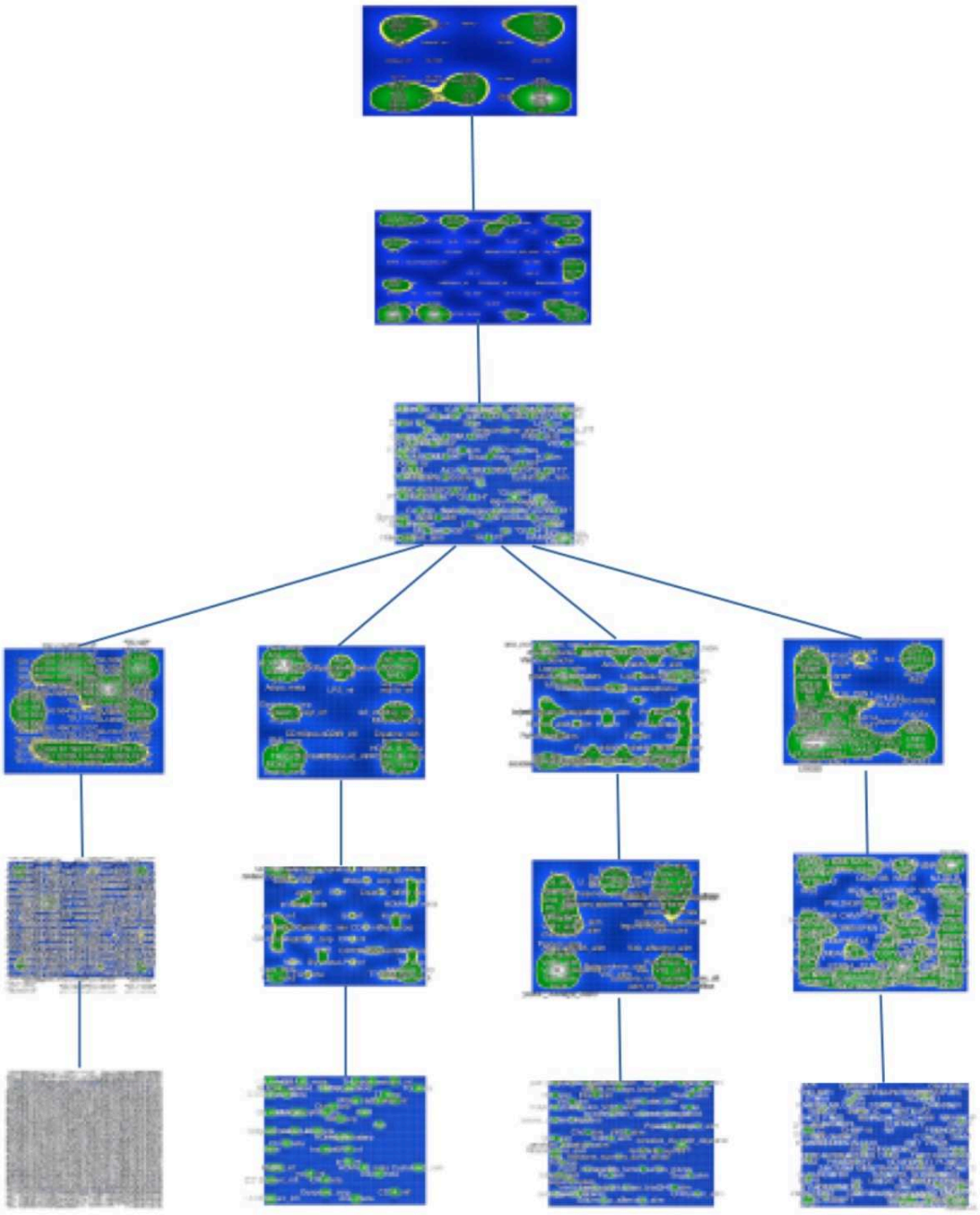

Figure II.1 *The hierarchy of SOM. For three lower levels, from left to right: MGS, environmental variables, host, and adipose tissue microarray data. Three upper layers perform data integration from four data sources.*

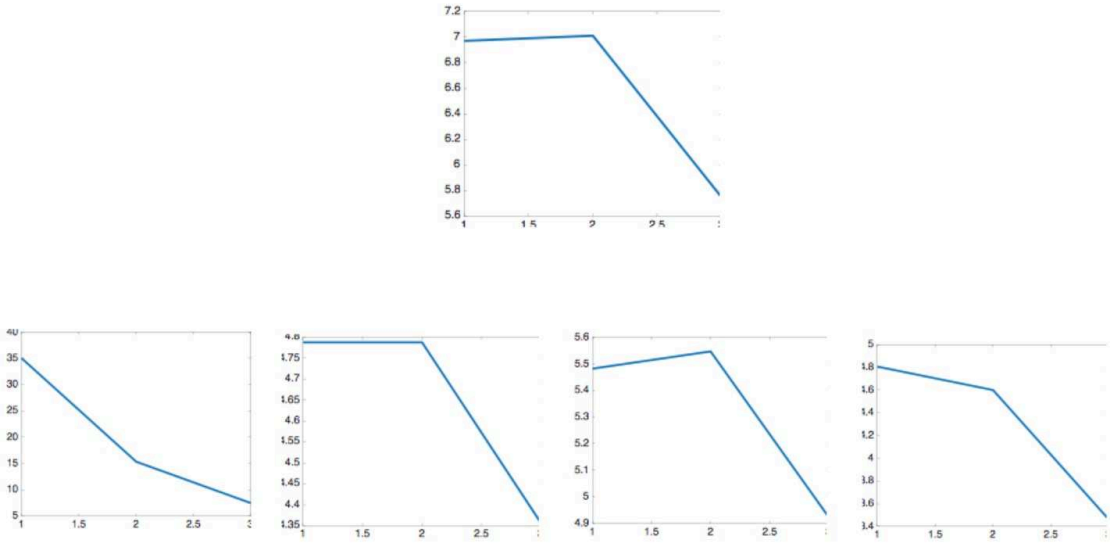

Figure II.2 *Quantization error. Above: the error associated with three highest levels of the SOM deep architecture; below: the quantization error for each data source, associated with the lower three levels of the SOM hierarchy on Figure II.1*

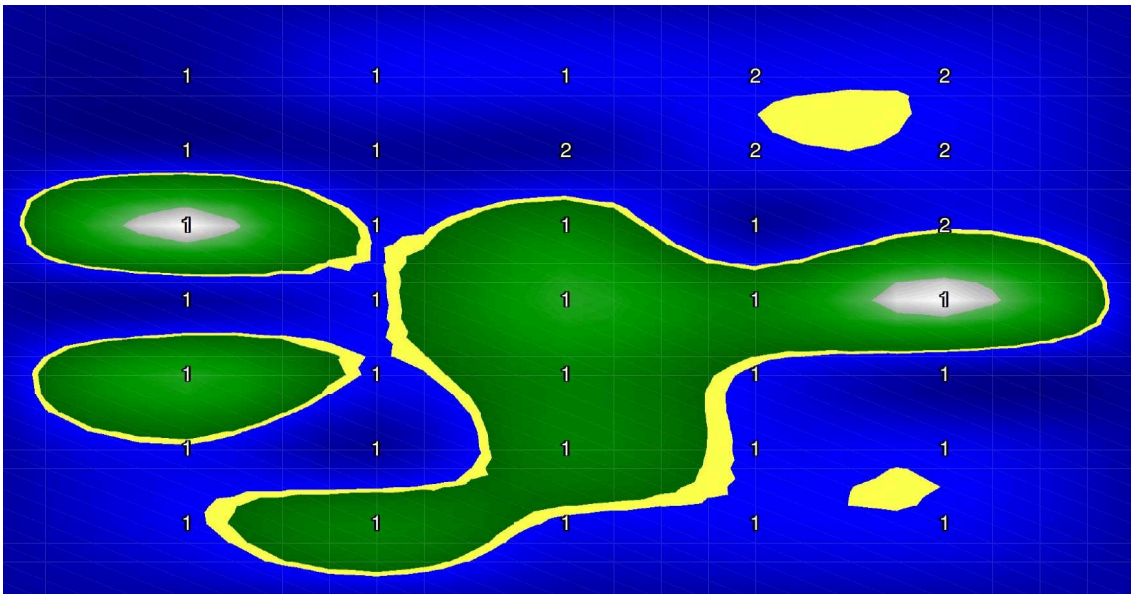

Figure II.3 *Separation of patients with the selected features. 1– high gene count patients, 2 – low gene count patients.*

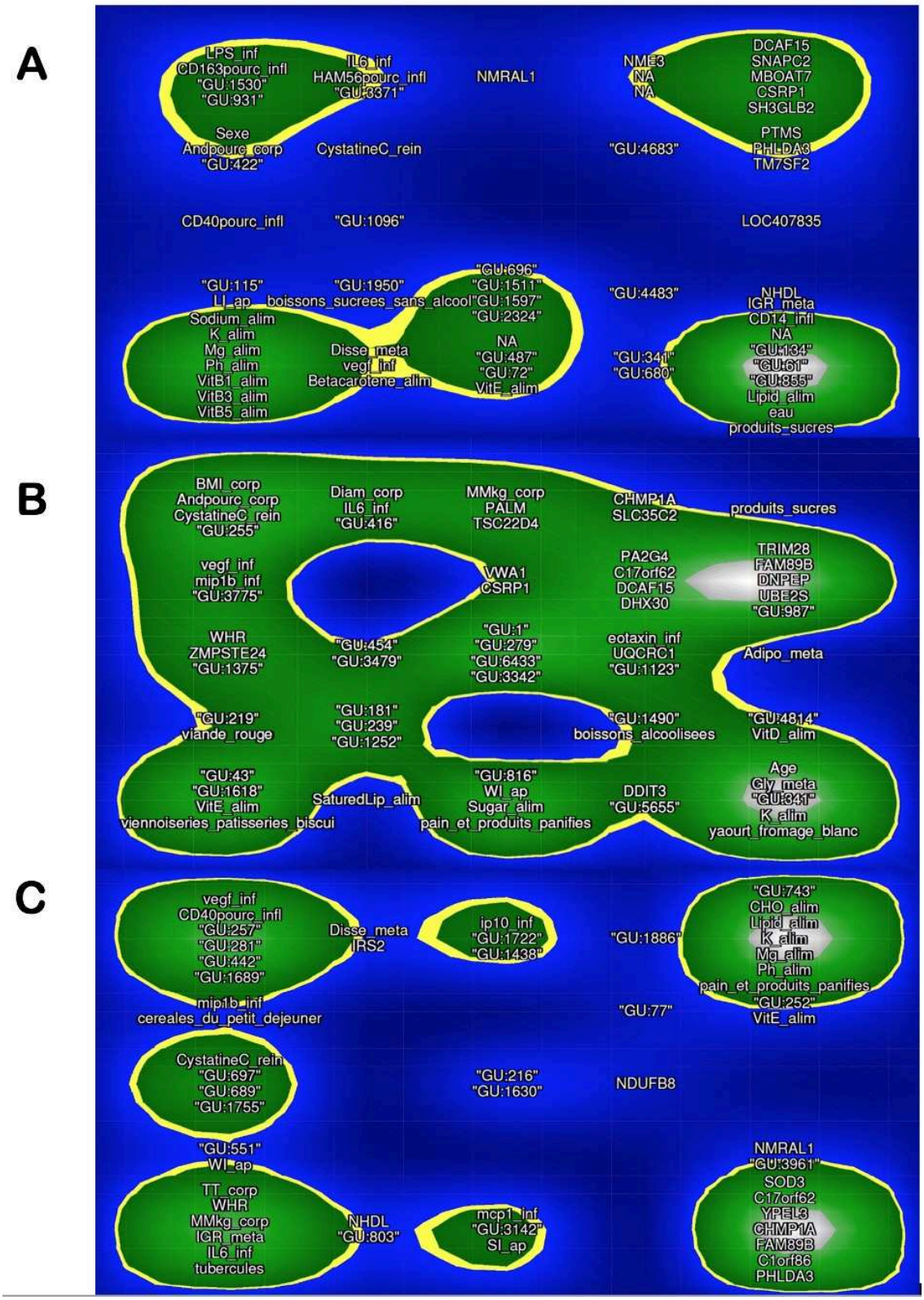

Figure II.4 *(A): Signature of the high gene count group which is associated with a better health. (B): Signature of the low gene count group which is associated with higher inflammation. (C): A signature which discriminates high gene count and low gene count groups.*

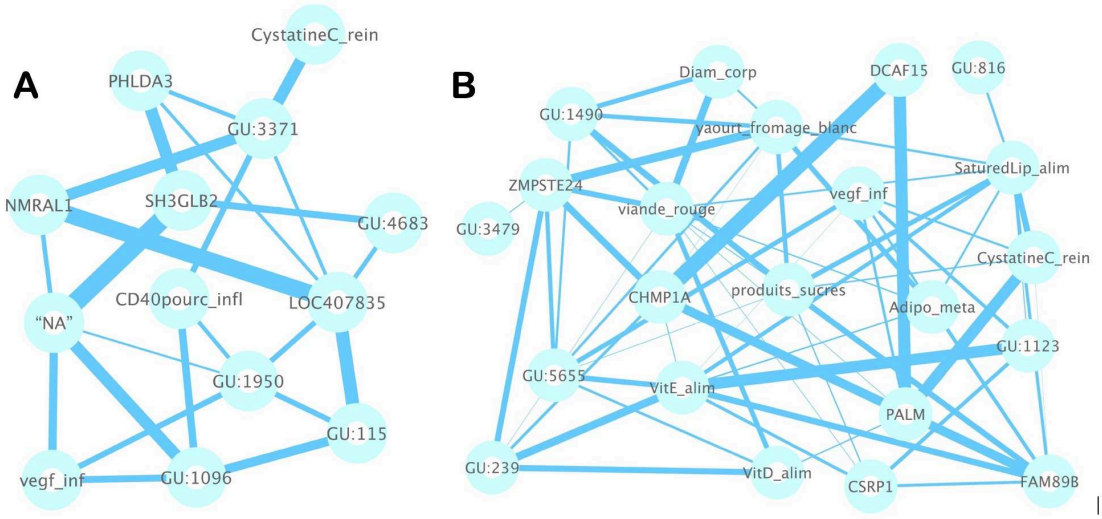

Figure II.5 *(A): Bayesian network of the selected features associated with the HGC (A) and LGC (B) group*

than for the HGC. Analyzing Figure Figure II.4B and II.5B, we do the same conclusion: the SOM and the network reveal the same structure in data, with several interesting exceptions. The biomedical analysis of the results is out of scope of this study, and will be done by fundamental biologists and clinicians.

II.5.3 Comparison with State-of-the-art Methods

In this section we show that the proposed feature selection method is efficient compared to the state-of-the art methods such as lasso and elastic net. In our experiments we use the glmnet R package [74]. Figure II.6 shows the 10- folds cross validation error rate on the MicroObese data as a function of the number of non-zero features in the model. We have done unsupervised feature pre-selection with the deep SOM, the structure drafted on Figure II.1. Then we apply the lasso to the pre-selected set of features, and compare the result with the lasso applied to the whole set of parameters (more than 2000 features).

Figure II.6A on the left demonstrates the performance for the lasso applied to all features, without the unsupervised pre-selection step. On the right of Figure II.6A, we show the performance of our framework. Note that since the pre-selection step is unsupervised, we do not do any overfitting. The accuracy of both methods is comparable, and taking into consideration that the prediction task is quite challenging, the error rate around 0.3 – 0.33 is acceptable. However, the proposed framework applies lasso to a reduced data set, with a hundred of parameters, and not with thousands as the initial set.

Figure II.6B illustrates our result (with the elastic net) which is similar to the one of Figure II.6A. It shows the performance as function of the number of selected features. On the left, we show the error rate for the elastic net ($\alpha = 0.5$ in the *glmnet* R package) on all features, and on the right of II.6B, for the proposed approach. The elastic net achieves a higher accuracy than the lasso. As in the previous lasso experiment, we run the elastic

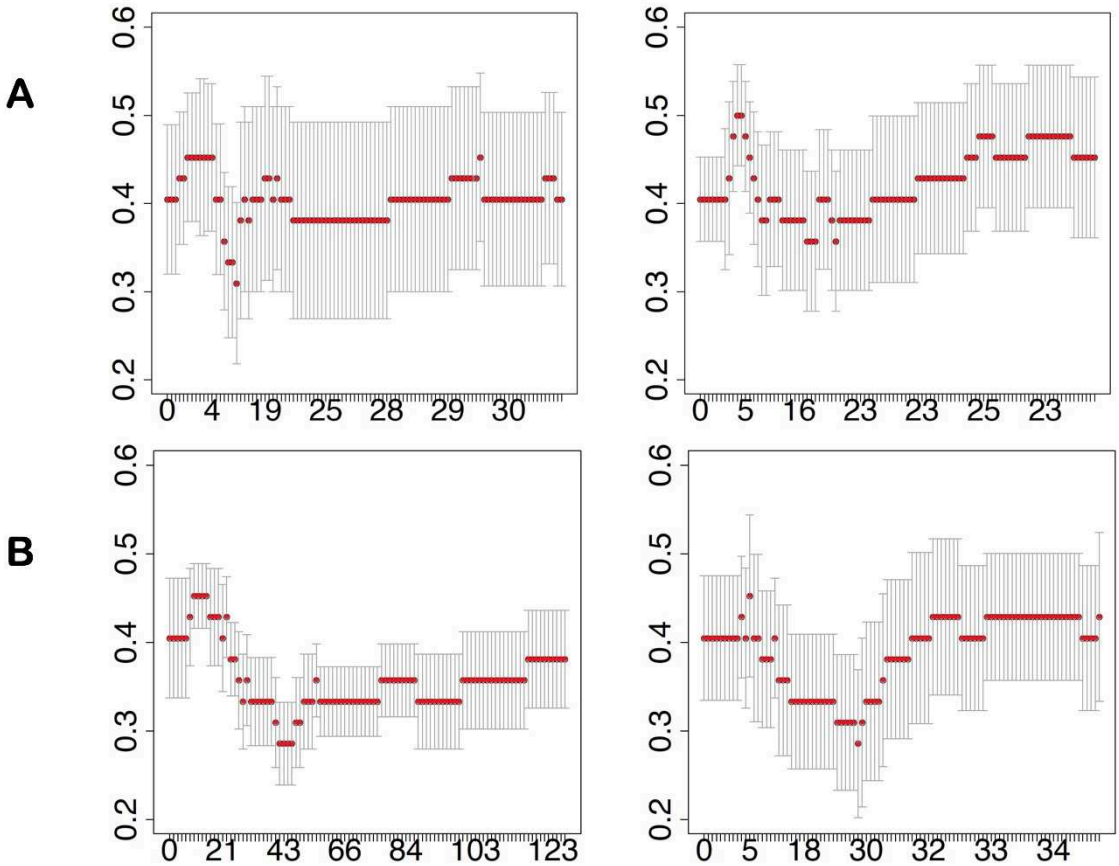

Figure II.6 *The 10-folds cross validation error rate on the MicroObese data as a function of the number of active features. On the left: the lasso (A) and the elastic net (B); on the right: the lasso (A) and the elastic net (B) after the unsupervised feature selection.*

net on about 100 variables for our approach, instead of thousands (the result on the left), and we see that the best model is one with about 30 parameters chosen among 100 from the features pre-selected in unsupervised manner. Note that the best model learned from all parameters (on the left of Figure II.6B), but with the comparable error rate, has more, namely, 43 features.

II.6 Closing and remarks

Data integration is a challenge, especially in applications where data are high-dimensional, e.g., metagenomics and the metagenomic species, and where the number of observations (patients) is small. We have proposed to reduce dimensionality by a deep approach which is based on SOM, and which learns new compact data layer-wise, in a hierarchical way. We have considered supervised and unsupervised feature selection frameworks, as well as we considered a real data integration challenge. We show that the considered deep SOM approach is efficient on a real medical complex data set, and it is beneficial

to combine it with the lasso and the elastic net approaches. The unsupervised feature selection diminishes the computational burden of the standard methods and also leads to the state-of-the art performance. Although the detailed biomedical discussion of the features clustering and of the quality of the obtained signatures is out of scope of this study, and is to be done by biologists doing pre-clinical research, we expect that our framework can help to better stratify patients, develop methods of personalized medicine, and improve diagnosis and prognosis.

Chapter III

Visualization Approaches for Metagenomics

Contents

Abstract: In this chapter, we introduce the **Met2Img** framework based on *Fill-up* and dimensionality reduction methods to visualize features which reflect the abundance of different levels of OTUs such as species, genus, order, family, class, phylum. A simple way as Fill-up arranges abundance/presence values into a square matrix sizing by the number of features. We also consider more sophisticated methods that carry out dimensionality reduction and map the data into a 2-dimensional space. The dimensionality reduction are used in this study, both supervised and unsupervised methods such as Linear Discriminant Analysis (LDA), Isomap, and t-Distributed Stochastic Neighbor (t-SNE). We propose three binning methods which are essential to map abundance/presence values into images. Furthermore, we consider a range of diverse colormaps provided from Python library.

Our approach is evaluated on six metagenomic datasets using either abundance or presence values. We report that the proposed **Met2Img** not only shows improvements in the performance but also allows to visualize biomedical signatures.

III.1 Introduction

The visualization of metagenomic data is still a challenging issue in computational biology due to its very large dimensionality, as well as complex interactions among microbes. In addition, metagenomic data also shows complicated correlations with confounding environmental factors [165] (for example, total organic carbon, total nitrogen and pH[166]). As illustrated in numerous studies, data visualization is considered as an indispensable technique for the exploratory data analysis and becomes a key for discoveries [153]. A good visualization should discriminate between specific groups to extract characteristics of these groups. In addition, an ideal visualization method enables us to analyze such large-scale data efficiently.

In [153], the authors stated that Metagenomics visualization has become an attractive domain with a vast of publications introducing numerous novel approaches every year, and presenting new techniques based on visualization for generating and verifying novel biological hypotheses. They presented an overview of the existing approaches to visualize metagenomic data. Furthermore, the study also pointed out the best-known visualization of compositional data is a pie chart which shaped like a circular graphic separated into pieces. Each such piece represents a group of corresponding data in percent. Pie chart is available, implemented popularly to a variety of softwares and platforms such as Python, R [154], Excel, and so on. Krona [155] is one of such popular tools commonly-used in the research community. The software presents a metagenome as nested concentric rings shaping a circle together. Each ring matches to a taxonomic rank. This visualization reveals a multi-level view of the structures of metagenome data. MG-RAST [157] is a type of the web-based display, represents metagenomics in the hierarchy regardless of magnitude. MEGAN is a software that enables us to analyze and explore the taxonomic content of large metagenomic data. A comparison among 3 these common methods are presented in [155]. Figure III.1 exhibits the taxonomic classification using MG-RAST, MEGAN and Krona. As seen from the figure, all levels of the hierarchy of the most abundance taxa are visualized clearly by Krona while MEGAN and MG-RAST is limited the scope of their overview. MEGAN and MG-RAST expand hierarchical node-link diagrams with log-scaled, quantitative charts while Krona utilizes a radial space-filling display to visualizes abundance and hierarchy concurrently. Krona uses a red-green color gradient revealing average e-values of Basic Local Alignment Search Tool (BLAST) [159] hits inside each taxon. Red represents the highest observed e-value (least significant) while green depicts the lowest (most significant) one.

A bar chart that is also a useful tool to represent a data distribution, forms as rectangular colorful bars for each group of data. The height of these bars reflects the values of corresponding groups. A vast of tools provided include *AmphoraVizu* [160], package of *metricsgraphics* [162] and *gplots* in R [161]. *Phinch* [163] is also a helpful software to show taxonomic composition of microbial communities.

As [153] presented, an abundance table, where the rows representing the samples and columns corresponding to features, is considered as a standard method to represent the community structure inferred from metagenomic datasets. In this table, each cell contains the value of the relative abundance corresponding taxa in the sample. A heat map table is an extended version of abundance table. Each cell in this table is filled by a color. The different abundance between 2 cells is identified by distinct colors. The R package

d3heatmap [164] provides a variety of options to build a vast of kinds of heatmaps. In addition, *Anvi'o* is also able to figure a heat map of nucleotide positions. Heatmaps table is a key idea that we conduct for the Fill-up approach.

Find the global structure of microbial community using metagenomic data is really a substantial challenge. Besides charts and tables, due to very large dimensionality of data, researchers also have attempted recently dimension reduction algorithms such as Isomap, Principle component analysis (PCA), t-SNE in numerous metagenomic studies [167, 168, 169, 170]. Each sample is characterized by hundreds or thousands of features (relative abundance of individual species or genus), and a dimensionality reduction method can be applied. A new reduced data can be presented as a point (or dot, pixel) on the scatter plot of two (2D) or three (3D) principal components.

As stated in [153], recent discoveries in microbial data using metagenomic data have substantially increased our understanding on the structure and possible functions of bacterial in the human body. These discoveries are incredibly difficult to complete without a good visual tool to go deeply into the analysis. However, although the benefits of visualization on metagenenomic data was demonstrated by a vast of studies, applying these visualizations for prediction tasks using deep learning techniques are not taken into account completely. In this chapter, we explore and investigate different approaches for visualizing features of metagenomic data. Our objective is to propose efficient data representations which produce compact and informative images, which not only present data distributions visually but also can be used for classification.

The chapter is organized as follows. Related work on visualization approaches is presented in III.2. In Section III.3, we describe six species abundance datasets (group A) used for performing visualizations. The visualization methods are introduced in Section III.4 including the methods of visualizing feature abundance (the Fill-up and manifold learning approaches) as well as the comparison results of each method. A number of binning methods including binning based on abundance distribution, transformation and one-hot encoding along with a variety of colormaps are also taking into account in this chapter. Finally, conclusion is presented in the Section III.5.

III.2 Dimensionality reduction algorithms

In this study, we consider a number of dimensionality reduction approaches for feature visualization including both supervised and unsupervised learning algorithms. As stated in [81] and [88], manifold learning methods are a robust framework for reducing the dimension. The idea for these algorithms is to find a more compact space that is embedded in a higher dimensional space. Ingwer et al. presented Multidimensional scaling (MDS) that create a mapping from a high-dimensional space to a more compact space and try to preserve the distance between pairs of points while Locally Linear Embedding (LLE) and Isometric Mapping (Isomap) were considered as a new generation of dimensionality reduction algorithms and applied to synthetic and real data with great achievements. We recall some information on some of the prominent dimensionality reductions.

Linear Discriminant Analysis (LDA) [172, 171] is a supervised dimensionality reduction method that fits a Gaussian density to each class. This algorithm projects the input to the most discriminative directions to reduce the dimension. As described

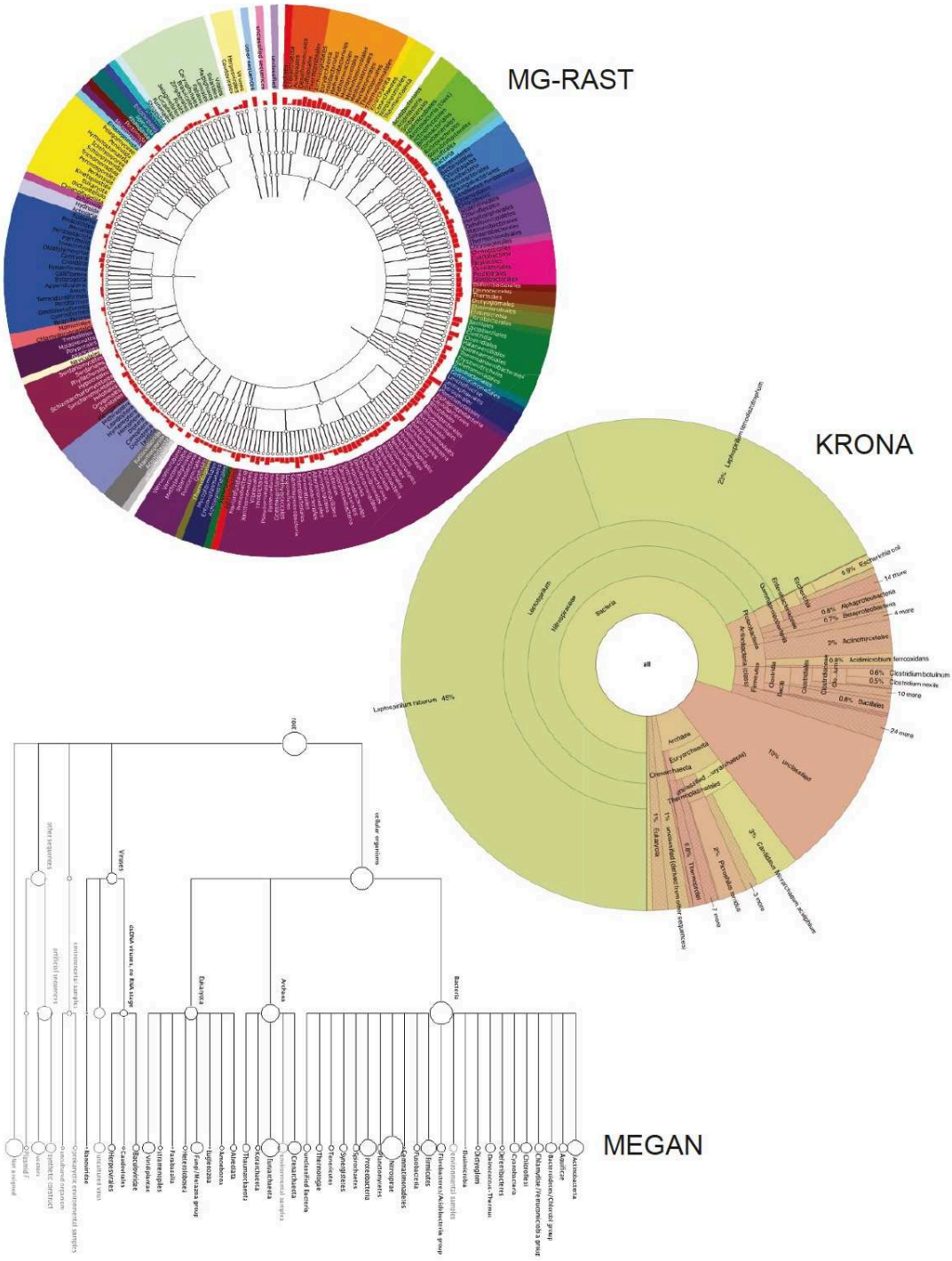

Figure III.1 *The acid drainage metagenome [156] was visualized by MG-RAST [157], MEGAN [158], and Krona [155]. The figure is provided from [155].*

in [173], classical LDA puts the data into a lower-dimensional vector space with the maximum proportion between *between-class* distance and the *within-class* distance to ensure a maximum discrimination. Given a data $A \in \mathbb{R}^{N\times n}$ with n is the number of sample (rows) and N is the number of features or dimensions (columns). Given l is the desired dimension ($l < N$), a transformation $G \in \mathbb{R}^{N\times l}$ where $G : a_i \in \mathbb{R}^N \rightarrow b_i = G^T a_i \in \mathbb{R}^l$ aims to map each column a_i of A (1$\leq i \leq$n) with N-dimensional space to a vector b_i in l-lower-dimensional space. Suppose that samples in A is grouped into k categories: A $= \{\Pi_1, \Pi_2, ..., \Pi_k\}$ where the ith class Π_i consists of n_i samples (with $\sum_{n=1}^{k} n_i = n$). Two scatter matrices, namely *within-class* $S_w = \sum_{i=1}^{k} \sum_{x\in\Pi_i (x-m_i)(x-m_i)}{}^T$ and *between-class* $S_b = \sum_{i=1}^{k} n_i(x - m_i)(x - m_i)^T$, where $m_i = \frac{1}{n_i}\sum_{x\in\Pi_i} x$ depicts the mean of the ith category and $m = \frac{1}{n}\sum_{i=1}^{k}\sum_{x\in\Pi_i} x$ reveals the global mean, aim to evaluate the quality of the clusters. While trace(S_w) estimates the closeness of the vectors within the classes, trace(S_b) represents the separation between classes. The *within-class* and *between-class* matrices turn to $S_b^L = G^T S_b G$ and $S_w^L = G^T S_w G$ in the low-dimensional space with the linear transformation G, respectively. To optimal G, we need to maximize trace(S_b^L) and minimize trace(S_w^L) which leads to (Eq. III.1)

$$\max_G \{trace((S_w^L)^- 1 S_b^L)\} \text{ and } \min_G \{trace((S_b^L)^- 1 S_w^L)\} \tag{III.1}$$

[172]. Eq. III.1 is corresponding to the generalized eigenvalue problem: $S_b x = \lambda S_w x$ with $\lambda \neq 0$. We employ an eigen-decomposition to $S_w^{-1} S_b$, if S_x is nonsingular, or $S_b^{-1} S_w$, if S_b is nonsingular, to solve this problem. The rank of the matrix S_b is limited from above by $k - 1$, so there exists at most $k - 1$ eigenvectors being equivalent to nonzero eigenvalues. As a result, the most reduced dimensionality is $k - 1$. LDA has been used commonly in many applications regarding to metagenomics [175, 174].

Random Projection is an algorithm to reduce the dimensionality with the core idea was introduced in [120] and widely-used in applications [114, 115, 116, 117, 127, 128, 129]. As described in [114], Random Projection (Rn_pro) utilizes a random matrix with unit Euclidean column norms to find a lower-dimensional subspace that approximately preserves the distances between the points. Given an original d-dimensional data $X \in R^{N\times d}$, where N is the number of points. Rn_pro transforms X to the k-lower dimensional space $X_{k\times N}^{RP}$ (with $k \ll d$) calculated by $X_{k\times N}^{RP} = R_{k\times d} X_{d\times N}$ where $R \in R^{d\times k}$ is a random matrix with unit Euclidean column norms that can be generated from a Gaussian distribution. As in [121] presented the first row is a random unit vector uniformly selected from S^{d-1} while the second is a random unit vector from the space orthogonal to the first one and the third vector is a random unit vector from the space orthogonal to the first two.

Non-negative matrix factorization (NMF) [178, 179] also has been carried out to investigate the complicated structure embedded metagenomic data [176, 181, 180, 182, 167, 177]. As detailed in [176, 182], NMF decomposition calculates a decomposition of samples A into two matrices of W and H with non-negative elements. The algorithm optimizes the distance d between X and the matrix product WH. Given A $\in R^{m\times n}$, and the selected number $k < min(m, n)$, we need to calculate W $\in R_+^{m\times k}$ and H $\in R_+^{k\times n}$ to solve $min_{W\geq 0, H\geq 0}\Phi(A, WH)$ where $\Phi(A, WH) : R^{m\times n} \times R^{m\times n} \longrightarrow R_+$ is some appropriate (distance) metric such as the Frobenius norm with $||A - WH||_F$. The approach aims

to calculate the minimum difference between A and WH to find W and H (Eq. III.2):

$$\underset{min}{W,H} f(W,H) \equiv \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij}-(WH)_{ij})^2 \tag{III.2}$$

Principal Component Analysis (PCA) [77] is one of the most commonly-used linear dimension reduction for numerous applications in exploratory data analysis for performing predictive models. A key idea of PCA aims to find the low-dimensional linear subspace where catches the maximum proportion of variation within data [81]. A disadvantage of PCA is that the embedded subspace should be linear. PCA is a technique to find the largest eigenvalues. Supposing that the data $x_1, x_2, ..., x_n$, PCA calculates the eigenvalues and corresponding eigenvectors of the (r×r) sample covariance matrix. In order to embed to k-lower-dimensional space, PCA preserves only those eigenvalue-eigenvector combinations for which the k largest eigenvalues interpret a high proportion of the total variation in the data.

Multidimensional scaling (MDS) is a form of non-linear dimensionality reduction presented in [122, 123]. The technique aims to place each point in N-dimensional space where distances between points are preserved as well as possible. Classical MDS uses Euclidean distances. As in described classical MDS in [126, 80], the coordinate matrix can be derived using eigenvalue decomposition $B = XX^{'}$ computed by proximity matrix D by using double centering with steps as follows.

- Set up the squared proximity matrix $D^{(2)} = [d_{ij}^2]$
- employ double centering: $B = -\frac{1}{2}JD^{(2)}$, where $J = I_n - \frac{1}{n}J_n$ and $J_n = 1_n 1_n^t$ is matrix of ones.
- Indicate the m largest eigenvalues $\lambda_1, ..., \lambda_m$ and corresponding eigenvectors $e_1, ..., e_m$ of B. m denotes the number of dimensions required for output.
- Compute $X = E_m \bigwedge_m^{\frac{1}{2}}$, where E_m is the matrix of m eigenvectors and $\bigwedge_m$ is the diagonal matrix of m eigenvalues of B.

Isomap was presented in [79, 91] as an extended version of MDS by changing the Euclidean distances with another type of distance (geodesic distances). This method aims to solve high dimensional data using the measured local metric information to learn the structure of data with numerous applications. As in [134], authors applied Isomap to predict key clinical variables. The author in [88] stated that there are two various types of manifold learning including local and global methods. Isomap is a global manifold learning approach with the embeddings based on geodesic distances between all pairs of points. There are three steps in Isomap as follows:

- Consider N points $X_i, i = 1, 2, , ..., N$. For each point X_i, compute the distances $d_{ij}^X = d^X(x_i, x_j) = \|x_i - x_j\|_X$ between all pairs of points $x_i, x_j \in X$, then determine its neighbors that can be either using k-nearest neighbors of each point X_i or an ϵ-neighborhood that consists all of the points that are no more than ϵ-distance away from X_i.

- Calculate a weighted graph of neighborhoods $G = G(V, \epsilon)$, where the vertices, $V = x_1, ..., x_n$ are the input points and the edges, $\epsilon = e^{ij}$ denote the relationships of the neighbor points. The edge e^{ij} which connects between the neighbor points of x_i and x_j owns a weight w_{ij} equivalent to the distance d_{ij}^X between considering two points. The corresponding weight is zero (0) if there is no edge between any two points. The true geodesic distances d_{ij}^M on the manifold are calculated by d_{ij}^G, the graph distances, that are the shortest edges between all pairs of points in the graph, G can be calculated efficiently via Dijkstra's algorithm [89] or Floyd's algorithm [90].

- A low dimensional projection is created using classical MDS.

Locally Linear Embedding (LLE) is a type of local manifold learning method introduced in [124]. As in [88] stated that LLE looks a manifold as a collection of potentially overlapping areas. In the case where the manifold is smooth and the number of points in a neighborhood is minor, the regions look like as locally linear. LLE and Isomap have the main difference that is the way LLE performs in the second step. LLE supposes that every manifold as locally linear . Therefore, each point can be approximated by linear function of its K nearest neighbors. The details of the algorithm can be found in [124, 88, 81].

Laplacian Eigenmaps- (or called as Spectral Embedding (SE)): applies spectral decomposition to the corresponding graph Laplacian. Similar to LLE, SE focus only o retaining local neighborhood relationships in the input space [125]. In some cases, SE is equivalent to LLE. Details of the algorithms are presented in [125, 88]

t-SNE [3, 190] is one of the best technique for dimensionality reduction used more and more recently [130, 133, 132, 131, 132]. t-SNE performs well and seems to like to an appropriate method for visualizing data in 2-3D compared to other manifold methods. Additionally, t-SNE not only illustrates the local structure of data but also preserves the global structures of the data like clusters.

III.3 Metagenomic data benchmarks

We work with metagenomic abundance data, i.e. data indicating how present (or absent) is an OTU (Operational taxonomic unit) in human gut. In this chapter, we evaluated our method on 6 different datasets related to 5 various diseases (Table III.1).

Group A consists of datasets including bacterial species related to various diseases, namely: liver cirrhosis (CIR), colorectal cancer (COL), obesity (OBE), inflammatory bowel disease (IBD) and Type 2 Diabetes (T2D) [18, 19, 20, 36, 2, 10, 32, 141, 142], with CIR (n=232 samples with 118 patients), COL (n=48 patients and n=73 healthy individuals), OBE (n=89 non-obese and n=164 obese individuals), IBD (n=110 samples of which 25 were affected by the disease) and T2D (n=344 individuals of which n=170 are T2D patients). In addition, one dataset, namely WT2, that includes 96 European women with n=53 T2D patients and n=43 healthy individuals is also considered. The abundance datasets are transformed to obtain another representation based on feature presence when the abundance is greater than zero. These data were obtained using the default parameters of MetaPhlAn2 [30] as detailed in Pasolli et al. [18].

Group A	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**
#features	542	503	443	465	572	381
#samples	232	121	110	253	344	96
#patients	118	48	25	164	170	53
#controls	114	73	85	89	174	43
Ratio of patients	0.51	0.40	0.23	0.65	0.49	0.552
Ratio of controls	0.49	0.60	0.77	0.35	0.51	0.448
Autofit size for images	24×24	23×23	22×22	22×22	24×24	20×20

Table III.1 – Information on 6 datasets in group A.

For each sample, species abundance is a relative proportion and is represented as a real number which is the total abundance of all species summing to 1.

III.4 Met2Img approach

In this section, we describe how to map the microbial taxonomic abundance profiles represented by the Operational Taxonomic Units (OTUs) into synthetic images. Our approach consists of the following steps. First, a set of colors is chosen and applied to different binning approaches (see III.4.1). The binning can be performed on a logarithmic scale or a transformation. Then, data are mapped into images by one of two different approaches, i.e. either by the ***Fill-up*** (using either phylogenetic-sorting or random ordering) or based on manifold learning methods such as t-Distributed Stochastic Neighbor Embedding (t-SNE) [3] (see III.4.2). The t-SNE technique is useful to find faithful representations for high-dimensional points visualized in a more compact space, typically the 2D plane. For the Fill-up with phylogenetic-sorting, the features which are bacterial species are arranged based on their taxonomic annotation ordered alphabetically by concatenating the strings of their taxonomy (i.e. phylum, class, order, family, genus and species). This ordering of the variables embeds into the image external biological knowledge, which reflects the evolutionary relationship between these species. In all experiments in Section III.4, we use the Fill-up with phylogenetic-sorting. The performances were evaluated by 10-fold-stratified-cross validation, repeated and averaged on 10 independent runs, compared to MetAML [18]. The results with p-values < 0.05 are significant improvements. Each visualization method is employed to either represent abundance or presence data. The presence/abundance representation serves as a control and is the 1D of the raw data (with the species also sorted phylogenetically). For the representation based on manifold learning approaches, we use only training sets to generate global maps, images of training and test set are created from these global maps.

III.4.1 Abundance Bins for metagenomic synthetic images

In order to discretize abundances, choose a color for them, and construct synthetic images, we use different methods of binning (or discretization). On the artificial images, each bin is illustrated by a distinct color extracted from a color strip of heat map colormaps in Python library such as *jet*, *viridis*, etc. In [38], authors stated that *viridis* showed a good

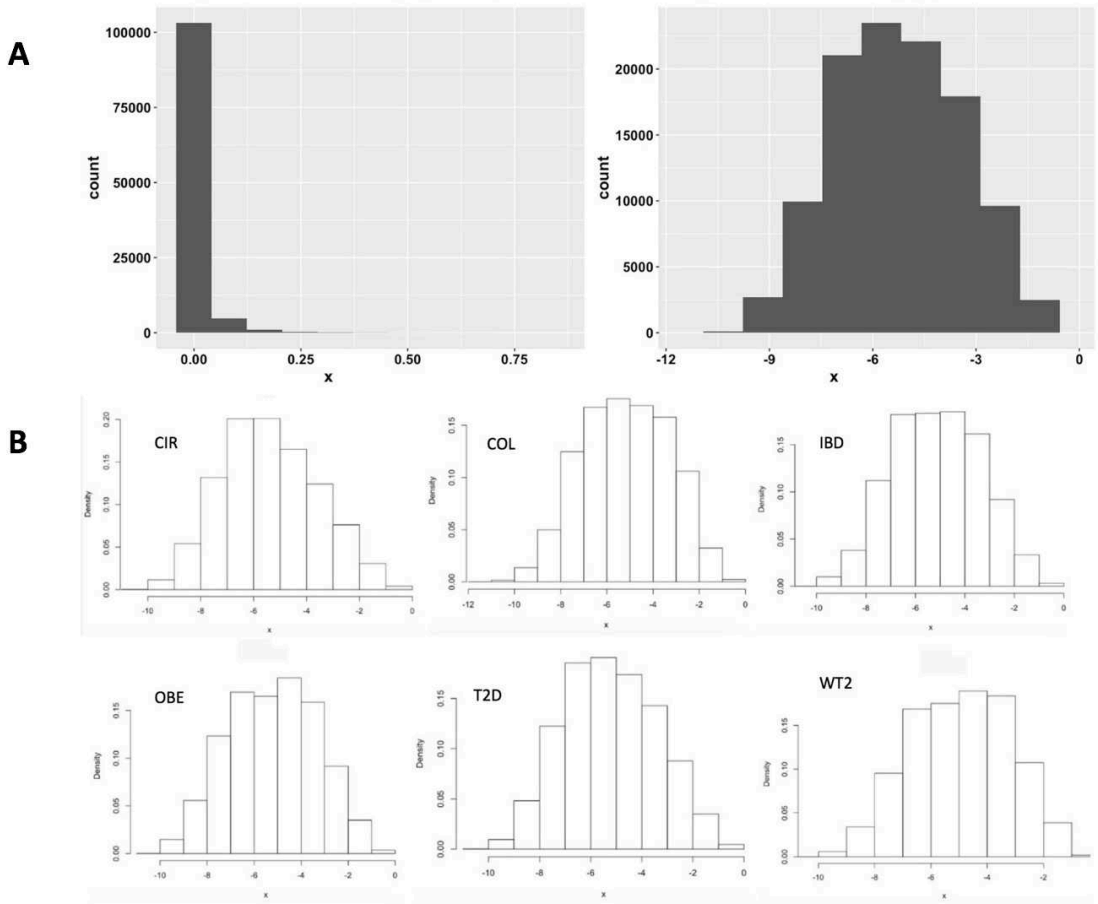

Figure III.2 *(A): Histogram of whole six datasets of group A (Left: original data, Right: Log-histogram (base 4)). (B): Log histogram of each dataset*

performance in terms of time and error. The binning method we used in the project is unsupervised binning which does not use the target (class) information. In this part, we use **EQual Width binning** (EQW) with ranging **[Min, Max]**. We test with $k = 10$ bins (for color distinct images, and gray images), width of intervals is $w = 0.1$, if Min=0 and Max = 1, for example.

III.4.1.1 Binning based on abundance distribution

On Figure III.2A, on the *left* we show the histogram of the original data. The original data follow the zero-inflated distribution what is typical for metagenomic data [186]. On the *right* we show the log-transformed distribution of the data where the logarithm (base 4) is taken, and we notice that the data are more normally-distributed. In the logarithmic scale, the width of each break is 1 being equivalent to a 4-fold increase from the previous bin. As observed from Figure III.2B, histograms of six datasets of group A with the logarithmic scale (base 4) share the same distributions. From our observations, we propose a hypothesis that the models will perform better with such breaks owning values of breaks from $0, 10^{-7}, 4*10^{-7}, 1.6*10^{-6}, ..., 0.0065536, 1$. The first break is from 0

Model	Bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN	EQW	0.859	0.737	**0.847**	**0.684**	0.569	0.617	0.719
CNN	SPB	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
FC	EQW	0.582	0.622	0.775	0.648	0.494	0.608	0.622
FC	SPB	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760

Table III.2 – Performance comparison (in ACC) between EQW and SPB using gray images. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3

to 10^{-7} which is the minimum value of species abundance known in 6 datasets of group A, each multiplies four times preceding one. We called this binning "SPecies Bins" (**SPB**) in our experiments.

In order to evaluate the efficiency of SPB, we compare the performance between SPB and EQW in Table III.2. EQW bins are performed in a range of [**Min,Max**]. For each k fold, EQW indicates maximum and minimum values of original abundances in training set, then dividing $k = 10$ bins that have the equal width. The bins determined using training set, after that, we apply these bins to test set. We observed that the SPB outperforms the EQW in most of the datasets. Prominently, the EQW exhibits a great performance in CNN compared to FC model (see the models in IV.4.3). The EQW using CNN performs slightly better than the SPB on the OBE dataset.

III.4.1.2 Binning based on Quantile Transformation (QTF)

We test another approach to bin the data, based on a scaling factor which is learned in the training set and then applied to the test set. With different distributions of data, standardization is a commonly-used technique for numerous ML algorithms. Quantile TransFormation (QTF), a Non-Linear transformation, is considered as a strong preprocessing technique because of reducing the outliers effect. Values in new/unseen data (for example, test/validation set) which are lower or higher the fitted range will be set to the bounds of the output distribution. In the experiments, we use this transformation to fit the features' signal to a uniform distribution.

We also illustrate another scaler, a linear scaler, MinMaxScaler (MMS), for a comparison between scaler algorithms. This algorithm scales each feature to a given range with the formulas (III.3) and (III.4):

$$X_{std} = \frac{X - X.min}{X.max - X.min} \tag{III.3}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{III.4}$$

QTF and MMS implementations are provided from the scikit-learn library [16] in Python.

As shown in Table III.3, QTF transformation defeats MMS in most of cases. Comparing to EQW (in Table III.2), we see that MMS and EQW share a pattern in the results, but MMS outperforms EQW for FC model.

Model	Scaler	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN	MMS	0.862	0.733	**0.839**	**0.669**	0.617	0.599	0.720
CNN	QTF	**0.906**	0.768	0.821	**0.672**	0.660	0.659	0.748
FC	MMS	0.824	0.749	0.780	**0.659**	0.518	0.651	0.697
FC	QTF	**0.897**	0.771	**0.843**	**0.680**	0.666	0.687	0.757

Table III.3 – Performance comparison (in ACC) between MMS and QTF using gray images. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Models are detailed in Section IV.4.3

Model	#bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN	255	**0.907**	0.767	0.823	**0.671**	0.655	0.668	0.748
CNN	10	**0.906**	0.768	0.821	**0.672**	0.660	0.659	0.748
FC	255	**0.896**	0.768	**0.838**	**0.678**	0.666	0.684	0.755
FC	10	**0.897**	0.771	**0.843**	**0.680**	0.666	0.687	0.757

Table III.4 – Performance comparison (in ACC) between the chosen number of bins. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3

Another evaluation on the effect of selecting the number of bins in QTF is carried out in Table III.4. There are only some insignificant differences between them that come from CNN models on T2D, and WT2; and IBD datasets for FC.

III.4.1.3 Binary Bins

Beside the previous two methods described above, we also use the binary bins to indicate PResence/absence (PR), respectively corresponding to black/white (BW) values in subsequent images. Note that this method is equivalent to one-hot encoding.

III.4.2 Generation of artificial metagenomic images: Fill-up and Manifold learning algorithms

III.4.2.1 Fill-up

Images are created by arranging abundance/presence values into a matrix in a right-to-left order by row top-to-bottom. The order to arrange species can be either phylogenetic or random. As an example for a dataset containing 542 features (i.e. bacterial species) in the cirrhosis dataset, we need a matrix of 24×24 to fill up 542 values of species into this square. The first row of pixels is arranged from the first species to the 24th species, the second row includes from the 25th to the 48th and so on till the end. We use distinct colors in binning scale with SPB, QTF and PR to illustrate abundance values of species and black and white for presence/ absence, where white represents absent values (see examples at Figure III.3 and Figure III.4).

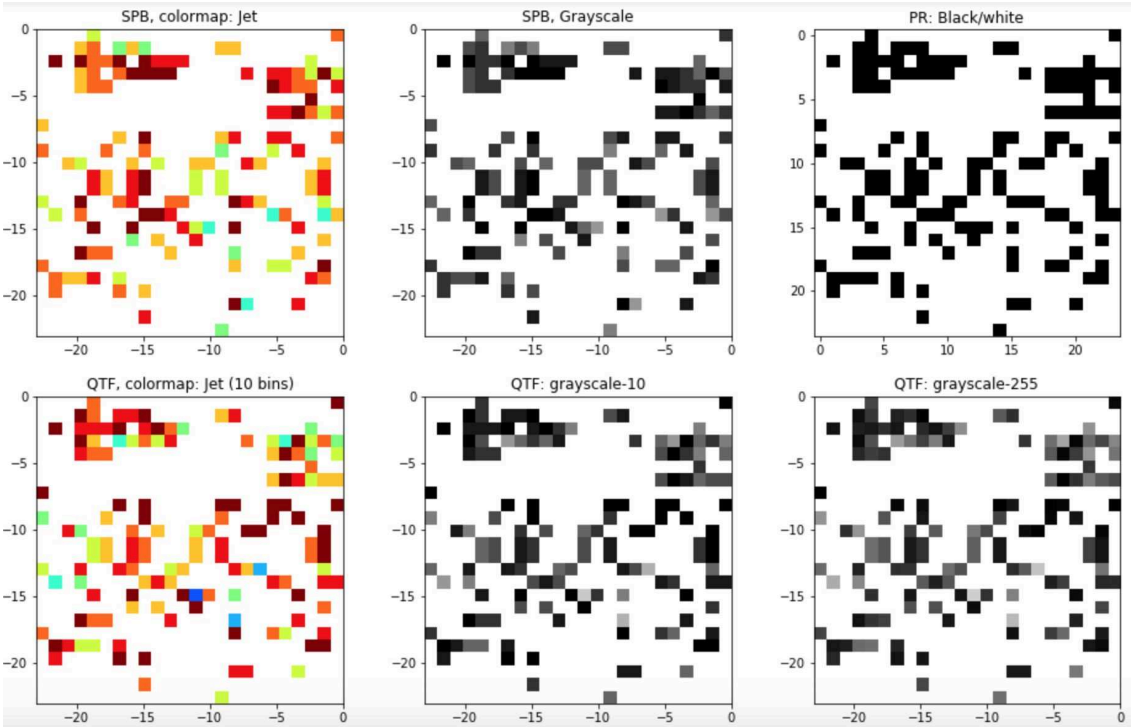

Figure III.3 *Examples of Fill-up on a sample of CIR dataset (images of 24×24), Top: Left-Right* ***SPB*** *with jet colormap,* ***SPB*** *with gray images, and* ***PR*** *with black/white images; Down: Left-Right:* ***QTF*** *(10 bins) with jet color images,* ***QTF*** *(10 bins) with gray images and* ***QTF*** *(255 bins) with gray images.*

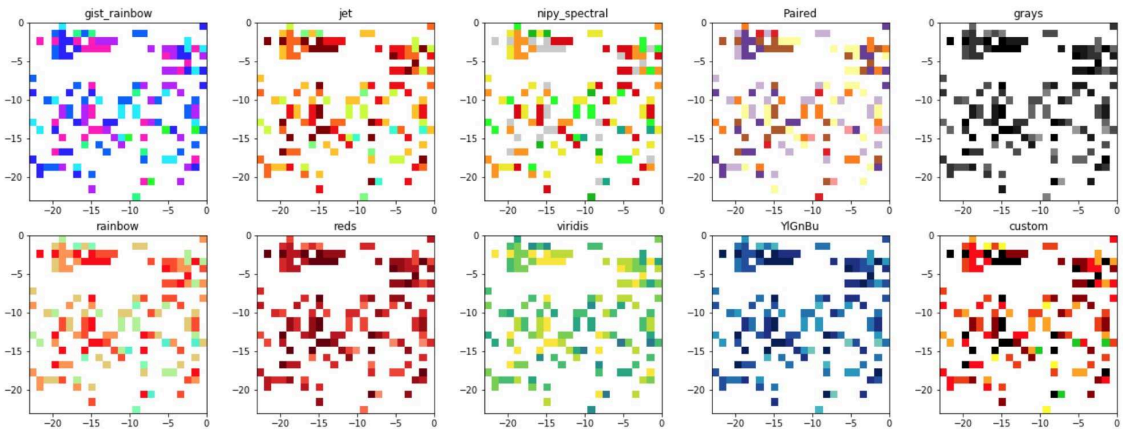

Figure III.4 *Examples of Fill-up on a sample of CIR dataset using* ***SPB*** *with 10 different colormaps.*

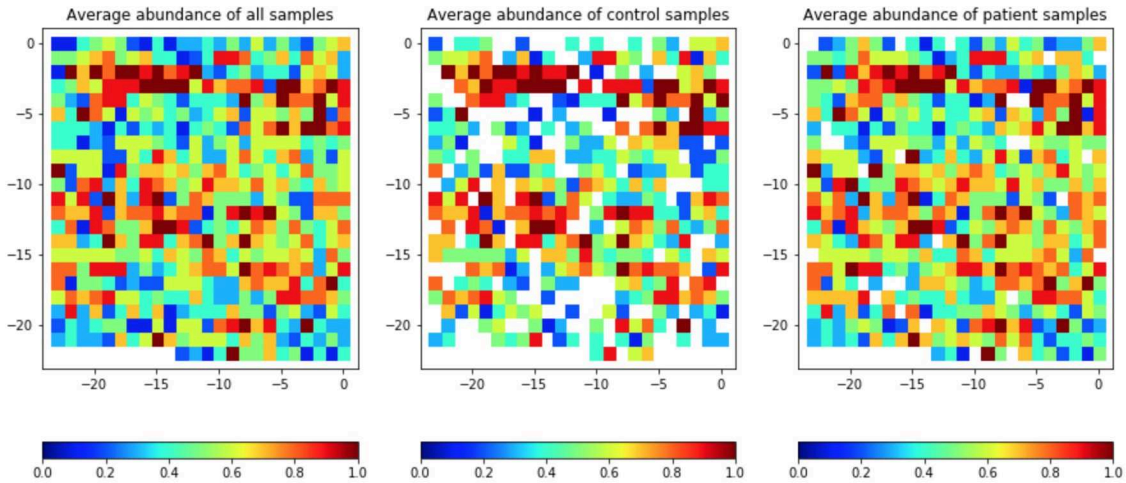

Figure III.5 *Average abundance of species visualized by Fill-up using jet colormap and phylogenetic ordering; relative species abundance binned by SBP*

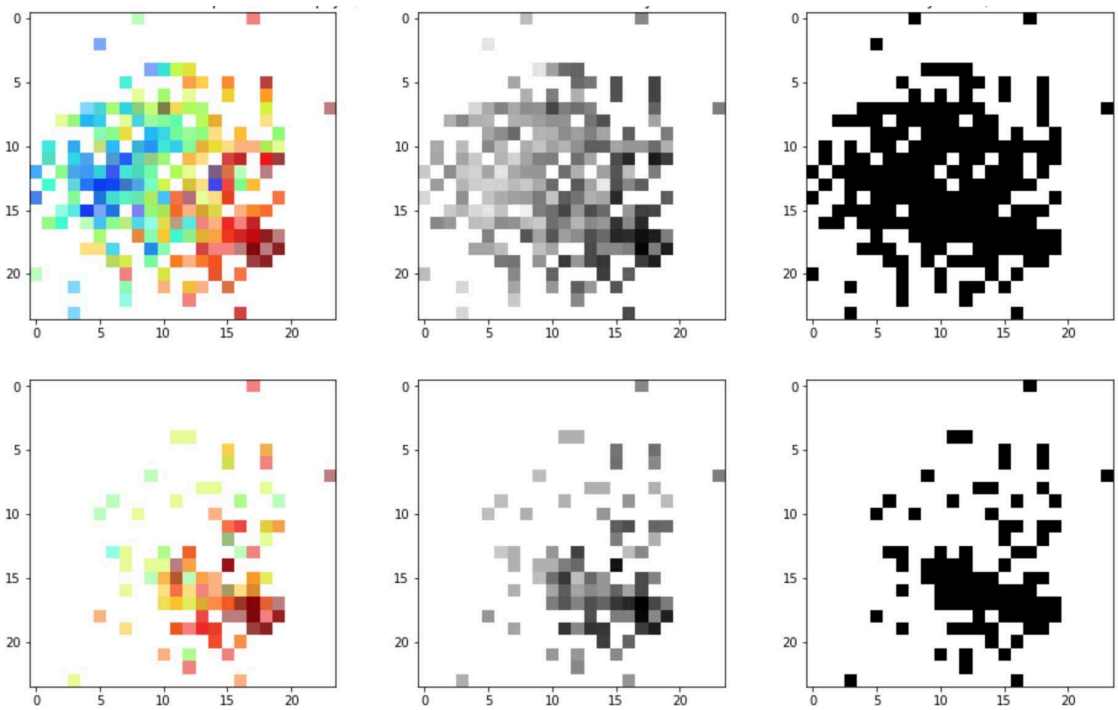

Figure III.6 *Examples of t-SNE representation, Left-Right (types of images): color images (using jet colormap), gray images, black/white images; Top-Down: Global t-SNE maps and images of samples created from the global t-SNE map*

Phylum name	**Color_index**	#species	**abundance**	
			Sample 78	Sample 93
Euryarchaeota	1	4	0.000000	0.000000
Acidobacteria	2	1	0.000000	0.000000
Actinobacteria	3	53	0.002369	0.001485
Bacteroidetes	4	100	0.485673	0.526229
Candidatus_Saccharibacteria	5	3	0.000000	0.000000
Chlorobi	6	1	0.000000	0.000000
Deinococcus_Thermus	7	1	0.000000	0.000000
Firmicutes	8	247	0.451369	0.305555
Fusobacteria	9	13	0.000890	0.013925
Proteobacteria	10	105	0.035651	0.152767
Spirochaetes	11	5	0.000000	0.000000
Synergistetes	12	3	0.000000	0.000000
Tenericutes	13	1	0.000000	0.000000
Verrucomicrobia	14	1	0.024047	0.000039
Ascomycota	15	4	0.000000	0.000000
Total		542	1.000000	1.000000

Table III.5 – Information on the number of species belonging to phyla in CIR dataset along with the index color and phylum abundance visualized in 2D or 3D in Figure III.7, III.8.

Figure III.5 shows 542-species abundance distribution using SBP. High abundance is dense at rows of $-3, -4, -11, -12$ (dark red areas) while rows of $0, -7$ contain many colors of blue what corresponds to a low abundance. As seen from the Figure III.5, there are some differences on the average abundance between control and patient samples. Control samples contain more white cells where abundance is close to 0. Furthermore, the row -9 of patient samples reveals more high abundance than control samples.

On Figure III.7, we show species distribution corresponding to phylum, class and order. The phylum Firmicutes which is colored in green (index 8, see Table III.5) contains the most species (247), following by Proteobacteria (index 10, yellow), and Bacteroidetes (index 4, the blue which has coordinates from -5 to -1) with 105 and 100 species, respectively. In order to illustrate the abundance into these 2-D figures, we plot them into Figure III.9 and Figure III.8 taking 2 samples of 78 (control) and 93 (patient) as an example. We exhibit phylum abundance in Figure III.8 while Figure III.9 reveals species abundance (see detail in Table III.5). Although Bacteroidetes (blue) only consists of 100 species, it shows the highest abundance in both samples. Verrucomicrobia is a significant difference between 2 these samples. In Figure III.8, we observe Verrucomicrobia as a small red dot in the right of Figure III.8A. Going deeply into species abundance in Figure III.9, we notice a yellow bar in patient (the red bar) is one of highest abundance in sample 93 while in sample 78, this species reveals a low abundance as a tiny yellow dot in the 3D plot.

Table III.6 exposes results of Fill-up with random ordering and phylogenetic sorting. We can see that there is no significant difference between 2 types of arranging species

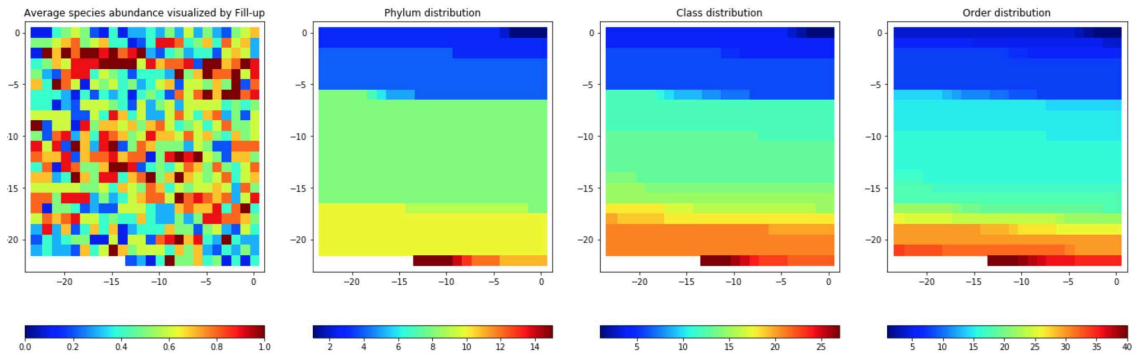

Figure III.7 *Visualization based on Fill-up of average species abundance visualized by phylogenetic ordering (binned by SPB) and distributions of 16 Phylum OTUs, 27 Class OTUs, and 40 Order OTUs on CIR dataset (The names and indexes of color of phylum, class, order taxa are listed in Table III.5, B.1, B.2). Each OTU is presented by a distinct color*

	Bins	Model	Color	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
Fill-up ∗	PR	CNN	BW	0.880	0.747	**0.837**	**0.674**	0.673	0.724	0.756
Fill-up	PR	CNN	BW	0.880	0.763	**0.841**	**0.669**	0.666	0.667	0.748
Fill-up ∗	SPB	CNN	grays	**0.903**	0.768	**0.840**	**0.673**	0.666	**0.739**	0.765
Fill-up	SPB	CNN	grays	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
Fill-up ∗	PR	FC	BW	0.867	0.744	**0.834**	**0.664**	0.649	0.699	0.743
Fill-up	PR	FC	BW	0.863	0.735	**0.842**	**0.672**	0.656	0.712	0.747
Fill-up ∗	SPB	FC	grays	0.887	0.758	**0.853**	**0.677**	0.646	0.711	0.755
Fill-up	SPB	FC	grays	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760

Table III.6 – Comparison between random sorting (Fill-up∗) and phylogenetically ordering. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3.

for CIR, OBE and T2D. We calculated paired t-tests for comparison, and found that IBD in phylogenetic order shows a significant improvement compared to sorting species randomly for SPB and gray images, while for other approaches, phylogenetic arrangement also shows a slight increase in performance. COL also reveals similar patterns to IBD, while WT2 works well on random sorting when using CNN.

III.4.2.2 Visualization based on dimensionality reduction algorithms

Besides the Fill-up, we also use dimensionality reduction algorithms to visualize features. Data visualization is a good way to see the structures of data so that with the visualized shape of data probably enhances the learning to get a better performance. High-dimensional datasets such as metagenomics are usually very difficult to explore and to interpret corresponding machine learning models. However, we can try to investigate data while plotting them in two or three dimensional space. A key idea of this approach

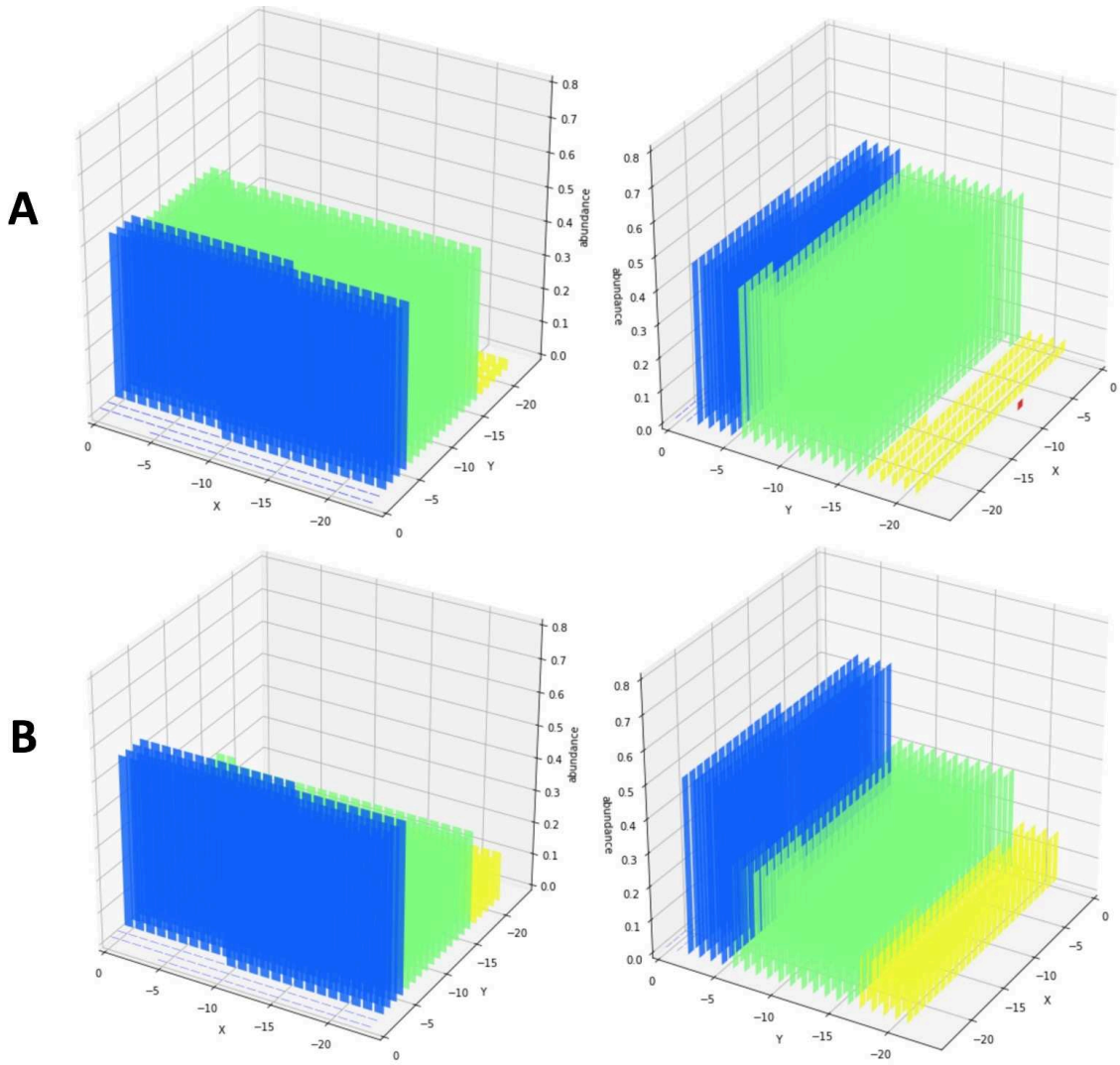

Figure III.8 *Visualization in 3D of phylum distribution consisting of coordinates and phylum abundance from (A) a control sample (78th sample) and (B) a patient sample (93rd sample) of CIR dataset with 2 different viewing angles. Each phylum is presented by a color. Abundance and name of phylum are in detail in Table III.5*

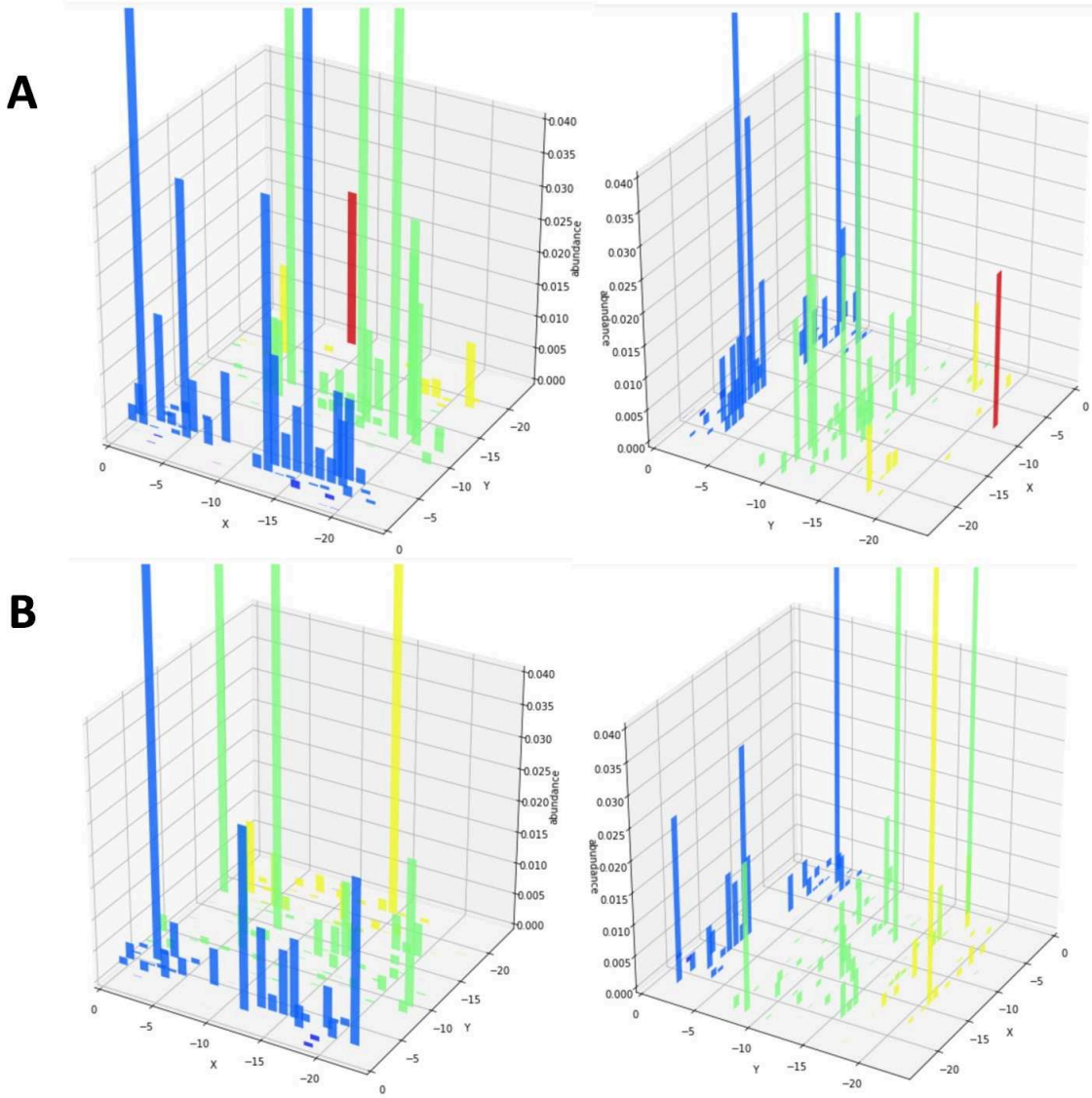

Figure III.9 *Visualization in 3D of phylum distribution including positions and species abundance from (A) the control sample (78th) and (B) the patient sample (93rd) of CIR dataset with 2 different viewing angles. Each phylum is presented by a color. Abundance and name of phylum are in detail in Table III.5*

is that we can find the structures of high-dimensional data shaped in 2D images where deep learning techniques for images can be applied to improve prediction results. Besides unsupervised dimensionality reduction as above, we also apply a supervised version with Linear Discriminant Analysis (LDA) [148, 146, 145, 147] algorithm which fits a Gaussian density to each class, and supposing that all classes reveal the same covariance matrix. Each group can be a higher relative level of a group of organisms in a taxonomic hierarchy such as genus, family, order and so on.

Figure III.10 visualizes average species abundance of 232 samples in CIR dataset with different methods such as MDS, PCA, Isomap, T-SNE, NMF and LDA with various labels using different levels of OTUs. We observe that the t-SNE performs well in both SPB (using original abundance) and QTF (transformed data). For original abundance, t-SNE reveals the best among considered algorithms. MDS seems to take the second place, while the others exhibit poor visualization with numerous overlapped points. For transformed abundance, t-SNE meets a problem with large values (red and yellow points in the maps) while MDS, Isomap and PCA show similar visualizations that formed as a ball. NMF apparently extends points to corners. While MDS, PCA, T-SNE, show the lowest-highest from left-right, isomap performs oppositely. LDA in QTF that uses family or genus as labels show better results compared to SPB. This proves that LDA is very sensitive to outliers. Observed abundance distribution with various levels of OTUs are shown on Figure III.11 and Figure III.12. The authors stated in [153] that "taxa relative abundance values are distributed in a non-normal way", we also see that abundance distribution seems to not follow taxa information at higher levels than species. When we provide the labels for the supervised dimensionality reduction algorithm LDA with various levels, only the result with labels with the lowest level, genus, obtains some relative clear clusters of genus to distinct a vast of species. We suppose that the newly created images illustrated on Figure III.10, can be of a help to improve the classification. We hope that arranging together species that have the similar magnitude of abundance is informative, and the produced synthetic images are a reasonable input for a deep learning algorithm.

In order to apply dimensionality reduction methods to visualize high-dimensional data, first, the features of all samples in training set from raw data are transformed by a dimensionality reduction algorithm such as PCA, NMF, Random Projection, t-SNE, MDS, Isomap, LLE, LDA, or Spectral Embedding (SE) into a global map. This map is then used to generate images for training and testing data. Each species is represented as a point on the map and only species that are present are shown either in abundance or presence using the same color schemes as above. All of nonlinear manifold learning approaches such as Isomap, LLE, and so on. depend on the size of neighborhoods around each point [88] with parameters of K and ϵ and finding an optimal value for numerous situations have been still investigating. The global maps are built based on training sets with the number of neighbors to consider for each point of 5 (this equivalents to the parameter of perplexity for t-SNE), learning rate = 100, and the number of iterations set to 300 (see examples of t-SNE images in Figure III.6). The packages for manifold learning approaches are available in scikit-learn (Python library) [16].

As seen from our experiments with the dimensionality reduction on the CIR dataset with different manifold learning algorithms at Figure III.13 and III.14, for SPB on original data, t-SNE gives the best visualization with less overlapped points while there are numerous hidden points in the others. In order to reduce the effects of overlapping problem,

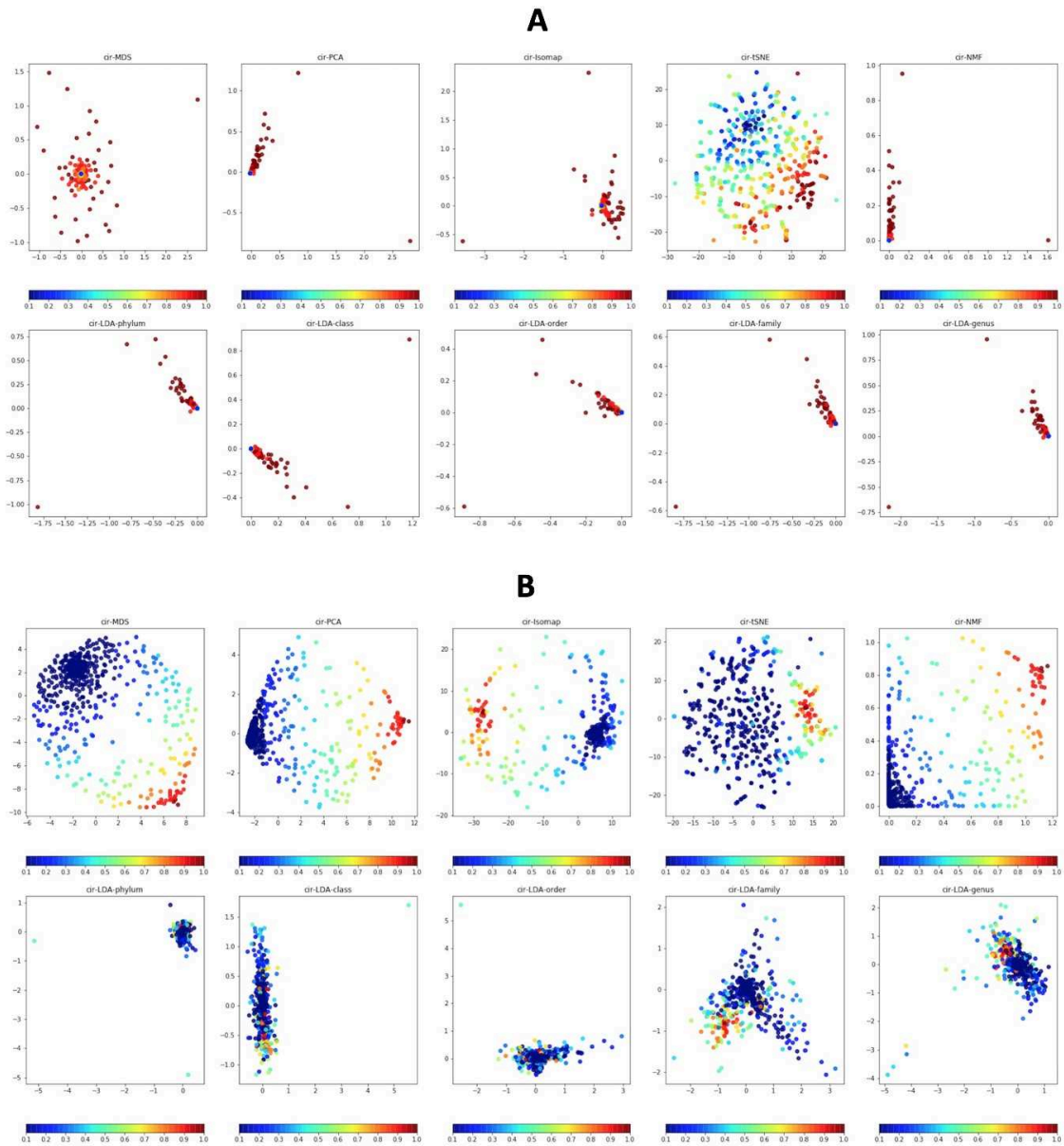

Figure III.10 *Average abundance map of CIR dataset visualized by MDS, PCA, Isomap, t-SNE, NMF, and LDA with SPB (A) and QTF (B). For LDA (the second rows of each A and B), we use different level OTUs for label inputs (Left-Right: phylum, class, order, family, genus). Colors are presented in the maps showing the magnitude of average species abundance with the lowest abundance colored dark blue and the largest revealed red.*

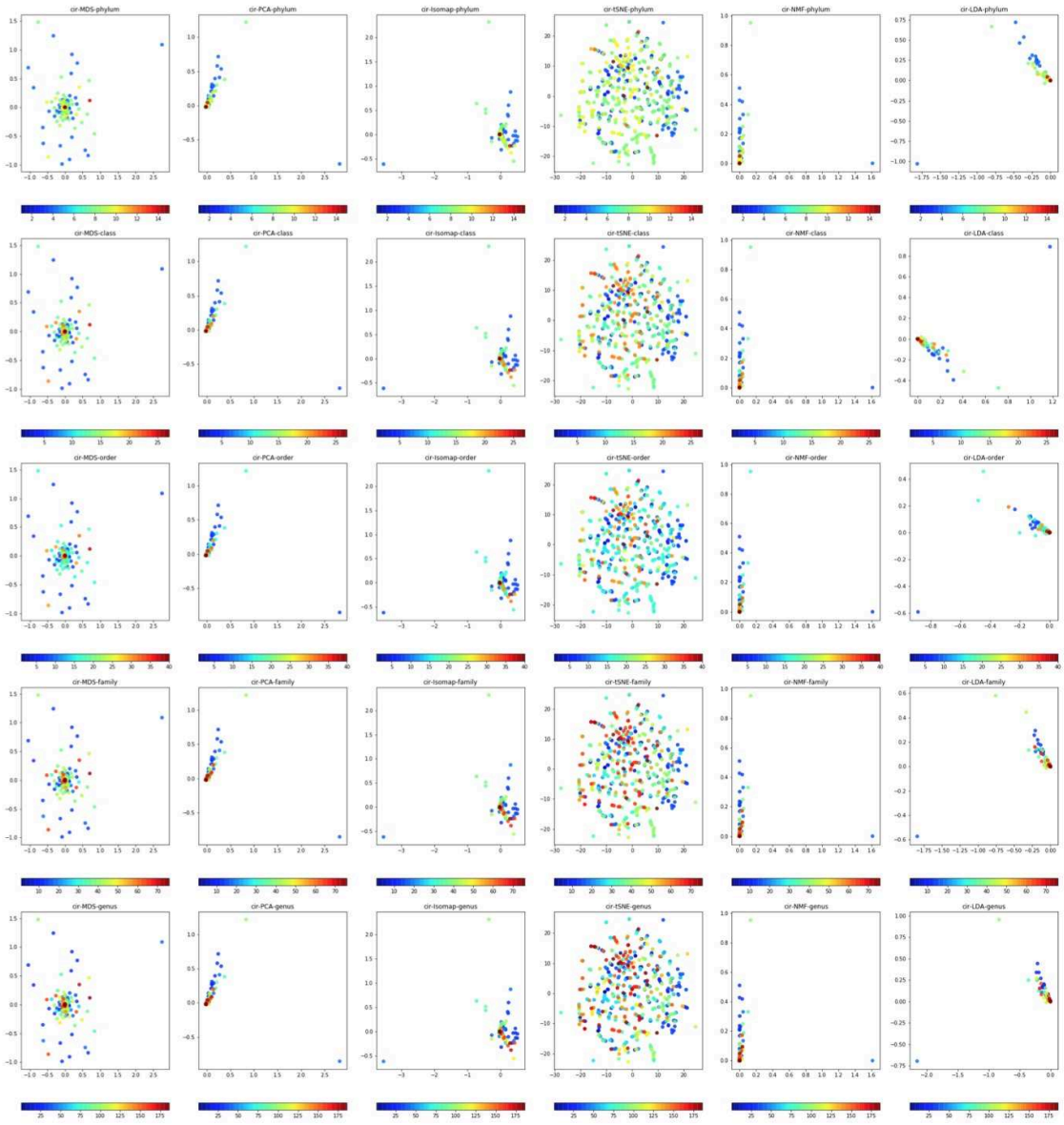

Figure III.11 *Visualization Comparison among of MDS, PCA, Isomap, t-SNE, NMF, and LDA with different levels of taxa using original abundance. Rows of 1, 2, 3, 4, and 5 correspond to phylum, class, order, family, genus, respectively.*

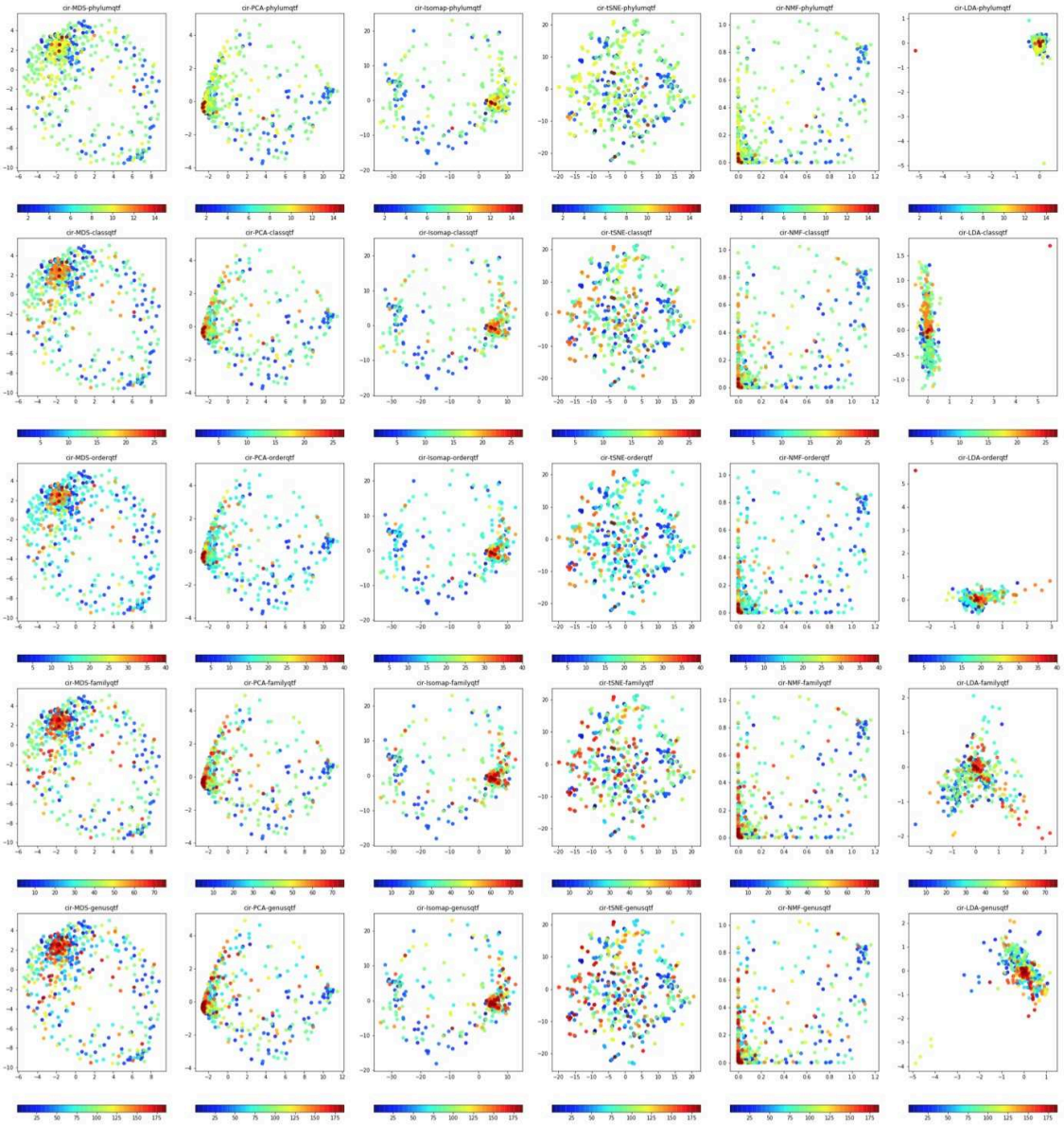

Figure III.12 *Visualization Comparison among oof MDS, PCA, Isomap, t-SNE, NMF, and LDA with different levels of taxa using transformed abundance using QTF. Rows of 1, 2, 3, 4, and 5 correspond to phylum, class, order, family, genus, respectively.*

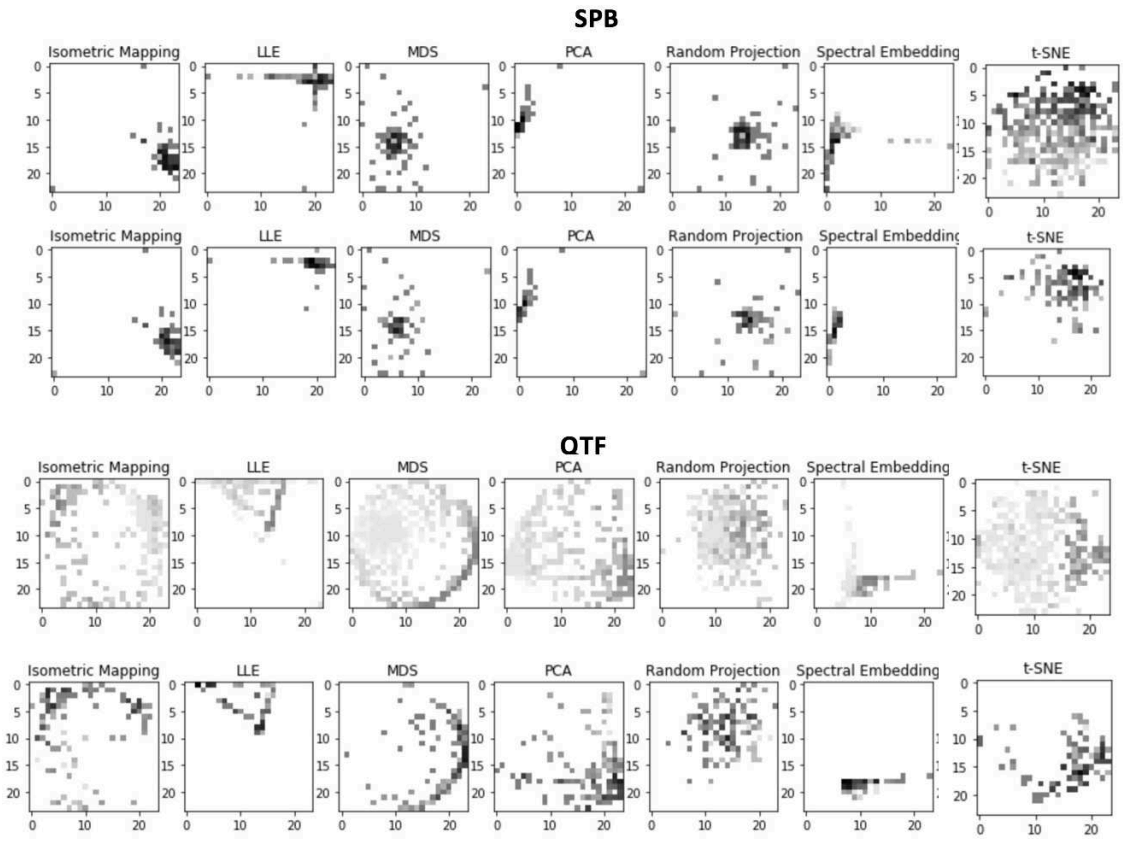

Figure III.13 *Examples of Global maps (1st and 3rd rows) and samples (2nd and 4th rows) created from manifold methods (Isomap, LLE, MDS, PCA, Random projection, Spectral Embedding, and t-SNE) with SPB and QTF bins using CIR dataset. Top: Global maps. Down: images of samples created from the global maps above.*

we use the "transparency trick" that suffers from a large amount of overlapped points. The transparency parameter can be fixed in the interval between 0 and 1, and it indicates the transparency of the markers plotted. A value of 0 determines all points are opaque while 1 means that transparency is not applied. In our experiments, for visualizations by manifold learning methods, we set the transparent value to 0.5.

Similar to observations from visualizations, Tables III.7, III.9, and III.8 also reveal the same results. For data without transformation, using FC model, t-SNE obtain the best on all considered datasets while the others reveal the similar results together with all average performances on 6 datasets being under 0.7. In addition, t-SNE also achieves the best performance (5 out of 6 datasets) and MDS takes the second place while other methods show poor performances. NMF exhibits the worst among all considered algorithms. There is a slight difference when we change the learning rate of t-SNE.

For BW images (PR bins), we obtain the same pattern to SPB. Furthermore, t-SNE defeats the other on most of datasets.

However, for data transformed using QTF with a uniform distribution, Isomap pro-

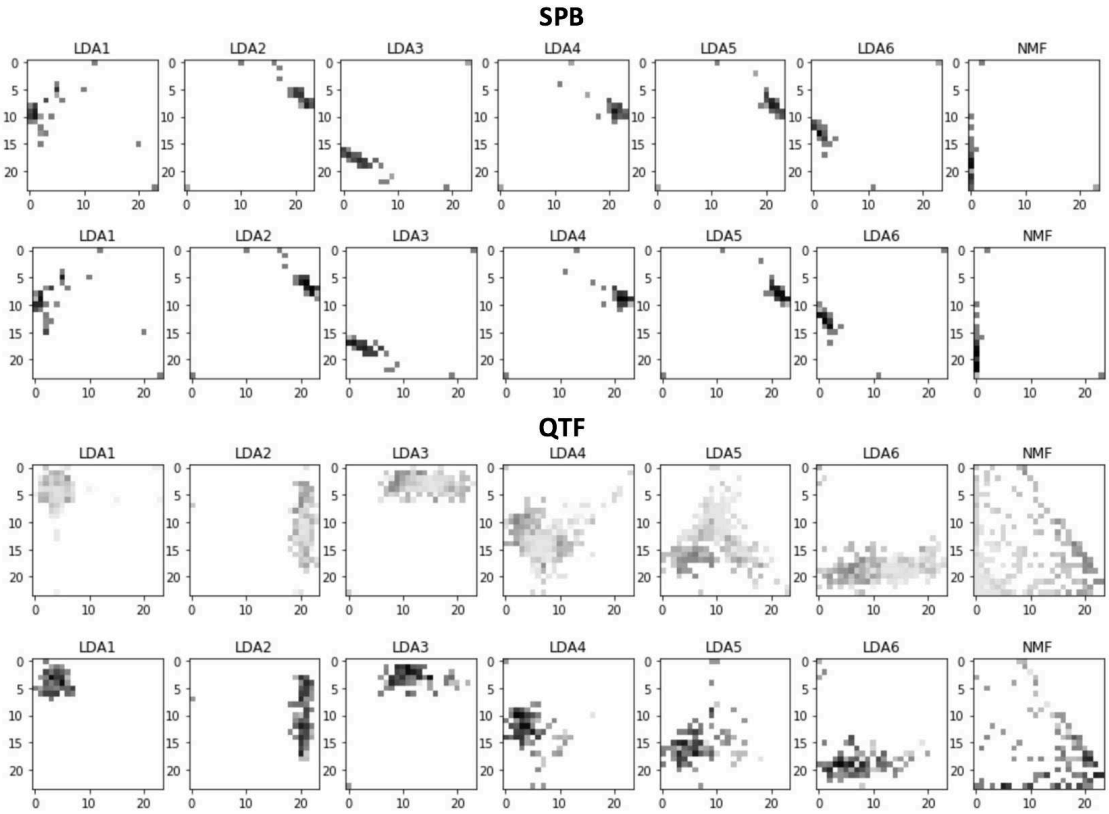

Figure III.14 *Examples of Global maps (1st and 3rd rows) and samples (2nd and 4th rows) created from manifold methods (NMF and LDA with level different labels of OTUs) with SPB and QTF bins using CIR dataset. Top: Global maps. Down: images of samples created from the global maps above. LDA1, LDA2, LDA3, LDA4, LDA5, LDA6 correspond to the labels of Kingdom, Phylum, Class, Order, Family and genus, respectively.*

duces better results than t-SNE. Two of them are LLE and SE show the worst performance in both CNN and FC models. We also illustrate results of LDA with 6 levels of OTUs for labels. As seen from Table III.11, the levels of Order, Family, Genus that are near to species reveal better performances. In Table III.9 and III.7, we show the results with the level of Family which show the best result in III.11.

Interestingly, MDS with SPB seems like to be efficient for IBD with average ACC of 0.848 (p-value < 0.0005) while, using QTF, t-SNE, NMF and PCA perform the best on T2D and Isomap seems like to be an appropriate solution to OBE and WT2. Furthermore, it is worth noting that LDA obtains the best performance on CIR.

III.4.3 Colormaps for images

We test a variety of colormaps, namely *gist_rainbow, jet, nipy_spectral, paired, grays, rainbow, reds, viridis, and YlGnBu* for images. These colormaps are classified into two types including single-hue such as Gray and Reds, and multi-hue colormaps consisting

Algorithms	Model	Bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
ISOMAP	CNN	SPB	0.818	0.695	0.806	**0.656**	0.583	0.590	0.691
LDA	CNN	SPB	0.832	0.679	0.804	**0.652**	0.585	0.588	0.690
LLE	CNN	SPB	0.798	0.749	0.807	0.651	0.631	0.636	0.712
MDS	CNN	SPB	0.850	0.724	**0.848**	**0.656**	0.607	0.677	0.727
NMF	CNN	SPB	0.758	0.660	0.777	0.649	0.541	0.621	0.668
PCA	CNN	SPB	0.778	0.651	0.790	**0.651**	0.561	0.609	0.673
RD_PRO	CNN	SPB	0.856	0.686	**0.827**	**0.660**	0.595	0.658	0.714
SE	CNN	SPB	0.799	0.754	0.796	**0.659**	0.621	0.679	0.718
TSNE	CNN	SPB	0.885	0.765	**0.828**	**0.676**	0.655	0.681	0.748
TSNE∗	CNN	SPB	0.877	0.761	0.809	**0.679**	0.664	0.686	0.746
ISOMAP	FC	SPB	0.762	0.684	0.775	0.647	0.495	0.639	0.667
LDA	FC	SPB	0.678	0.651	0.775	0.648	0.495	0.608	0.643
LLE	FC	SPB	0.731	0.744	0.775	**0.660**	0.535	0.646	0.682
MDS	FC	SPB	0.800	0.690	0.776	**0.651**	0.533	0.660	0.685
NMF	FC	SPB	0.532	0.654	0.775	0.648	0.494	0.648	0.625
PCA	FC	SPB	0.616	0.649	0.775	0.648	0.495	0.626	0.635
RD_PRO	FC	SPB	0.782	0.696	0.775	**0.652**	0.518	0.651	0.679
SE	FC	SPB	0.743	0.742	0.776	**0.652**	0.504	0.667	0.681
TSNE	FC	SPB	0.873	0.773	0.791	**0.678**	0.632	0.708	0.743
TSNE∗	FC	SPB	0.871	0.770	0.790	**0.678**	0.633	0.702	0.741

Table III.7 – Performance comparison (in ACC) of manifold methods using gray images and SPB. The significant results (compared to MetAML) are reported in **bold**. TSNE∗ use a learning rate of 100 while this parameter of TSNE set to 200. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3

of other colormaps. We also propose another colormap based on jet combining to black, namely *custom*. In Table III.10 and III.12, we shows the results with colormaps supported in Python library, Matplotlib [82] and the proposed colormap. Visualizations of 10 various colormaps with 10 distinct colors (10 bins) for each colormap are illustrated in Figure III.15.

For SPB bins, we see that *jet, custom*, and *grays* perform encouraging results both FC and CNN model. Significantly, *jet* obtains 3 significant results and perform approximately to other datasets both FC and CNN model compared to MetAML while *grays* shares the same results. Notably, *grays, jet* and *rainbow* appear as good solutions for CNN where the performance of CNN is greater than FC. Besides, *YlGnBu* also is a promising colormap with high performances for both CNN and FC. On another hand, *viridis* shows lower performance in both FC and CNN. In addition, the colormaps of *nipy_spectral* and *paired* reveal the worst performance among all considered colormaps.

For QTF bins, *viridis* obtains dominant results for CNN with the best performance among all considered colormaps, while the others expose a decrease in performance of CNN compared to FC models. The single-hue colormaps (such as *grays* and *reds*), and perceptually uniform colormap (*viridis*) yield better results compared to the others. Be-

Algorithms	Model	Bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
ISOMAP	CNN	PR	0.601	0.639	0.808	0.649	0.572	0.593	0.644
LLE	CNN	PR	0.729	0.709	0.799	0.647	0.628	0.646	0.693
MDS	CNN	PR	0.724	0.640	**0.836**	**0.658**	0.614	0.682	0.692
NMF	CNN	PR	0.540	0.634	0.785	0.647	0.530	0.613	0.625
PCA	CNN	PR	0.552	0.622	0.786	0.650	0.558	0.586	0.626
RD_PRO	CNN	PR	0.716	0.635	0.825	0.652	0.588	0.654	0.678
SE	CNN	PR	0.736	0.711	0.787	**0.664**	0.619	0.673	0.698
TSNE	CNN	PR	0.865	0.701	0.815	**0.659**	0.640	0.680	0.727
TSNE∗	CNN	PR	0.862	0.712	0.815	**0.664**	0.660	0.672	0.731
ISOMAP	FC	PR	0.605	0.634	0.791	0.647	0.539	0.610	0.638
LLE	FC	PR	0.717	0.716	0.782	**0.653**	0.590	0.658	0.686
MDS	FC	PR	0.723	0.644	0.806	**0.659**	0.587	0.662	0.680
NMF	FC	PR	0.531	0.624	0.778	0.648	0.502	0.613	0.616
PCA	FC	PR	0.559	0.625	0.781	0.648	0.527	0.623	0.627
RD_PRO	FC	PR	0.708	0.652	0.804	**0.654**	0.571	0.655	0.674
SE	FC	PR	0.726	0.709	0.803	**0.658**	0.575	0.655	0.688
TSNE	FC	PR	0.859	0.713	0.818	**0.662**	0.633	0.681	0.728
TSNE∗	FC	PR	0.857	0.710	0.813	**0.662**	0.640	0.688	0.728

Table III.8 – Performance comparison (in ACC) of manifold methods using gray images and PR. TSNE∗ uses a learning rate of 200 while this parameter of TSNE set to 100. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3

sides, such single-hue colormaps share the same pattern in performance. Interestingly, our proposed colormap outperforms jet and takes the second place among all performance of CNN using QTF bins.

As observed from Figure III.15 and descriptions in [82], we can see clearly that *gist_rainbow* and *paired* colormaps are not perceptually uniform; these palettes tend to appear several "kinks" where the apparent color changes quickly over a short range of values while viridis stands out for its large perceptual range. The characteristics of viridis enables us to leverage the available color space as much as possible while maintaining uniformity. The results suggest that multi-hue colormaps can provide improved discrimination and classification. We propose to use *jet*, and *grays* for SPB and *viridis* for QTF bins that shaped as a uniform distribution.

We also examined some colormaps on visualizations based on dimensionality reduction methods (see Table III.13). As observed from Table III.13, t-SNE reveals an better overall performance in SPB compared to QTF. Color images seems to improve slight results, but the performance of CNN is lower than FC, while, in contrast to gray images, CNN achieve better FC using SPB. It is worthy to note that t-SNE using QTF with jet colormap reach to significant results on T2D while Fill-up performs poorly on this dataset. Other results on LDA, NMF, MDS, PCA, and Isomap are presented in Table C.4, C.6, C.7, C.5, respectively, of Appendices.

Algorithms	Model	Bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
ISOMAP	CNN	QTF	0.889	0.743	0.815	**0.691**	0.657	0.712	0.751
LDA	CNN	QTF	**0.900**	0.746	0.809	**0.671**	0.661	0.675	0.744
LLE	CNN	QTF	0.846	0.719	0.806	**0.661**	0.615	0.648	0.716
MDS	CNN	QTF	0.881	0.741	0.816	**0.685**	0.667	0.670	0.743
NMF	CNN	QTF	0.883	0.742	**0.826**	**0.665**	**0.683**	0.635	0.739
PCA	CNN	QTF	0.890	0.744	0.815	**0.661**	0.678	0.686	0.745
RD_PRO	CNN	QTF	0.883	0.733	0.824	**0.673**	0.660	0.675	0.741
SE	CNN	QTF	0.866	0.697	0.791	**0.665**	0.592	0.673	0.714
TSNE	CNN	QTF	0.865	0.737	0.824	**0.659**	0.674	0.662	0.737
TSNE∗	CNN	QTF	0.875	0.747	0.809	**0.664**	0.672	0.651	0.736
ISOMAP	FC	QTF	0.871	0.742	0.793	**0.688**	0.646	0.689	0.738
LDA	FC	QTF	0.873	0.736	0.792	**0.679**	0.631	0.687	0.733
LLE	FC	QTF	0.850	0.714	0.776	**0.660**	0.564	0.627	0.699
MDS	FC	QTF	0.889	0.767	0.782	**0.668**	0.637	0.685	0.738
NMF	FC	QTF	**0.893**	0.756	0.790	**0.676**	0.655	0.686	0.743
PCA	FC	QTF	**0.893**	0.749	0.796	**0.669**	0.654	0.684	0.741
RD_PRO	FC	QTF	0.876	0.751	0.785	**0.678**	0.624	0.694	0.734
SE	FC	QTF	0.851	0.702	0.776	0.651	0.512	0.675	0.695
TSNE	FC	QTF	0.867	0.771	0.791	**0.663**	0.646	0.701	0.740
TSNE∗	FC	QTF	0.873	0.764	0.787	**0.664**	0.647	0.686	0.737

Table III.9 – Performance comparison (in ACC) of manifold methods using gray images and QTF. The significant results (compared to MetAML) are reported in **bold**. TSNE∗ uses a learning rate of 200 while this parameter of TSNE set to 100. The average performance of six datasets is computed and shown in the last column (AVG). Models are described in Section IV.4.3

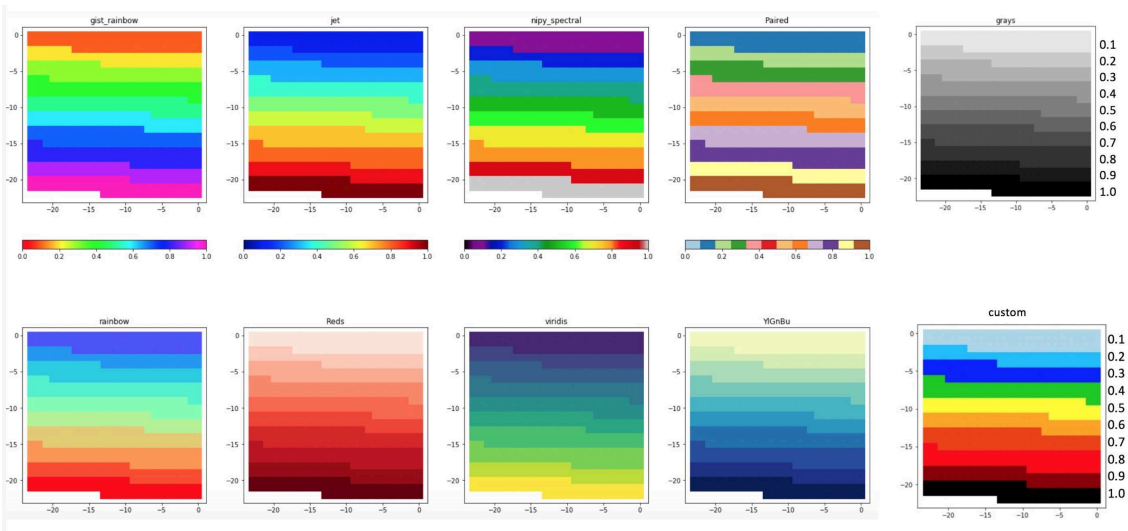

Figure III.15 *Visualization of 10 bins using a variety of colormaps to visualize features on CIR dataset (24×24 images).*

Model	Color	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN	custom	**0.897**	0.782	**0.857**	**0.673**	0.658	0.707	0.762
CNN	gist_rainbow	0.886	0.791	**0.835**	**0.666**	0.653	0.670	0.750
CNN	Grays	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
CNN	jet	**0.903**	0.798	**0.863**	**0.681**	0.649	0.713	0.768
CNN	nipy_spectral	0.872	0.757	0.811	0.652	0.647	0.657	0.733
CNN	Paired	0.838	0.728	0.798	0.655	0.637	0.622	0.713
CNN	rainbow	**0.893**	0.775	**0.865**	**0.668**	0.635	0.712	0.758
CNN	reds	0.890	0.773	**0.851**	**0.664**	0.660	0.704	0.757
CNN	viridis	0.880	0.765	0.823	**0.662**	0.645	0.651	0.738
CNN	YlGnBu	**0.895**	0.774	**0.852**	**0.666**	0.654	0.704	0.758
FC	custom	**0.904**	0.805	**0.835**	**0.676**	0.647	0.709	0.763
FC	gist_rainbow	**0.894**	0.791	**0.834**	**0.679**	0.638	0.695	0.755
FC	Grays	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760
FC	jet	**0.905**	0.794	**0.837**	**0.679**	0.659	0.713	0.764
FC	nipy_spectral	0.880	0.753	0.822	**0.665**	0.627	0.732	0.747
FC	Paired	0.831	0.714	0.806	**0.666**	0.598	0.684	0.716
FC	rainbow	**0.899**	0.759	**0.847**	**0.677**	0.631	0.705	0.753
FC	reds	**0.893**	0.782	**0.845**	**0.676**	0.647	0.731	0.762
FC	viridis	**0.895**	0.764	**0.841**	**0.683**	0.636	0.704	0.754
FC	YlGnBu	**0.895**	0.776	**0.847**	**0.676**	0.650	0.723	0.761

Table III.10 – Performance (in ACC) comparison of a variety of colormaps using Fill-up and SPB. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are described in IV.4.3

	Level of Label	**Cir**	**Col**	**Ibd**	**Obe**	**T2D**	**Wt2**	AVG
Lda	1	0.810	0.709	0.774	**0.654**	0.504	0.648	0.683
Lda	2	0.838	0.762	0.788	**0.660**	0.512	0.681	0.707
Lda	3	0.868	0.739	0.793	**0.667**	0.581	0.673	0.720
Lda	4	0.873	0.746	0.796	**0.669**	0.603	0.688	0.729
Lda	5	0.873	0.736	0.792	**0.679**	0.631	0.687	0.733
Lda	6	0.867	0.749	0.786	**0.667**	0.577	0.707	0.726

Table III.11 – Performance (in ACC) comparison of different levels of OTUs using LDA, QTF, and FC model. Levels of 1,2,3,4,5,6 correspond to the labels of Kingdom, Phylum, Class, Order, Family and genus, respectively. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

Model	Color	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN	Custom	**0.903**	0.776	0.818	**0.660**	0.679	0.654	0.748
CNN	gist_rainbow	**0.896**	0.777	**0.827**	0.655	0.671	0.661	0.748
CNN	Grays	**0.906**	0.768	0.821	**0.672**	0.660	0.659	0.748
CNN	jet	**0.909**	0.769	0.825	0.648	0.665	0.650	0.744
CNN	nipy_spectral	0.878	0.718	0.807	0.647	0.602	0.623	0.713
CNN	Paired	0.854	0.720	0.796	**0.668**	0.652	0.679	0.728
CNN	rainbow	**0.896**	0.767	0.816	0.646	0.664	0.665	0.743
CNN	reds	**0.901**	0.762	0.818	**0.659**	0.676	0.669	0.747
CNN	viridis	**0.897**	0.781	**0.837**	**0.659**	0.664	0.690	0.755
CNN	YlGnBu	**0.902**	0.760	0.822	**0.656**	0.666	0.682	0.748
FC	Custom	**0.896**	0.789	**0.831**	**0.665**	0.673	0.683	0.756
FC	gist_rainbow	0.887	0.783	**0.828**	**0.661**	0.639	0.678	0.746
FC	Grays	**0.897**	0.771	**0.843**	**0.680**	0.666	0.687	0.757
FC	jet	**0.903**	0.779	**0.845**	**0.661**	0.673	0.640	0.750
FC	nipy_spectral	0.872	0.697	0.822	**0.657**	0.591	0.677	0.719
FC	Paired	0.836	0.728	**0.829**	**0.663**	0.613	0.726	0.733
FC	rainbow	**0.906**	0.782	**0.845**	**0.660**	0.667	0.657	0.753
FC	reds	**0.898**	0.788	**0.841**	**0.664**	0.656	0.694	0.757
FC	viridis	0.887	0.765	**0.853**	**0.657**	0.640	0.711	0.752
FC	YlGnBu	**0.905**	0.784	**0.844**	**0.664**	0.661	0.698	0.759

Table III.12 – Performance (in ACC) comparison of a variety of colormaps using Fill-up and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

III.5 Closing remarks

In this chapter, we introduced **Met2Img** framework with several dimensionality reduction and projection into 2D. We discussed the way to generate synthetic images from metagenomic abundance data. First, our numerical results show that the Fill-up method outperforms the approaches based on dimensionality reduction because all points using Fill-up are visible while other dimensionality reduction techniques suffer from the fact that some points are overlapping in 2D. Second, the Fill-up approach integrates prior knowledge on the phylogenetic classification of the features. In addition, the images based on manifolds learning are more complex than the Fill-up images. However, we got some encouraging results using the dimensionality reduction methods on T2D and OBE data sets, that are in fact better than the ones obtained with the Fill-up.

We also discussed five binning methods that are conducted from three approaches: abundance distribution, transformation and one-hot encoding. The bins and images generation approaches are constructed and produced using training data only, thus avoiding over-fitting issues. For methods based on abundance distribution, we compare two ways including using original abundance (EQW) and utilizing logarithm (SPB). Transformation algorithms are also studied with linear (MMS) and non-linear approaches (QTF). As

	Model	Bins	Colors	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
TSNE	CNN	SPB	custom	0.882	0.777	0.821	**0.685**	0.661	0.678	0.751
TSNE	CNN	SPB	jet	0.887	0.779	0.819	**0.680**	0.650	0.685	0.750
TSNE	CNN	SPB	viridis	0.881	0.746	0.814	**0.664**	0.650	0.667	0.737
TSNE	CNN	SPB	grays	0.885	0.765	**0.828**	**0.676**	0.655	0.681	0.748
TSNE	FC	SPB	custom	**0.892**	0.784	**0.827**	**0.685**	0.634	0.694	0.752
TSNE	FC	SPB	jet	**0.893**	0.783	**0.826**	**0.687**	0.621	0.711	0.753
TSNE	FC	SPB	viridis	**0.894**	0.751	0.814	**0.676**	0.610	0.692	0.740
TSNE	FC	SPB	grays	0.873	0.773	0.791	**0.678**	0.632	0.708	0.743
TSNE∗	CNN	SPB	custom	0.885	0.760	0.804	**0.686**	0.671	0.669	0.746
TSNE∗	CNN	SPB	jet	0.886	0.769	0.802	**0.682**	0.658	0.685	0.747
TSNE∗	CNN	SPB	viridis	0.875	0.743	0.823	**0.675**	0.650	0.674	0.740
TSNE∗	CNN	SPB	grays	0.877	0.761	0.809	**0.679**	0.664	0.686	0.746
TSNE∗	FC	SPB	custom	0.885	0.772	0.825	**0.682**	0.632	0.691	0.748
TSNE∗	FC	SPB	jet	**0.890**	0.778	0.820	**0.689**	0.615	0.705	0.749
TSNE∗	FC	SPB	viridis	0.881	0.757	0.818	**0.683**	0.614	0.699	0.742
TSNE∗	FC	SPB	grays	0.871	0.770	0.790	**0.678**	0.633	0.702	0.741
TSNE	CNN	QTF	viridis	0.869	0.753	0.821	0.646	0.673	0.670	0.739
TSNE	CNN	QTF	custom	0.854	0.745	**0.827**	**0.656**	0.661	0.664	0.735
TSNE	CNN	QTF	jet	0.870	0.748	0.820	0.654	**0.690**	0.660	0.741
TSNE	CNN	QTF	grays	0.865	0.737	0.824	**0.659**	0.674	0.662	0.737
TSNE	FC	QTF	viridis	0.881	0.739	0.821	**0.661**	0.610	0.687	0.733
TSNE	FC	QTF	custom	0.860	0.759	**0.836**	**0.676**	0.614	0.668	0.735
TSNE	FC	QTF	jet	0.883	0.768	**0.825**	**0.670**	0.648	0.661	0.742
TSNE	FC	QTF	grays	0.867	0.771	0.791	**0.663**	0.646	0.701	0.740
TSNE∗	CNN	QTF	custom	0.869	0.747	0.816	0.643	0.646	0.655	0.729
TSNE∗	CNN	QTF	jet	0.878	0.746	0.811	**0.658**	**0.684**	0.648	0.737
TSNE∗	CNN	QTF	viridis	0.870	0.762	0.821	**0.658**	0.669	0.651	0.739
TSNE∗	CNN	QTF	grays	0.875	0.747	0.809	**0.664**	0.672	0.651	0.736
TSNE∗	FC	QTF	custom	0.872	0.761	**0.844**	**0.667**	0.607	0.661	0.735
TSNE∗	FC	QTF	viridis	0.878	0.741	0.812	**0.658**	0.604	0.675	0.728
TSNE∗	FC	QTF	jet	0.889	0.770	**0.825**	**0.660**	0.638	0.650	0.739
TSNE∗	FC	QTF	grays	0.873	0.764	0.787	**0.664**	0.647	0.686	0.737

Table III.13 – Performance (in ACC) comparison of a variety of colormaps using t-SNE. TSNE∗ uses a learning rate of 200 while this parameter of TSNE set to 100. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

demonstrated in results, QTF and SPB are promising binning methods for metagenomic data.

A diverse range of colormaps are investigated. The *jet, rainbow, viridis* are good choices, but CNN often underperform FC for color images, while *grays* usually yields better performance in CNN.

Chapter IV

Deep Learning for Metagenomics

Contents

Abstract: Deep learning (DL) techniques have shown unprecedented success when applied to images, waveforms, and text. Generally, when the sample size (N) is much bigger than the number of features (d), DL often outperforms other machine learning (ML) techniques, often through the use of Convolutional Neural Networks (CNN). However, in many bioinformatics fields (including metagenomics), we encounter the opposite situation where d is significantly greater than N. In these situations, applying DL techniques would lead to severe over-fitting. Here we aim to improve classification of various diseases with metagenomic data through the use of CNN. The proposed **Met2Img** approach relies on taxonomic and dimensionality reduction embeddings to transform abundance data into "synthetic images" presented in Chapter III. In this chapter, we investigate various architectures of CNN to perform classification tasks on the proposed visualizations.

We applied our approach to 25 benchmark data sets divided into 5 groups (A,B,C, D and E) including more than 5000 metagenomic samples. Our results show significant improvements over the state-of-the-art algorithms (Random Forest (RF), Support Vector Machine (SVM)). We observe that the integration of phylogenetic information alongside abundance data improves the classification. The proposed approach is not only important in classification setting but also allows to visualize complex metagenomic data. The Met2Img is implemented in Python.

IV.1 Introduction

High throughput data acquisition in the biomedical field has revolutionized research and applications in medicine and biotechnology. Also known as "omics" data, they reflect different aspects of systems biology (genomics, transcriptomics, metabolomics, proteomics, etc.) but also whole biological ecosystems acquired through metagenomics. There is an increasing number of datasets which are publicly available. Different statistical methods have been applied to classify patients from controls[57] and some have also performed meta-analyses on multiple datasets [18]. However, exploring omics data is challenging, since the number of features d is very large, and the number of observations N is small. Up to now, the most successful techniques applied to omics datasets have been mainly Random Forest (RF), and sparse regression.

In this chapter, we evaluated all proposed representations on six metagenomic datasets in group A, which reflect bacterial species abundance and presence in the gut of diseased patients and healthy controls. Then, we apply our method to additional datasets of colorectal cancer (group B), additional datasets with *genus* abundance (group C), and datasets on gene families abundance (group D). Since DL performs particularly well for image classification, we focused in this work in the use of CNN applied to images. For this purpose, we first searched for ways to represent metagenomic data as "synthetic" images (see Chapter III) and second applied CNN techniques to learn representations of the data that could be used next for classification purposes.

Here, we present the **Met2Img** framework based on *Fill-up* and manifold learning methods (such as *t-SNE* [3]) to visualize features into images. Our objectives were to propose efficient representations which produce compact and informative images, and to prove DL techniques as efficient tools for prediction tasks in the context of metagenomics. Our contribution is multi-fold:

- We propose a *visualization approach for metagenomic data* (see Chapter III) which shows significant results on 5 out of 6 datasets (except for colorectal cancer (COL)) compared to MetAML [18].

- Although, Met2Img shows not so good results for COL in group A compared to shallow learning algorithms such as RF, the framework reveals significant improvements compared to analysis in [37] (using SVM) and 4 out of 5 datasets for RF model when increasing the number of considered features.

- We illustrate that the proposed method not only performs competitively on species abundance data but also shows significant improvements compared to the state-of-

the-art, Ph-CNN, on genus-level taxonomy abundance with six various classification tasks on IBD dataset published in [11].

- The results on gene families abundance also expose a potential benefits in term of accuracy and execution time. The performance on the data reduced dimension gives a reasonable result compared to the original data although the dimension is reduced to more 2,000 times. Furthermore, gene families abundance in very compact images also demonstrates high efficiency with 3 significant results compared to RF.

- The experiments show that CNN outperform standard shallow learning algorithms such as RF and SVM. This proves that deep learning is a promising ML technique for metagenomics.

The chapter is organized as follows. Related work on machine learning for metagenomics and literature review on Convolutional Neural Networks are presented in IV.2. In Section IV.3, we describe five groups consisting of 25 considered datasets in details. The proposed methods are introduced in Section IV.4 including the proposed architectures for CNN are conducted from datasets in group A. We show our results in Section IV.5. In this section, we evaluate all representations on six datasets in group A including 1D and 2D data, and apply algorithms such as LIME [29], Saliency Maps [113, 34] and Gradient-weighted Class Activation Mapping (Grad-CAM [33]) to find explanations for the prediction tasks. Furthermore, an analysis on gene abundance (group D) is also taken in account using dimensionality reduction algorithms and very highly compressed images. Additionally, we illustrate the performance of our approach on five datasets in group B related to colorectal cancer disease with species-level abundance and another additional datasets consisting of six classification tasks on IBD with genus-level abundance (group C). We also compare Met2Img to selbal [194] on datasets with read counts (group E). Finally, conclusion is presented in the Section IV.6.

IV.2 Related work

IV.2.1 Machine learning for Metagenomics

The development of complexity and variability of metagenomics has been increasing in recent years. This leads to high requirements for computations. An increase of use of modern machine learning to solve the problems of metagenomics is more and more extended. There are five important metagenomics problems that were presented in [48] including Operational Taxonomic Unit (OTU)-clustering [107, 95, 108], binning, taxonomic profiling and assignment [94], comparative metagenomics [103, 102, 100, 101] and gene prediction [104, 105, 106]. These problems are related to three types of machine learning tasks such as classification, clustering and dimensionality reduction. These three types are listed in Figure IV.1. A large number of studies have attempted to propose and develop computational methods based on machine learning applying to metagenomics. In fact, machine learning is considered as a good methodology to explorer patterns and build predictive models for classification in metagenomics. Nowadays, machine learning has been providing numerous promising tools for designing predictive models to perform prediction tasks on biological data.

	Representations	Learning	Applications example	Tools example
Classification/ Regression	**Instances**	*k*-NN	binning, OTU clustering	DOTUR
		Support Vector Machines	gene prediction, Comparative MG	MetaDistance
	Linear models	linear regression	binning	Tetra
		logistic regression	gene prediction, Comparative MG	MetaGene, MetaPhyl
		large scale linear model	taxonomic assignment	Vowpal Wabbit
	Decision trees	Random Forest	taxonomic assignment	16S Classifier
	Neural networks	neural networks	gene prediction	Orphelia
	Network	Hidden Markov Model	gene prediction	FragGeneScan
		Interpolated Markov Model	binning, gene prediction	Glimmer-MG, SCIMM
Dimension reduction	**Linear combinations of features**	PCA	binning	CONCOCT
		SVD	binning	LSA
Clustering	**Means**	*k*-means	binning	SCIMM, MetaCluster
	Medoids	*k*-medoids	binning	MultiBin
	Dendrogram	hierarchical clustering	OTU clustering	ESPRIT, MOTUR
	Mixtures / Soft partitions / Likelihoods	Gaussian mixture models	binning	CONCOCT
		Bayesian clustering / stochastic search clustering	OTU clustering, binning	CROP,BACDNAS, BEBaC, LikelyBin
		spectral clustering	binning	CompostBin
	NA	greedy heuristic clustering	binning, OTU clustering	DNAClust, USEARCH

Figure IV.1 *Some examples of approaches based on machine learning to metagenomics. The figure is provided from [48].*

The authors in [48] also provided general principles in machine learning fundamentals. A typical machine learning program includes experience (data), task (in the form of an output of the algorithm), and objective (formed as performance measurement of a given output. Flach in [44] pointed out some requirements to build a program to learn how to revolve a given problem. First of all, data for learning needs to be mapped to features formed representation as input. A *model* receives the input and maps to an output. Some example data (called as *training data*) is provided to a learning algorithm to attempt to indicate an optimal model from a *hypothesis space*. In [4, 78] , a typical pipeline for classification tasks was suggested including preprocessing, then feature extraction, followed by classification or clustering. In feature extraction, raw data was transformed by mathematical transformation. A new shape of data helps predictors to perform prediction task easier. In some cases, if scientists have obtained knowledge from the domain, they are able to apply several ways such as heuristics and/or hand picking to select a subset from these features to be more informative to the models (feature selection). In the following step, one or more algorithms of classification (such as SVM, k-NN, neural networks) and/or clustering (k-means, spectral clustering) are applied to such features. Several other studies on metagenomics are introduced as below.

Figure IV.2 *A typical pipeline suggested in [4].*

The authors in [78] experimented using a deep belief network and recursive neural network and proved these as appropriate approach for metagenomics. In [18], Edoardo et al. proposed a computational framework for prediction tasks using quantitative microbiome profiles consisting of species-level relative abundances and presence of strain-specific markers. The potential benefits of microbiome analysis are promising tools for diagnostics. Additionally, indicating the relationships between the microbiome and specific diseases is crucial for next machine learning studies.

The computational framework (MetAML tool) [18] was designed based on machine learning classifiers such as support vector machines (SVMs), random forest (RFs), Lasso, and Elastic Net (ENet). Cross-validation and cross-study were to measure the prediction strength of metagenomic data and compare the generalization of the model between various studies. For cross-validation, accuracies were measured by 10-fold cross validation, repeated. A set of evaluations including Overall Accuracy (OA), precision, recall, F1, and area under the curve (AUC) and other tools such as receiver operating characteristics (ROC) curve plots, confusion matrices, plots of the most relevant features in addition to average relative abundances, and heat map figures were to measure the performance. In cross-validation, samples were divided randomly into k subsets (k folds) with the same size. A subset aimed to test and the training used the remaining k-1 subsets. This is repeated k times until each fold all used for testing with only one time. The performance was measured by a average value of k results of testing.

A metagenomic dataset usually contains bacterial DNA sequences obtained from an ecosystem such as the intestinal microbiome for instance. These sequences are usually transformed using Bioinformatics pipelines into a vector representing the abundance of a set of features (species or other taxonomic groups). One approach of conversion from sequences to vectors removed short and low quality score reads from the whole collection of sequences. Then, the remainder is clustered with CD-HIT [6], for example into non redundant representative sequences - usually called a catalog. Reads are typically aligned to the catalog and counted. Annotation algorithms are applied to the catalog allowing to infer abundance on different taxonomic levels. NAST [9] was used to align these representative sequences, then they were collected into taxonomic classifications by algorithms such as Ribosomal Database Project's or Naïve Bayes [13]. Other groups have also tackled the use of DL in classifying metagenomic data. For instance, some authors introduced approaches to design a tree like structure for data sets of metagenomics learned by neural networks. A study in [37] identified a group of bacteria that is consistently related to Colorectal Cancer (COL) and revealed potential diagnosis of this disease across multiple populations and classified COL cases with a SVM model using the seven COL-enriched bacterial species. However, applying DL to metagenomics is a challenge as the authors in

[78] stated: "deep learning approaches may not be suitable for metagenomics application" due to the lack of significant results. The authors in [11] introduced a novel DL approach, Ph-CNN, for classification of metagenomics using hierarchical structure of Operational Taxonomic Unit (OTU). The performance of Ph-CNN is promising compared to SVM, RF and a fully connected neural network. Derek Reiman et al. [83] proposed PopPhy-CNN with 2D matrix generated by embeddings based on phylogenetic tree for predicting host phenotype from metagenomic samples. This framework show the superior performance compared to RF, SVM, Lasso, and 1 baseline CNN-1D on three metagenomic datasets using relative abundance of microbial taxa. For the DNA sequencing process, deep learning have also been playing a important role to speed up this process with a review presented in [92]. In [187], the authors introduced DeepVirFinder including a convolutional layer, a pooling layer, a fully connected layer along with some dropout layers to predict viral sequences. Yoshua B. stated [45] that the performance of prediction tasks depends on selecting representations, numerous studies have attempted to propose efficient representations. Montavon et al [144] presented invariant representations of molecules to perform prediction tasks on atomization energy. In our study, we use representations based on images to perform classification tasks.

IV.2.2 Convolutional Neural Networks

Image classification based on ML has obtained great achievements and there are numerous studies attempting to propose architectures to improve the performance. In 1990, LeCun et al. in [84] established a modern framework of CNN. They also proposed LeNet-5 that used to classify handwritten digits [85]. However, because of lacking of large training data and limitations of computer processors, CNN were not able to show good performances on more complex problems [65] such as large-scale images and video recognitions. Since 2006, numerous studies have been proposing and developing a variety of techniques to overcome the challenges in training deep CNN. Alex et al. in [41] introduced a deep and big CNN (including 60 million parameters) which performed on a very large dataset with 1.2 million color images of 224x224 in ImageNet LSVRC-2010. The architectures consisted of 5 convolutional and 3 fully connected layers. The network achieved top-1 and top-5 error rates of 37.5% and 17% , respectively. The authors in [31] presented a novel technique (ZFNet) for visualizing feature maps through convolutional layers, and investigated how to improve the performance of Convolutional Neural Networks (convnet). A team at Google in 2014 proposed GoogLeNet [40], a very deep architecture of CNN with 22 layers that won the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). In [60], the authors presented how depth affects the performance of CNN (VGGNet). The authors designed deep architectures of CNN with very small convolutions (3x3) and achieved top-1 validation error and top-5 validation error were 23.7% and 6.8% ,respectively. The authors in [25] introduced a residual learning framework (ResNet) that was up to 152 layers but being less complex. ResNet won the 1st places on the ILSVRC 2015 and COCO 2015 competitions. In [97], authors introduced a set of information to design deep neural networks more efficient with an analysis of important metrics in practical applications including accuracy, memory footprint, parameters, operations count, inference time and power consumption. Among their most interesting results are the ideas to use batches for learning, and a hyperbolic relationship between accuracy and inference time where

a small increase in accuracy led to a significant growth in computational time. Hao Li et al. investigated the structure of neural loss functions. They found the effects of loss landscapes on generalization and introduced some visualizations for loss functions[98].

In the following sections, we describe several famous architectures in recent years such as AlexNet, ZFNet, GooLeNet, VGGNet and ResNet.

IV.2.2.1 AlexNet, ImageNet Classification with Deep Convolutional Neural Networks

Alex et al. in [41] presented a deep and big CNN (including 60 million parameters) which performed on a huge dataset with 1.2 million color images of 224×224 in ImageNet LSVRC-2010. The architectures consisted of 5 convolutional layers and 3 fully connected layers. They used some novel and unusual features for their architecture:

- ReLU Nonlinearity: The authors stated that the non-saturating nonlinearity $f(x) = max(0, x)$ (referred as Rectified Linear Units (ReLUs) [86]) makes the learning faster than the saturating nonlinearities such as Tanh $f(x) = tanh(x)$ or Sigmoid $f(x) = \frac{1}{(1+e^{-x})}$. An experiment that was to compare the training time between ReLU and Tanh showed that the network with ReLUs was six times faster than the network with Tanh.

- Using multi-GPUs: Due to limitations of GPU's memory (only 3GB for a single GTX 580 GPU) and the large training examples (1.2 million images of 224×224), one GPU was not enough to fit dataset. Hence, authors distributed the network to both 2 GPUs (Each GPU was installed half of the kernels/neurons and communicated in some specific layers).

- Local Response Normalization: The authors observed the local normalization scheme that was able to improve generalization.

- Overlapping Pooling: The traditional pooling was $z \times z$ with $step = z$ that usually appears in commonly CNN. In the proposed architecture, the authors used the pooling of 3x3 with step=2 that reduced the error rates by 0.4%.

In overall architecture (Figure IV.3), the first convolutional layer contained 96 filters of 3×11×11 with a stride of 4 pixels. The second convolutional layer included 256 kernels of 48×5×5 (on each GPU). After performing max pooling, the output of feature maps was 27×27. The third, fourth, fifth convolutional layer both had the size of 3×3 with the number of filter 384, 384, 256 respectively. There were 4096 neurons for each fully-connected layer. Softmax in the last layer was to classify 1000 class labels. To reduce overfitting for a huge convolutional neural network with a high number of parameters (60 million parameters), authors applied two primary approaches including Data Augmentation and Dropout [28, 55]. Two fully connected layers were applied dropout with probability 0.5. It took a double of iterations to converge when applying dropout.

Setting of learning: Stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005 were installed in the network. The weights were initialized in each layer with a zero-mean Gaussian distribution (standard deviation of 0.01), while the biases in the second, fourth, fifth convolutional layers and fully-connected

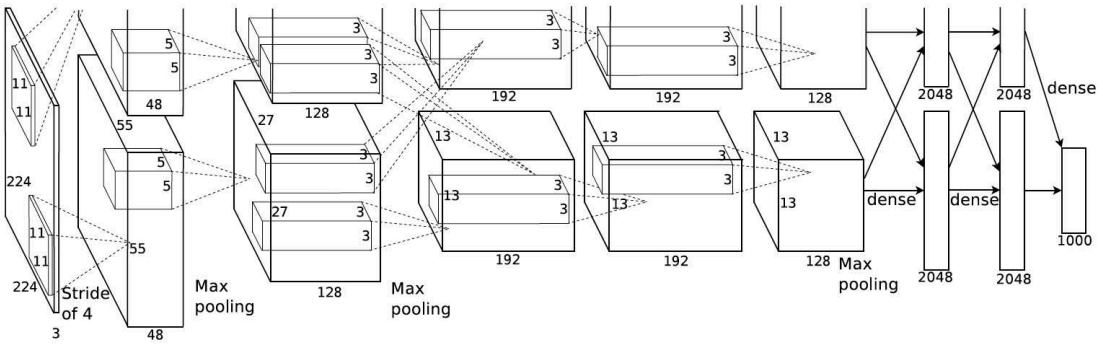

Figure IV.3 *The architecture of ImageNet on 2 GPUs with input including images of 3×224×224, and 1000 outputs of classes proposed in [41].*

layers were set to 1. They started the learning rate at 0.01, then dividing by 10 when the validation error did not decrease.

Results: The ILSVRC classification challenge is a prediction task to predict 1000 categories with 1.2 million images for training, 50,000 for validation and 100,000 for testing. The descriptions and evaluation of ILSVRC were described in detail in [40]. The highest scoring classifier was used to evaluate the performance with 2 reported results: the top-1 accuracy rate that compared the ground truth with the first predicted category, while the top-5 error rate compared the ground truth with the first 5 predicted categories. In the top-5 error rate, an image was considered as a correct prediction if the ground truth is among the top-5. ILSVRC used this measurement (the top-5) to rank architectures. The network achieved top-1 and top-5 error rates of 37.5% and 17%, respectively.

IV.2.2.2 ZFNet, Visualizing and Understanding Convolutional Networks

In this study [14], the authors presented a novel technique for Visualizing feature maps through convolutional layers, and investigated how to improve the performance of Convolutional Neural Networks (convnet). Deconvolutional Network (deconvnet) [7] was used to visualize the results from convolutions. Each convolutional layer attached a deconvnet to back to image pixels (Figure IV.4). The authors used unpool, rectify and filter to reconstruct the activity in the layer. These actions were looped until input pixel space is reached.

- Unpooling: In the convnet, the max pooling operation was non-invertible, so the authors recorded the locations where values was maximum within each pooling area to invert approximately. For the deconvnet, these switches put the reconstructions from the given layer to appropriate positions.

- Rectification: In the convnet, ReLU made values in feature maps to be always positive. In the deconvnet, the reconstructed signal was passed through a ReLU to get valid feature reconstructions.

- Filtering: In the convnet, feature maps were generated by convolutions of filters. In the deconvnet, transposed filters (were flipped vertically and horizontally) did not

apply to the output from the under layer, but applied to the rectified feature maps.

The architecture used to visualize in Figure IV.5 is similar to AlexNet [41]. The network took images of 224x224 as input with mini batch size of 128. The learning rate was initialized at 0.01 with momentum 0.9. Dropout rate of 0.5 was applied to fully connected layers. The results showed that with smaller stride and filter size enabled to produce distinctive features and fewer "dead" features. There was a relatively small difference when eliminating Layer 6, 7 (fully connected layers). Also, the network with removing two of the middle convolutional (Layer 3,4) gave a slight decrease in accuracy but removing both the layer 3,4 (middle layers) and Layer 6, 7 (fully connected layers) caused the network to perform a poor performance. Additionally, changing the number of units in fully connected layer or expanding the width of middle convolutional layer might improve the performance, but increasing both these leads to overfitting.

IV.2.2.3 Inception Architecture

The primary idea was finding the optimal local construction and repeating it spatially. Current incarnations of the Inception architecture used restrictedly kernels with sizes 1×1, 3×3 and 5×5 keep away from patch-alignment problems. In addition, pooling operations were useful in previous studies, so adding an alternative parallel pooling path in each such stage might obtain additional positive points. Higher layers explored higher abstractions, the concentrated region was expected to decline. Therefore, the ratio of 3×3 and 5×5 convolutions were suggested to rise when shifting to higher layers. However, this was very expensive, even for a modest number of 5x5 convolutions. This issue is easy to find when pooling units are added to the architectures. Therefore, the authors suggested to apply dimension reduction and projections where the requirement for computation might rise. They installed 1×1 convolutions to calculate reductions before convolutions of 3×3 and 5×5, and added Rectified linear activation for dual-purpose. An Inception network included these components stacked upon each other with max-pooling operations used infrequently with stride 2 to reduce the dimension of the grid. In order to use memory efficiency, using Inception modules should be only at higher layers while the others should preserve traditional convolutions. This architecture helped to increase the number of units at each stage while the complicated computation would not jump uncontrollably. Another advantage of this architecture was that data should be processed at different scales. After that, information was accumulated so that the next stage was able to explore features from various scales simultaneously.

IV.2.2.4 GoogLeNet, Going Deeper with Convolutions

A team of Google in 2014 proposed GoogLeNet [40], a very deep architecture of CNN with 22 layers that won the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). GooLeNet was a specific incarnation of Inception architecture that will be described as below.

GooLeNet was a specific incarnation of the Inception architecture that was used to submit to the ILSVRC14 competition. GooleNet was developed from Inception architecture as described in Figure IV.6. Rectified linear activation was installed in all the

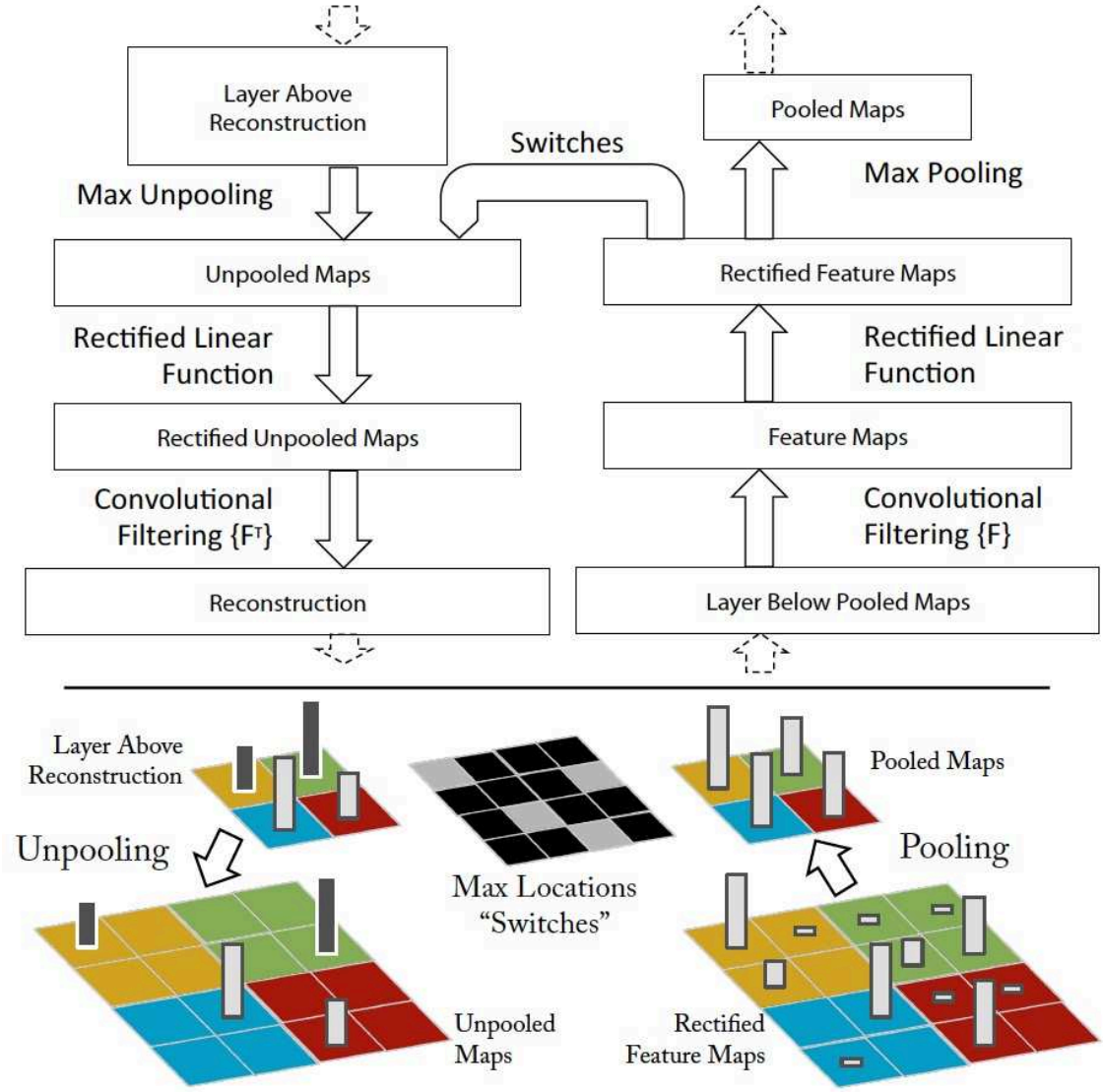

Figure IV.4 *Top: Attachment scheme between a deconvnet and a convnet. Bottom: An example of the unpooling operation. Max locations "switches" saved the location of local max when pooling operation performed in the convnet. The figure is provided from [14].*

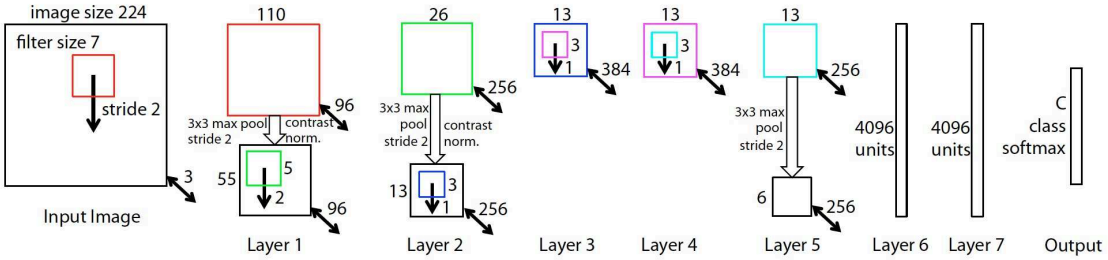

Figure IV.5 *The eight-layer architecture of ZFNet including 96 kernels of 7x7 (a stride of 2) in the first layer took images of 224×224 as input. After being convoluted by the first convolutional layer, the feature maps passed through by a rectified linear function, performed by max pooling of 3x3, then contrast normalized across feature maps (stride 2) to generate 96 feature maps of 55x55. Layers 2,3,4,5 looped the same operations as Layer 1. Layer 6, 7 were fully connected layers, and the final layer used softmax with C outputs (classes). The figure is provided from [14].*

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure IV.6 *GoogLeNet from Inception architecture introduced in [40].*

convolutions. The input has the size of 224x224 with RGB color channels. In the FigureIV.6, "3x3 reduce" and "5x5 reduce" showed the number of 1x1 kernels (with 128 filters for dimension reduction) in the reduction layer deployed before the 3x3 and 5x5 convolutions. The network included totally 27 layers (22 of 27 layers contained parameters) with 5 pooling layers. Additionally, the authors installed an average Pooling with 5x5 filter size and stride 3 before the classifier instead of Fully connected layers. Although, these fully connected layers were eliminated, a dropout layer with 70% ratio of dropped outputs still remained to use. A fully connected layer with 1024 units and rectified linear activation put before the classifier that used softmax loss to predict 1000 classes (see Figure IV.7).

Setting of learning, and results: The network used asynchronous stochastic gradient descent with 0.9 momentum [96], while the learning rate declined by 4% after each 8 epochs. GooLeNet won the challenge with a top-5 error of 6.67% on both the validation and testing data.

IV.2.2.5 VGGNet, very deep convolutional networks for large-scale image recognition

In this paper [60], the authors presented how depth affects the performance of CNN. The authors designed deep architectures of CNN with very small convolutions (3×3). They stated that the depth to 16-19 layers was able to achieve a considerable improvement.

Configurations: All configuration for layers used the same rules. The architecture included a stack of convolutional layers with filters of 3×3 (stride 1). The authors also used padding to border at input layer to preserve image's resolution after convolution. There were 5 max pooling layers that followed several convolutional layers with the size of 2×2 and stride 2. Three fully connected layers (the first two contained 4096 neurons and the last one performed the prediction task with 1000 classes), and the softmax layer followed the stack of convolutional layers. The networks used ReLU as activation functions for all layers.

The configurations were depicted in Figure IV.8. The size of kernels and the number of kernels in each layer were denoted as "conv< the size of filters > - < the number of filters>". For instance, *conv3-64* depicted that the convolutional layer contained 64 filters of 3×3. Figure IV.8 showed 5 configurations named from A to E. The configurations varied only the depth with 11 weight layers at Configuration A, and 19 weight layers at Configuration E. The widths of layers increased from 64 to 512 according to the depth after each max-pooling. Although, the networks were rather deep, the number of parameters was not high when comparing to some shallow architectures with larger widths and receptive fields [21]. The study in [22] also deployed small kernels but used for small images. Another interesting information that the authors pointed out is why we should use a stack of three 3×3 convolutional layers instead of a single 7×7 convolution. Firstly, they combined three non-linear rectification layers instead of a single one, so that the decision function to discriminate efficiently. Additionally, this helped to reduce the number of parameters. For example, we installed K filters for each convolutional layers of 3×3 (a stack of 3 convolutional layers of 3×3), this required $3(3^2K^2) = 27K^2)$ parameters while a single 7×7 convolution needs $7^2K^2 = 49K^2$.

Setting of learning, and results: The training procedure was similar to [41] using mini-batch with size of 256, gradient descent with momentum of 0.9, and was regularized

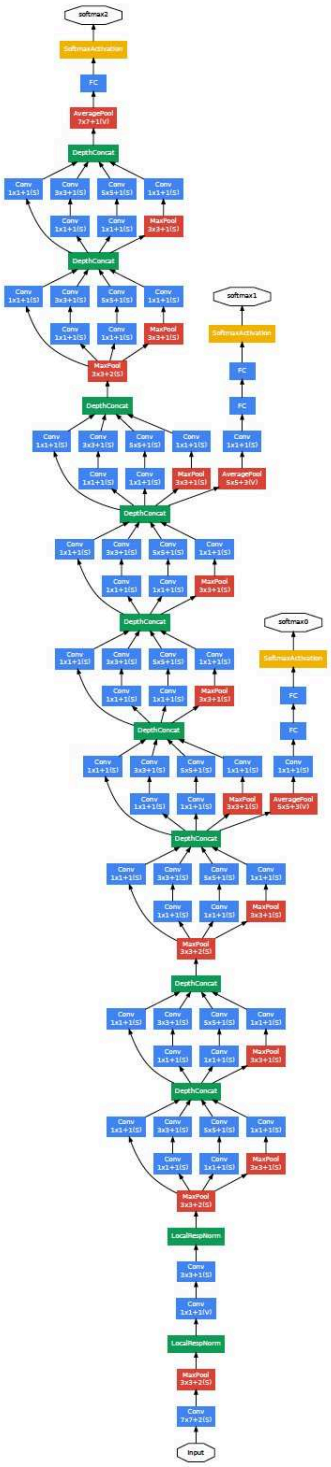

Figure IV.7 *The architecture of GoogLeNet with components presented in [40].*

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure IV.8 *The configurations of VGGNet. The figure is provided from [60].*

by weight decay [23] of 5.10^{-4}. Dropout was applied to the first two fully connected layers (with ratio of 0.5). The learning rate was initialized at 0.01, and declining by a factor of 10 after the accuracy of validation set did not increase. Comparing to [41], the network required less epoch to converge (74 epochs). The networks were implemented in C++ Caffe [24]. VGGNet achieved top-1 validation error and top-5 validation error were 23.7% and 6.8%, respectively.

IV.2.2.6 ResNet, Deep Residual Learning for Image Recognition

The authors in [25] introduced a residual learning framework (ResNet) that was up to 152 layers but being less complex. ResNet won the 1st places on the ILSVRC 2015 and COCO 2015 competitions.

It seemed that it was not simple to improve the learning by stacking more layers because of vanishing/exploding gradients [27]. However, the use of normalized initialization and intermediate normalization layers with batch normalization [26] helped the networks (using stochastic gradient descent with back-propagation [84]) to converge with the number of layers up to 10 layers. Additionally, the authors mentioned a degradation problem where the network with increased depth leading to saturation in accuracy, but then accuracy declined quickly. Overfitting was not the reason for degradation, a deep model get a higher error in training by adding more layers. The authors introduced ResNet to address the degradation problem.

The architecture included mostly kernels of 3×3 with 2 design principles: (1) the layers consisted of the same of filters for the same output feature size, (2) if the size of feature maps reduced to a half, the number of kernels increased to twice to preserve the time complexity per layer. The last layers of the architecture included a global average pooling layer and softmax with 1000 outputs (Figure IV.9). The network was implemented with batch size 256, the learning rate was initialized at 0.1 and reduced to 10 times if the error plateaued, and used to $60 * 10^4$ iterations. A weight decay of 0.0001 and a momentum of 0.9 was set in the learning. Remarkably, dropout was not implemented, but followed Batch Normalization [26]. ResNet with 152 layers reached 3.57% top-5 error on the test set.

IV.3 Metagenomic data benchmarks

We evaluated our method on 25 different datasets (see Table IV.1) divided into **five** groups (A, B, C, D and E). The 6 datasets in group A are described in III.3, in this part, we introduce additional datasets in the groups of B, C, D, and E.

Group B contains 526 metagenomic samples of colorectal cancer (COL) disease from Chinese, Austrian, American, German and French cohorts namely C1, C2, C3, C4, respectively. These cohorts were analyzed in [37]. Additional dataset (C5) was created by merging C1, C2, C3 and C4. These datasets includes shotgun metagenomic sequencing sequences (using Illumina Hiseq 2000/2500271 sequencing platform with similar sequencing depths (read length 100 bp, and target sequencing depth 5GB)) with 271 controls and 255 COL cases. Low quality sequences were eliminated using Trimmomatic v_0.36 [37, 192].

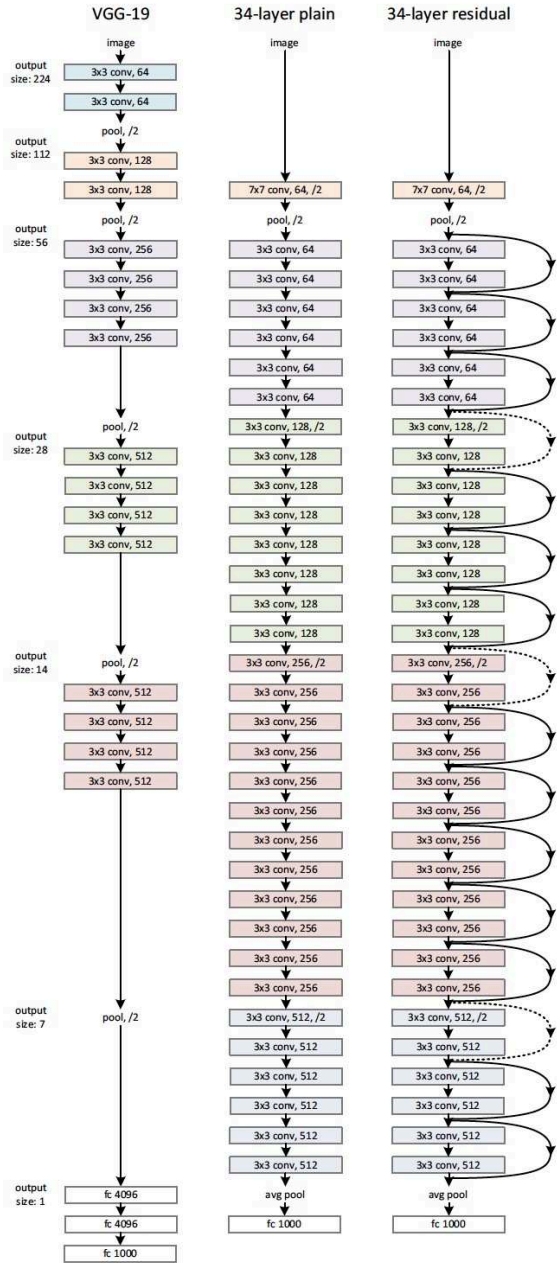

Figure IV.9 *Architecture comparison between VGG19, ResNet plain and ResNet residual [25]. Left: the architecture of VGG-19 [60]. Middle: a plain network as a baseline. Right: a residual network (with the same number of parameter layers (34) to the plain network). The figure is provided from [25].*

Group C includes Sokol's lab data [8] consisting of microbiome information of 38 healthy subjects (HS) and 222 IBD patients. The abundance includes 306 OTUs with genus level abundance. Patients in this data are classified into two categories according to the disease phenotype Ulcerative colitis (UC) and Crohn's disease (CD). Each category is divided into two conditions (flare (f), if patients got worse or reappeared the symptoms and remission (r), if patients' symptoms were decreased or disappear). The dataset was partitioned into subset with ileal Crohn's disease (iCD) and colon Crohn's disease (cCD). The detail description of the data was presented in [11].

Group D has the same samples of CIR [10], COL [32], IBD [141], OBE [19], T2D [142], WT2D [2] as Group A, but the data includes abundance of gene families generated by HMP Unified Metabolic Analysis Network (HUMAnN2) [140] with very high dimension being up to more than one million features. The data are downloaded from the package ***curatedMetagenomicData*** [193] in R.

For each sample, species/genus/gene abundance is a relative proportion and is represented as a real number, and the total abundance of all species/genus/gene sums to 1.

We also investigate our approach on two datasets (group E) in selbal analysis (Crohn and HIV) [194] with the number of counts for microbial taxa at genus level. HIV contains 155 samples while Crohn includes 662 patients of Crohn's disease and 313 controls. HIV dataset includes 62 features with the last feature indicating an HIV risk factor, MSM: "Men who has Sex with Men". We use one-hot-encoding with 1 if MSM is true, and 0 if MSM is false.

IV.4 CNN architectures and models used in the experiments

IV.4.1 Convolutional Neural Networks

In [17], the authors described convolutional neural networks (CNN) as a specialized type of neural networks aiming to process shaped in a known grid-like topology such as time-series data that can be formed as 1-D grid and images that can be formatted in a 2-D grid of pixels. The name of the network implies the network performs mathematical operation called as **convolution**, a specialized type of linear operation.

Although there are a variety of variants of CNN architectures as described in the literature (Section IV.2.2), their basic components are really similar. As presented in [65] and [17], a CNN includes three types of layers, i.e. convolutional layer, pooling layer and fully-connected layer.

The goal of convolutional layer is to learn feature representations from the inputs. A convolutional layer consists of one or more kernels (or called as *filters*) which generate different feature maps. The value of the feature at location (i, j) in the k-th feature map of l-th layer, $z^l_{i,j,k}$ is computed by the formula:

$$z^l_{i,j,k} = {w^l_k}^T x^l_{i,j} + b^l_k \tag{IV.1}$$

where w^l_k and b^l_k are the weight vector and bias term of the k-th kernel of the l-th layer respectively while $x^l_{i,j}$ denotes the input patch centered at position (i, j). Besides, the

Group A	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**
#features	542	503	443	465	572	381
#samples	232	121	110	253	344	96
Ratio of patients	0.51	0.40	0.23	0.65	0.49	0.552
Ratio of controls	0.49	0.60	0.77	0.35	0.51	0.448
Minimum size	24×24	23×23	22×22	22×22	24×24	20×20
Group B	**C1**	**C2**	**C3**	**C4**	**C5**	
#features	1976	1981	1932	1980	1985	
#samples	100	109	165	152	526	
Ratio of patients	0.48	0.42	0.44	0.58	0.49	
Ratio of controls	0.52	0.58	0.56	0.42	0.52	
Minimum size	45×45	45×45	44×44	45×45	45×45	
Group C	**CDf**	**CDr**	**iCDf**	**iCDr**	**UCf**	**UCr**
#features	259	237	247	257	250	237
#samples	98	114	82	97	79	82
Ratio of patients	0.61	0.67	0.54	0.61	0.52	0.537
Ratio of controls	0.39	0.33	0.46	0.39	0.48	0.463
Minimum size	17×17	16×16	16×16	17×17	16×16	16×16
Group D	**Cirgene**	**Colgene**	**Ibdgene**	**Obegene**	**T2Dgene**	**WT2gene**
#features	1,747,534	1,796,274	1,730,384	1,519,375	1,690,774	1,415,610
#samples	232	121	110	253	344	96
Ratio of patients	0.51	0.40	0.23	0.65	0.49	0.552
Ratio of controls	0.49	0.60	0.77	0.35	0.51	0.448
Minimum size	1322×1322	1341×1341	1316×1316	1233×1233	1301×1301	1190×1190
Group E	**Crohn**	**HIV**				
#features	48	61				
#samples	975	155				
Ratio of patients	0.68	0.83				
Ratio of controls	0.32	0.17				
Minimum size	7×7	8×8				

Table IV.1 – Information on 5 groups of datasets. *Minimum size* is the minimum resolution (shaped as a square) for images to fit the number of features.

filter w_k^l that creates the feature map $z_{:,:,k}^l$ is shared. These sharing weights allow the network to reduce the complexity and train easier.

Beside layers above, CNN also use nonlinear activation functions after each convolution. Let $a(.)$ is the activation function. The value $a_{i,j,k}^l$ after each convolution of $z_{:,:,k}^l$ of is calculated by

$$a_{i,j,k}^l = a(z_{i,j,k}^l) \tag{IV.2}$$

Activation function commonly-used are Sigmoid, Tanh [87], and ReLU [86].

The pooling layer helps to reduce the resolution of feature maps and usually follows a stack of one or more convolutional layers. Each feature maps generated from a pooling layer is connected to its corresponding feature maps from the preceding layer of convolution. Let denote the pooling layer as pool(.), for each feature map $a_{:,:,k}^l$, the value after pooling layer is computed by

$$y_{i,j,k}^l = pool(a_{m,n,k}^l) \forall (m,n) \in R_{ij} \tag{IV.3}$$

where R_{ij} is a local neighborhood around position (i,j). Pooling operations commonly-used are max pooling and average pooling.

After a stack of convolutional and pooling layers, CNN usually contain one or more fully-connected layers to perform high-level reasoning [14, 60, 28]. As in [65], the fully-connected layer can be replaced by a convolution of 1×1.

The final layer is an output layer that can be softmax or sigmoid operator depending on the number of outputs. There are two approaches for binary classification including using either two output neurons (two-node technique) using softmax function or one output neuron (one-node technique) with sigmoid operator.

Denoting θ as all parameters of a CNN (weights, bias, etc.). In order to optimize θ for a specific learning task, we need to minimize loss function defined on that task. Assuming that we have N of samples with input-output relations $(x^{(n)}, y^{(n)}); n \in [1, ..., N]$, where $x^{(n)}$ denotes the n-th input data while $y^{(n)}$ is its corresponding label, and $o^{(n)}$ is predicted score from CNN. The loss of the network can be computed by the formula (IV.4)

$$L = \frac{1}{N} \sum_{n=1}^{N} l(\theta; y^{(n)}, o^{(n)}) \tag{IV.4}$$

In order to find the best fitting set of parameters, we have to minimize the loss function. Stochastic Gradient Decent [137, 138, 139], Adagrad [135], and Adam [136] are widely used algorithms for optimizing CNN networks.

IV.4.2 One-dimensional case

In order to predict disease using 1D data, we use a Fully Connected neural network (FC) and one 1D convolutional neural network (CNN1D). FC model includes one fully-connected layer and gives one output. This is a very simple model but the performance is relatively high. The structure of this network contains a fully connected layer with sigmoid function. CNN1D includes one 1D convolutional layer with 64 filters and one max pooling of 2. We fit data into classical learning algorithms such RF [118] (with 50 trees) and SVM [119] (kernels of Sigmoid, Radial, Linear) for 1D data.

```
OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####     24   24    3
InputLayer    |   --------------------        0     0.0%
            #####     24   24    3
  Flatten   ||||| --------------------        0     0.0%
            #####        1728
    Dense   XXXXX --------------------     1729   100.0%
  sigmoid   #####           1
```

A

```
OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####     24   24    1
InputLayer    |   --------------------        0     0.0%
            #####     24   24    1
  Flatten   ||||| --------------------        0     0.0%
            #####         576
    Dense   XXXXX --------------------      577   100.0%
  sigmoid   #####           1
```

B

```
OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####     24   24    3
InputLayer    |   --------------------        0     0.0%
            #####     24   24    3
   Conv2D    \|/  --------------------     1792    16.0%
     relu   #####     24   24   64
MaxPooling2D Y max --------------------        0     0.0%
     relu   #####     12   12   64
  Flatten   ||||| --------------------        0     0.0%
            #####        9216
    Dense   XXXXX --------------------     9217    83.0%
  sigmoid   #####           1
```

C

```
OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####     24   24    1
InputLayer    |   --------------------        0     0.0%
            #####     24   24    1
   Conv2D    \|/  --------------------      640     6.0%
     relu   #####     24   24   64
MaxPooling2D Y max --------------------        0     0.0%
     relu   #####     12   12   64
  Flatten   ||||| --------------------        0     0.0%
            #####        9216
    Dense   XXXXX --------------------     9217    93.0%
  sigmoid   #####           1
```

D

Labels	Name	#weights
A	FC for three-channel images	1,729
B	FC for one-channel images	577
C	Proposed CNN for three-channel images	11,009
D	Proposed CNN for one-channel images	9,857
E	VGG-like conv net	213,537

```
OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####     24   24    3
   Conv2D    \|/  --------------------      896     0.0%
     relu   #####     24   24   32
   Conv2D    \|/  --------------------     9248     4.0%
     relu   #####     22   22   32
MaxPooling2D Y max --------------------        0     0.0%
            #####     11   11   32
   Conv2D    \|/  --------------------    18496     8.0%
     relu   #####      9    9   64
   Conv2D    \|/  --------------------    36928    17.0%
     relu   #####      7    7   64
MaxPooling2D Y max --------------------        0     0.0%
            #####      3    3   64
  Flatten   ||||| --------------------        0     0.0%
            #####        576
    Dense   XXXXX --------------------   147712    69.0%
     relu   #####        256
    Dense   XXXXX --------------------      257     0.0%
  sigmoid   #####           1
```

E

Figure IV.10 *Comparison of operations and weights of various architectures of CNN and FC models visualized using keras_sequential_ascii [109].*

IV.4.3 Two-dimensional case

Images constructed with the Fill-up vary in size from 16×16 to 24×24 (for datasets in group A) depending on the number of features while, in t-SNE and other dimensionality reduction algorithms, we use 24x24 images for all datasets.

The architecture is conducted from the results investigated on a variety of architectures of CNN applied to representations based on Fill-up, using SPB, gray images (Table IV.2 and Figure IV.12). We also compare to VGG-like convolutional networks (convnet) proposed in Keras (a high-level neural networks API, written in Python [109]) document `https://keras.io/getting-started/sequential-model-guide/` with a little modification as a baseline (see the architecture in Figure IV.11). The results show the performance is rising according to the width of the CNN. However, the performance decreases when we add more layer due to overfitting (see the number of weights of networks in IV.10). As seen the graph (Figure IV.12), CNN with one convolutional layer and a large number of filters outperform FC for CIR, COL, IBD while WT2, OBE seems like meets overfitting with CNN. In most of cases, the architectures of CNN with one convolutional layer obtain greater performance than two-convolutional-layer and VGG architectures. However, for WT2, the architectures with two layers seems to give a slight improvement (see Figure IV.12). For shallow CNN, dropout in FC seems like not improve

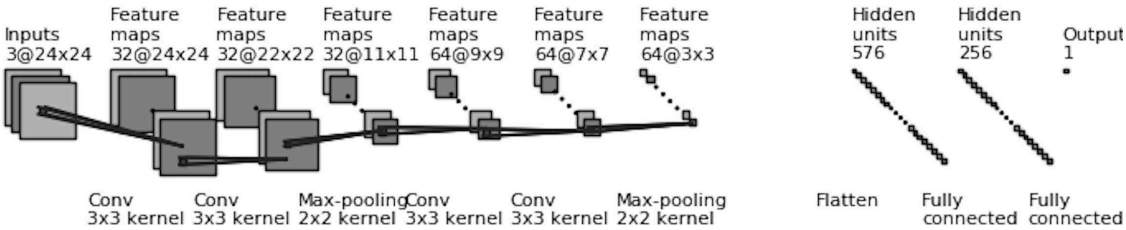

Figure IV.11 *The architecture of Vgg-like convnet.*

the performance but it works on deep networks such as VGG. VGG with dropout rate of 0.2 reveals the best performance among variations of VGG in our experiments. For some cases, combination between dropout in Convolutional layers and FC layers improve the performance compared to apply to only the FC layers (in CNN-l1f32, CNN-l2f16). For CIR, CNN with one convolutional layer outperform FC. CIR and IBD reach to the peak at CNN-l1f64 while OBE and T2D perform the best at CNNl1f16 and CNNl1f20. In other hand, WT2 faces overfitting problem with no better results in CNN compared to FC. The use of dropout is apparently not efficient.

Based on these results, we propose a CNN architecture including a stack of one convolutional layers with 64 kernels of 3×3 (stride 1), followed by a max pooling 2×2 (stride 2). ReLU is used after each convolution. Wide convolution is implemented to preserve the dimension of input after passing through the convolutional layer. We employ a sigmoid activation function at the final layer (see the CNN architecture for images of 24×24 in Figure IV.13).

IV.4.4 Experimental Setup

All networks use the same optimization function (Adam [39]), learning rate = 0.001, loss function: binary cross entropy, epoch = 500 (using Early Stopping to avoid over-fitting with the number of epoch patience of 5). The best results obtained in [93] carry out the batch size ranging 2 to 32, so our framework implements a batch size of 16. The prediction performance are assessed by 10-fold-stratified-cross validation, repeated and averaged on 10 independent runs (except for the analysis in IV.5.3, we use the same setting with [11] using 5-fold-cross validation repeated 10 times). **Met2Img** is implemented in Keras 2.1.3 [109], Tensorflow 1.5 [111, 110], libraries for visualizations in Matplotlib [82] using Python/Scikit-Learn[112, 16], and can run either in CPU or GPU architectures. Source code is publicly available at `https://git.integromics.fr/Deepomics/deepMG_tf`

Early Stopping [17] is used to avoid overfitting with epoch patience = 5. The epoch patience is defined as a number of epochs to wait before early stop if there is no progress on a validation set. Although Keras.Callback.callbacks.EarlyStopping is useful and really improves the performance, sometimes EarlyStopping of keras has the problem when the performance fluctuates during first epochs. I modified EarlyStopping of keras slightly to improve this situation.

The modification solves the following problem. For instance, the val_loss of 1 training running 14 epochs (without EarlyStopping, EarlyStopping (Keras), EarlyStopping (mod-

Model	*drcnn*	*drfc*	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
CNN-l1f08	0.0	0.0	**0.897**	0.766	**0.844**	**0.680**	0.647	0.667	0.750
CNN-l1f08	0.0	0.2	**0.894**	0.766	**0.853**	**0.667**	0.642	0.679	0.750
CNN-l1f08	0.1	0.2	**0.895**	0.762	**0.851**	**0.666**	0.637	0.676	0.748
CNN-l1f16	0.0	0.0	**0.897**	0.775	**0.850**	**0.684**	0.660	0.690	0.759
CNN-l1f16	0.0	0.2	**0.896**	0.763	**0.848**	**0.673**	0.653	0.675	0.752
CNN-l1f16	0.1	0.2	**0.897**	0.764	**0.852**	**0.669**	0.657	0.681	0.754
CNN-l1f20	0.0	0.0	**0.899**	0.782	**0.849**	**0.690**	0.649	0.699	0.761
CNN-l1f20	0.0	0.2	**0.895**	0.782	**0.850**	**0.668**	0.654	0.686	0.756
CNN-l1f20	0.1	0.2	**0.896**	0.789	**0.850**	**0.663**	0.652	0.684	0.756
CNN-l1f32	0.0	0.0	**0.901**	0.790	**0.853**	**0.683**	0.656	0.693	0.763
CNN-l1f32	0.0	0.2	**0.903**	0.801	**0.850**	**0.674**	0.652	0.673	0.759
CNN-l1f32	0.1	0.2	**0.902**	0.799	**0.851**	**0.671**	0.655	0.687	0.761
CNN-l1f48	0.0	0.0	**0.896**	0.786	**0.855**	**0.685**	0.650	0.698	0.762
CNN-l1f48	0.0	0.2	**0.897**	0.774	**0.858**	**0.681**	0.652	0.698	0.760
CNN-l1f64	0.0	0.0	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
CNN-l1f64	0.0	0.2	**0.900**	0.787	**0.860**	**0.676**	0.646	0.696	0.761
CNN-l1f64	0.1	0.2	**0.901**	0.783	**0.860**	**0.675**	0.644	0.693	0.759
CNN-l2f08	0.0	0.0	0.889	0.758	**0.836**	**0.677**	0.646	0.679	0.748
CNN-l2f08	0.0	0.2	0.885	0.741	**0.828**	**0.664**	0.645	0.672	0.739
CNN-l2f08	0.1	0.2	0.881	0.741	**0.831**	**0.662**	0.643	0.667	0.738
CNN-l2f16	0.0	0.0	**0.892**	0.749	**0.842**	**0.677**	0.646	0.699	0.751
CNN-l2f16	0.0	0.2	0.880	0.752	**0.828**	0.650	0.642	0.658	0.735
CNN-l2f16	0.1	0.2	0.883	0.759	**0.842**	**0.672**	0.642	0.678	0.746
CNN-l2f20	0.0	0.0	**0.899**	0.761	**0.845**	**0.670**	0.640	0.705	0.753
CNN-l2f20	0.0	0.2	0.886	0.754	**0.836**	**0.660**	0.647	0.685	0.745
CNN-l2f20	0.1	0.2	0.888	0.744	**0.830**	**0.669**	0.648	0.670	0.742
CNN-l2f32	0.0	0.0	**0.894**	0.772	**0.846**	**0.666**	0.645	0.713	0.756
CNN-l2f32	0.0	0.2	0.884	0.757	**0.827**	**0.661**	0.639	0.686	0.742
CNN-l2f32	0.1	0.2	0.881	0.748	**0.832**	**0.658**	0.633	0.693	0.741
CNN-l2f64	0.0	0.2	0.880	0.762	**0.838**	**0.658**	0.626	0.710	0.746
CNN-l2f64	0.1	0.2	0.887	0.735	**0.841**	**0.656**	0.622	0.695	0.739
VGG	0.0	0.0	0.838	0.769	0.820	0.614	0.539	0.674	0.709
VGG	0.0	0.1	0.843	0.755	**0.827**	0.619	0.526	0.681	0.709
VGG	0.0	0.2	0.850	0.775	**0.827**	0.618	0.522	0.676	0.711
VGG	0.0	0.3	0.841	0.768	0.813	0.609	0.538	0.668	0.706
VGG	0.0	0.4	0.849	0.751	0.817	0.614	0.534	0.666	0.705
VGG	0.0	0.5	0.845	0.746	0.812	0.622	0.534	0.671	0.705
FC	0.0	0.0	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760

Table IV.2 – Performance (in ACC) comparison of different architectures of CNN. CNN-l$\boldsymbol{x}$f$\boldsymbol{y}$ denotes to the CNN contains $\boldsymbol{x}$ convolutional layer(s), $\boldsymbol{y}$ filters for each convolutional layer and a max pooling after all convolutional layers (see an example of CNN-l1f64 in Figure IV.13) while ***drfc***, ***drcnn*** reveal dropout rate in FC layers, Convolutional layers, respectively. The significant results are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG)

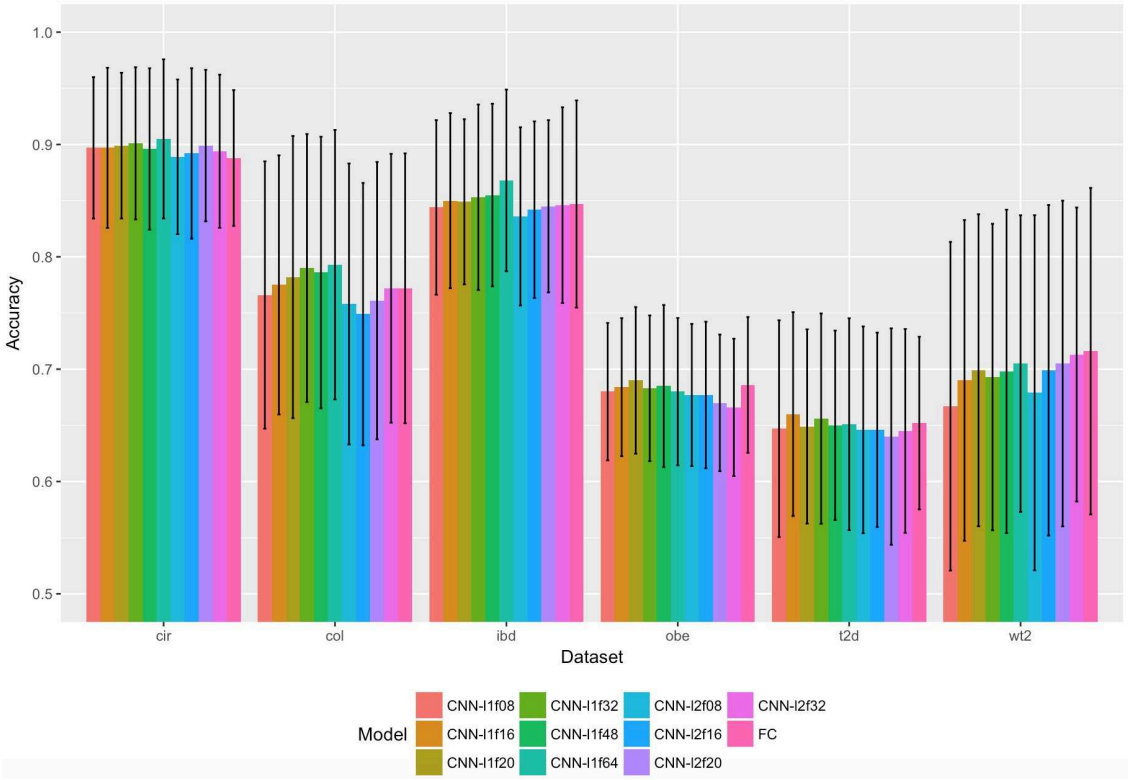

Figure IV.12 *Performance comparison (in ACC) of a variety of CNN (without dropout) and FC models. Error bars are standard deviations*

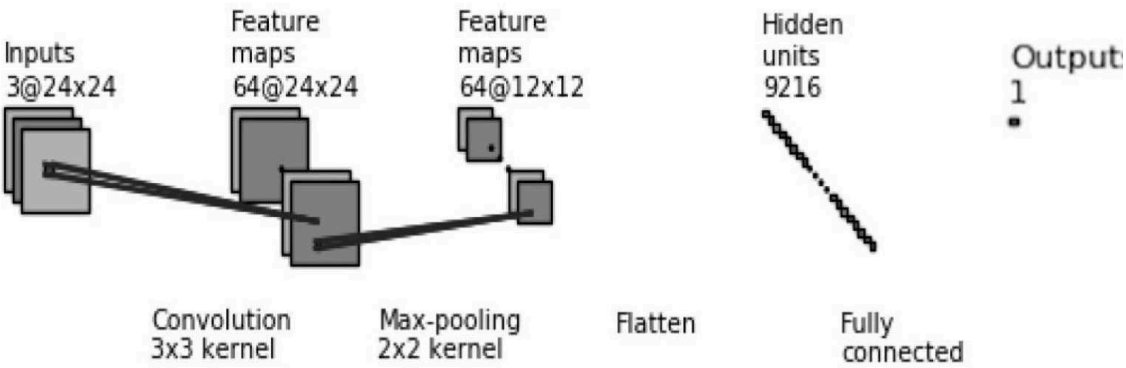

Figure IV.13 *The CNN architecture includes a stack of one convolutional layer with 64 filters of 3x3 and a max pooling of 2x2, followed by one fully connected layer. The input includes either images of one channel for gray (or black/white) images or three channels for color images*

without Early Stopping	**EarlyStopping (Keras)**	**EarlyStopping (modification)**
epoch 1: Val_loss 0.250 epoch 2: Val_loss 0.100 epoch 3: Val_loss 0.200 epoch 4: Val_loss 0.192 epoch 5: Val_loss 0.182 epoch 6: Val_loss 0.188 epoch 7: Val_loss 0.120 epoch 8: Val_loss 0.080 epoch 9: Val_loss 0.050 epoch 10: Val_loss 0.055 epoch 11: Val_loss 0.055 epoch 12: Val_loss 0.058 epoch 13: Val_loss 0.060 epoch 14: Val_loss 0.068	epoch 1: Val_loss 0.250 epoch 2: Val_loss 0.100 epoch 3: Val_loss 0.200 (0.100 < 0.200 patience 1) epoch 4: Val_loss 0.192 (0.100 < 0.192 patience 2) -->Stopped at epoch 4 with val_loss=0.192	epoch 1: Val_loss 0.250 epoch 2: Val_loss 0.100 epoch 3: Val_loss 0.200 (0.100 < 0.200 patience 1) epoch 4: Val_loss 0.192 epoch 5: Val_loss 0.182 epoch 6: Val_loss 0.188 (0.182 < 0.188 patience 1) epoch 7: Val_loss 0.120 epoch 8: Val_loss 0.080 epoch 9: Val_loss 0.050 epoch 10: Val_loss 0.055 (0.055 < 0.050 patience 1) epoch 11: Val_loss 0.055 (0.055 = 0.055 patience 2) -->Stopped at epoch 11 with val_loss=0.055

Figure IV.14 *Examples on effects of 2 types of Early Stopping and without using Early Stopping (patience = 2)*

ification)) shown in Figure IV.14. Assume that patience = 2 is number of epochs with no improvement after which training will be stopped. In the modification version, patience is consecutive number of times of 2 epochs of previous-next with no improvement after which training will be stopped), with Keras.Callback.callbacks.EarlyStopping the training stops at epoch 4 but with EarlyStopping with modification the training will continue until epoch 11. As illustrated in Tab . IV.3, EarlyStopping with modification improves slightly performance of the overall performance while it also shows significant improvement on T2D dataset increasing by 0.04 for T2D on FC. We see that EarlyStopping with modification run more some epochs compared to classical EarlyStopping of Keras leading to the performance improved. Sometimes, there is some situation where the network meets slight overfitting (FC for IBD) but this has no significant difference in the performance.

IV.5 Results

In order to compare results between methods, the *p-values* are computed based on function tsum.test (**PASWR**- package for Probability and Statistics with R) to compare results from the-state-of-the-arts. The results which have *p-values* < 0.05 are significant improvements.

IV.5.1 Comparing to the-state-of-the-art (MetAML)

To compare to the MetAML [18], we use accuracy (ACC) to measure model performances. Classification accuracies were assessed by 10-fold cross validation, repeated and averaged on 10 independent runs using stratified cross validation approach.

model	Type	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
Accuracy								
FC	Keras	0.888	0.771	**0.849**	**0.686**	0.612	0.707	0.752
FC	Modification	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760
CNN	Keras	**0.903**	0.793	**0.865**	**0.680**	0.644	0.703	0.765
CNN	Modification	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
Epochs stopped								
FC	Keras	383.25	217.31	374.77	107.28	161.44	281.30	254.225
FC	Modification	383.25	218.02	402.78	112.07	234.82	295.83	274.462
CNN	Keras	131.16	83.32	136.53	21.98	41.45	72.01	81.075
CNN	Modification	162.87	84.07	144.39	24.53	68.29	73.41	92.927

Table IV.3 – Accuracy and the average number of epochs during training. A comparison between Early Stopping (patience = 5) of Keras and Early Stopping with modifications. The significant results are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG)

IV.5.1.1 Execution time

Met2Img is implemented using Python library running on a cluster system using CentOS Linux operating (we set up to use the job scheduler aiming to measure execution time with 4 cpu cores per a job or running on a NVIDIA-SMI 387.26 GPU with a memory of 12GB) with a RAM memory of 512 GB and a local macbook (OS X 10.11.6) with 4 CPU cores and 16GB of RAM.

The comparison between CPU and GPU running a CNN including one stack of one convolutional (64 filters) and one max pooling layers, and FC models, is revealed in Figure IV.15. The resolutions of images range from 20×20 to 24×24. For simple model such as FC, there is no significant difference on inference time between GPU and CPU. However, GPU speeds up notably on CNN models. The improved time is up to four to five, even more, compared to CPU. Extraordinarily, CNN using GPU run faster than simple FC model.

Another comparison on various bins and visualizations used images and models as above (for t-SNE, we use 24×24 images for all datasets) shown in Figure IV.16, we see clearly that SPB is the fastest while t-SNE consumes a large mount of time. SPB only creates the set of images for training and testing once at the beginning, while t-SNE and QTF generate a new set of training and test images for each fold. Beside the time for generating images at each fold, t-SNE takes time to build global t-SNE maps that contains coordinates of data points while QTF also run for transforming the data. As observed, T2D is the largest dataset in terms of the number of features and samples, so the execution time is the highest. The small datasets such as COL, WT2 share the same pattern with execution time being around 30 minutes.

IV.5.1.2 The results on 1D data

As shown in the Table IV.4, the Random Forest both in MetAML and our framework outperforms other models, and share the same pattern in results. In 1D data, CNN1D

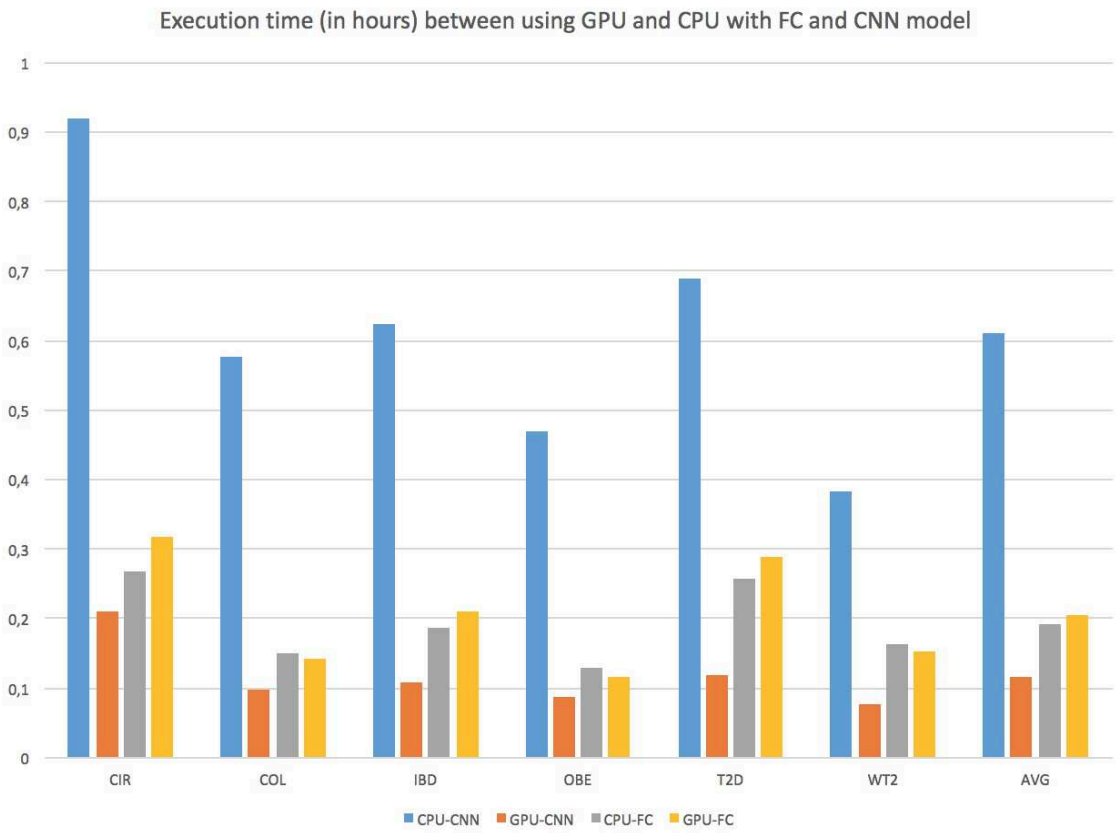

Figure IV.15 *Execution time comparison between using GPU and CPU for different models (Fill-up)*

gives better FC while results for the SVM models are the worst. In SVM models, Linear kernel shows the best performance. From these results, standard and shallow learning algorithms (such as RF) are more robust than deep learning for 1D data. Remarkably, CNN1D obtains two significant results on IBD, and OBE, and show an approximate performance on T2D. Furthermore, RF in our framework also provides a slight increase in ACC compared to RF in MetAML.

IV.5.1.3 The results on 2D data

Table IV.5 shows the performance in ACC (see the standard deviation in Table C.1 in Appendices) of Met2Img framework including visualizations (Fill-up and manifold learning approaches) using a FC model and a CNN with the architecture described in Figure IV.13 and Figure IV.10. The table also reveals the results of a variety of types of color and different bins. The performances of MetAML [18] shown are results of RF model, the best model in MetAML. As seen from the table, the performances with QTF bins share the same pattern with "Species bins" (SPB), but the performance of QTF is slightly lower than **SPB**, while PR reveals poor performance in most of cases both Fill-up and t-SNE. Noticeably, although PR is the worst among all considered bins, it shows an increase compared to SPB for T2D. As shown for Fill-up using random ordering and phylogenetic

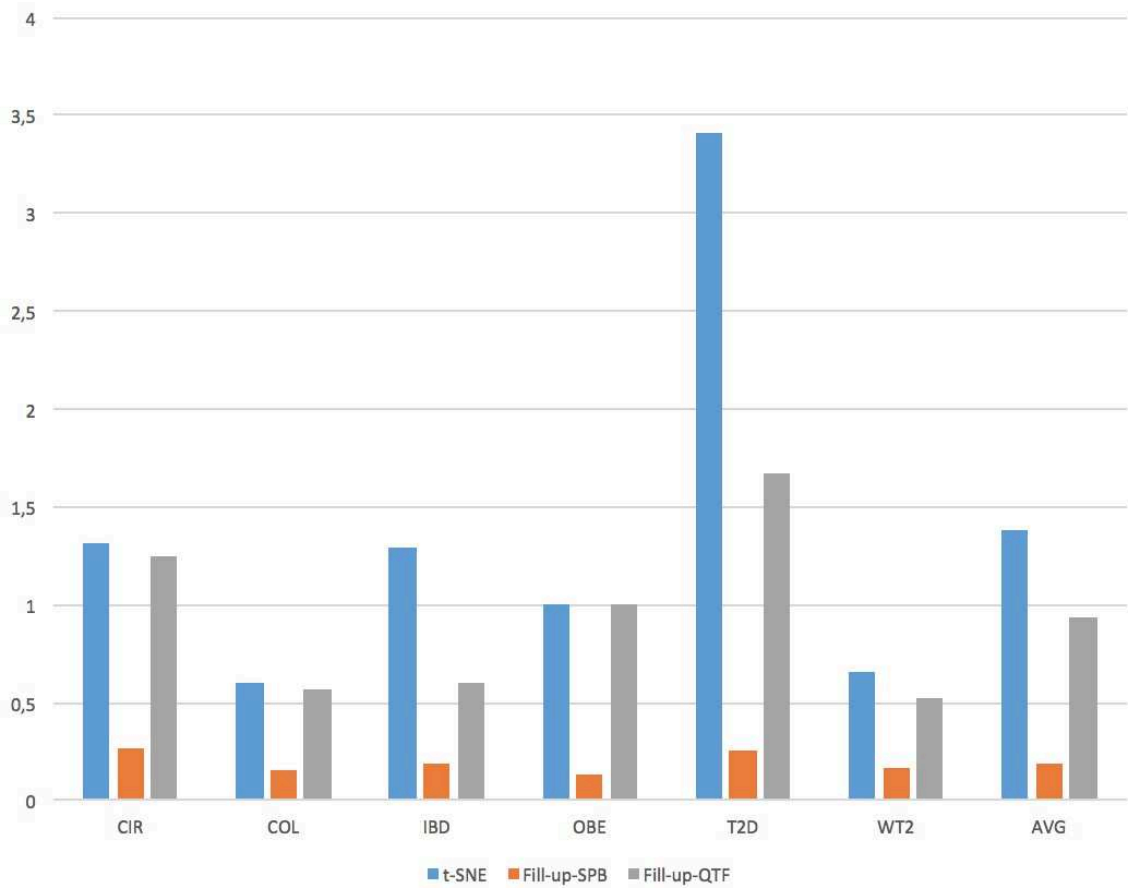

Figure IV.16 *Execution time comparison of types of embedding (in hours)*

Framework	Model	CIR	COL	IBD	OBE	T2D	WT2	AVG
MetAML	RF	0.877	0.805	0.809	0.644	0.664	0.703	0.750
	SVM	0.834	0.743	0.809	0.636	0.613	0.596	0.705
Met2Img	RF	0.877	0.812	0.808	0.645	0.672	0.703	0.753
	SVM- Sigmoid	0.509	0.603	0.775	0.648	0.515	0.553	0.600
	SVM- Radial	0.529	0.603	0.775	0.648	0.593	0.553	0.617
	SVM- Linear	0.766	0.666	0.792	0.612	0.634	0.676	0.691
	FC	0.776	0.685	0.775	**0.656**	0.665	0.607	0.694
	CNN1D	0.775	0.722	**0.842**	**0.663**	0.668	0.618	0.715

Table IV.4 – Performance (in ACC) comparison on 1D data. The significant results are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG).

sorting, T2D with PR achieves better performance SPB with corresponding OTUs sorting. The most predictable disease seems to be **liver cirrhosis** with the best ACC **over 0.9**. As proven from the results, CNN is a promising model for predicting liver cirrhosis with a majority of high ACCs coming from CNN. Especially, Fill-up using color images is appropriate to CIR dataset with both FC and CNN models obtaining significant results. Another point here is that Fill-up reveals as a promising tool for predicting IBD. The performance of **IBD** follows CIR with the best result of **0.868**. All of results using Fill-up achieve significant improvement compared to MetAML for IBD.

The proposed framework is a good solution for the OBE data with all results shown in Table IV.5 being significant results, while the COL data seem to be a challenge with only SPB using color images or SPB with CNN using gray images giving equivalent results compared to classical shallow algorithms (RF). There is a trivial difference between using color and gray images for CNN while color images helps to improve performance considerably for FC model.

The performance of Fill-up is better compared to visualizations based on manifold learning approaches. An explanation is that the representations based on manifold learning methods perform worse due to many overlapped points, while in contrast every features in Fill-up is visible. It is noteworthy that t-SNE using QTF gives significant results for **T2D** with the best ACC of **0.690** while Fill-up shows poor performance on this dataset. Additionally, Isomap also exhibits the best performance on **OBE** with a ACC of **0.691**.

Another important point is that integration of phylogenetic information helps to improve the performance on IBD and COL datasets. Only a significant results of random ordering compared to phylogenetic sorting is on WT2 dataset (ACC=**0.739**).

Additionally, we carry out a comparison chart between a CNN and a robust classical learning algorithm (RF) in Figure IV.18. The CNN model achieves either significant results with CIR, IBD, OBE, WT2 (p-value < 0.0005) or comparative performance to the others including COL, T2D compared to the best model (RF) in MetAML. This proves that CNN is a promising approach for metagenomic data.

We repeat the experiments 10 times using 10-fold cross-validation, the same folds are used for all classifiers, so we can compare the difference in performance of any two classifiers within each test fold. On Figure IV.17, six charts show results in detail of the performance between RF (of Met2Img) and CNN model (with SPB, gray images). It is noteworthy that RF in our framework gives a slight increase in performance compared to MetAML for COL, T2D while performance in other datasets remains nearly the same. As seen from Figure IV.17, the performance we see in most of cases that CNN outperforms RF for CIR (p-value= 0.001794), IBD (p-value=2.506×10^{-08}), OBE (p-value=6.186×10^{-6}). Numerous folds in IBD reach to the accuracy of 100% while CIR almost range in 0.9 to over 0.95 and several folds achieve the 100%. CNN applied to OBE obtains over 15% of all folds that surpasses the peak of RF (0.75). There are a few outliers results for T2D where the performance is either very high (3 folds) or extreme low (1 fold). Most of results of CNN are in bottom of the chart as proven that RF (ACC=0.672) outperforms CNN (ACC=0.651) with p-value=0.04636. For other 2 datasets, CNN is approximate to RF. The maximum performance of CNN is over 0.9 while for RF is near to 0.9. For WT2, there are numerous folds that obtain the same results. RF achieves many performances of 100% while CNN reaches the peak at about 0.9.

Figure IV.19 reveals the number of epochs the learning stopped during training phrase.

	Bins	Model	Color	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
MetAML - RF				0.877	0.805	0.809	0.644	0.664	0.703	0.750
Fill-up	PR	CNN	BW	0.880	0.763	**0.841**	**0.669**	0.666	0.667	0.748
Fill-up	QTF	CNN	viridis	**0.897**	0.781	**0.837**	**0.659**	0.664	0.690	0.755
Fill-up	SPB	CNN	custom	**0.899**	0.782	**0.857**	**0.678**	0.656	0.710	0.764
Fill-up	SPB	CNN	gray	**0.905**	0.793	**0.868**	**0.680**	0.651	0.705	0.767
Fill-up	SPB	CNN	jet	**0.903**	0.798	**0.863**	**0.681**	0.649	0.713	0.768
Fill-up	PR	FC	BW	0.863	0.735	**0.842**	**0.672**	0.656	0.712	0.747
Fill-up	QTF	FC	viridis	0.887	0.765	**0.853**	**0.657**	0.640	0.711	0.752
Fill-up	SPB	FC	grays	0.888	0.772	**0.847**	**0.686**	0.652	0.716	0.760
Fill-up	SPB	FC	jet	**0.905**	0.794	**0.837**	**0.679**	0.659	0.713	0.764
Fill-up	SPB	FC	custom	**0.904**	0.805	**0.835**	**0.676**	0.645	0.709	0.762
Fill-up ∗	PR	CNN	BW	0.880	0.747	**0.837**	**0.674**	0.673	0.724	0.756
Fill-up ∗	SPB	CNN	grays	**0.903**	0.768	**0.840**	**0.673**	0.666	**0.739**	0.765
Fill-up ∗	PR	FC	BW	0.867	0.744	**0.834**	**0.664**	0.649	0.699	0.743
Fill-up ∗	SPB	FC	grays	0.887	0.758	**0.853**	**0.677**	0.646	0.711	0.755
isomap	QTF	CNN	grays	**0.890**	0.740	0.817	**0.689**	0.657	0.714	0.751
isomap	QTF	CNN	grays	0.889	0.743	0.815	**0.691**	0.657	0.712	0.751
isomap	QTF	FC	grays	0.871	0.742	0.791	**0.688**	0.646	0.693	0.738
isomap	QTF	FC	grays	0.871	0.742	0.793	**0.688**	0.646	0.689	0.738
TSNE	PR	CNN	BW	0.862	0.712	0.815	**0.664**	0.660	0.672	0.731
TSNE	PR	FC	BW	0.857	0.710	0.813	**0.662**	0.640	0.688	0.728
TSNE	QTF	CNN	viridis	0.869	0.753	0.821	0.646	0.673	0.670	0.739
TSNE	QTF	CNN	jet	0.870	0.748	0.820	0.654	**0.690**	0.660	0.741
TSNE	QTF	CNN	grays	0.865	0.737	0.824	**0.659**	0.674	0.662	0.737
TSNE	QTF	FC	viridis	0.881	0.739	0.821	**0.661**	0.610	0.687	0.733
TSNE	QTF	FC	jet	0.883	0.768	**0.825**	**0.670**	0.648	0.661	0.742
TSNE	QTF	FC	grays	0.867	0.771	0.791	**0.663**	0.646	0.701	0.740
TSNE	SPB	CNN	jet	0.887	0.779	0.819	**0.680**	0.650	0.685	0.750
TSNE	SPB	CNN	viridis	0.881	0.746	0.814	**0.664**	0.650	0.667	0.737
TSNE	SPB	CNN	grays	0.885	0.765	**0.828**	**0.676**	0.655	0.681	0.748
TSNE	SPB	FC	jet	**0.893**	0.783	**0.826**	**0.687**	0.621	0.711	0.753
TSNE	SPB	FC	viridis	**0.894**	0.751	0.814	**0.676**	0.610	0.692	0.740
TSNE	SPB	FC	grays	0.873	0.773	0.791	**0.678**	0.632	0.708	0.743

Table IV.5 – Performance (in ACC) comparison on 2D data (Results of Fill-up with random ordering are marked "*"). The significant results are reported in **bold**. The average performance of six datasets is computed and shown in the last column (AVG). Standard deviation is presented in Table C.1

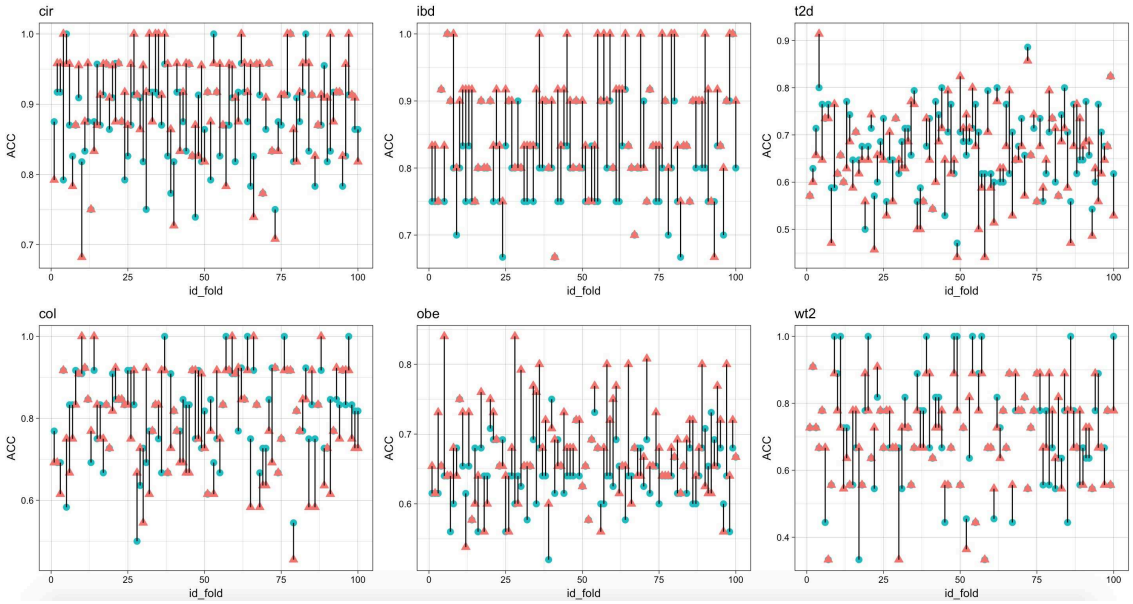

Figure IV.17 *Performance comparison in detail (ACC) between CNN (illustrated by red triangles) model and RF (shown in blue circles).*

Comparing between FC and CNN, we see that CNN stopped earlier than FC due to overfitting. For models with one-channel input, the learning needs more epochs to converge while models for color (jet) images face the overfitting sooner. OBE meets overfitting only after few epochs leading to low performances (ACC<0.7), while high ACCs appear to CIR, IBD when the training consumes numerous epochs for learning samples.

IV.5.1.4 The explanations from LIME and Grad-CAM

In this part, we apply Local Interpretable Model-Agnostic Explanations (LIME [29]), Saliency Maps [113, 34] and Gradient-weighted Class Activation Mapping (Grad-CAM [33]) implemented in *keras-vis* [35] to extract the explanations from the output of predictions. The figures in this part come from:

- The dataset: CIR dataset with 24x24 images.
- The model: CNN (see the architecture in Figure IV.13).
- Paired-images: Original - Output of explanation from LIME/Saliency Maps/Grad-CAM.

88_l1 denotes the sample of 88 labeled 1 while 88_p[0.45] indicates the sample of 88 is predicted with a probability of 0.45 infected disease. We can see the top super pixels that are most positive towards the class (infected cirrhosis disease).

In order to be easier than in comparisons, we perform a visualization using Fill-up on important features extracted using lasso for feature selection in MetAML on CIR dataset (Figure IV.20). The left image (A) in IV.20 is the ranking (0-541) of the important features where the highest value (541, dark red) reveals the most important features.

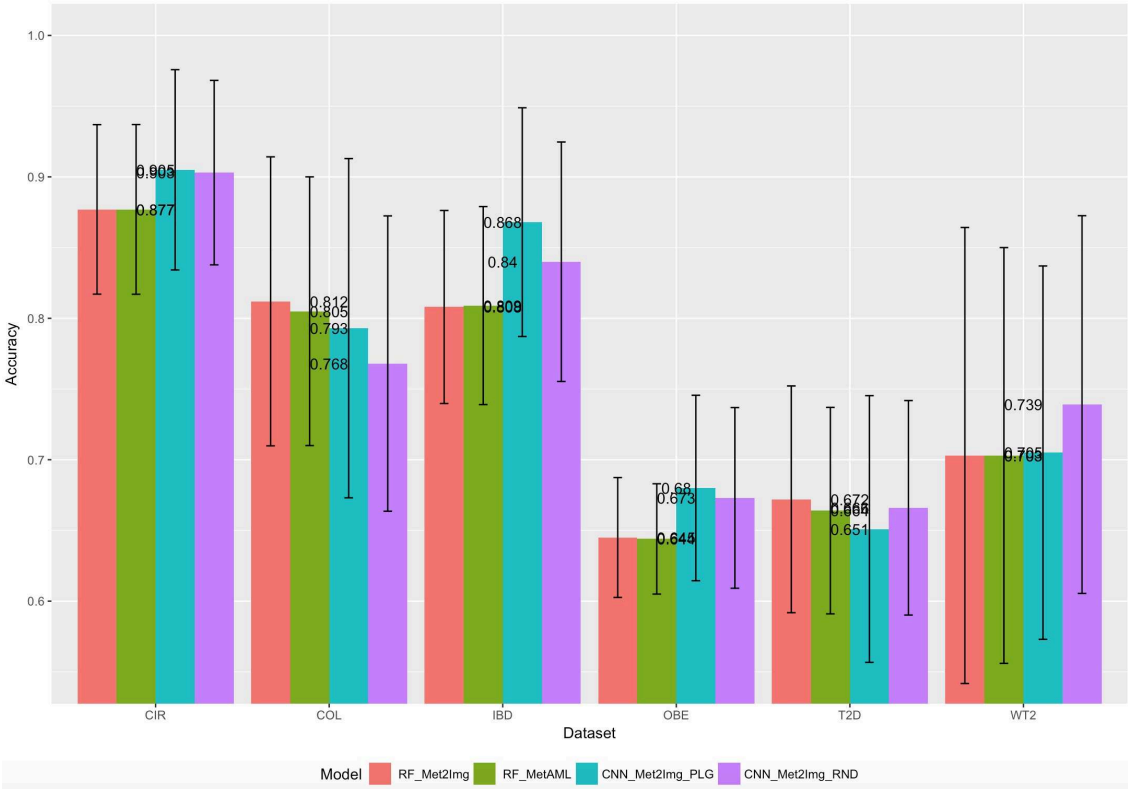

Figure IV.18 *Performance comparison in ACC between CNN model (using Fill-up (Phylogenetic order (PLG) and random sorting (RND)) with SPB and gray images) and shallow learning algorithm (RF) of MetAML and our framework. The standard deviations are shown in error bars.*

The other plot (B) shows the magnitude of the importance of the features. We also compute p-values to show significant differences between in species abundance of control and patient samples with the magnitudes visualized in (C) and p-values that are less than 0.05 (significant differences) illustrated by "red" in (D).

As visualized in Figure IV.21, the regions of the image where are covered by red, are negative score for cirrhosis disease while green areas in images where are the explanation that the patient infected the disease. Additionally, we can find the explanations for two prediction results the algorithm got wrong (sample 79 gave p=0.69 while the label is 0, and sample 88 where the classification yielded a score of 0.47 whereas its label is 1).

Also, we exploit Saliency Maps that aim to find the pixels of an image which contribute most towards to the disease class. As exhibited in Figure IV.22, Saliency maps indicates pixels that are the most importance (red regions) for images being classified as a cirrhosis patient. Similar results are found in Figure IV.20B.

Besides LIME and Saliency Maps, Grad-CAM [33] also is leveraged to make the explanations for the output. Grad-CAM visualizes more clear than LIME in illustrating important locations for the prediction and contains more details than Saliency Maps since Grad-CAM benefits from Convolution and Pooling features that contain more spatial detail which is lost in Dense layers. Figure IV.23 gives explanations for several control

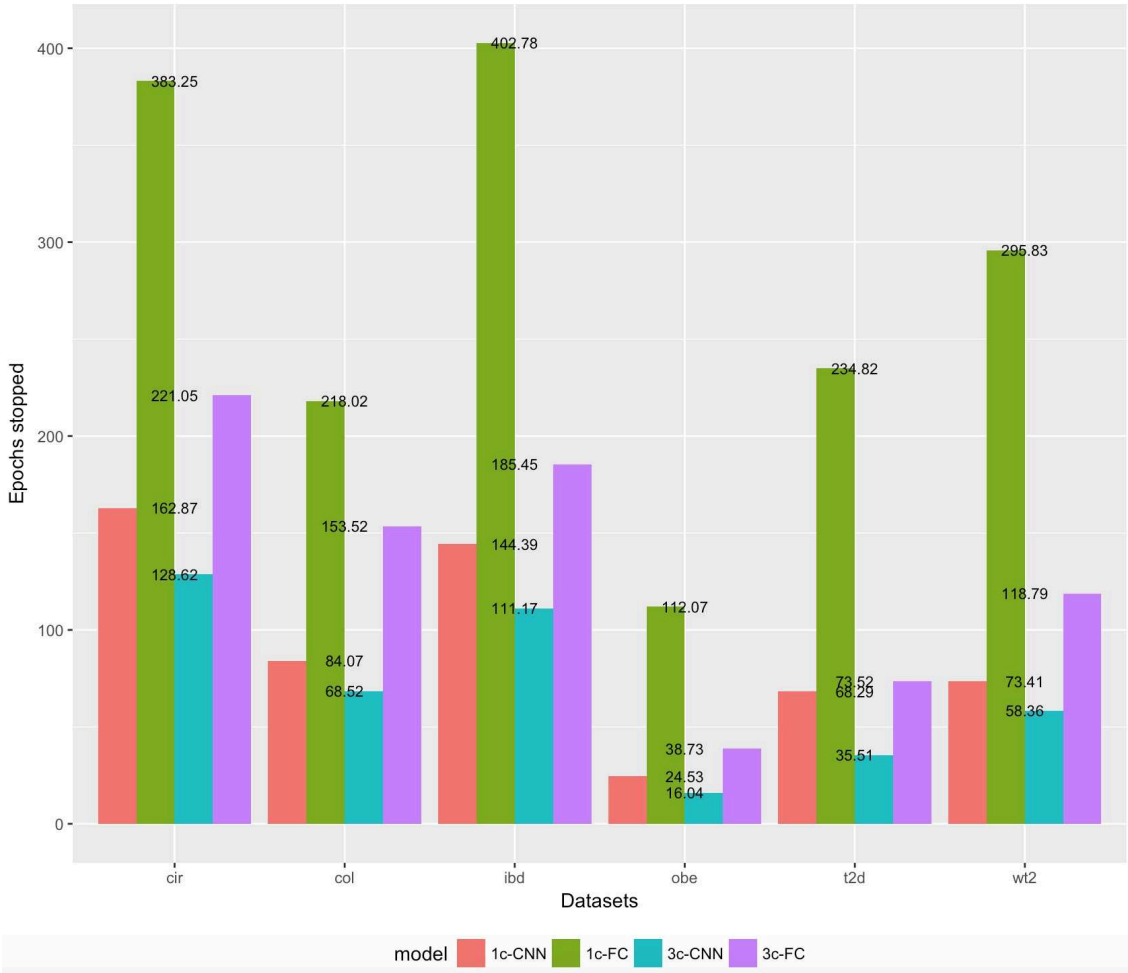

Figure IV.19 *The average epochs stopped in Early Stopping technique of models applied to datasets using Fill-up with SPB and phylogenetic ordering. 3c-CNN and 3c-FC denote the CNN and FC models, respectively, for color images with 3 channels while 1c-CNN, 1c-FC reveal the CNN and FC with one-channel input for gray images.*

and patient samples. The decision of the classification seems to conduct from red areas in images (such as top-left area) while blue regions correspond to a decrease in score for the disease class. These (red) regions are suitable to area where important features are dense (Figure IV.20A) and some of them are fitted to significant difference areas in (Figure IV.20D).

IV.5.2 Comparing to shallow learning algorithms

We also evaluate **Met2Img** on five datasets including samples of C1, C2, C3, C4, and C5 in [37]. Met2Img shows poor performance for COL dataset in group A, but the results are improved when increasing the number of features. As in [37], we use AUC as a measurement performance for the experiments in this part. Dai et al. in [37] (the code is available at `https://github.com/DAIZHENWEI/meta-analysis-microbiome`) achieved average AUC of 0.75 on the testing fold using the seven CRC-enriched bacterial species to classify 255 CRC cases from controls. Without feature selection as in [37], we obtains significant results on the dataset with full OTUs provided in [37] compared to the results in [37] reproduced.

In addition, we fit OTUs of each dataset in group C into a RF and SVM model for comparisons. As revealed in Table IV.6, SVM performed better on the seven bacterial species. For high dimension data, SVM shows poor performance while, another classical learning, RF, outperforms SVM in all situations.

Although, there is no significant improvement to colorectal cancer for COL dataset in group A, our framework with gray images using QTF illustrates substantial results for CNN compared to shallow learning algorithms. Remarkably, Fill-up using QTF with *viridis* shows significant improvements of 4 out of 5 datasets compared to RF. SPB seems to be appropriate to the cohort from America (C3), while QTF can be a good approach for cohorts coming from Chinese (C1), German and French (C4).

Another comparison between classifiers shown in Figure IV.24, Our framework shows improvements to RF for C1, C3, C4 while the framework performs nearly the same to RF for other datasets. In addition, except for C1,C5, CNN surpass the other, the networks face overfitting problems. For the larger number of features compared to datasets in group A, the architectures need to be more sophisticated to improve the performance.

IV.5.3 Applying Met2Img on Sokol's lab data

In this section, we evaluate the performance of Met2Img on Sokol's lab data [8], which used to evaluate Ph-CNN in [11]. The authors performed the six classification tasks on this data to classify HS versus the six partitions CDf, CDr, iCDf, iCDr, UCf, UCr. They proposed Ph-CNN considered as a novel DL architecture using phylogenetic structure of metagenomics for classification tasks. The structure of the model includes a stack of two 1D convolutional layers of 4x16 and a max pooling of 2x1, followed by a fully connected layer with 64 neurons and a dropout of 0.25. In order to compare to Ph-CNN, we use the same experimental procedure with 5-fold stratified cross validation repeated 10 times (internal validation) and compute Matthews Correlation Coefficient (MCC) as a performance measurement in [11], then applying the best model to the external validation datasets. In [99], MCC was considered as a good performance evaluation score for

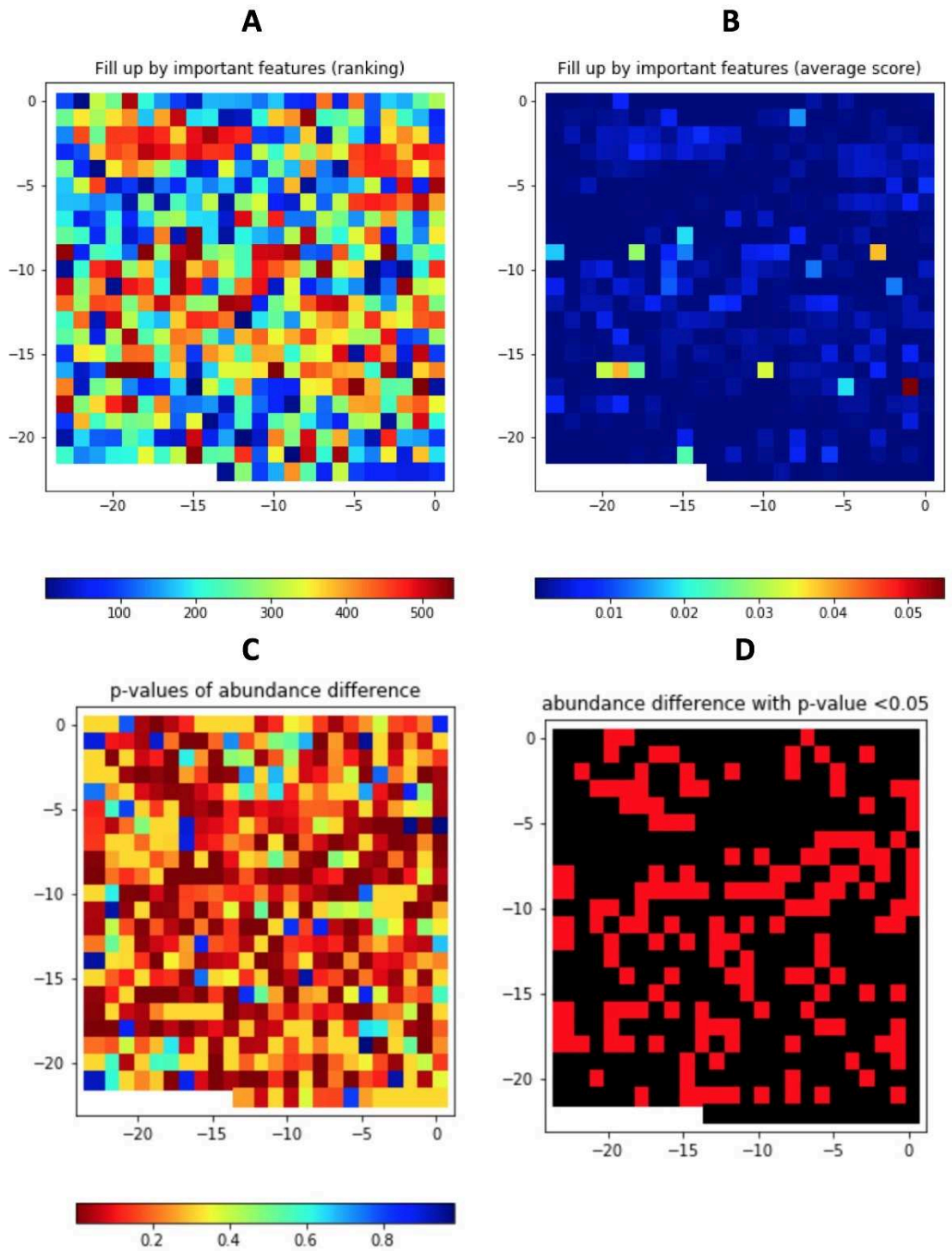

Figure IV.20 *Important features extracted from RF model of MetAML visualized by Fill-up including ranking (A) and average scores (B). The p-values reveal the differences in species abundance between control and patient samples in (C) and p-values that is less than 0.05 (significant differences) colored "red" in (D).*

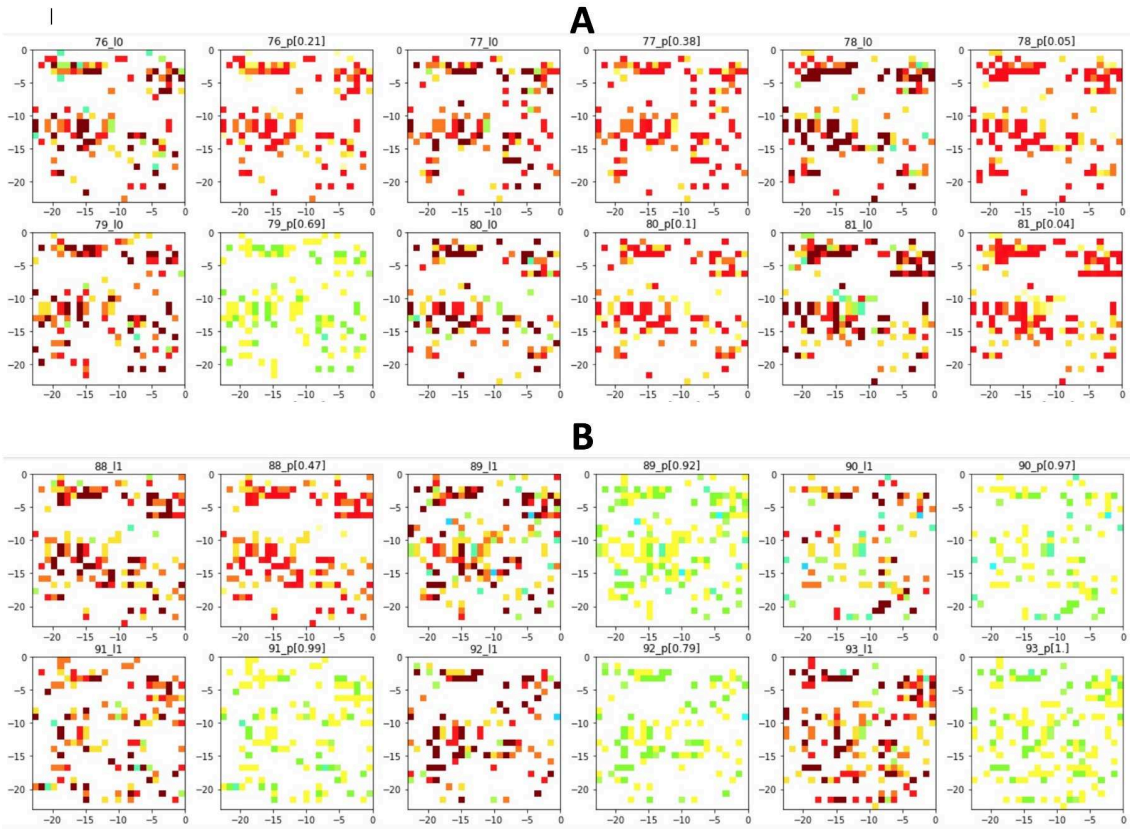

Figure IV.21 *Explanation Visualizations for the samples of controls (A) and patients (B) using LIME. Pros in green areas where are evidence for the disease class while Cons in red is against the class.*

imbalanced datasets and helps to evaluate whether the model is going well or not. The internal and external validation sets evaluated in our experiments are the same as in [11].

The performance in Table IV.7 using Fill-up with color images shows significant results compared to Ph-CNN. Especially, Fill-up using QTF outperforms most of datasets in the internal validation and get better performance in 5/6 external validation set. SPB is conducted from species abundance distribution also shows promising results with four significant results (using rainbow) on the internal validation sets. Interestingly, for QTF, although CNN model shows worse performance compared to FC, this model (with *viridis*) reveals encouraging results with better performance on 5/6 external validation sets. Additionally, viridis performs the best with CNN with an average performance of 0.817 on the external validation. This result is similar to the performance shown for datasets in group A.

As it is shown on Figure IV.25, we compare the best results of SPB (rainbow) and QTF (with *viridis*) compared to Ph-CNN. In most situations, our approaches outperform Ph-CNN. FC shows better MCC in numerous cases. For the external validation, Ph-CNN shows the worst performance on 4 out of 6 datasets including CDf, iCDf, UCf and UCr while FC and CNN-SBP-rainbow exhibit the best on 3 datasets.

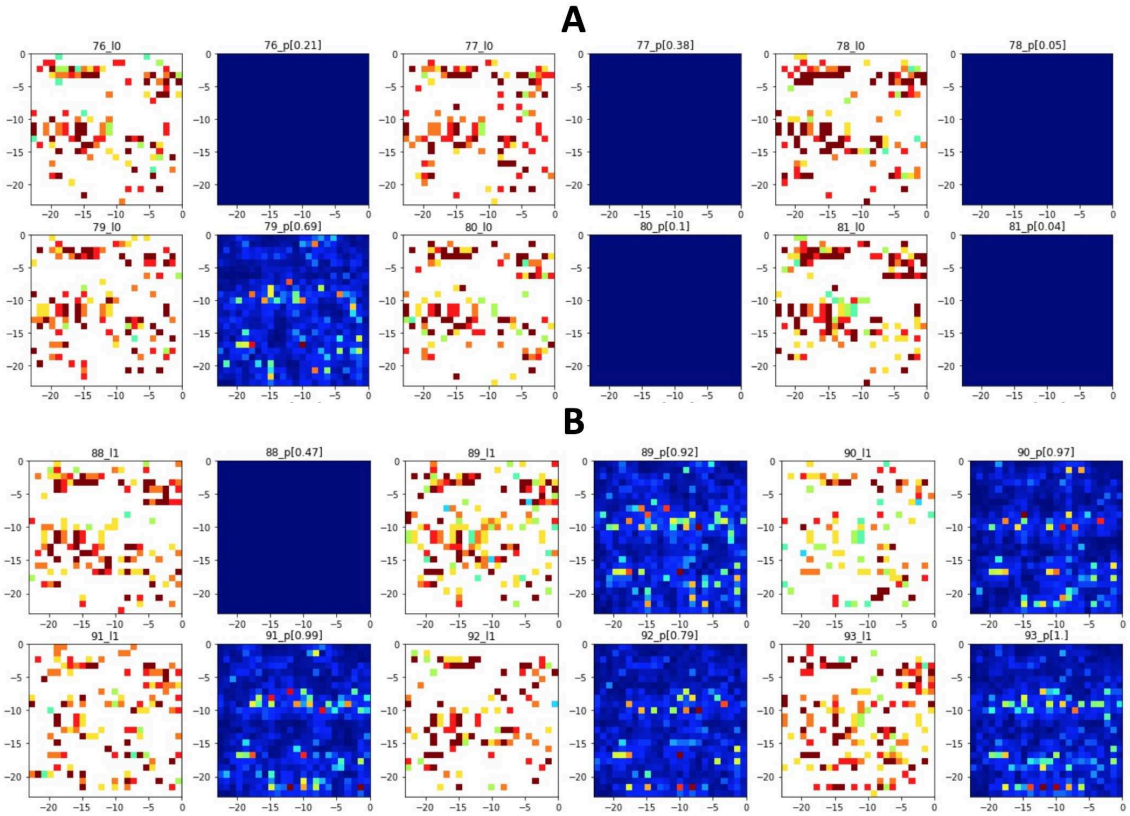

Figure IV.22 *Explanation Visualizations for the samples of controls (A) and patients (B) using Saliency Maps*

IV.5.4 Applying Met2Img on selbal's datasets

We also compare our approach to selbal method [194] in Figure IV.26. There are sinificant results compared to selbal from the CNN with Fill-up using QTF and gray images both datasets and one significant improvement on Crohn training with FC. HIV and Crohn datasets use counts of reads including the values being greater than 1 with different scopes. For Crohn dataset, we meet the situation where n (the number of samples) is greater than d (dimension of data) while other dataset is characterized with the problem of extreme class Imbalance with 83% samples belonging to Class 1. The results exhibit that our approach not only performs well on range [0,1] such as abundance data, but also obtains substantial achievements on different bounds of data.

IV.5.5 The results with gene-families abundance

IV.5.5.1 Applying dimensionality reduction algorithms

An important problem for nonlinear manifold learning is the way to solve new unseen data. These algorithms usually require to rerun the whole algorithm on the data including new data and old data. Therefore, some of them cannot be generalizable [88], although several studies have explored to solve this issue but the implementation is not commonly-used. In

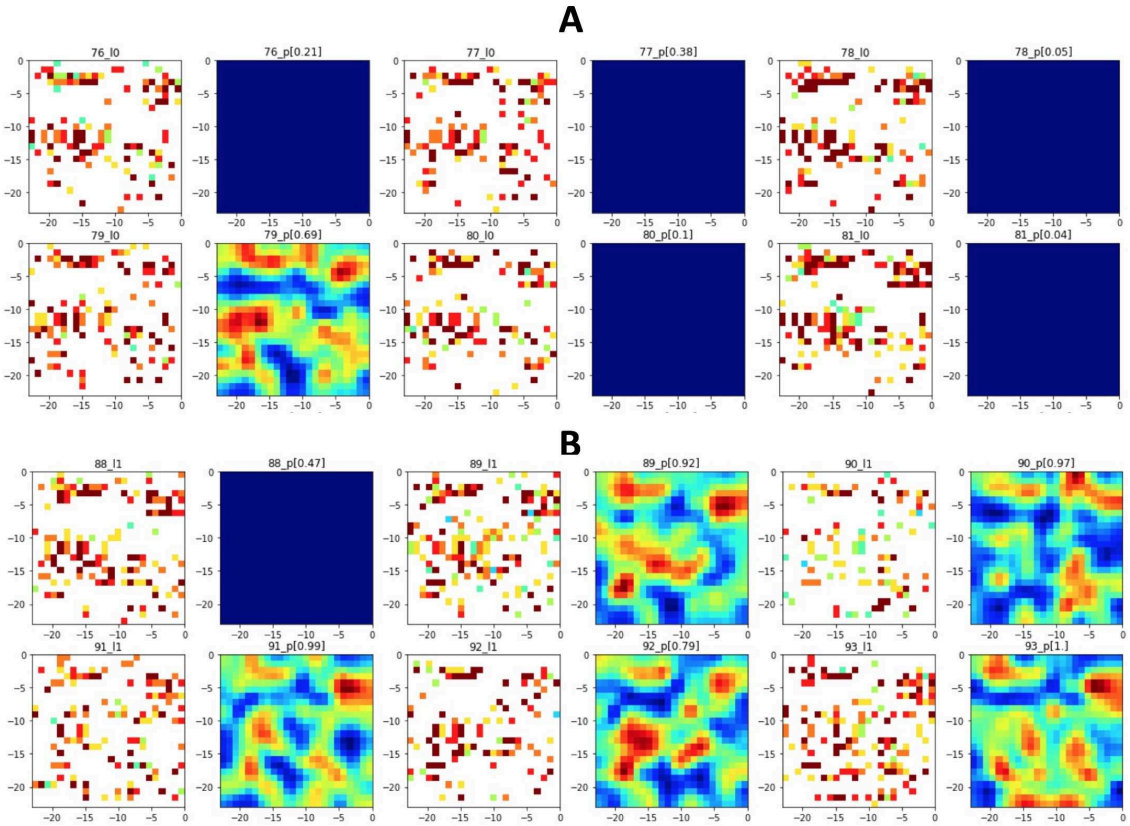

Figure IV.23 *Explanation Visualizations for the samples of controls (A) and patients (B) using Grad-CAM*

our study, we employ Random Projection [12, 117, 114, 116, 115] and PCA [77] algorithms to reduce dimension for gene abundance with their capabilities of generalization on unseen new data, other manifold learning approaches have still been investigating.

For gene families abundance, the number of features can be reached to more one million, so dimensionality reduction is very necessary. Here, we apply two well-known dimensionality reduction algorithms including Random Projection [12, 117, 114, 116, 115] and Principal Component Analysis (PCA) [77]. These are simple, and computationally efficient methods to reduce the data dimension in a unsupervised manner. For the type of data like gene abundance, the number of features may be up to over one million, these techniques seem to be useful methods. Random Projection helps to reduce the number of features as well as the size of the model while PCA captures well the variance of the original features and gives good combinations of features.

The results with dimensionality reduction: FC model using reduction algorithms of Random Projection (revealed as "RD_PRO" in Table IV.8) and PCA (the number of features is reduced from over one million to 576 which is equivalent to an image of 24×24), then applying Fill up approach with QTF with 10 and 255 bins for gray scale images (24×24). As we can see from Table IV.8, after reducing dimension, the performance is poor on new data. However, the performance is able to be improved to near to the performance of the original data. In addition, PCA shows a slight better performance

Bins	Color	#Bins	Model	C1	C2	C3	C4	C5	AVG
SPB	jet	10	CNN	0.688	0.754	0.716	0.774	0.749	0.736
SPB	viridis	10	CNN	0.653	0.722	**0.786**	0.782	0.730	0.734
SPB	gray	10	CNN	0.657	0.739	**0.794**	0.799	0.758	0.749
QTF	jet	10	CNN	**0.763**	0.730	0.718	0.791	0.762	0.753
QTF	viridis	10	CNN	**0.735**	0.741	0.731	0.805	0.763	0.755
QTF	jet	255	CNN	**0.772**	0.727	0.718	0.789	∗0.766	0.754
QTF	viridis	255	CNN	**0.724**	0.743	0.729	0.794	0.756	0.749
QTF	gray	10	CNN	**0.741**	0.739	**0.765**	0.809	∗0.769	0.765
QTF	gray	255	CNN	**0.746**	0.737	**0.765**	0.814	∗**0.776**	0.768
PR	gray	2	CNN	0.637	0.652	**0.754**	0.737	0.694	0.695
SPB	jet	10	FC	0.652	0.773	**0.768**	0.797	0.755	0.749
SPB	viridis	10	FC	**0.705**	0.756	**0.804**	0.805	0.747	0.763
SPB	gray	10	FC	0.693	0.767	**0.782**	**0.823**	0.757	0.764
QTF	jet	10	FC	**0.743**	0.710	**0.749**	0.804	0.732	0.748
QTF	viridis	10	FC	**0.708**	0.733	**0.761**	**0.824**	∗0.769	0.759
QTF	jet	255	FC	**0.755**	0.700	**0.753**	0.804	0.731	0.749
QTF	viridis	255	FC	**0.709**	0.735	**0.762**	**0.818**	∗**0.785**	0.762
QTF	gray	10	FC	0.689	0.763	**0.791**	0.815	0.758	0.763
QTF	gray	255	FC	0.689	0.763	**0.794**	**0.818**	0.762	0.765
PR	gray	2	FC	0.682	0.702	0.746	0.766	0.719	0.723
			RF	0.663	0.762	0.719	0.791	0.759	0.739
			SVM	0.526	0.643	0.545	0.750	0.649	0.623
SVM - the seven CRC-enriched bacterial species in [37]								0.750	

Table IV.6 – AUC performance for datasets in group C. The significant results (compared to RF) are reported in **bold**. The results marked ∗ are significant to [37] for C5. The average performance of five datasets is computed and shown in the last column (AVG)

comparing to Random Projection.

The average performance on raw data without any dimensionality reduction gives an accuracy of 0.681, while our approach using fill-up after reducing dimension with PCA improves the performance over 2% comparing to raw data. Moreover, PCA outperforms RD_PRO for fill-up and improves slightly for raw data. These results also reveal the ability of our framework with fill-up which increases the accuracy from 0.605 (raw- PCA) to 0.696/0.697 (fill up - PCA 10 bins / 255 bins). RD_PRO shows a worse performance, but it also shares the same pattern with PCA with the improvement in Fill-up. Also, we gain the benefits on execution time from dimensionality reduction (Table IV.9). The execution times are reduced even up to 20 times (T2D, raw data with 67.76 hours and after using RD_PRO: 3.37 hours!) Using more bins, this might help to reduce the time for learning. As we can see that images with 255 bins, the total execution time reduced from 36.35 hours to 29.58 hours for PCA while for RD_PRO, this decrease is trivial.

Internal Validation									
Color	Model	Bins	CDf	CDr	iCDf	iCDr	UCf	UCr	AVG
Ph-CNN			0.630	0.241	0.704	0.556	0.668	0.464	0.544
jet	CNN	QTF	0.678	**0.414**	0.758	0.550	**0.779**	0.489	0.611
rainbow	CNN	QTF	0.693	**0.344**	**0.766**	0.556	**0.767**	0.495	0.603
viridis	CNN	QTF	0.694	**0.357**	0.746	0.618	**0.811**	0.467	0.616
jet	CNN	SPB	**0.755**	**0.341**	**0.800**	0.433	0.730	0.519	0.596
rainbow	CNN	SPB	**0.808**	**0.369**	**0.862**	0.508	**0.790**	**0.610**	0.658
viridis	CNN	SPB	0.590	**0.377**	0.722	0.565	0.697	0.407	0.560
jet	FC	QTF	0.697	**0.409**	**0.792**	0.616	**0.791**	0.529	0.639
rainbow	FC	QTF	**0.731**	**0.389**	0.757	**0.631**	**0.837**	0.544	0.648
viridis	FC	QTF	**0.743**	**0.362**	**0.815**	**0.642**	**0.796**	0.519	0.646
jet	FC	SPB	**0.771**	**0.360**	**0.854**	0.508	**0.753**	**0.580**	0.638
rainbow	FC	SPB	**0.791**	0.292	**0.829**	0.531	**0.795**	**0.632**	0.645
viridis	FC	SPB	**0.768**	**0.374**	**0.822**	0.592	0.709	0.503	0.628
External Validation									
Ph-CNN			0.858	0.853	0.842	0.628	0.741	0.583	0.751
jet	CNN	QTF	***0.930***	0.805	***0.920***	***0.669***	0.565	0.329	0.703
rainbow	CNN	QTF	***0.930***	0.795	***0.920***	0.591	0.477	0.438	0.692
viridis	CNN	QTF	***0.930***	***0.868***	***0.919***	0.580	***0.826***	***0.777***	0.817
jet	CNN	SPB	***1.000***	0.425	***0.920***	0.497	***0.742***	0.329	0.652
rainbow	CNN	SPB	***1.000***	0.270	***1.000***	***0.664***	***0.840***	0.580	0.726
viridis	CNN	SPB	0.705	0.361	***1.000***	0.330	***0.826***	0.329	0.592
jet	FC	QTF	0.854	0.802	***1.000***	***0.919***	***0.837***	0.410	0.804
rainbow	FC	QTF	***0.930***	***0.868***	***0.919***	***0.698***	***0.837***	0.580	0.805
viridis	FC	QTF	***1.000***	***0.868***	0.842	0.514	***0.916***	***0.713***	0.809
jet	FC	SPB	0.854	0.654	***1.000***	0.628	***0.826***	0.580	0.757
rainbow	FC	SPB	0.854	0.654	***1.000***	***0.669***	***0.916***	***0.585***	0.780
viridis	FC	SPB	0.779	0.669	***1.000***	***0.749***	***0.826***	0.329	0.725

Table IV.7 – Classification performance (in MCC) compared to Ph-CNN on six classification tasks on IBD. The better results on the external validation sets are formatted ***bold-Italic***. The significant results are reported in **bold**. The average performances are revealed in the last column (AVG).

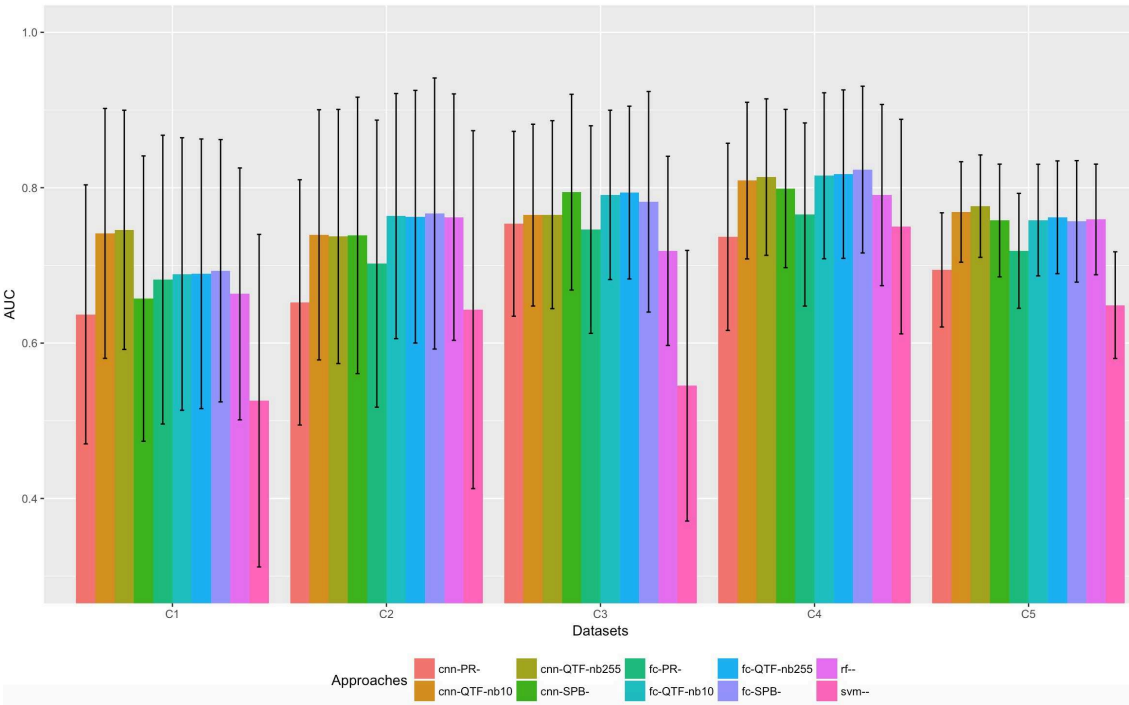

Figure IV.24 *Performance comparison of shallow learning algorithms and deep learning approaches on Colorectal cancer datasets (in AUC) using gray images of Fill-up. Error bars are standard deviations. Approaches: Model-Type of Bins-Number of bins (CNN-QTF-nb10 means using the model of CNN with 10 bins of QTF).*

IV.5.5.2 Comparing to standard machine learning methods

In order to evaluate the efficiency of our approach on gene families abundance compared to standard learning algorithms such as RF and SVM, we generate images from gene families abundance without dimensionality reduction algorithms to apply CNN. Due to some limitations of computational resource, we build small images with a resolution of 48×48 (See Figure IV.28 to see a comparison between a minimum size of images fitting all features and the compressed images with 48×48).

Table IV.10 and Figure IV.27 demonstrate the comparison results between classical learning methods (RF and SVM) and Fill-up approach. It is worthy to note that we only use images of 48×48 for gene families abundance while this data requires a minimum resolution of images being to up to more than 1000 pixels for each dimension (see Table IV.1). This means we compress data more than 20 times (see Figure IV.28). Interestingly, although we employ very small images compared to the minimum requirement, Fill-up obtains 3 significant results compared to RF. Another substantial point is that PR in gene families abundance reveals a similar result in comparison with species abundance. In addition, RF performs very well on OBE that it shows a poor performance on species abundance, while SVM exposes the worst among the considered approaches. Another noteworthy result is that PR outperforms SPB. A reason for this is SPB suffers a large problem of a enormous number of overlapped points with different colors in a very compact

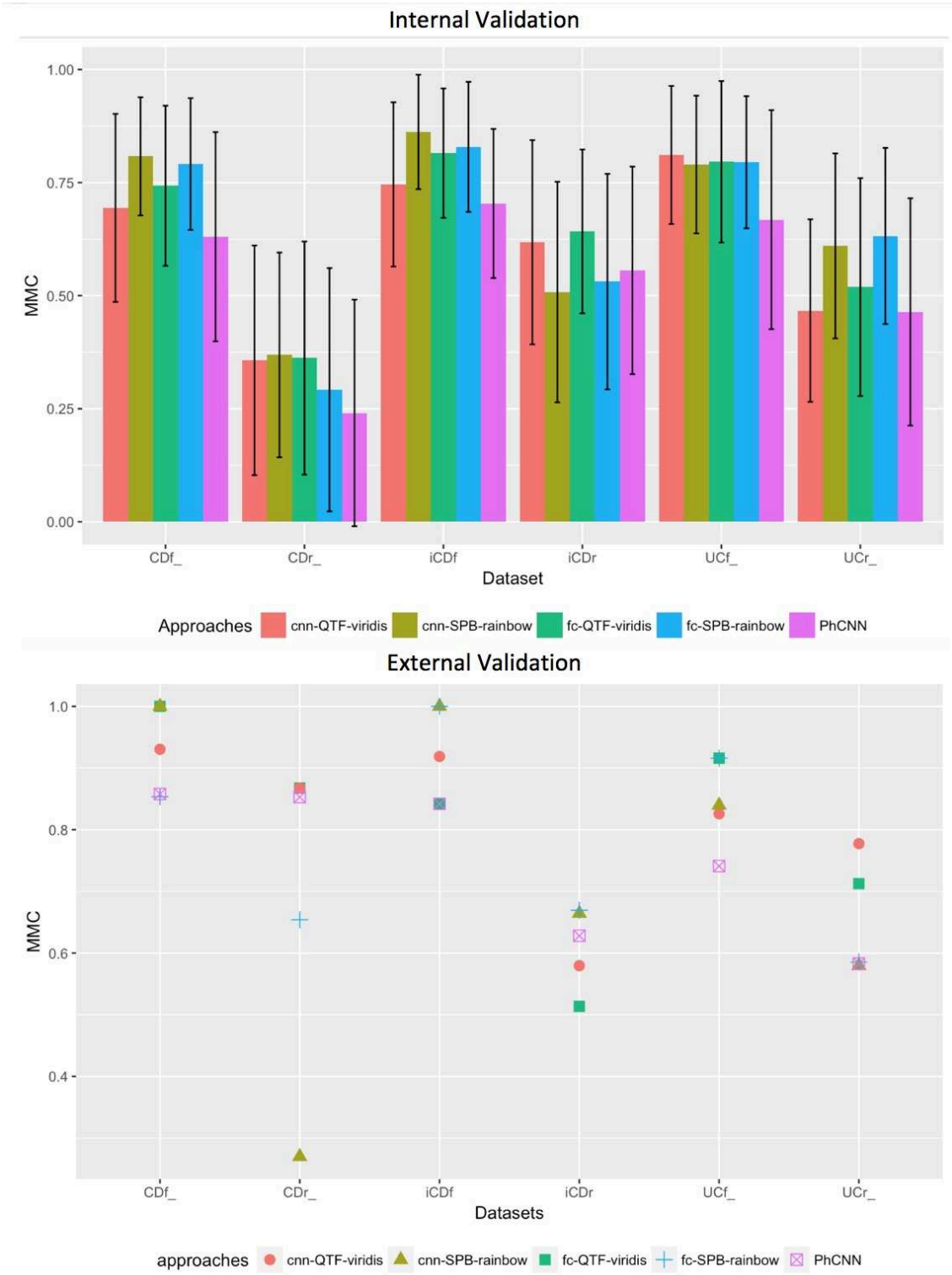

Figure IV.25 *Performance comparison (in MCC) of approaches on the internal and external validations. Error bars are standard deviations.*

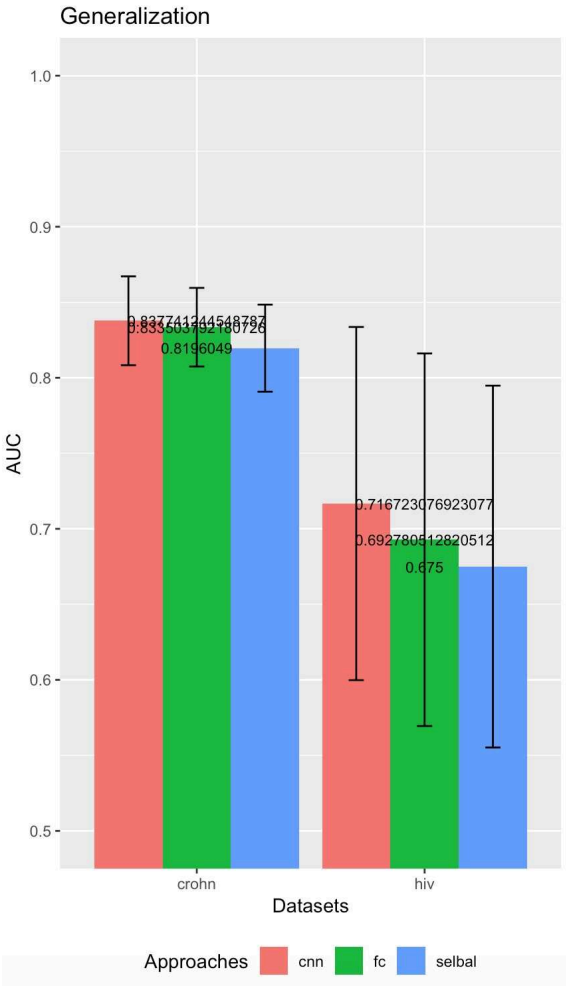

Figure IV.26 *Performance comparison (in AUC) of our approach and selbal. Error bars are standard deviations.*

space, while PR only needs to mark a black if the abundance is greater than 0, so it does not care how many points are overlapped.

From these results, our approach is really a promising technique for very high dimensional data both species and gene abundance.

IV.6 Closing remarks

In this chapter, we explored a the architecture of CNN to learn and classify complex metagenomic data using visual representations constructed with Fill-up and dimensionality reduction methods including t-SNE, Isomap, MDS, etc. Our results show that a simple architecture such as one-convolutional-layer achieves a better performance than a more complex one, and deeper architectures such as VGG-like.

We have observed that the Fill-up approach outperforms other dimensionality reduction methods tested to construct synthetic images. This may be due to several factors.

Model	#bins	Rep		**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
FC	10	Fill-up	PCA	0.757	**0.709**	**0.832**	**0.663**	0.626	0.593	0.697
FC	10	Fill-up	RD_PRO	0.701	**0.709**	0.785	0.658	0.581	0.526	0.660
FC	255	Fill-up	PCA	0.755	**0.707**	**0.829**	**0.662**	0.627	0.593	0.696
FC	255	Fill-up	RD_PRO	0.695	**0.706**	0.784	0.660	0.592	0.543	0.663
FC		Raw	PCA	0.547	0.604	0.775	0.648	0.514	0.540	0.605
FC		Raw	RD_PRO	0.555	0.605	0.775	0.648	0.496	0.530	0.602
FC		Raw		0.761	0.628	0.775	0.648	0.655	0.620	0.681

Table IV.8 – Results of gene families (Unit: Accuracy) Comparison between raw data and reduced dimensionality data using Random Projection (RD_PRO) and Principal Component Analysis (PCA) with the same model FC. The significant results compared to original abundance are reported in **bold** . The average performance of six datasets is computed and shown in the last column (AVG)

#bins	Rep		**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	SUM	AVG
10	Fill-up	PCA	6.87	5.22	4.92	5.86	9.68	3.80	36.35	6.06
10	Fill-up	RD_PRO	4.44	3.54	3.29	4.19	5.22	3.28	23.96	3.99
255	Fill-up	PCA	5.86	4.24	3.90	5.03	8.05	2.50	29.58	4.93
255	Fill-up	RD_PRO	3.29	4.04	4.13	2.66	3.75	3.87	21.74	3.62
	Raw	PCA	6.19	3.56	3.05	5.44	8.44	1.78	28.46	4.74
	Raw	RD_PRO	3.06	2.83	2.57	2.86	3.37	2.38	17.07	2.85
	Raw		37.63	19.41	18.85	20.99	67.76	13.67	178.31	29.72
		sum	67.34	42.85	40.71	47.03	106.26	31.27		
		avg	9.62	6.12	5.82	6.72	15.18	4.47		

Table IV.9 – Results of gene families (Unit: hours). Comparison time inference between raw data and reduced dimensionality data using RD_PRO (Random Projection) and Principal Component Analysis (PCA) with the same model FC. The average execution time of six datasets is computed and shown in the last column (AVG)

First, the features in the Fill-up are all visible while features in the other methods are often overlapping. Second, the Fill-up approach integrates prior knowledge on the phylogenetic classification of the features. In addition, the images based on manifolds learning are more complex than the Fill-up images. It is noteworthy that with the Fill-up we show significant improvements on four data sets, while the t-SNE reveals significant improvement on one data set, namely on the T2D. The FC model outperforms the CNN model in color images while CNN model achieves a better performance than the FC for gray and black/white images. Besides, the representations based on 2D images yield better results compared to 1D data. In general, the proposed Met2Img method outperforms the state-of-the-art both on species and genus abundance data.

Our approach is also evaluated on a number of different benchmarks including species abundance, genus abundance, and gene families abundance. These datasets are characterized by various dimensions ranging from thousand to millions features. The results show that increasing the number of considered attributes is also potential to enhance the

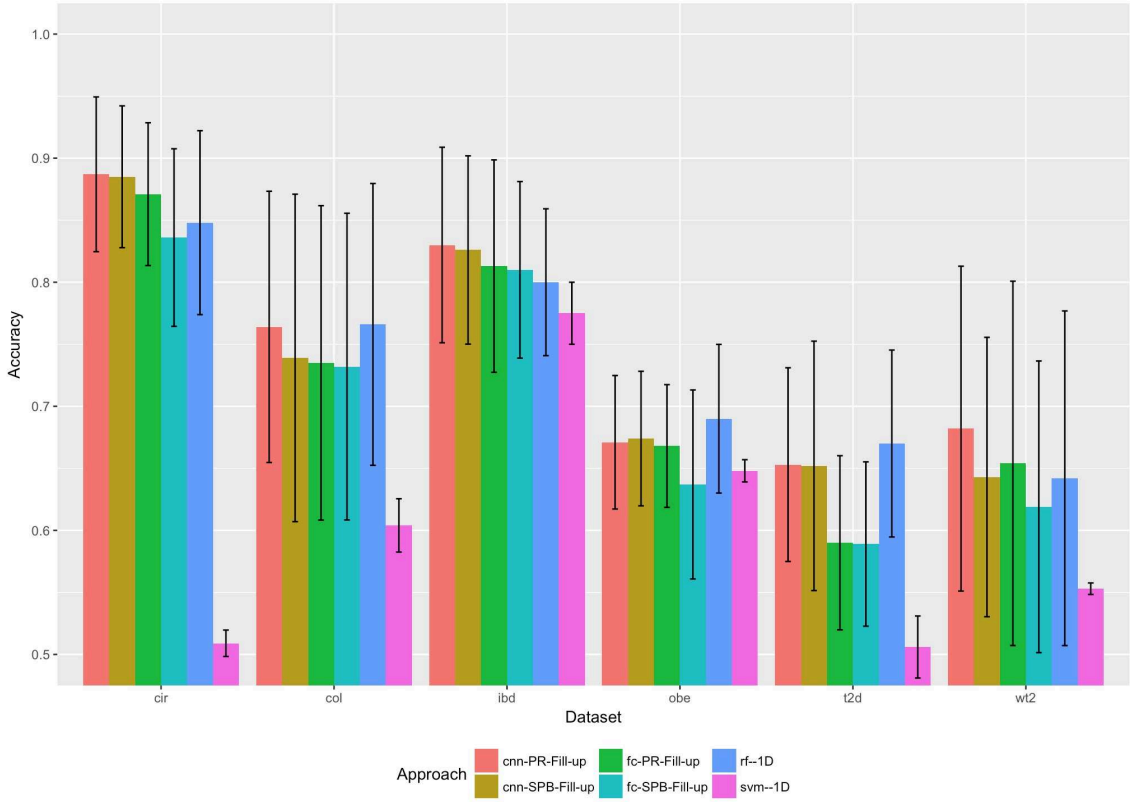

Figure IV.27 *The ACC performances of classical learning and our approach. Standard deviations are shown in error bars.*

	Model	Bins	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
Fill-up	CNN	PR	**0.887**	0.764	**0.830**	0.671	0.653	**0.682**	0.748
Fill-up	CNN	SPB	**0.885**	0.739	**0.826**	0.674	0.652	0.643	0.737
Fill-up	FC	PR	**0.871**	0.735	0.813	0.668	0.590	0.654	0.722
Fill-up	FC	SPB	0.836	0.732	0.810	0.637	0.589	0.619	0.704
1D	FC		0.761	0.628	0.775	0.648	0.655	0.620	0.681
1D	RF		0.848	0.766	0.800	0.690	0.670	0.642	0.736
1D	SVM		0.509	0.604	0.775	0.648	0.506	0.553	0.599

Table IV.10 – Performance comparison between classical learning algorithms and our approach. The average performance of six datasets is computed and shown in the last column (AVG). The results in bold is significant improvement compared to RF.

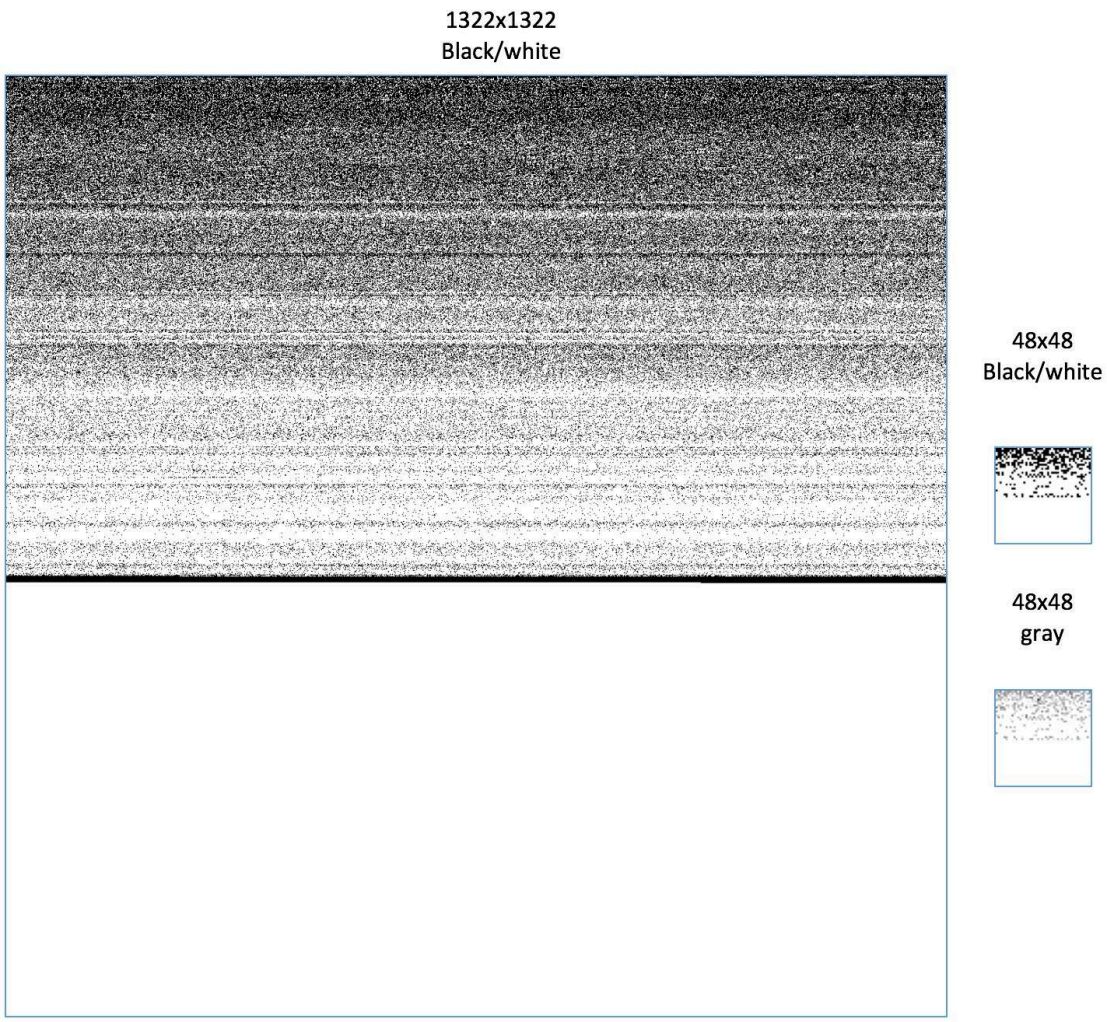

Figure IV.28 *An illustration of Fill-up from a sample of cirrhosis gene families abundance dataset with images of 1322×1322 and 48×48. 48×48 Black/white and 1322×1322 Black/white are generated with PR while 48×48 gray is applied by SPB.*

performance compared to classical machine learning methods.

Computational time of GPU for CNN is improved greatly compared to CPU, but there is no considerable difference for FC models. This indicates that GPU seems to only run efficiently with complex models.

Currently we are investigating various deep learning architectures, and also explore integration of other heterogeneous "omics" data.

Chapter V

Conclusion and Perspectives

V.1 Conclusion

The main contribution of this thesis is a framework called **Met2Img** that provides a novel approach for visualization of metagenomic data (Chapter III), and the corresponding architecture of the CNN applied to the visualizations (Chapter IV) to perform a classification. We also presented a feature selection framework to integrate heterogeneous high-dimensional sources based on the powerful visualization capabilities of the Self-Organising Maps (SOM) and predictive accuracy of Support Vector Machines (SVM) in Chapter II.

In Chapter II, we proposed a framework to reduce dimensionality by a deep approach based on SOM and a SVM. The deep structure aims to visualize clusters in 2D based on the robust capability of SOM. The method achieves reasonable results compared to the-state-of-the-art performance. The framework is expected to help to better stratify patients, and to develop approaches for personalized medicine.

In Chapters III and IV we presented a framework Met2Img that constructs synthetic images from metagenomic data, and it enables the powerful capabilities of deep learning in image classification. We explored a variety of visualization methods including the fill-up and manifold learning based approaches using various kinds of bins to reduce the effects of minor observation errors, and to get rid of noise in the data.

The experimental results are promising. On the public benchmark including 6 species abundance datasets (group A) related to five diseases with the number of features ranging from 381 to 572, Fill-up achieves significant improvements on 4 out of 6 datasets such as CIR, IBD, OBE, WT2 while visualization based on dimensionality reduction learning methods such as t-SNE, and NMF outperform on T2D dataset compared to the-state-of-the-art [18]. Although we cannot obtain any significant results on COL dataset in group A compared to RF, the performances on colorectal cancer datasets in group B are improved with 4 out of 5 datasets achieving significant results compared to RF as increasing the number of considered features (the number of considered OTUs is up to nearly 2000). Noteworthy, our approach shows significant improvements both FC and CNN models compared to the-state-of-the-art that used only seven enriched species found in [37]. Comparing to another state-of-the-art, Ph-CNN [11], on genus-level abundance (group C) with smaller number of features ranging from 237 to 257 features, our framework also

obtains encouraging results with most of cases of classification being significant results. Moreover, the framework performs well on the external validation sets. Comparing the results from gene families abundance (group D) with the number of features being up to more than one million, we also achieve some greater performances compared to RF, even though only very small images are carried out in this analysis. The results on gene families abundance using dimensionality reduction algorithms also exhibit considerably improvements for FC model in terms of execution time and accuracy. The results on read counts (group E) reveal that the proposed framework is efficient not only for abundance data, but also read counts.

The models of Fill-up integrated phylogenetic ordering information outperform random sorting on COL, IBD datasets. This exposes that the phylogenetic information integrated in the images can improve the performance. Furthermore, sorting species that has a relation on phylogenetic ordering side by side can be useful information for prediction.

We proposed and tested three efficient binning algorithms, namely **SPB** (Species Bin), **QTF** (quantile transformation) and **PR** (Presence/absence). The SPB is more efficient than the others for species abundance in most of situations, but QTF helps to improve remarkably the performance for Isomap and exhibits a substantial results for genus abundance. For the QTF bins, for each fold, data is transformed to a uniform distribution then creating images based on transformed data. Therefore, the QTF tends to consume more time in execution compared to SPB where the set of images is created once at the beginning. Although in general the PR seems to be less efficient than the SPB and the QTF, its performance also reaches the-art-of-the-art. Remarkably, the PR outperforms the SPB when we consider very highly compressed images for gene families abundance.

We explored about ten quantitative colormaps for the synthetic images. For data shaped as a uniform distribution, *viridis* performs well in the term of accuracy while *jet*, *rainbow* seem to be appropriate for distributions shaped as a bell-shaped. Furthermore, a gray scale is an appropriate choice, it is noteworthy that gray images present substantial results for both the FC (fully connected) and the CNN. *Grays* also often yields a better performance in CNN compared to FC. However, *grays* needs more epochs to converge compared to the color images. We found that perceptual multi-hue colormap such as *viridis* and single-hue colormaps such as grays perform well on big data. Our results suggest that *viridis* and *grays* can provide a sufficient discrimination for classification on large scale data.

The Fill-up which is a simple approach, outperforms manifold learning approaches in most of cases. The first reason for it, is that images generated by manifold learning methods contain overlapped points while all features in the Fill-up are visible. Second, images visualized by manifold learning are rather complex what also requires more complex architectures to achieve a good performance. As shown in the thesis, the CNN perform better than the fully connected in the majority of our experiments for gray images based on manifold learning approaches. Although the Fill-up usually exposes poor performance on T2D, the t-SNE obtains a significant improvement on this dataset. Additionally, the t-SNE performs well on original data while the Isomap shows the best efficiency on transformed data formed a uniform distribution.

The analysis of various CNN structures reveals that the performance can degrade as the number of convolutional layer increases. In this case, instead of increasing the depth

of the network, it is better to expand the width of the network by increasing the number of filters to attempt to improve the performance. Moreover, we modified the EarlyStopping criterion in Keras.callback that leads to a substantial improvement on the T2D, and also on all other tested datasets.

The Random Forest performs implicit feature selection and usually produces a model with the most accurate performance, but it is still a black box. For high-dimensional data, it is a challenge to explain and interpret the data and the estimated models. Our framework does not only provide visualization methods for features mapped into 2-dimensional images which can be investigated by human experts, but also, the framework produces an outstanding performance compared to original data. Additionally, we also explored recent methods to interpret complex machine learning models such as LIME, Grad-Cam, and Saliency to find explanations for the output of the CNN. We were able to discover patterns in images that can be evidences and biomedical signatures in prediction tasks. Comparing the extracted patterns to important features obtained by the RF of MetAML visualized using Fill-up, we have noticed an important overlap between them. These biological markers are expected to discriminate patients, and to help to develop approaches of personalized medicine.

We analyzed the execution time, and compared between CPU and GPU. We noticed that the GPU works very efficient for the CNN while on simple models such as FC, the GPU does not show improvements in the speed of processing. We could propose to utilize GPU for CNN and CPU for modest learning algorithm such as FC.

We also introduce a publicly available module, namely **Met2Img**, at https://git.integromics.fr/published/deepMG_tf. It is implemented in Python, and it provides not only a framework to visualize the OTUs as synthetic images (in order to further apply deep learning techniques), but it also facilitates follow-up studies and evaluation of new methods for disease classification. The module also supports a variety of different measures such as ACC (predictive empirical accuracy), AUC, MCC (Matthews Correlation Coefficient), loss value that are available for all folds, and other scores such as F1-score, precision, and confusion matrix. This information enables to evaluate the efficiency of new methods. We introduced a number of options that can be used within the module. These parameters help the users to use Met2Img. The users are welcome to test variations of deep learning techniques such CNN1D, CNN2D, VGG, FC, Long Short Term Memory networks (LSTM), as well classical learning algorithms, for instance, SVM and RF. There are a number of possibilities to generate synthetic images that are supplied such as different resolution, and a diversity of visualization methods based on manifold learning. In this thesis, we report that the Met2Img is a useful and promising tool to assess and to comparing various human diseases based on microbiome features.

V.2 Future Research Directions

Our preliminary studies on visualization of microbiome data could lead to several directions for future work. The proposed framework explores potential use on metagenomic data. However, although the Met2Img is only studied for metagenomics, the method can be applied directly to any other "omics" data, in order to perform a more advanced function-based analysis.

The increasing amount of data from multiple sources leads to more requirements for heterogeneous data integration. The prediction of various diseases can be more accurate if the classifiers combine these heterogeneous data more efficiently. A combination of metagenomic data and any other data such as dietary, environmental factors, and etc. enables a better diagnostic accuracy. Moreover, the deep learning in general, and the convolutional neural networks in particular, is an active domain of research where exciting performance results are reported regularly. More and more architectures of CNN are proposed to improve the performance and perform better than humans in numerous applications [183]. If the performance of the deep learning methods applied to image processing will improve further, the ideas to present data as images for efficient analysis have a good soil to come true. Additionally, the studies on interpretability such as Grad-CAM, LIME that have been currently developed and investigated enable us to get understanding on the decision of the models and extract signatures from explanations. Combining such tools with the trained models in real-time application might help biologists to stratify patients and find signals of diseases better.

Nowadays, the speed of metagenomic data processing (gene families abundance) is rather slow, and recent advances in distributed algorithms and parallel computations are badly needed to reduce the execution time. Currently, generating synthetic images consumes a considerable time and memory for high-dimensional data such as gene abundance matrices, especially for the QTF. In [37], the authors computed the significance of available bacterial abundance changes in colorectal cancer datasets to extract seven bacterial markers for diagnosis of early-stage colorectal cancer. This feature extraction method [37] that identified seven bacterial species revealing differential abundance in CRC compared to controls across all the four cohorts, is a great potential to reduce the dimensionality of gene families abundance data leading to smaller resolution images, and the training procedure will be much less complex. Such results encourage us to go deeper in visualization of significant features in settings where the number of features is extremely high. However, these methods can demand much more time for calculation of statistical significance of the OTUs. Therefore, there is a need to find a trade off between inference time and accuracy.

The CNN with quite deep architectures still face overfitting while increasing the width of the architecture can improve the performance. In this thesis, we only considered small images ranging from 16×16 to 48×48 so requirements on CNN are not quite complex. However, larger data such as gene abundance with the number of features up to millions, deeper architectures should be investigated precisely. Hyper-parameters tuning for CNN also is a limitation and consumes so much time. Our study has not evaluated the efficiency of applying the pre-trained network in natural images such as VGG16, VGG19, ResNet50, etc. A reason is that most of analysis in this thesis focuses on small images while input for those networks usually require very large images such as 224×224. Furthermore, there are some limitations of computational resources, so such heavy networks have not been examined completely yet. Therefore, this problem should explored more advanced strategies to further improve the classification performance.

The t-SNE is a widely used approach for visualization. However, it suffers from not being generalizable, i.e how to handle new data. When we performed the t-SNE on training set, we also have to rerun the entire algorithm on the dataset if new data were added. Another problem related to the t-SNE is that the standard t-SNE procedure needs

a huge amount of memory [190], and the execution time is also very high. The program might crash when we build the t-SNE maps on big data. As in [190], applicability of standard t-SNE is limited to a few thousand samples. The use of Barnes-Hut [191] in t-SNE enables to embed datasets with more than a million data points [190]. However, Barnes-Hut-SNE is only able to embed data into two or three dimensions because the size of tree in Barnes-Hut algorithm develops exponentially as increasing the dimensionality of data [190]. The next step would be to improve Met2Img in terms of memory consumption, and inference time using Model-Parallel, distributed and scalable Frameworks such as Apache Spark [188], PySpark [189].

Appendices

Appendix A

The contributions of the thesis

The contributions of this thesis include three publications:

- N. Sokolovska, H. T. Nguyen, K. Clément, J.-D. Zucker. *Deep Self-Organizing Maps for Efficient Heterogeneous Biomedical Signatures Extraction.* International Joint Conference on Neural Networks (IJCNN), 2016, pages 5079-5086, IEEE, Vancouver, Canada.
- T. H. Nguyen, Y. Chevaleyre, E. Prifti, N. Sokolovska, J.-D. Zucker. *Deep Learning for Metagenomic Data: using 2D Embeddings and Convolutional Neural Networks.* NIPS 2017 Workshop on Machine Learning for Healthcare. In Proceedings of the NIPS ML4H 2017 Workshop in Long Beach, CA, USA.
- T. H. Nguyen, E. Prifti, Y. Chevaleyre, N. Sokolovska, J.-D. Zucker. *Disease Classification in Metagenomics with 2D Embeddings and Deep Learning.* In Proceedings of Conférence d'Apprentissage (CAP) 2018, Rouen, France.

A manuscript presenting the results of chapters III and IV is planed to submit to the *Scientific Reports*.

Appendix B

Taxonomies used in the example illustrated by Figure III.7

Class name	**Color_index**	**#species**
Methanobacteria	1	3
Methanococci	2	1
Acidobacteriia	3	1
Actinobacteria	4	53
Bacteroidetes_noname	5	2
Bacteroidia	6	93
Flavobacteriia	7	3
Sphingobacteriia	8	2
Candidatus_Saccharibacteria_noname	9	3
Chlorobia	10	1
Deinococci	11	1
Bacilli	12	85
Clostridia	13	109
Erysipelotrichia	14	23
Negativicutes	15	30
Fusobacteriia	16	13
Alphaproteobacteria	17	3
Betaproteobacteria	18	24
Deltaproteobacteria	19	5
Epsilonproteobacteria	20	8
Gammaproteobacteria	21	65
Spirochaetia	22	5
Synergistia	23	3
Mollicutes	24	1
Verrucomicrobiae	25	1
Eurotiomycetes	26	1
Saccharomycetes	27	3
		542

Table B.1 – Information on the number of species of classes in CIR dataset along with the index color and abundances shown in Figure III.9

Order name	**Color_index**	**#species**
Methanobacteriales	1	3
Methanococcales	2	1
Acidobacteriales	3	1
Actinomycetales	4	21
Bifidobacteriales	5	14
Coriobacteriales	6	18
Bacteroidetes_noname	7	2
Bacteroidales	8	93
Flavobacteriales	9	3
Sphingobacteriales	10	2
Candidatus_Saccharibacteria_noname	11	3
Chlorobiales	12	1
Deinococcales	13	1
Bacillales	14	13
Lactobacillales	15	72
Clostridiales	16	109
Erysipelotrichales	17	23
Selenomonadales	18	30
Fusobacteriales	19	13
Rhizobiales	20	2
Rhodospirillales	21	1
Burkholderiales	22	12
Gallionellales	23	1
Neisseriales	24	11
Desulfovibrionales	25	5
Campylobacterales	26	8
Aeromonadales	27	4
Alteromonadales	28	1
Cardiobacteriales	29	3
Enterobacteriales	30	38
Oceanospirillales	31	1
Pasteurellales	32	13
Pseudomonadales	33	4
Xanthomonadales	34	1
Spirochaetales	35	5
Synergistales	36	3
Mycoplasmatales	37	1
Verrucomicrobiales	38	1
Eurotiales	39	1
Saccharomycetales	40	3
		542

Table B.2 – Information on the number of species of classes in CIR dataset along with the index color shown in Figure III.8

Appendix C

Some other results on datasets in group A

	Bins	MODEL	Color	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**
	MetAML - RF			0.060	0.095	0.070	0.039	0.073	0.147
Fill-up	PR	CNN	BW	0.077	0.122	0.081	0.053	0.083	0.136
Fill-up	QTF	CNN	viridis	0.069	0.117	0.084	0.058	0.083	0.150
Fill-up	SPB	CNN	custom	0.072	0.119	0.086	0.058	0.092	0.140
Fill-up	SPB	CNN	gray	0.071	0.120	0.081	0.066	0.094	0.132
Fill-up	SPB	CNN	jet	0.068	0.104	0.082	0.063	0.087	0.146
Fill-up	PR	FC	BW	0.077	0.117	0.083	0.061	0.090	0.143
Fill-up	QTF	FC	viridis	0.067	0.125	0.087	0.062	0.087	0.142
Fill-up	SPB	FC	grays	0.060	0.120	0.092	0.060	0.077	0.145
Fill-up	SPB	FC	jet	0.059	0.128	0.089	0.065	0.077	0.127
Fill-up	SPB	FC	custom	0.058	0.119	0.089	0.064	0.077	0.129
Fill-up $*$	PR	CNN	BW	0.069	0.108	0.082	0.059	0.080	0.132
Fill-up $*$	SPB	CNN	grays	0.065	0.104	0.085	0.064	0.076	0.134
Fill-up $*$	PR	FC	BW	0.077	0.122	0.084	0.062	0.086	0.153
Fill-up $*$	SPB	FC	grays	0.064	0.123	0.089	0.056	0.078	0.154
isomap	QTF	CNN	grays	0.067	0.111	0.085	0.067	0.081	0.150
isomap	QTF	CNN	grays	0.064	0.103	0.087	0.063	0.074	0.156
isomap	QTF	FC	grays	0.073	0.119	0.057	0.056	0.078	0.132
isomap	QTF	FC	grays	0.073	0.119	0.056	0.057	0.077	0.133
TSNE	PR	CNN	BW	0.074	0.107	0.077	0.055	0.079	0.128
TSNE	PR	FC	BW	0.077	0.116	0.078	0.054	0.083	0.147
TSNE	QTF	CNN	viridis	0.080	0.123	0.076	0.057	0.085	0.136
TSNE	QTF	CNN	jet	0.074	0.127	0.082	0.061	0.083	0.144
TSNE	QTF	CNN	grays	0.075	0.129	0.081	0.062	0.080	0.138
TSNE	QTF	FC	viridis	0.074	0.117	0.072	0.042	0.079	0.155
TSNE	QTF	FC	jet	0.073	0.120	0.076	0.056	0.097	0.157
TSNE	QTF	FC	grays	0.074	0.117	0.049	0.050	0.083	0.144
TSNE	SPB	CNN	jet	0.065	0.102	0.087	0.061	0.076	0.129
TSNE	SPB	CNN	viridis	0.062	0.108	0.074	0.062	0.087	0.152
TSNE	SPB	CNN	grays	0.070	0.107	0.086	0.062	0.077	0.142
TSNE	SPB	FC	jet	0.061	0.117	0.080	0.064	0.075	0.167
TSNE	SPB	FC	viridis	0.063	0.117	0.071	0.058	0.072	0.139
TSNE	SPB	FC	grays	0.064	0.115	0.055	0.056	0.074	0.157

Table C.1 – Standard deviation (of ACC) results of Fill-up and manifold learning methods presented in Table IV.5

Model	Color	CIR	COL	IBD	OBE	T2D	WT2
CNN	custom	**1.1E-02**	0.952	**1.3E-06**	**4.3E-06**	0.727	0.413
CNN	gist_rainbow	0.144	0.875	**3.4E-03**	**5.0E-04**	0.844	0.965
CNN	Grays	**4.8E-04**	0.805	**1.9E-09**	**7.3E-07**	0.891	0.450
CNN	jet	**7.6E-04**	0.726	**4.5E-08**	**2.2E-07**	0.933	0.295
CNN	nipy_spectral	0.735	1.000	0.418	0.100	0.965	0.994
CNN	Paired	1.000	1.000	0.911	0.092	0.995	1.000
CNN	rainbow	**2.1E-02**	0.987	**6.7E-09**	**1.1E-04**	0.998	0.294
CNN	reds	0.077	0.989	**1.7E-05**	**1.4E-03**	0.663	0.483
CNN	viridis	0.372	0.997	0.066	**2.4E-03**	0.965	0.998
CNN	YlGnBu	**1.9E-02**	0.986	**2.6E-05**	**6.5E-04**	0.841	0.474
FC	custom	**1.6E-04**	0.497	**5.1E-03**	**4.6E-06**	0.969	0.364
FC	gist_rainbow	**6.9E-03**	0.848	**6.9E-03**	**1.6E-06**	0.996	0.664
FC	Grays	0.075	0.991	**1.9E-04**	**1.6E-09**	0.892	0.233
FC	jet	**6.9E-05**	0.785	**3.2E-03**	**1.4E-06**	0.718	0.272
FC	nipy_spectral	0.338	1.000	0.082	**4.1E-04**	1.000	0.067
FC	Paired	1.000	1.000	0.651	**3.0E-04**	1.000	0.836
FC	rainbow	**4.9E-04**	0.999	**4.3E-05**	**6.1E-07**	0.999	0.447
FC	reds	**2.8E-02**	0.940	**2.9E-04**	**1.2E-06**	0.975	0.053
FC	viridis	**8.8E-03**	0.999	**2.7E-04**	**7.3E-09**	0.999	0.483
FC	YlGnBu	**1.1E-02**	0.981	**1.2E-04**	**7.2E-06**	0.942	0.117

Table C.2 – P-values of Fill-up SPB using different colormaps compared t-test to MetAML's results (in ACC).

		CIR	COL	IBD	OBE	T2D	WT2
cnn	Custom	**9.2E-04**	0.981	0.187	**9.5E-03**	0.075	0.997
cnn	gist_rainbow	**6.8E-03**	0.975	**3.9E-02**	0.053	0.238	0.988
cnn	Grays	**2.7E-04**	0.996	0.099	**1.9E-05**	0.638	0.992
cnn	jet	**4.0E-05**	0.996	0.056	0.287	0.459	0.999
cnn	nipy_spectral	0.435	1.000	0.571	0.244	1.000	1.000
cnn	Paired	0.998	1.000	0.916	**1.7E-04**	0.886	0.921
cnn	rainbow	**1.5E-02**	0.997	0.218	0.359	0.506	0.983
cnn	reds	**1.7E-03**	0.999	0.203	**6.6E-03**	0.116	0.967
cnn	viridis	**6.9E-03**	0.961	**2.2E-03**	**1.0E-02**	0.510	0.757
cnn	YlGnBu	**1.1E-03**	0.998	0.102	**3.6E-02**	0.404	0.885
fc	Custom	**6.3E-03**	0.885	**1.2E-02**	**3.1E-03**	0.173	0.874
fc	gist_rainbow	0.089	0.941	**2.6E-02**	**1.2E-02**	0.995	0.922
fc	Grays	**5.4E-03**	0.987	**5.2E-04**	**2.2E-07**	0.415	0.823
fc	jet	**3.2E-04**	0.968	**3.7E-04**	**5.2E-03**	0.174	1.000
fc	nipy_spectral	0.756	1.000	0.100	**9.7E-03**	1.000	0.913
fc	Paired	1.000	1.000	**2.1E-02**	**1.4E-03**	1.000	0.087
fc	rainbow	**1.2E-04**	0.929	**1.0E-04**	**1.1E-02**	0.395	0.995
fc	reds	**3.1E-03**	0.880	**1.0E-03**	**3.6E-03**	0.811	0.692
fc	viridis	0.111	0.997	**9.7E-06**	**3.0E-02**	0.991	0.333
fc	YlGnBu	**1.9E-04**	0.913	**3.6E-04**	**2.5E-03**	0.639	0.616

Table C.3 – P-values of Fill-up QTF using different colormaps compared t-test to MetAML's results (in ACC).

Algorithm	Model	Colors	CIR	COL	IBD	OBE	T2D	WT2	AVG
LDA	CNN	custom	**0.899**	0.771	0.803	**0.674**	0.656	0.664	0.744
LDA	CNN	grays	**0.900**	0.746	0.809	**0.671**	0.661	0.675	0.744
LDA	CNN	jet	**0.896**	0.743	0.794	**0.665**	0.661	0.652	0.735
LDA	CNN	viridis	0.883	0.764	0.818	**0.660**	0.654	0.689	0.745
LDA	FC	custom	0.875	0.779	0.819	**0.689**	0.604	0.682	0.741
LDA	FC	grays	0.873	0.736	0.792	**0.679**	0.631	0.687	0.733
LDA	FC	jet	0.875	0.761	0.818	**0.669**	0.637	0.649	0.735
LDA	FC	viridis	0.858	0.740	**0.829**	**0.666**	0.603	0.716	0.735

Table C.4 – Performance (in ACC) comparison of a variety of colormaps using LDA (with labels from a (family) taxonomic group and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

Algorithm	Model	Colors	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
PCA	CNN	custom	0.886	0.746	**0.824**	0.645	0.663	0.677	0.740
PCA	CNN	grays	0.890	0.744	0.815	**0.661**	0.678	0.686	0.745
PCA	CNN	jet	0.890	0.750	0.822	**0.659**	0.678	0.674	0.746
PCA	CNN	viridis	0.883	0.745	**0.838**	0.653	0.670	0.687	0.746
PCA	FC	custom	0.889	0.776	**0.837**	**0.673**	0.642	0.684	0.750
PCA	FC	grays	**0.893**	0.749	0.796	**0.669**	0.654	0.684	0.741
PCA	FC	jet	**0.899**	0.763	**0.838**	**0.672**	0.656	0.670	0.750
PCA	FC	viridis	0.880	0.760	**0.842**	**0.663**	0.621	0.688	0.742

Table C.5 – Performance (in ACC) comparison of a variety of colormaps using PCA and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

Algorithm	Model	Colors	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
NMF	CNN	custom	0.883	0.756	**0.832**	0.646	0.675	0.639	0.739
NMF	CNN	grays	0.883	0.742	**0.826**	**0.665**	**0.683**	0.635	0.739
NMF	CNN	jet	0.890	0.747	**0.831**	**0.654**	0.670	0.626	0.736
NMF	CNN	viridis	0.881	0.750	**0.826**	0.653	0.668	0.649	0.738
NMF	FC	custom	**0.890**	0.780	**0.837**	**0.674**	0.644	0.659	0.747
NMF	FC	grays	**0.893**	0.756	0.790	**0.676**	0.655	0.686	0.743
NMF	FC	jet	**0.898**	0.757	**0.826**	**0.670**	0.648	0.662	0.744
NMF	FC	viridis	0.886	0.761	**0.842**	**0.666**	0.619	0.702	0.746

Table C.6 – Performance (in ACC) comparison of a variety of colormaps using NMF and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

Algorithm	Model	Colors	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
MDS	CNN	custom	0.878	0.753	**0.833**	**0.670**	0.669	0.659	0.744
MDS	CNN	grays	0.881	0.741	0.816	**0.685**	0.667	0.670	0.743
MDS	CNN	jet	0.879	0.749	0.811	**0.670**	0.663	0.660	0.739
MDS	CNN	viridis	0.876	0.763	0.818	**0.672**	0.653	0.660	0.740
MDS	FC	custom	0.882	0.776	**0.843**	**0.683**	0.634	0.687	0.751
MDS	FC	grays	0.889	0.767	0.782	**0.668**	0.637	0.685	0.738
MDS	FC	jet	**0.897**	0.763	**0.824**	**0.678**	0.636	0.661	0.743
MDS	FC	viridis	0.883	0.738	**0.832**	**0.664**	0.597	0.683	0.733

Table C.7 – Performance (in ACC) comparison of a variety of colormaps using MDS and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

Algorithm	Model	Colors	**CIR**	**COL**	**IBD**	**OBE**	**T2D**	**WT2**	AVG
ISOMAP	CNN	grays	0.889	0.743	0.815	**0.691**	0.657	0.712	0.751
ISOMAP	CNN	jet	0.887	0.746	0.824	**0.683**	0.658	0.687	0.747
ISOMAP	CNN	viridis	0.881	0.744	**0.837**	**0.661**	0.651	0.713	0.748
ISOMAP	CNN	custom	0.883	0.757	**0.830**	**0.675**	0.638	0.706	0.748
ISOMAP	FC	grays	0.871	0.742	0.793	**0.688**	0.646	0.689	0.738
ISOMAP	FC	jet	0.887	0.752	0.815	**0.671**	0.629	0.662	0.736
ISOMAP	FC	viridis	0.862	0.745	**0.829**	**0.662**	0.605	0.705	0.735
ISOMAP	FC	custom	0.880	0.761	**0.838**	**0.694**	0.624	0.709	0.751

Table C.8 – Performance (in ACC) comparison of a variety of colormaps using Isomap and QTF. The significant results (compared to MetAML) are reported in **bold**. The average performance of six datasets are computed and shown in the last column (AVG). Models are detailed in IV.4.3

List of Figures

List of Tables

Bibliography

[1] R. Collobert, K. Kavukcuoglu & C. Farabet; «Torch7: A Matlab-like Environment for Machine Learning»; dans «BigLearn, NIPS Workshop», (2011). Cited on page 5

[2] F. H. Karlsson, V. Tremaroli, I. Nookaew, G. Bergström, C. J. Behre, B. Fagerberg, J. Nielsen & F. Bäckhed; «Gut metagenome in European women with normal, impaired and diabetic glucose control»; Nature **498**, p. 99–103 (2013). Cited on pages xviii, 27, and 67

[3] L. v. d. Maaten & G. Hinton; «Visualizing Data using t-SNE»; J. Mach. Learn. Res. **9**, p. 2579–2605 (2008). Cited on pages xiv, 27, 28, and 52

[4] G. Ditzler, G. Rosen & R. Polikar; «Forensic identification with environmental samples»; dans «2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)», p. 1861–1864. Cited on pages 54, 55, and 119

[5] L. Deng & D. Yu; «Deep Learning: Methods and Applications»; **7**, p. 197–387 (2014). ISSN 1932-8346, 1932-8354. https://nowpublishers.com/article/Details/SIG-039. Cited on pages x, 3, and 5

[6] W. Li & A. Godzik; «Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences»; **22**, p. 1658–1659. ISSN 1367-4803. https://academic.oup.com/bioinformatics/article/22/13/1658/194225/Cd-hit-a-fast-program-for-clustering-and-comparing. Cited on page 55

[7] M. D. Zeiler, G. W. Taylor & R. Fergus; «Adaptive Deconvolutional Networks for Mid and High Level Feature Learning»; dans «Proceedings of the 2011 International Conference on Computer Vision», ICCV '11; p. 2018–2025 (IEEE Computer Society, Washington, DC, USA) (2011); ISBN 978-1-4577-1101-5. http://dx.doi.org/10.1109/ICCV.2011.6126474. Cited on page 58

[8] H. Sokol, V. Leducq, H. Aschard, H.-P. Pham, S. Jegou, C. Landman, D. Cohen, G. Liguori, A. Bourrier, I. Nion-Larmurier, J. Cosnes, P. Seksik, P. Langella, D. Skurnik, M. L. Richard & L. Beaugerie; «Fungal microbiota dysbiosis in IBD»; Gut **66**, p. 1039–1048 (2016). Cited on pages xviii, 65, and 83

[9] T. Z. DeSantis, Jr, P. Hugenholtz, K. Keller, E. L. Brodie, N. Larsen, Y. M. Piceno, R. Phan & G. L. Andersen; «NAST: a multiple sequence alignment

server for comparative analysis of 16S rRNA genes»; Nucleic Acids Res. **34**, p. W394–9 (2006). Cited on page 55

[10] N. Qin, F. Yang, A. Li, E. Prifti, Y. Chen, L. Shao, J. Guo, E. Le Chatelier, J. Yao, L. Wu, J. Zhou, S. Ni, L. Liu, N. Pons, J. M. Batto, S. P. Kennedy, P. Leonard, C. Yuan, W. Ding, Y. Chen, X. Hu, B. Zheng, G. Qian, W. Xu, S. D. Ehrlich, S. Zheng & L. Li; «Alterations of the human gut microbiome in liver cirrhosis»; Nature **513**, p. 59–64 (2014). Cited on pages xviii, 27, and 67

[11] D. Fioravanti, Y. Giarratano, V. Maggio, C. Agostinelli, M. Chierici, G. Jurman & C. Furlanello; «Phylogenetic convolutional neural networks in metagenomics»; **19**, p. 49 (2018). ISSN 1471-2105. https://doi.org/10.1186/s12859-018-2033-5. Cited on pages xviii, 5, 53, 56, 67, 71, 83, 85, and 97

[12] S. S. Vempala; *The Random Projection Method* (American Mathematical Soc.) (2005). Cited on page 87

[13] J. R. Cole, Q. Wang, E. Cardenas, J. Fish, B. Chai, R. J. Farris, A. S. Kulam-Syed-Mohideen, D. M. McGarrell, T. Marsh, G. M. Garrity & J. M. Tiedje; «The Ribosomal Database Project: improved alignments and new tools for rRNA analysis»; Nucleic Acids Res. **37**, p. D141–5 (2009). Cited on page 55

[14] M. D. Zeiler & R. Fergus; «Visualizing and Understanding Convolutional Networks»; CoRR **abs/1311.2901** (2013)http://arxiv.org/abs/1311.2901; 1311.2901. Cited on pages 58, 60, 61, 69, and 119

[15] H. W. Virgin & J. A. Todd; «Metagenomics and personalized medicine»; Cell **147**, p. 44–56 (2011). Cited on page 2

[16] R. Garreta & G. Moncecchi; *Learning scikit-learn: Machine Learning in Python* (Packt Publishing Ltd) (2013). Cited on pages 30, 38, and 71

[17] I. Goodfellow, Y. Bengio & A. Courville; *Deep Learning* (MIT Press) (2016). Cited on pages x, 3, 67, and 71

[18] E. Pasolli, D. T. Truong, F. Malik, L. Waldron & N. Segata; «Machine Learning Meta-analysis of Large Metagenomic Datasets: Tools and Biological Insights»; PLoS Comput. Biol. **12**, p. e1004977 (2016). Cited on pages ix, x, xvii, xviii, xxi, 3, 5, 27, 28, 52, 55, 74, 76, and 97

[19] E. Le Chatelier, T. Nielsen, J. Qin, E. Prifti, F. Hildebrand, G. Falony, M. Almeida, M. Arumugam, J.-M. Batto, S. Kennedy, P. Leonard, J. Li, K. Burgdorf, N. Grarup, T. Jørgensen, I. Brandslund, H. B. Nielsen, A. S. Juncker, M. Bertalan, F. Levenez, N. Pons, S. Rasmussen, S. Sunagawa, J. Tap, S. Tims, E. G. Zoetendal, S. Brunak, K. Clément, J. Doré, M. Kleerebezem, K. Kristiansen, P. Renault, T. Sicheritz-Ponten, W. M. de Vos, J.-D. Zucker, J. Raes, T. Hansen, MetaHIT consortium, P. Bork, J. Wang, S. D. Ehrlich & O. Pedersen; «Richness of human gut microbiome correlates with metabolic markers»; Nature **500**, p. 541–546 (2013). Cited on pages xviii, 12, 27, and 67

[20] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada, D. R. Mende, J. Li, J. Xu, S. Li, D. Li, J. Cao, B. Wang, H. Liang, H. Zheng, Y. Xie, J. Tap, P. Lepage, M. Bertalan, J.-M. Batto, T. Hansen, D. Le Paslier, A. Linneberg, H. B. Nielsen, E. Pelletier, P. Renault, T. Sicheritz-Ponten, K. Turner, H. Zhu, C. Yu, S. Li, M. Jian, Y. Zhou, Y. Li, X. Zhang, S. Li, N. Qin, H. Yang, J. Wang, S. Brunak, J. Doré, F. Guarner, K. Kristiansen, O. Pedersen, J. Parkhill, J. Weissenbach, MetaHIT Consortium, P. Bork, S. D. Ehrlich & J. Wang; «A human gut microbial gene catalogue established by metagenomic sequencing»; Nature **464**, p. 59–65 (2010). Cited on pages xviii and 27

[21] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus & Y. LeCun; «OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks»; CoRR **abs/1312.6229** (2013)http://arxiv.org/abs/1312.6229. Cited on page 62

[22] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella & J. Schmidhuber; «Flexible, High Performance Convolutional Neural Networks for Image Classification»; dans «Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two», IJCAI'11; p. 1237–1242 (AAAI Press) (2011); ISBN 978-1-57735-514-4. http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210. Cited on page 62

[23] *A Simple Weight Decay Can Improve Generalization* (Advances in Neural Information Processing Systems 4) (1992). Cited on page 65

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama & T. Darrell; «Caffe: Convolutional Architecture for Fast Feature Embedding»; arXiv preprint arXiv:1408.5093 (2014). Cited on page 65

[25] K. He, X. Zhang, S. Ren & J. Sun; «Deep Residual Learning for Image Recognition»; CoRR **abs/1512.03385** (2015). Cited on pages 56, 65, 66, and 119

[26] S. Ioffe & C. Szegedy; «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift»; CoRR **abs/1502.03167** (2015)http://arxiv.org/abs/1502.03167. Cited on page 65

[27] X. Glorot & Y. Bengio; «Understanding the difficulty of training deep feedforward neural networks»; dans Y. W. Teh & D. M. Titterington (rédacteurs), «Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)», , tome 9p. 249–256 (2010). http://www.jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf. Cited on page 65

[28] *Improving neural networks by preventing co-adaptation of feature detectors* (arXiv preprint arXiv:1207.0580) (2012). Cited on pages 57 and 69

[29] M. Ribeiro, S. Singh & C. Guestrin; «"Why Should I Trust You?": Explaining the Predictions of Any Classifier»; dans «Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations», (2016). Cited on pages 53 and 80

[30] D. T. Truong, E. A. Franzosa, T. L. Tickle, M. Scholz, G. Weingart, E. Pasolli, A. Tett, C. Huttenhower & N. Segata; «MetaPhlAn2 for enhanced metagenomic taxonomic profiling»; Nat. Methods **12**, p. 902–903 (2015). Cited on pages xviii and 27

[31] M. D. Zeiler & R. Fergus; «Visualizing and Understanding Convolutional Networks»; dans «Lecture Notes in Computer Science», p. 818–833 (2014). Cited on page 56

[32] G. Zeller, J. Tap, A. Y. Voigt, S. Sunagawa, J. R. Kultima, P. I. Costea, A. Amiot, J. Böhm, F. Brunetti, N. Habermann, R. Hercog, M. Koch, A. Luciani, D. R. Mende, M. A. Schneider, P. Schrotz-King, C. Tournigand, J. Tran Van Nhieu, T. Yamada, J. Zimmermann, V. Benes, M. Kloor, C. M. Ulrich, M. von Knebel Doeberitz, I. Sobhani & P. Bork; «Potential of fecal microbiota for early-stage detection of colorectal cancer»; Mol. Syst. Biol. **10**, p. 766 (2014). Cited on pages xviii, 27, and 67

[33] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh & D. Batra; «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization»; dans «2017 IEEE International Conference on Computer Vision (ICCV)», (2017). Cited on pages 53, 80, and 81

[34] A. Mahendran & A. Vedaldi; «Salient Deconvolutional Networks»; dans B. Leibe, J. Matas, N. Sebe & M. Welling (rédacteurs), «Computer Vision – ECCV 2016», , tome 9910p. 120–135 (Springer International Publishing); ISBN 978-3-319-46465-7 978-3-319-46466-4. http://link.springer.com/10.1007/978-3-319-46466-4_8. Cited on pages 53 and 80

[35] R. Kotikalapudi & contributors; «keras-vis»; https://github.com/raghakot/keras-vis (2017). Cited on page 80

[36] J. Qin, Y. Li, Z. Cai, S. Li, J. Zhu, F. Zhang, S. Liang, W. Zhang, Y. Guan, D. Shen, Y. Peng, D. Zhang, Z. Jie, W. Wu, Y. Qin, W. Xue, J. Li, L. Han, D. Lu, P. Wu, Y. Dai, X. Sun, Z. Li, A. Tang, S. Zhong, X. Li, W. Chen, R. Xu, M. Wang, Q. Feng, M. Gong, J. Yu, Y. Zhang, M. Zhang, T. Hansen, G. Sanchez, J. Raes, G. Falony, S. Okuda, M. Almeida, E. LeChatelier, P. Renault, N. Pons, J.-M. Batto, Z. Zhang, H. Chen, R. Yang, W. Zheng, S. Li, H. Yang, J. Wang, S. D. Ehrlich, R. Nielsen, O. Pedersen, K. Kristiansen & J. Wang; «A metagenome-wide association study of gut microbiota in type 2 diabetes»; Nature **490**, p. 55–60 (2012). Cited on pages xviii and 27

[37] Z. Dai, O. O. Coker, G. Nakatsu, W. K. K. Wu, L. Zhao, Z. Chen, F. K. L. Chan, K. Kristiansen, J. J. Y. Sung, S. H. Wong & J. Yu; «Multi-cohort analysis of colorectal cancer metagenome identified altered bacteria across populations and universal bacterial markers»; **6**, p. 70 (2018). ISSN 2049-2618. https://doi.org/10.1186/s40168-018-0451-2. Cited on pages xviii, xxi, xxii, 5, 52, 55, 65, 83, 88, 97, 100, and 122

[38] Y. Liu & J. Heer; «Somewhere Over the Rainbow: An Empirical Assessment of Quantitative Colormaps»; dans «Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems», CHI '18; p. 598:1–598:12 (ACM) (2018); ISBN 978-1-4503-5620-6. http://doi.acm.org/10.1145/3173574.3174172. Cited on pages xv and 28

[39] D. P. Kingma & J. Ba; «Adam: A Method for Stochastic Optimization»; CoRR **abs/1412.6980** (2014)http://arxiv.org/abs/1412.6980; 1412.6980. Cited on page 71

[40] C. Szegedy, W. Liu, Y. J. andVGGNet Pierre Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke & A. Rabinovich; «Going Deeper with Convolutions»; CoRR **abs/1409.4842** (2014)http://arxiv.org/abs/1409.4842. Cited on pages 56, 58, 59, 61, 63, and 119

[41] *ImageNet Classification with Deep Convolutional Neural Networks* (The Neural Information Processing Systems Conference 2012) (2012). Cited on pages 56, 57, 58, 59, 62, 65, and 119

[42] A. L. Dallora, S. Eivazzadeh, E. Mendes, J. Berglund & P. Anderberg; «Machine learning and microsimulation techniques on the prognosis of dementia: A systematic literature review»; PLoS One **12**, p. e0179 804 (2017). Cited on page 3

[43] T. Zhang; «Adaptive Forward-Backward Greedy Algorithm for Sparse Learning with Linear Models»; dans D. Koller, D. Schuurmans, Y. Bengio & L. Bottou (rédacteurs), «Advances in Neural Information Processing Systems 21», p. 1921–1928 (Curran Associates, Inc.) (2009). Cited on pages xiii and 11

[44] P. Flach; *Machine Learning: The Art and Science of Algorithms that Make Sense of Data* (Cambridge University Press) (2012). Cited on page 54

[45] Y. Bengio, A. Courville & P. Vincent; «Representation learning: a review and new perspectives»; IEEE Trans. Pattern Anal. Mach. Intell. **35**, p. 1798–1828 (2013). Cited on page 56

[46] A. Cotillard, S. P. Kennedy, L. C. Kong, E. Prifti, N. Pons, E. Le Chatelier, M. Almeida, B. Quinquis, F. Levenez, N. Galleron, S. Gougis, S. Rizkalla, J.-M. Batto, P. Renault, ANR MicroObes consortium, J. Doré, J.-D. Zucker, K. Clément & S. D. Ehrlich; «Dietary intervention impact on gut microbial gene richness»; Nature **500**, p. 585–588 (2013). Cited on pages 8 and 12

[47] A. Coates, A. Ng & H. Lee; «An analysis of single-layer networks in unsupervised feature learning»; of the fourteenth international conference on ... (2011). Cited on page 8

[48] H. Soueidan & M. Nikolski; «Machine learning for metagenomics: methods and tools»; Metagenomics **1** (2017). Cited on pages 53, 54, and 119

[49] N. Sokolovska, S. Rizkalla, K. Clément & J.-D. Zucker; «Continuous and Discrete Deep Classifiers for Data Integration»; dans «Lecture Notes in Computer Science», p. 264–274 (2015). Cited on page 9

[50] H. Lee, R. Grosse, R. Ranganath & A. Y. Ng; «Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations»; dans «Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09», (2009). Cited on page 8

[51] L. Kaufman & P. Rousseeuw; *Clustering by Means of Medoids* (1987). Cited on page 9

[52] S. Srivastava, S. Soman, A. Rai & P. K. Srivastava; «Deep learning for health informatics: Recent trends and future directions»; dans «2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)», (2017). Cited on pages x and 3

[53] G. E. Hinton; «Reducing the Dimensionality of Data with Neural Networks»; Science **313**, p. 504–507 (2006). Cited on page 8

[54] E. Pampalk; *Islands of music: Analysis, organization, and visualization of music archives* (na) (2001). Cited on page 12

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov; «Dropout: A Simple Way to Prevent Neural Networks from Overfitting»; Journal of Machine Learning Research 15 p. 1929–1958 (2014). Cited on page 57

[56] Y. Tang; «Deep Learning using Linear Support Vector Machines»; (2013)1306.0239. Cited on pages 8 and 10

[57] G. S. Ginsburg & H. F. Willard; «Genomic and personalized medicine: foundations and applications»; Transl. Res. **154**, p. 277–287 (2009). Cited on pages xvii and 52

[58] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio & X. Muller; «The Manifold Tangent Classifier»; dans J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira & K. Q. Weinberger (rédacteurs), «Advances in Neural Information Processing Systems 24», p. 2294–2302 (Curran Associates, Inc.) (2011). Cited on page 8

[59] Q. Song, J. Ni & G. Wang; «A Fast Clustering-Based Feature Subset Selection Algorithm for High-Dimensional Data»; IEEE Trans. Knowl. Data Eng. **25**, p. 1–14 (2013). Cited on page 9

[60] *Very Deep Convolutional Networks for Large-Scale Image Recognition* (3rd International Conference on Learning Representations (ICLR2015)) (2015). Cited on pages 56, 62, 64, 66, 69, and 119

[61] Y. Cho; *Kernel Methods for Deep Learning* (2012). Cited on page 9

[62] E. Pampalk, W. Goebl & G. Widmer; «Visualizing changes in the structure of data for exploratory feature selection»; dans «Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining», p. 157–166 (ACM) (2003). Cited on page 12

[63] Y. Zhao, B. C. Healy, D. Rotstein, C. R. G. Guttmann, R. Bakshi, H. L. Weiner, C. E. Brodley & T. Chitnis; «Exploration of machine learning techniques in predicting multiple sclerosis disease course»; PLoS One **12**, p. e0174 866 (2017). Cited on page 3

[64] E. K. Costello, C. L. Lauber, M. Hamady, N. Fierer, J. I. Gordon & R. Knight; «Bacterial community variation in human body habitats across space and time»; Science **326**, p. 1694–1697 (2009). Cited on page 3

[65] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang & G. Wang; «Recent Advances in Convolutional Neural Networks»; CoRR **abs/1512.07108** (2015). Cited on pages x, 3, 5, 56, 67, and 69

[66] Fu Jie Huang, F. J. Huang & Y. LeCun; «Large-scale Learning with SVM and Convolutional for Generic Object Categorization»; dans «2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)», . Cited on page 9

[67] P. Sollich; «Probabilistic interpretations and bayesian methods for support vector machines»; dans «9th International Conference on Artificial Neural Networks: ICANN '99», (1999). Cited on page 10

[68] L. Kaufman & P. J. Rousseeuw; *Finding Groups in Data: An Introduction to Cluster Analysis* (John Wiley & Sons) (2009). Cited on page 9

[69] T. Kohonen; «The self-organizing map»; Proc. IEEE **78**, p. 1464–1480 (1990). Cited on pages xii, 8, and 10

[70] J. Iivarinen, K. Valkealahti, A. Visa & O. Simula; «Feature Selection with Self-Organizing Feature Map»; dans «ICANN '94», p. 334–337 (Springer London) (1994). Cited on page 9

[71] J. Bien, J. Taylor & R. Tibshirani; «A LASSO FOR HIERARCHICAL INTERACTIONS»; Ann. Stat. **41**, p. 1111–1141 (2013). Cited on page 9

[72] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis & D. I. Fotiadis; «Machine learning applications in cancer prognosis and prediction»; Comput. Struct. Biotechnol. J. **13**, p. 8–17 (2015). Cited on page 3

[73] R. Butterworth, G. Piatetsky-Shapiro & D. A. Simovici; «On Feature Selection through Clustering»; dans «Fifth IEEE International Conference on Data Mining (ICDM'05)», . Cited on page 9

[74] J. Friedman, T. Hastie & R. Tibshirani; «Regularization Paths for Generalized Linear Models via Coordinate Descent»; J. Stat. Softw. **33**, p. 1–22 (2010). Cited on page 17

[75] A.-C. Haury, P. Gestraud & J.-P. Vert; «The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures»; PLoS One **6**, p. e28 210 (2011). Cited on page 13

[76] Y. Bengio, P. Lamblin, D. Popovici & H. Larochelle; «Greedy Layer-Wise Training of Deep Networks»; dans B. Schölkopf, J. C. Platt & T. Hoffman (rédacteurs), «Advances in Neural Information Processing Systems 19», p. 153–160 (MIT Press) (2007). Cited on page 8

[77] K. Pearson; *On Lines and Planes of Closest Fit to Systems of Points in Space* (1901). Cited on pages 26 and 87

[78] G. Ditzler, R. Polikar & G. Rosen; «Multi-Layer and Recursive Neural Networks for Metagenomic Classification»; IEEE Trans. Nanobioscience **14**, p. 608–616 (2015). Cited on pages x, xi, 2, 3, 5, 54, 55, and 56

[79] J. B. Tenenbaum, V. de Silva & J. C. Langford; «A Global Geometric Framework for Nonlinear Dimensionality Reduction»; Science **290**, p. 2319 (2000). Cited on page 26

[80] I. Borg, P. J. Groenen & P. Mair; *Applied Multidimensional Scaling* (Springer Publishing Company, Incorporated) (2012); ISBN 3642318479, 9783642318474. Cited on page 26

[81] X. Huo, X. S. Ni & A. K. Smith; *A Survey of Manifold-Based Learning Methods*; p. 691–745 (WORLD SCIENTIFIC) (2011). https://www.worldscientific.com/doi/abs/10.1142/9789812779861_0015; https://www.worldscientific.com/doi/pdf/10.1142/9789812779861_0015. Cited on pages 23, 26, and 27

[82] J. D. Hunter; «Matplotlib: A 2D graphics environment»; Computing In Science & Engineering **9**, p. 90–95 (2007). Cited on pages 43, 44, and 71

[83] D. Reiman, A. A. Metwally & Y. Dai; «PopPhy-CNN: A Phylogenetic Tree Embedded Architecture for Convolution Neural Networks for Metagenomic Data»; p. – (2018). http://biorxiv.org/lookup/doi/10.1101/257931. Cited on pages x, 2, and 56

[84] Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard & L. Jackel; «Backpropagation applied to handwritten zip code recognition»; Neural computation **1**, p. 541–551 (1989)ISSN 0899-7667. Cited on pages 56 and 65

[85] Y. LeCun, L. Bottou, Y. Bengio & P. H. ner; «Gradient-Based Learning Applied to Document Recognition»; p. 46. Cited on page 56

[86] V. Nair & G. E. Hinton; «Rectified Linear Units Improve Restricted Boltzmann Machines»; dans «Proceedings of the 27th International Conference on International Conference on Machine Learning», ICML'10; p. 807–814 (Omnipress, USA) (2010); ISBN 978-1-60558-907-7. http://dl.acm.org/citation.cfm?id=3104322.3104425. Cited on pages 57 and 69

[87] «Neural networks: tricks of the trade»; OCLC: 246316889. Cited on page 69

[88] A. J. Izenman; «Introduction to manifold learning»; **4**, p. 439–446. ISSN 1939-0068. https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1222. Cited on pages 23, 26, 27, 38, and 86

[89] E. W. Dijkstra; «A Note on Two Problems in Connexion with Graphs»; Numer. Math. **1**, p. 269–271 (1959). ISSN 0029-599X. http://dx.doi.org/10.1007/BF01386390. Cited on page 27

[90] R. W. Floyd; «Algorithm 97: Shortest Path»; Commun. ACM **5**, p. 345– (1962). ISSN 0001-0782. http://doi.acm.org/10.1145/367766.368168. Cited on page 27

[91] J. McQueen, M. Meila, J. VanderPlas & Z. Zhang; «megaman: Manifold Learning with Millions of points»; http://arxiv.org/abs/1603.02763; 1603.02763. Cited on page 26

[92] F. Celesti, A. Celesti, J. Wan & M. Villari; «Why Deep Learning Is Changing the Way to Approach NGS Data Processing: a Review»; p. 1–1 (2018). ISSN 1937-3333. Cited on page 56

[93] D. Masters & C. Luschi; «Revisiting Small Batch Training for Deep Neural Networks»; http://arxiv.org/abs/1804.07612; 1804.07612. Cited on page 71

[94] K. Sedlar, K. Kupkova & I. Provaznik; «Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics»; **15**, p. 48–55. ISSN 2001-0370. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5148923/. Cited on page 53

[95] M. Arumugam, J. Raes, E. Pelletier, D. L. Paslier, T. Yamada, D. R. Mende, G. R. Fernandes, J. Tap, T. Bruls, J.-M. Batto, M. Bertalan, N. Borruel, F. Casellas, L. Fernandez, L. Gautier, T. Hansen, M. Hattori, T. Hayashi, M. Kleerebezem, K. Kurokawa, M. Leclerc, F. Levenez, C. Manichanh, H. B. Nielsen, T. Nielsen, N. Pons, J. Poulain, J. Qin, T. Sicheritz-Ponten, S. Tims, D. Torrents, E. Ugarte, E. G. Zoetendal, J. Wang, F. Guarner, O. Pedersen, W. M. d. Vos, S. Brunak, J. Doré, M. C. a. Members), M. Antolín, F. Artiguenave, H. M. Blottiere, M. Almeida, C. Brechot, C. Cara, C. Chervaux, A. Cultrone, C. Delorme, G. Denariaz, R. Dervyn, K. U. Foerstner, C. Friss, M. v. d. Guchte, E. Guedon, F. Haimet, W. Huber, J. v. Hylckama-Vlieg, A. Jamet, C. Juste, G. Kaci, J. Knol, K. Kristiansen, O. Lakhdari, S. Layec, K. L. Roux, E. Maguin, A. Mérieux, R. M. Minardi, C. M'rini, J. Muller, R. Oozeer, J. Parkhill, P. Renault, M. Rescigno, N. Sanchez, S. Sunagawa, A. Torrejon, K. Turner, G. Vandemeulebrouck, E. Varela, Y. Winogradsky, G. Zeller, J. Weissenbach, S. D. Ehrlich & P. Bork; «Enterotypes of the human gut microbiome»; **473**, p. 174–180. ISSN 1476-4687. https://www.nature.com/articles/nature09944. Cited on page 53

[96] I. Sutskever, J. Martens, G. Dahl & G. Hinton; «On the Importance of Initialization and Momentum in Deep Learning»; dans «Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28», ICML'13; p. III–1139–III–1147 (JMLR.org) (2013). http://dl.acm.org/citation.cfm?id=3042817.3043064. Cited on page 62

[97] A. Canziani, A. Paszke & E. Culurciello; «An Analysis of Deep Neural Network Models for Practical Applications»; http://arxiv.org/abs/1605.07678; 1605.07678. Cited on page 56

[98] H. Li, Z. Xu, G. Taylor, C. Studer & T. Goldstein; «Visualizing the Loss Landscape of Neural Nets»; http://arxiv.org/abs/1712.09913; 1712.09913. Cited on page 57

[99] D. Chicco; «Ten quick tips for machine learning in computational biology»; **10**. ISSN 1756-0381. https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0155-3. Cited on page 83

[100] S. Nayfach & K. S. Pollard; «Toward Accurate and Quantitative Comparative Metagenomics»; **166**, p. 1103–1116. ISSN 0092-8674. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5080976/. Cited on page 53

[101] S. M. Dabdoub, S. M. Ganesan & P. S. Kumar; «Comparative metagenomics reveals taxonomically idiosyncratic yet functionally congruent communities in periodontitis»; **6**, p. 38 993. ISSN 2045-2322. https://www.nature.com/articles/srep38993. Cited on page 53

[102] L.-x. Chen, M. Hu, L.-n. Huang, Z.-s. Hua, J.-l. Kuang, S.-j. Li & W.-s. Shu; «Comparative metagenomic and metatranscriptomic analyses of microbial communities in acid mine drainage»; **9**, p. 1579–1592. ISSN 1751-7370. https://www.nature.com/articles/ismej2014245. Cited on page 53

[103] D. H. Huson, D. C. Richter, S. Mitra, A. F. Auch & S. C. Schuster; «Methods for comparative metagenomics»; **10**, p. S12. ISSN 1471-2105. http://www.biomedcentral.com/1471-2105/10/S1/S12. Cited on page 53

[104] C. Mathé, M.-F. Sagot, T. Schiex & P. Rouzé; «SURVEY AND SUMMARY: Current methods of gene prediction, their strengths and weaknesses»; **30**, p. 4103–4117ISSN 0305-1048. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC140543/. Cited on page 53

[105] Z. Wang, Y. Chen & Y. Li; «A Brief Review of Computational Gene Prediction Methods»; **2**, p. 216–221. ISSN 1672-0229. http://www.sciencedirect.com/science/article/pii/S1672022904020285. Cited on page 53

[106] N. Goel, S. Singh & T. C. Aseri; «A Review of Soft Computing Techniques for Gene Prediction»; . https://www.hindawi.com/journals/isrn/2013/191206/. Cited on page 53

[107] N.-P. Nguyen, T. Warnow, M. Pop & B. White; «A perspective on 16S rRNA operational taxonomic unit clustering using sequence similarity»; **2**, p. 16 004. ISSN 2055-5008. https://www.nature.com/articles/npjbiofilms20164. Cited on page 53

[108] S. Park, H. s. Choi, B. Lee, J. Chun, J. H. Won & S. Yoon; «hc-OTU: A Fast and Accurate Method for Clustering Operational Taxonomic Units Based on Homopolymer Compaction»; **15**, p. 441–451. ISSN 1545-5963. Cited on page 53

[109] F. Chollet *et al.*; «Keras»; https://keras.io (2015). Cited on pages xix, 70, 71, and 119

[110] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu & X. Zheng; «TensorFlow: A System for Large-scale Machine Learning»; dans «Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation», OSDI'16; p. 265–283 (USENIX Association, Berkeley, CA, USA) (2016); ISBN 978-1-931971-33-1. http://dl.acm.org/citation.cfm?id=3026877.3026899. Cited on page 71

[111] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu & X. Zheng; «TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems»; (2015). https://www.tensorflow.org/; software available from tensorflow.org. Cited on page 71

[112] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt & G. Varoquaux; «API design for machine learning software: experiences from the scikit-learn project»; dans «ECML PKDD Workshop: Languages for Data Mining and Machine Learning», p. 108–122 (2013). Cited on page 71

[113] K. Simonyan, A. Vedaldi & A. Zisserman; «Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps»; http://arxiv.org/abs/1312.6034; 1312.6034. Cited on pages 53 and 80

[114] X. Z. Fern & C. E. Brodley; «Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach»; p. 8. Cited on pages 25 and 87

[115] C. Grellmann, J. Neumann, S. Bitzer, P. Kovacs, A. Tönjes, L. T. Westlye, O. A. Andreassen, M. Stumvoll, A. Villringer &

A. Horstmann; «Random Projection for Fast and Efficient Multivariate Correlation Analysis of High-Dimensional Data: A New Approach»; **7**. ISSN 1664-8021. https://www.frontiersin.org/articles/10.3389/fgene.2016.00102/full. Cited on pages 25 and 87

[116] P. Li, M. Mitzenmacher & A. Shrivastava; «Coding for Random Projections»; http://arxiv.org/abs/1308.2218; 1308.2218. Cited on pages 25 and 87

[117] E. Bingham & H. Mannila; «Random Projection in Dimensionality Reduction: Applications to Image and Text Data»; dans «Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining», KDD '01; p. 245–250 (ACM, New York, NY, USA) (2001); ISBN 1-58113-391-X. http://doi.acm.org/10.1145/502512.502546. Cited on pages 25 and 87

[118] L. Breiman; «Random Forests»; **45**, p. 5–32. ISSN 0885-6125, 1573-0565. https://link.springer.com/article/10.1023/A:1010933404324. Cited on pages xix and 69

[119] C. Cortes & V. Vapnik; «Support-vector networks»; **20**, p. 273–297. ISSN 0885-6125, 1573-0565. http://link.springer.com/10.1007/BF00994018. Cited on pages xix and 69

[120] W. Johnson & J. Lindenstrauss; «Extensions of Lipschitz mappings into a Hilbert space»; dans «Conference in modern analysis and probability (New Haven, Conn., 1982)», , *Contemporary Mathematics*, tome 26p. 189–206 (American Mathematical Society) (1984). Cited on page 25

[121] N. Ailon & B. Chazelle; «The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors»; SIAM J. Comput. **39**, p. 302–322 (2009). ISSN 0097-5397. http://dx.doi.org/10.1137/060673096. Cited on page 25

[122] J. Kruskal & M. Wish; «Multidimensional Scaling»; Sage University Paper Series on Quantitative Applications in the Social Sciences (1978)http://dx.doi.org/10.4135/9781412985130. Cited on page 26

[123] Borg & Groenen; «Modern Multidimensional Scaling - Theory and Applications»; (2005)https://www.springer.com/fr/book/9780387251509. Cited on page 26

[124] S. T. Roweis & L. K. Saul; «Nonlinear Dimensionality Reduction by Locally Linear Embedding»; **290**, p. 2323–2326. ISSN 0036-8075, 1095-9203. http://science.sciencemag.org/content/290/5500/2323. Cited on page 27

[125] A. Talwalkar, S. Kumar & H. Rowley; «Large-scale manifold learning»; dans «2008 IEEE Conference on Computer Vision and Pattern Recognition», p. 1–8. Cited on page 27

[126] F. WICKELMAIER; «An Introduction to MDS»; p. 26 (2003). Cited on page 26

[127] C. Hegde, M. Wakin & R. Baraniuk; «Random Projections for Manifold Learning»; dans J. C. Platt, D. Koller, Y. Singer & S. T. Roweis (rédacteurs), «Advances in Neural Information Processing Systems 20», p. 641–648 (Curran Associates, Inc.). http://papers.nips.cc/paper/3191-random-projections-for-manifold-learning.pdf. Cited on page 25

[128] S. Lahiri, P. Gao & S. Ganguli; «Random projections of random manifolds»; http://arxiv.org/abs/1607.04331; 1607.04331. Cited on page 25

[129] S. Dasgupta; «Experiments with Random Projection»; p. 9 (2000). Cited on page 25

[130] I. Gashi, V. Stankovic, C. Leita & O. Thonnard; «An Experimental Study of Diversity with Off-the-Shelf AntiVirus Engines»; dans «2009 Eighth IEEE International Symposium on Network Computing and Applications», p. 4–11. Cited on page 27

[131] P. Hamel & D. Eck; «LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS»; p. 6. Cited on page 27

[132] A. R. Jamieson, M. L. Giger, K. Drukker, H. Li, Y. Yuan & N. Bhooshan; «Exploring nonlinear feature space dimension reduction and data representation in breast CADx with Laplacian eigenmaps and -SNE»; **37**, p. 339–351. ISSN 2473-4209. https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3267037. Cited on page 27

[133] I. Wallach & R. Lilien; «The protein–small-molecule database, a non-redundant structural resource for the analysis of protein-ligand binding»; **25**, p. 615–620. ISSN 1367-4803. https://academic.oup.com/bioinformatics/article/25/5/615/183421. Cited on page 27

[134] H. Park; «ISOMAP induced manifold embedding and its application to Alzheimer's disease and mild cognitive impairment»; **513**, p. 141–145. ISSN 0304-3940. http://www.sciencedirect.com/science/article/pii/S0304394012002030. Cited on page 26

[135] J. Duchi, E. Hazan & Y. Singer; «Adaptive Subgradient Methods for Online Learning and Stochastic Optimization»; **12**, p. 2121–2159ISSN ISSN 1533-7928. http://jmlr.org/papers/v12/duchi11a.html. Cited on page 69

[136] D. P. Kingma & J. Ba; «Adam: A Method for Stochastic Optimization»; http://arxiv.org/abs/1412.6980; 1412.6980. Cited on page 69

[137] H. Robbins & S. Monro; «A Stochastic Approximation Method»; **22**, p. 400–407. ISSN 0003-4851, 2168-8990. https://projecteuclid.org/euclid.aoms/1177729586. Cited on page 69

[138] J. Kiefer & J. Wolfowitz; «Stochastic Estimation of the Maximum of a Regression Function»; **23**, p. 462–466. ISSN 0003-4851, 2168-8990. https://projecteuclid.org/euclid.aoms/1177729392. Cited on page 69

[139] L. Bottou, F. E. Curtis & J. Nocedal; «Optimization Methods for Large-Scale Machine Learning»; http://arxiv.org/abs/1606.04838; 1606.04838. Cited on page 69

[140] S. Abubucker, N. Segata, J. Goll, A. M. Schubert, J. Izard, B. L. Cantarel, B. Rodriguez-Mueller, J. Zucker, M. Thiagarajan, B. Henrissat, O. White, S. T. Kelley, B. Methé, P. D. Schloss, D. Gevers, M. Mitreva & C. Huttenhower; «Metabolic Reconstruction for Metagenomic Data and Its Application to the Human Microbiome»; **8**, p. e1002 358. ISSN 1553-7358. http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002358. Cited on pages xviii and 67

[141] H. B. Nielsen, M. Almeida, A. S. Juncker, S. Rasmussen, J. Li, S. Sunagawa, D. R. Plichta, L. Gautier, A. G. Pedersen, E. L. Chatelier, E. Pelletier, I. Bonde, T. Nielsen, C. Manichanh, M. Arumugam, J.-M. Batto, M. B. Q. d. Santos, N. Blom, N. Borruel, K. S. Burgdorf, F. Boumezbeur, F. Casellas, J. Doré, P. Dworzynski, F. Guarner, T. Hansen, F. Hildebrand, R. S. Kaas, S. Kennedy, K. Kristiansen, J. R. Kultima, P. Léonard, F. Levenez, O. Lund, B. Moumen, D. L. Paslier, N. Pons, O. Pedersen, E. Prifti, J. Qin, J. Raes, S. Sørensen, J. Tap, S. Tims, D. W. Ussery, T. Yamada, M. Consortium, H. B. Nielsen, M. Almeida, A. S. Juncker, S. Rasmussen, J. Li, S. Sunagawa, D. R. Plichta, L. Gautier, A. G. Pedersen, E. L. Chatelier, E. Pelletier, I. Bonde, T. Nielsen, C. Manichanh, M. Arumugam, J.-M. Batto, M. B. Q. d. Santos, N. Blom, N. Borruel, K. S. Burgdorf, F. Boumezbeur, F. Casellas, J. Doré, P. Dworzynski, F. Guarner, T. Hansen, F. Hildebrand, R. S. Kaas, S. Kennedy, K. Kristiansen, J. R. Kultima, P. Leonard, F. Levenez, O. Lund, B. Moumen, D. L. Paslier, N. Pons, O. Pedersen, E. Prifti, J. Qin, J. Raes, S. Sørensen, J. Tap, S. Tims, D. W. Ussery, T. Yamada, P. Renault, T. Sicheritz-Ponten, P. Bork, J. Wang, S. Brunak, S. D. Ehrlich, A. Jamet, A. Mérieux, A. Cultrone, A. Torrejon, B. Quinquis, C. Brechot, C. Delorme, C. M'Rini, W. M. d. Vos, E. Maguin, E. Varela, E. Guedon, F. Gwen, F. Haimet, F. Artiguenave, G. Vandemeulebrouck, G. Denariaz, G. Khaci, H. Blottière, J. Knol, J. Weissenbach, J. E. T. v. H. Vlieg, J. Torben, J. Parkhill, K. Turner, M. v. d. Guchte, M. Antolin, M. Rescigno, M. Kleerebezem, M. Derrien, N. Galleron, N. Sanchez, N. Grarup, P. Veiga, R. Oozeer, R. Dervyn, S. Layec, T. Bruls, Y. Winogradski, Z. E. G, P. Renault, T. Sicheritz-Ponten, P. Bork, J. Wang, S. Brunak & S. D. Ehrlich; «Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes»; **32**, p. 822–828. ISSN 1546-1696. https://www.nature.com/articles/nbt.2939. Cited on pages xviii, 27, and 67

[142] J. Oh, A. L. Byrd, C. Deming, S. Conlan, N. C. S. Program, B. Barnabas, R. Blakesley, G. Bouffard, S. Brooks, H. Coleman, M. Dekhtyar, M. Gregory, X. Guan, J. Gupta, J. Han, S.-l. Ho, R. Legaspi, Q. Maduro, C. Masiello, B. Maskeri, J. McDowell, C. Montemayor, J. Mullikin, M. Park, N. Riebow, K. Schandler, B. Schmidt, C. Sison,

M. Stantripop, J. Thomas, P. Thomas, M. Vemulapalli, A. Young, H. H. Kong & J. A. Segre; «Biogeography and individuality shape function in the human skin metagenome»; **514**, p. 59–64. ISSN 1476-4687. https://www.nature.com/articles/nature13786. Cited on pages xviii, 27, and 67

[143] J. Ning & R. G. Beiko; «Phylogenetic approaches to microbial community classification»; **3**. ISSN 2049-2618. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4593236/. Cited on page 3

[144] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, O. A. von Lilienfeld & K.-R. Müller; «Learning Invariant Representations of Molecules for Atomization Energy Prediction»; dans «Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1», NIPS'12; p. 440–448 (Curran Associates Inc.). http://dl.acm.org/citation.cfm?id=2999134.2999184. Cited on page 56

[145] M. Li & B. Yuan; «2D-LDA: A statistical linear discriminant analysis for image matrix»; **26**, p. 527–532. ISSN 0167-8655. http://www.sciencedirect.com/science/article/pii/S0167865504002272. Cited on page 35

[146] S. Mika, G. Ratsch, J. Weston, B. Scholkopf & K. R. Mullers; «Fisher discriminant analysis with kernels»; dans «Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)», p. 41–48. Cited on page 35

[147] P. Xanthopoulos, P. M. Pardalos & T. B. Trafalis; «Linear Discriminant Analysis»; dans «Robust Data Mining», SpringerBriefs in Optimization; p. 27–33 (Springer, New York, NY); ISBN 978-1-4419-9877-4 978-1-4419-9878-1. https://link.springer.com/chapter/10.1007/978-1-4419-9878-1_4. Cited on page 35

[148] R. A. Fisher; «The Use of Multiple Measurements in Taxonomic Problems»; **7**, p. 179–188 (1936). ISSN 2050-1439. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x. Cited on page 35

[149] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo & G. Z. Yang; «Deep Learning for Health Informatics»; **21**, p. 4–21. ISSN 2168-2194. Cited on pages xi and 3

[150] V. Kuznetsov, H. K. Lee, S. Maurer-Stroh, M. J. Molnár, S. Pongor, B. Eisenhaber & F. Eisenhaber; «How bioinformatics influences health informatics: usage of biomolecular sequences, expression profiles and automated microscopic image analyses for clinical needs and public health»; **1**. ISSN 2047-2501. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4336111/. Cited on pages x and 3

[151] J. Li, S. K. Halgamuge, C. I. Kells & S.-L. Tang; «Gene function prediction based on genomic context clustering and discriminative learning: an application to bacteriophages»; **8**, p. S6. ISSN 1471-2105. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1892085/. Cited on pages x and 3

[152] G. H. Fernald, E. Capriotti, R. Daneshjou, K. J. Karczewski & R. B. Altman; «Bioinformatics challenges for personalized medicine»; **27**, p. 1741–1748. ISSN 1367-4803. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3117361/. Cited on pages x and 3

[153] K. Sudarikov, A. Tyakht & D. Alexeev; «Methods for The Metagenomic Data Visualization and Analysis»; p. 37–58. ISSN 14673037. http://www.caister.com/cimb/abstracts/v24/37.html. Cited on pages xiii, xiv, 3, 22, 23, and 38

[154] R Development Core Team; «R: A Language and Environment for Statistical Computing»; (2008)http://www.R-project.org; ISBN 3-900051-07-0. Cited on pages xiv and 22

[155] B. D. Ondov, N. H. Bergman & A. M. Phillippy; «Interactive metagenomic visualization in a Web browser»; **12**, p. 385. ISSN 1471-2105. https://doi.org/10.1186/1471-2105-12-385. Cited on pages xiv, 22, 24, and 117

[156] G. W. Tyson, J. Chapman, P. Hugenholtz, E. E. Allen, R. J. Ram, P. M. Richardson, V. V. Solovyev, E. M. Rubin, D. S. Rokhsar & J. F. Banfield; «Community structure and metabolism through reconstruction of microbial genomes from the environment»; **428**, p. 37–43. ISSN 1476-4687. https://www.nature.com/articles/nature02340. Cited on pages 24 and 117

[157] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening & R. Edwards; «The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes»; **9**, p. 386. ISSN 1471-2105. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2563014/. Cited on pages xiv, 22, 24, and 117

[158] D. H. Huson, A. F. Auch, J. Qi & S. C. Schuster; «MEGAN analysis of metagenomic data»; **17**, p. 377–386. ISSN 1088-9051. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1800929/. Cited on pages 24 and 117

[159] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis & T. L. Madden; «NCBI BLAST: a better web interface»; **36**, p. W5–W9. ISSN 0305-1048. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2447716/. Cited on page 22

[160] C. Kerepesi, D. Bánky & V. Grolmusz; «AmphoraNet: The webserver implementation of the AMPHORA2 metagenomic workflow suite»; **533**, p. 538–540. ISSN 0378-1119. http://www.sciencedirect.com/science/article/pii/S0378111913014091. Cited on pages xiv and 22

[161] L. B. R. G. W. H. A. L. T. L. M. M. A. M. S. M. M. S. B. V. Gregory R. Warnes, Ben Bolker; «Package 'gplots', CRAN repository»; https://CRAN.R-project.org/package=gplots. Cited on pages xiv and 22

[162] A. A. U. H. Rudis, B.; «Package 'metricsgraphics', CRAN repository»; (2015)https://CRAN.R-project.org/package=metricsgraphics. Cited on pages xiv and 22

[163] H. Bik; «Phinch: An interactive, exploratory data visualization framework for metagenomic datasets»; (2014). https://figshare.com/articles/Phinch_An_interactive_exploratory_data_visualization_framework_for_metagenomic_datasets/951915. Cited on pages xiv and 22

[164] J. Cheng; «Package 'd3heatmap', CRAN repository»; (2016)https://CRAN.R-project.org/package=d3heatmap. Cited on pages xiv and 23

[165] L. Jiang, M. Song, L. Yang, D. Zhang, Y. Sun, Z. Shen, C. Luo & G. Zhang; «Exploring the Influence of Environmental Factors on Bacterial Communities within the Rhizosphere of the Cu-tolerant plant, Elsholtzia splendens»; **6**. ISSN 2045-2322. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5080579/. Cited on pages xiii, 3, and 22

[166] J. T. Morton, J. Sanders, R. A. Quinn, D. McDonald, A. Gonzalez, Y. Vázquez-Baeza, J. A. Navas-Molina, S. J. Song, J. L. Metcalf, E. R. Hyde, M. Lladser, P. C. Dorrestein & R. Knight; «Balance Trees Reveal Microbial Niche Differentiation»; **2**, p. e00 162–16. ISSN 2379-5077. http://msystems.asm.org/content/2/1/e00162-16. Cited on pages xiii, 3, and 22

[167] X. Jiang, X. Hu, H. Shen & T. He; «Manifold learning reveals nonlinear structure in metagenomic profiles»; dans «2012 IEEE International Conference on Bioinformatics and Biomedicine», p. 1–6. Cited on pages xiv, 23, and 25

[168] A. Gisbrecht, B. Hammer, B. Mokbel & A. Sczyrba; «Nonlinear Dimensionality Reduction for Cluster Identification in Metagenomic Samples»; dans «2013 17th International Conference on Information Visualisation», p. 174–179. Cited on pages xiv and 23

[169] M. Alshawaqfeh, A. Bashaireh, E. Serpedin & J. Suchodolski; «Consistent metagenomic biomarker detection via robust PCA»; **12**, p. 4. ISSN 1745-6150. https://doi.org/10.1186/s13062-017-0175-4. Cited on pages xiv and 23

[170] R. Vidal, Y. Ma & S. S. Sastry; «Generalized Principal Component Analysis»; p. 353 (2006). Cited on pages xiv and 23

[171] R. O. Duda, P. E. Hart & D. G. Stork; *Pattern Classification (2Nd Edition)* (Wiley-Interscience, New York, NY, USA) (2000); ISBN 0471056693. Cited on page 23

[172] K. Fukunaga; *Introduction to Statistical Pattern Recognition (2Nd Ed.)* (Academic Press Professional, Inc., San Diego, CA, USA) (1990); ISBN 0-12-269851-7. Cited on pages 23 and 25

[173] J. Ye, R. Janardan & Q. Li; «Two-Dimensional Linear Discriminant Analysis»; dans L. K. Saul, Y. Weiss & L. Bottou (rédacteurs), «Advances in Neural Information Processing Systems 17», p. 1569–1576 (MIT Press) (2005). http://papers.nips.cc/paper/2547-two-dimensional-linear-discriminant-analysis.pdf. Cited on page 25

[174] N. Segata, J. Izard, L. Waldron, D. Gevers, L. Miropolsky, W. S. Garrett & C. Huttenhower; «Metagenomic biomarker discovery and explanation»; **12**, p. R60 (2011). ISSN 1465-6906. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3218848/. Cited on page 25

[175] H. Zheng & H. Wu; «Short prokaryotic dna fragment binning using a hierarchical classifier based on linear discriminant analysis and principal component analysis»; **08**, p. 995–1011. ISSN 0219-7200. https://www.worldscientific.com/doi/abs/10.1142/S0219720010005051. Cited on page 25

[176] D. D. Lee & H. S. Seung; «Learning the parts of objects by non-negative matrix factorization»; **401**, p. 788–791. ISSN 1476-4687. https://www.nature.com/articles/44565. Cited on page 25

[177] N. Gillis; «The Why and How of Nonnegative Matrix Factorization»; http://arxiv.org/abs/1401.5226; 1401.5226. Cited on page 25

[178] P. Paatero & U. Tapper; «Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values»; **5**, p. 111–126. ISSN 1099-095X. https://onlinelibrary.wiley.com/doi/abs/10.1002/env.3170050203. Cited on page 25

[179] C. Févotte & J. Idier; «Algorithms for nonnegative matrix factorization with the beta-divergence»; http://arxiv.org/abs/1010.1763; 1010.1763. Cited on page 25

[180] C.-J. Lin; «Projected Gradient Methods for Nonnegative Matrix Factorization»; **19**, p. 2756–2779. ISSN 0899-7667, 1530-888X. http://www.mitpressjournals.org/doi/10.1162/neco.2007.19.10.2756. Cited on page 25

[181] P. O. Hoyer; «Non-negative Matrix Factorization with Sparseness Constraints»; **5**, p. 1457–1469ISSN 1532-4435. http://dl.acm.org/citation.cfm?id=1005332.1044709. Cited on page 25

[182] C. Boutsidis & E. Gallopoulos; «SVD Based Initialization: A Head Start for Nonnegative Matrix Factorization»; Pattern Recogn. **41**, p. 1350–1362 (2008). ISSN 0031-3203. http://dx.doi.org/10.1016/j.patcog.2007.09.010. Cited on page 25

[183] S. Dodge & L. Karam; «A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions»; http://arxiv.org/abs/1705.02498; 1705.02498. Cited on pages x, 3, and 100

[184] D. Pratiwi; «The Use of Self Organizing Map Method and Feature Selection in Image Database Classification System»; CoRR **abs/1206.0104** (2012). Cited on page 4

[185] J. Göppert & W. Rosenstiel; «Self-organizing Maps vs. Backpropagation: An Experimental Study»; dans «Proc. of Workshop on Disign Methodologies for Microelectronis and Signal Processing», p. 153–162 (1993). Cited on page 4

[186] J. N. Paulson, M. Pop & H. C. Bravo; «Metastats: an improved statistical method for analysis of metagenomic data»; **12**, p. P17. ISSN 1465-6906. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3439073/. Cited on page 29

[187] J. Ren, K. Song, C. Deng, N. A. Ahlgren, J. A. Fuhrman, Y. Li, X. Xie & F. Sun; «Identifying viruses from metagenomic data by deep learning»; http://arxiv.org/abs/1806.07810; 1806.07810. Cited on page 56

[188] D. Shrivastava, S. Chaudhury & D. Jayadeva; «A Data and Model-Parallel, Distributed and Scalable Framework for Training of Deep Networks in Apache Spark»; http://arxiv.org/abs/1708.05840; 1708.05840. Cited on page 101

[189] A. Nandi; *Spark for Python Developers* (Packt Publishing); ISBN 978-1-78439-969-6. Cited on page 101

[190] L. van der Maaten; «Barnes-Hut-SNE»; (2008)http://arxiv.org/abs/1301.3342; 1301.3342. Cited on pages 27 and 101

[191] J. Barnes & P. Hut; «A hierarchical O(N log N) force-calculation algorithm»; **324**, p. 446–449. ISSN 1476-4687. https://www.nature.com/articles/324446a0. Cited on page 101

[192] A. M. Bolger, M. Lohse & B. Usadel; «Trimmomatic: a flexible trimmer for Illumina sequence data»; **30**, p. 2114–2120. ISSN 1367-4811. Cited on page 65

[193] E. Pasolli, L. Schiffer, P. Manghi, A. Renson, V. Obenchain, D. T. Truong, F. Beghini, F. Malik, M. Ramos, J. B. Dowd, C. Huttenhower, M. Morgan, N. Segata & L. Waldron; «Accessible, curated metagenomic data through ExperimentHub»; **14**, p. 1023–1024 (2017). ISSN 1548-7105. https://www.nature.com/articles/nmeth.4468. Cited on page 67

[194] J. Rivera-Pinto, J. J. Egozcue, V. Pawlowsky-Glahn, R. Paredes, M. Noguera-Julian & M. L. Calle; «Balances: a New Perspective for Microbiome Analysis»; **3** (2018). ISSN 2379-5077. Cited on pages xix, 53, 67, and 86