Disease Prediction using Synthetic Image Representations of Metagenomic data and Convolutional Neural Networks

Thanh Hai Nguyen
College of Information & Communication Technology
Can Tho University
Can Tho, Vietnam
nthai@cit.ctu.edu.vn

Edi Prifti
Institute of Cardiometabolism and Nutrition,
Integromics,
Paris, France
e.prifti@ican-institute.org

Nataliya Sokolovska
Nutriomics team
Sorbonne University
Paris, France
nataliya.sokovska@upmc.fr

Jean-Daniel Zucker
UMMISCO
IRD, Sorbonne University
Paris, France
jean-daniel.zucker@ird.fr

Abstract — **Information from metagenomic data from human microbiome may improve diagnosis and prognosis for multiple human diseases. However, to achieve a prediction based on bacterial abundance information remains a challenge. Indeed, the number of features being much higher than the number of samples, we face difficulties related to high dimensional data processing, as well as overfitting. In this study, we investigate several convolutional neural network architectures for synthetic images and some experimental techniques to generate and train these synthetic images. We also explore supervised learning for visualizing high dimensional data that use data on genus, species and higher taxonomic level information. In addition, some dimensionality reduction approaches are examined on very high dimensional data such as gene families abundance. We evaluated our approach on six different metagenomic datasets including five types of diseases with more than 1000 samples. Our method displays promising results and can be used in different omics data settings, including integrative ones.**

Keywords — ***metagenomic, convolutional neural network, colormap, supervised manifold learning, synthetic images, disease prediction, image classification, dimensionality reduction***

I. INTRODUCTION

Metagenomics is a relatively novel field, which focuses on data obtained from ecosystem DNA. Raw data are usually treated bioinformatically in order to obtain the species composition, abundance and functional profiles [14][3]. A metagenomic sample is traditionally described by its microbial taxonomic composition that can be a relative abundance of microbial taxa of one of the taxonomic categories including kingdom, phylum, class, order, family, genus, or species. Determining relative abundance of a bacteria, and linking it to host diseases may allow us to improve diagnosis at its early stage. Such data could also provide a deeper understanding of the disease mechanism. However, the association of individual microbes with a particular type of disease has revealed inconsistent results due to different problems such as disease complexity, and the limited amount of observed data. Furthermore, biological data in general, and metagenomics in particular, are complex high-dimensional data, very hard to interpret by humans.

Important progress has been made in the metagenomics over the recent years. This leads to high requirements for computation. Machine learning has been increasingly used to solve problems in metagenomics. Several important problems faced in metagenomics are presented in [27] Operational Taxonomic Units (OTUs)-clustering, binning, taxonomic and assignment, comparative metagenomics and gene prediction. These problems are related to three types of machine learning tasks such as classification, clustering and dimensionality reduction.

High throughput data acquisition 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 today publicly available. Different statistical methods have been applied to classify patients from controls and some have also performed meta-analyses on multiple datasets [6]. 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 forests (RFs), and sparse regression.

Although deep learning (DL) has achieved significant success in many applying it to metagenomic data has exhibited inconsistent results [14]. Therefore, further research should investigate novel efficient approaches for deep learning. In this study we explore several architectures of Convolutional Neural Networks (CNN) applied to synthetic images generated using Met2Img [26]. Moreover, we present different techniques for generating synthetic images from metagenomic data. Our contribution is multifold:

- We illustrate a supervised approach for visualizing metagenomic based on Linear Discriminant Analysis (LDA) algorithm revealing a reasonable classification performance compared to t-SNE [26].
- We explore numerous deep learning architectures for metagenomic images generated by Met2Img and compare to VGG-like architecture [9].
- 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.
- A variety of colormaps including single-hue and

978-1-5386-9313-1/19/$31.00 ©2019 IEEE

multi-hue colormaps are evaluated to produce synthetic images efficiently.

- We also show that GPU can speed up the learning for CNN models, but it seems like not efficient for shallow learning such as Linear regression.

II. Related work

Machine learning algorithms have 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 human diseases [6][8][10][13][20], prevention of diseases [21][11], and personalized treatment [23]). Some authors used state of the art ML algorithms, classifiers such as support vector machines (SVMs), random forests (RFs), Lasso, and Elastic Net (ENet), implemented in a computational framework (MetAML) [6]. They used it in the context of metagenomic data generated form multiple studies where data were processed bioinformatically in the same standardized way. We systematically compared our approach with results from MetAML in the same benchmark datasets.

Among various methodological variants of DL networks, the CNN have been extensively studied [12], especially in the field of image processing. Noteworthy, CNN are able to perform better than humans in some applications [24]. 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 has led to several studies, see, [14] for instance, where it was reported that "the DL approaches may not be suitable for metagenomic applications". In this study, we challenge this question, and show that DL is an efficient approach, which achieves very reasonable results on metagenomic data compared to standard machine learning methods.

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. [18] 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) [19] and Isometric Mapping (Isomap) [15] were considered as a new generation of dimensionality reduction algorithms and applied to synthetic and real data with great achievements. Authors in [25] [26] leveraged t-SNE [1] as a strong tool for visualizing metagenomic data. However, t-SNE is an unsupervised method, visualizations learned from samples along with their labels have not investigated completely. Therefore, in this study, we explore Linear Discriminant Analysis (LDA) [28] to visualize data in a supervised manner using labels from higher taxonomic levels.

III. Deep learning for metagenomics using embeddings and convolutional neural network

In order to apply deep learning to metagenomic data, we, first, transform data to images using various methods such as Fill-up [25][26] or manifold learning algorithms (t-SNE, LDA, Isomap, etc.), then applying the proposed CNN to these images for disease classification. We evaluated our approach on six different datasets (detailed in Table I) including data projected at the species level from various diseases, namely: liver cirrhosis (**CIR**), colorectal cancer (**COL**), obesity (**OBE**), inflammatory bowel disease (**IBD**) and Type 2 Diabetes (**T2D**) [26]. In addition, one dataset, namely **WT2**, that includes 96 European women with n=53 T2D patients and n=43 healthy individuals was also considered [6][26]. Beside species abundance, we also evaluate on the data which include the abundance of gene families generated by HMP Unified Metabolic Analysis Network (HUMAnN2) [32] with very high dimension being up to more than one million features. The data are downloaded from *curatedMetagenomicData* [33] in R.

The abundance of species or gene families is a relative proportion represented as a real number. The total abundance of the species is summing to 1. The results with gene families abundance are shown in IV.D (*Experiments*), the results using species abundance are presented in the rest.

TABLE I. Information on 6 datasets. #features (1) is the number of features with species abundance while #features (2) shows the number of features of gene families abundance.

	CIR	COL	IBD	OBE	T2D	WT2
#features (1)	542	503	443	465	572	381
#features (2)	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

A. Generation of metagenomic images

1) Binning for synthetic images

Various types of bins implemented in Met2Img[26] aim to reduce the effects of minor observation errors, and to get rid of noise in the data. The Species Bin (**SPB**) for data formed in a bell-shape distribution and Quantile Transformation (**QTF**) to fit the features to a uniform distribution showed very promising performances [26]. In this study, we leverage these binnings to generate the images. SPB was conducted from species distribution with breaks owning values of breaks from [0, 10^{-7}, $4*10^{-7}$, $1.6*10^{-6}$, ..., $6.5*10^{-3}$, 1] while QTF based on a scaling factor which is learned in the training set and then applied to the test set. Quantile Transformation (QTF) is a Non-Linear transformation, which is considered as a strong preprocessing technique because of reducing the effect of outlier. Values in new/unseen data (test/validation set) which are lower or higher the fitted range will be set to the bounds of the output distribution.

2) Colormaps for images

We illustrate abundance of features into images, we use various color palettes depending on the abundance of the feature. We consider nine colormaps provided in Python and propose a custom colormap based on *jet* combined to *black*.

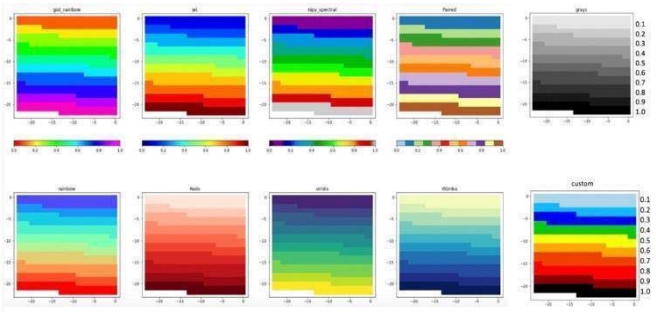

Fig. 1. Considered colormaps. Top: Left-right: gist_rainbow, jet, nipy_spectral, paired, grays. Bottom: Left-Right: rainbow, reds, viridis, YlGnBu and custom.

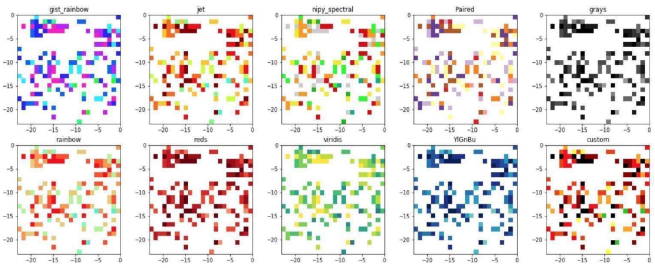

Fig. 2. Examples of Fill-up on a sample of CIR dataset using SPB with 10 different colormaps in Fig. 1. Top: Left-right: gist_rainbow, jet, nipy_spectral, paired, grays. Bottom: Left-Right: rainbow, reds, viridis, YlGnBu and custom.

Visualizations of 10 various colormaps with 10 distinct colors (10 bins) for each colormap are illustrated in Fig. 1. In Fig. 2, we illustrate a sample from the CIR dataset using 10 different colormaps.

3) Simple way to transform data to images with Fill-up

In [26], the authors proposed "Fill-up" to arrange features into a square matrix of a given size. Metagenomic images are generated 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 random or following a specific order such as for instance phylogenetic classification for the known species.

4) Visualizations based on supervised dimensionality reduction algorithm

Synthetic images offer a good way to visualize the structure of the data. High-dimensional datasets such as metagenomics are usually very difficult to explore and to interpret corresponding machine learning models. However, we can try to transform the raw data to smaller dimensions. A key idea of this approach is that we can find the structures of high-dimensional data shaped in 2D images, where optimized DL techniques can be applied to improve prediction results. We applied a supervised learning approach based on the Linear Discriminant Analysis (LDA) [28] algorithm which fits a Gaussian density to each class, while 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.

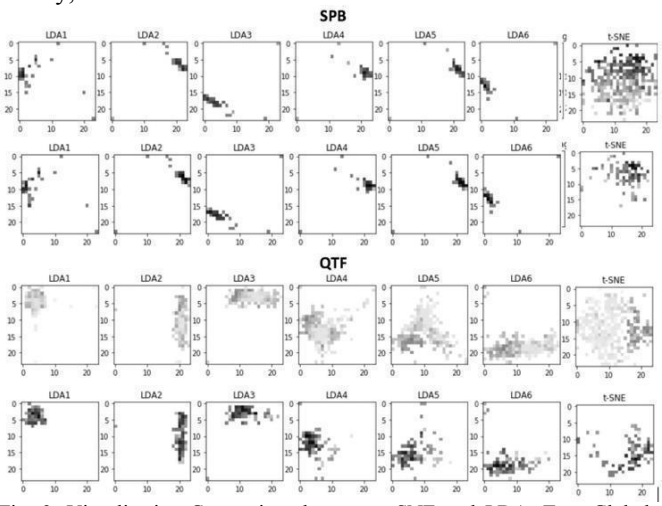

Fig. 3. Visualization Comparison between t-SNE and LDA. Top: Global maps. Down: images of samples created from the global maps above. LDA1, LDA2, LDA3, LDA4, LDA5, LDA6 correspond to the labels of Kingdom, Phylum, Class, Order, Family and genus, respectively.

The features of all samples in training set from raw data are transformed by LDA into a global map. This map is then used to generate images for training and testing data.

Fig. 3 compares visualizations between t-SNE and LDA. We demonstrate in Fig. 3 that t-SNE along with the SPB outperforms LDA. With QTF, LDA improves the visualization with less overlapped points than SPB. LDA in QTF that uses family or genus as labels show better results compared to the others. This proves that LDA is very sensitive to outliers. When we provide the labels for the supervised dimensionality reduction algorithm LDA with various taxonomic levels, only the result with labels with the lowest level, genus, obtains some relative clear clusters of genera to distinguish a vast number of species. The best visualizations focus on the taxonomic levels corresponding to order, family, genus levels. This reveals that with levels close to species, the relationship among features can be easily explored.

B. Deep learning architectures for metagenomic data

In this paper [9], the authors presented how network depth affects the performance of CNN. The authors designed deep architectures of CNN with very small convolutions (3x3). They stated that the depth up to 16-19 layers was able to achieve a considerable improvement. VGGNet [9] achieved top-1 validation error and top-5 validation error were 23.7% and 6.8%, respectively. We use VGG-like architecture as depicted in Fig. 4 and compared it to shallow CNN architectures as an example shown in Fig.5.

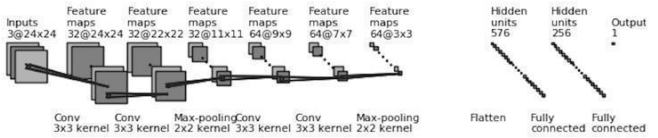

Fig. 4. The architecture of VGG-like convent.

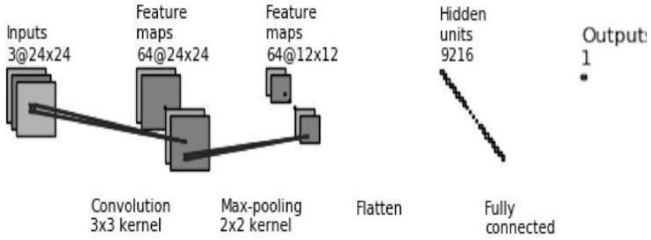

Fig. 5. An example of a shallow CNN architecture for metagenomic synthetic images.

C. Using dimensionality reduction algorithms for data with a very large number of features

For gene families abundance, the number of features can be reached to more than one million, so dimensionality reduction is very necessary. Here, we apply two well-known dimensionality reduction algorithms including Random Projection [30] and Principal Component Analysis (PCA) [31]. These are simple, and computationally efficient methods to reduce the data dimension in an unsupervised manner. 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.

IV. THE EXPERIMENTS

All networks use the same optimization function (Adam [7]), learning rate = 0.001, loss function: binary cross

entropy, epoch = 500 (using Early Stopping to avoid overfitting with the number of epoch patience of 5), and a batch size of 16. The performance of the classifier is assessed by measuring the accuracy (ACC) on 10-fold-stratified-cross validation, repeated and averaged on 10 independent runs. The significant results (p-*value* < 0.05) compared to MetAML [6] (using Random Forest) in the tables are reported in **bold.**

A. Performance of CNN with different architectures

Table II compares to VGG-like convolutional networks (VGG) proposed in Keras [17]. *CNNlxfy* in Table II denotes to the CNN contains *x* convolutional layer(s), *y* filters for each convolutional layer and a max pooling of 2x2 after all convolutional layers while *drfc*, *drcnn* reveal dropout rate in Fully Connected (FC) layers, Convolutional layers, respectively.

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. In most of cases, the architectures of CNN with one convolutional layer obtain greater performance than two-convolutional-layer and VGG-like architectures. However, for WT2, the architectures with two convolutional layers seem to give a slight improvement. Our results exhibit that the shallow architectures of CNN including one and two convolutional layer outperform deep architectures such as VGG-like.

TABLE II. PERFORMANCE ON DIFFERENT ARCHITECTURES FOR METAGENOMIC IMAGES USING GRAY IMAGES WITH FILL-UP AND SPB.

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

For shallow CNN, dropout in FC seems like it is not improving the performance but it works on deep networks such as VGG. In our experiments, VGG with dropout rate of 0.2 reveals the best performance among variations of VGG. For some cases, combination between dropout in convolutional layers and FC layers improves the performance compared when applying to only the FC layers (in CNN-***l1f32*** (***l1f32*** means the CNN includes one convolutional layer with 32 filters), CNN-l2f16). For CIR, CNN with one convolutional layer outperform the others. CIR and IBD reach to the peak at CNN-l1f64 while T2D and OBE perform the best at CNNl1f16 and CNNl1f20, respectively. The use of dropout is apparently not efficient.

B. Performance of CNN with various colormaps

We test a variety of colormaps, namely *gist_rainbow, jet, nipy_spectral, paired, grays, rainbow, reds, viridis, and YlGnBu* for images. We also propose another colormap based on *jet* combining to *black*, namely *custom*. In Table III and Table IV, we show the results with the colormaps supported in Python library, Matplotlib [29] and the proposed colormap. Visualizations of 10 various colormaps with 10 distinct colors (10 bins) for each colormap are illustrated in Fig. 1.

For the results of SPB in Table III, we see that *jet, custom*, and *grays* perform encouraging results both FC and CNN models. Significantly, *jet* obtains 3 significant results and perform approximately to other datasets both the architecture with only one Fully Connected layer (FC) and CNN architectures compared to MetAML [6] 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.

TABLE III. ACCURACY PERFORMANCE ON VARIOUS COLORMAPS FOR IMAGES WITH SPB USING FILL-UP

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

For QTF bins (Table IV), *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 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 Fig. 1, 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* enable 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.

TABLE IV. ACCURACY PERFORMANCE ON VARIOUS COLORMAPS FOR IMAGES WITH QTF USING FILL-UP

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

C. *Performance of supervised visualizations*

TABLE V. PERFORMANCE 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

	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 V exhibits the performance of LDA with many levels of OTUs as labels. In general, the results are rather low with FC model. We pick up the best result with level of family (level of 5) to perform prediction tasks with CNN.

As seen from Fig. 6, for visualizations based on LDA, CNN outperforms FC. Comparing to the performance of t-SNE in [26], we find quite interesting that although LDA exists many overlapped points in visualizations, LDA can outperform t-SNE on the CIR dataset. For other datasets, LDA with CNN performs comparable results compared to t-SNE.

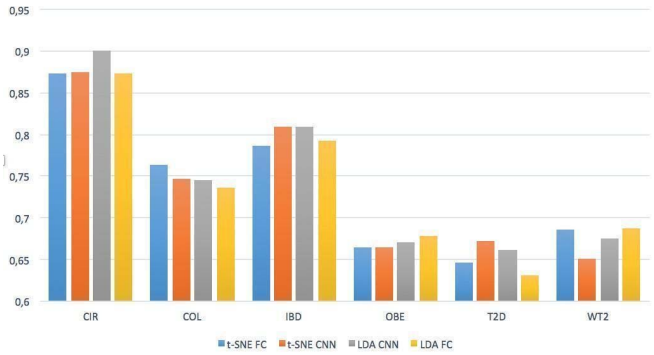

Fig. 6. Accuracy comparison between t-SNE in [26] and LDA.

D. *The results with Fill-up after applying dimensionality reduction algorithms to gene families abundance*

In this section, we explore some dimensionality reduction algorithms such as Random Projection (revealed as "RD_PRO" in Table VI) and PCA to apply to the original gene families abundance (raw data) to reduce the dimension from over one million to 576 which is equivalent to an image of 24x24, then we generate the gray images (24x24) using Fill up approach with QTF on the data reduced the dimension. As we can see the performance of FC model for gene families abundance from Table VI, 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 comparing to RD_PRO. The average performance on raw data without 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 average accuracy from 0.605 (raw-PCA) to 0.697 (fill up - PCA). RD_PRO shows a worse performance, but it also shares the same pattern with PCA with the improvement in Fill-up.

TABLE VI. COMPARISON (IN ACCURACY) BETWEEN RAW DATA AND REDUCED DIMENSIONALITY DATA USING RANDOM PROJECTION AND PCA WITH THE SAME MODEL FC ON GENE FAMALIES ABUNDANCE. THE SIGNIFICANT RESULTS COMPARED TO THE ORIGINAL ABUNDANCE ARE REPORTED IN **BOLD**.

Rep		CIR	COL	IBD	OBE	T2D	WT2	AVG
Fill-up	PCA	0.757	**0.709**	**0.832**	**0.663**	0.626	0.593	0.697
Fill-up	RD_PRO	0.701	**0.709**	0.785	0.658	0.581	0.526	0.660
Raw	PCA	0.547	0.604	0.775	0.648	0.514	0.540	0.605
Raw	RD_PRO	0.555	0.605	0.775	0.648	0.496	0.530	0.602
Raw		0.761	0.628	0.775	0.648	0.655	0.620	0.681

E. *Execution time*

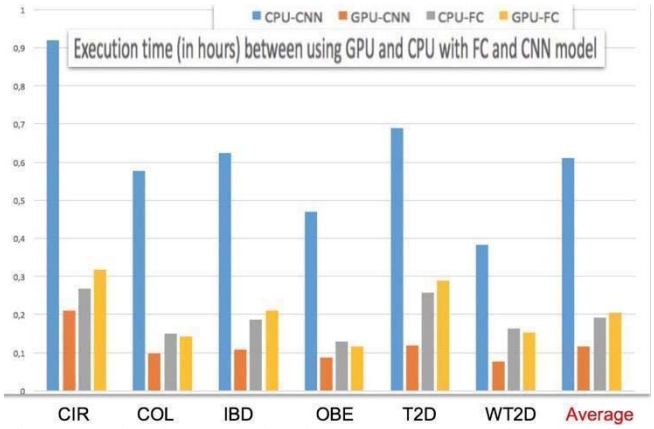

Fig. 7. Execution time comparison between using GPU and CPU for different models.

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 Fig. 7. The resolutions of images range from 20x20 to 24x24. For shallow 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 shallow FC model.

V. CONCLUSION

The result comparison between t-SNE and LDA exhibits a potential way to leverage information from higher taxonomic levels to improve the performance. However, LDA still suffers the overlapping issue. Further research to resolve this issue can improve the performance of LDA transformation.

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 its width by increasing the number of filters to attempt to improve the performance.

The results on gene families abundance after using dimensionality reduction algorithms also exhibit considerably improvements for FC model in terms accuracy. This shows that our approach can work with the data which reduced to a lower dimension.

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

Finally, we noticed that the GPU framework works very efficiently for the CNN while on simple models such as FC, the GPU does not show significant improvement compared to classical CPU setting. The software and the experimental framework are publicly available at https://git.integromics.fr/published/deepmg.

REFERENCES

[1] L. v. d. Maaten & G. Hinton; “Visualizing Data using t-SNE”; J. Mach. Learn. Res. 9, p. 2579–2605 (2008).

[2] L. Deng & D. Yu; “Deep Learning: Methods and Applications”; 7, p. 197–387. ISSN 1932-8346, 1932-8354. https://nowpublishers.com/article/Details/SIG-039.

[3] H. W. Virgin & J. A. Todd; “Metagenomics and personalized medicine”; Cell 147, p. 44–56 (2011).

[4] R. Garreta & G. Moncecchi; “Learning scikit-learn: Machine Learning in Python”. (Packt Publishing Ltd) (2013).

[5] I. Goodfellow, Y. Bengio & A. Courville; “Deep Learning” (MIT Press) (2016)

[6] 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. e1004 977 (2016).

[7] D. P. Kingma & J. Ba; "Adam: A Method for Stochastic Optimization"; CoRR abs/1412.6980 (2014)http://arxiv.org/abs/1412.6980; 1412.6980.

[8] 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).

[9] Very Deep Convolutional Networks for Large-Scale Image Recognition (3rd International Conference on Learning Representations (ICLR2015)) (2015).

[10] 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).

[11] 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).

[12] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang & G. Wang; "Recent Advances in Convolutional Neural Networks"; CoRRabs/1512.07108 (2015).

[13] 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).

[14] G. Ditzler, R. Polikar & G. Rosen; Multi-Layer and Recursive Neural Networks for Metagenomic Classification; IEEE Trans. Nanobioscience 14, p. 608–616 (2015).

[15] J. B. Tenenbaum, V. de Silva & J. C. Langford; "A Global Geometric Framework for Nonlinear Dimensionality Reduction"; Science 290, p. 2319 (2000).

[16] D. Masters & C. Luschi; "Revisiting Small Batch Training for Deep Neural Networks"; http://arxiv.org/abs/1804.07612; 1804.07612

[17] F. Chollet et al.; "Keras"; https://keras.io (2015).

[18] 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

[19] 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.

[20] J. Ning & R. G. Beiko; "Phylogenetic approaches to microbial community classification"; 3. ISSN 2049-2618. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4593236/.

[21] 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.

[22] 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.

[23] 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

[24] 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

[25] 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.

[26] 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.

[27] H. Soueidan & M. Nikolski; “Machine learning for metagenomics: methods and tools”; Metagenomics 1 (2017).

[28] P. Xanthopoulos, P. M. Pardalos & T. B. Trafalis; “Linear Discriminant Analysis”; SpringerBriefs in Optimization; p. 27–33 (Springer, New York, NY); https://link.springer.com/chapter/10.1007/978-1-4419-9878-1_4.

[29] J. D. Hunter; “Matplotlib: A 2D graphics environment”; Computing In Science & Engineering 9, p. 90–95 (2007).

[30] S. S. Vempala; The Random Projection Method (American Mathematical Soc.) (2005).

[31] K. Pearson; On Lines and Planes of Closest Fit to Systems of Points in Space (1901).

[32] S. Abubucker et al. “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.

[33] E. Pasolli et al. “Accessible, curated metagenomic data through ExperimentHub”; 14, p. 1023–1024 (2017). ISSN 1548-7105. https://www.nature.com/articles/nmeth.4468