Time Efficient and Novel Ways of Analyzing High-dimensional Multi-omics Datasets: Parallel Computing and Multi-view Learning

by

Rayhan Shikder

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science

University of Manitoba

Winnipeg

Abstract

Omics data (e.g., genomics, proteomics, microbiomics etc.) are generally very high-dimensional. Due to the advancements in high-throughput sequencing technologies, this type of data is rapidly increasing in number and require enhanced computational power to make sense. On the other hand, an integrated analysis of these omics data has the potential to reveal a comprehensive picture of any biological phenomenon than analyzing them separately (one omics data at a time).

In the first part of this thesis, I focused on developing time-efficient CPU-based parallel computing methods for computing longest common subsequence (LCS) of DNA sequence data. I developed shared, distributed, and hybrid memory algorithms for calculating the LCS between two DNA sequences and compared their performances. The shared memory implementation with OpenMP has been found to outperform others with an absolute speedup of 2 times than the best sequential LCS algorithm and a relative speedup of 7.

Multi-omics (multi-view) datasets measured on the same individuals are comprised of high-dimensional omics data, where different types of data from different views might have strong associations. Studying these associations is of paramount interest to biologists. Furthermore, these associations along with the view-specific omics data have the potential to improve the predictive power of the overall datasets. Recently, deep neural networks have received increased attention due to their ability to deal with high-dimensional data and keep the inherent relationships among and within the data views. Therefore, in the second part of this thesis, I focused on studying deep neural network (DNN)-based methods for integrating and analyzing a special type of multi-omics data consisting of gene expression and microbiome abundance data. At first, I performed a comprehensive comparison study of some existing correlation-based integration techniques (regularized canonical correlation analysis (RCC), deep canonical correlation analysis (DCCA), sparse canonical correlation

analysis (SCCA), etc.). The SCCA has been found to provide better correlation scores along with good classification performance than others. After that, I worked on improving the deep canonical correlation analysis by incorporating the class label information. I provided a new framework along with two new supervised versions of the deep canonical correlation analysis (namely DCCA S1, and DCCA S2). From the experimental results, I found that the new supervised versions provide better total correlation scores than the original DCCA method. Finally, I developed new DNN based methods for classifying multi-omics data. These methods are found to provide better or at-least similar results than the existing classical classification algorithms (e.g., support vector machine, random forest, etc.).

In summary, this thesis has two main contributions in the field of computer science and bioinformatics. First, the development of new CPU-based improved parallel algorithms for finding the LCS of DNA sequence data. Second, the thorough investigation of the existing canonical correlation analysis (CCA)-based multi-view learning methods for analyzing multi-omics data along with developing new multi-view learning frameworks and supervised DNN-based multi-view learning techniques.

Acknowledgements

At first, I would like to express my sincere gratitude to my supervisor Dr. Pingzhao Hu for providing continuous support and valuable guidelines throughout this two-year of my master's degree at the University of Manitoba. His thoughtful instructions and mentoring helped me extend my comfort zones and contributed to my overall academic growth. I am also grateful to my co-supervisor Dr. Pourang Irani for mentoring me in different steps of my thesis. He was always there when I needed any help.

I would also like to extend my gratitude to my advising committee members Dr. Carson Kai-Sang Leung, and Dr. Bob McLeod for their valuable time, and patience in reviewing my thesis work.

I would specially thank Dr. Parimala Thulasiraman for guiding me in designing the parallel algorithms of this thesis. I would also thank Dr. Hui Jiang for mentoring me in one of my projects (DNN Y model) related to this thesis during my internship at York University.

Special thanks to all my friends and colleagues at The Hu Lab: Md. Mohaiminul Islam, Jiyaing You, Dr. Svetlana Frenkel, Nikta Feizi, Qian Liu, Shuo Jia, Yong Won Jin, and Sujun Huang. Their friendly chit-chats in the break times and thoughtful insights not only helped me to enjoy this journey but also kept me motivated throughout the whole process.

Thanks to all the faculty members and staffs from the Computer Science department, and the Biochemistry and Medical Genetics department who helped me in various steps of my degree.

Finally, I would like to dedicate this thesis to my parents whose invaluable sacrifice, love, and support have made it possible for me to reach all the milestones I have achieved so far.

Publications

Shikder, R., Thulasiraman, P., Irani, P., & Hu, P. (2019). An OpenMP-based tool for finding longest common subsequence in bioinformatics. *BMC research notes*, *12*(1), 220.

Shikder, R., Irani, P., & Hu, P. (2019, May). Genome-Wide Canonical Correlation Analysis-Based Computational Methods for Mining Information from Microbiome and Gene Expression Data. In *Canadian Conference on Artificial Intelligence* (pp. 511-517). Springer, Cham.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

DNA: Deoxyribonucleic Acid

RNA: Ribonucleic Acid

mRNA: Messenger RNA

rRNA: Ribosomal RNA

SNV: Single-Nucleotide Variant

CNV: Copy Number Variation

CPU: Central Processing Unit

GPU: Graphics Processing Unit

LCS: Longest Common Subsequence

MPI: Message Passing Interface

OpenMP: Open Multi-Processing

CUDA: Compute Unified Device Architecture

PCAM: Partition, Communication, Agglomeration, and Mapping

UCR: University of California Riverside

NCBI: National Centre for Biotechnology Information

CCA: Canonical Correlation Analysis

DCCA: Deep CCA

RCC: Regularized CCA

SCCA: Sparse CCA

SCCA(S): Supervised SCCA

KCCA: Kernel CCA

GSCCA: Group Sparse CCA

DNN: Deep Neural Network

DCCA S1: Supervised DCCA Version 1

DCCA S2: Supervised DCCA Version 2

LDA: Linear Discriminant Analysis

SVM: Support Vector Machine

RF: Random Forest

PCA: Principal Component Analysis

OTU: Operational Taxonomic Unit

IBD: Inflammatory Bowel Disease

t-SNE: T-Stochastic Neighbor Embedding

FAP: Familial Adenomatous Polyposis

NP: No Pouchitis

AP: Acute Pouchitis

CDL: Crohn's Disease-Like Inflammation

PMA: Penalized Multivariate Analysis

ROC: Receiver Operating Characteristics

AUC: Area Under Curve

ReLU: Rectified Linear Unit.

Chapter 1: Introduction

1.1 Background and Literature Review

1.1.1 Omics and Multi-omics Data

Large scale studies focusing on the measurement of different characteristics of family of cellular molecules (e.g., genes, proteins, metabolites etc.) are known as omics studies and end with a suffix of *-omics*, such as genomics, proteomics, metabolomics etc. For instance, proteomics is a type of omics study where the composition, structure, and activity of proteins are studied in large scale. Studies (genomics, transcriptomics, proteomics) related to the central dogma (from DNA to mRNA to protein) in biology are mainly considered as part of the omics study. With the advancement of technologies, and computational powers, some other areas of biomedical science have started to become the part of omics level study [1]. Such fields are epigenomics, metabolomics, microbiomics etc.

Data arising from different parts of these omics studies are known as omics data. For example, DNA sequence data (strings of nucleotides: A, T, C, G), gene expression levels, mutation information are omics data originated from genomics. In proteomics, 3D structure of protein, protein-protein interactions, amino acid sequences are the omics data too. Multiple such omics data of the same individuals are aggregately referred to as multi-omics data. Omics data generated and analyzed in an isolated manner fail to provide a comprehensive picture of the situation compared to an integrated study of the multi-omics data.

In this thesis, I have mainly focused on multi-omics data consisting of gene expression and microbiome abundance counts. Gene expression data consists of continuous values and represents the level of expression for a particular gene. The protein coding regions of a human genome is only 2% of the whole genome and encodes around 20,000 genes (3

billion DNA base pairs) [2, 3]. The genomics data used in this thesis consists of gene expression from these genes. I also worked on another omics data consisting of microbiome abundance profiles. This data is generated by sequencing microbial genomes using high-throughput sequencing technologies (amplicon sequencing or 16S rRNA sequencing) and then counting the sequences of different taxa of microbial community [4]. Therefore, the data represents the number of sequences for a particular taxon in a sample.

1.1.2 Parallel Computing

The volume of data captured in a regular basis for different purposes by different technologies are increasing in an overwhelming rate and are commonly known as big data. Massive amount of computational power is required to make sense of this huge amount of data. The doubling of number of transistors in a single integrated circuit in every two years (Moore's law [5]) has led to a substantial growth in computational power of computers. In addition to this growth in computation power, a new paradigm of computing, parallel computing, has emerged where many calculations or process executions can be performed simultaneously [6]. Parallel computing coupled with the growth in computational power has made the impossible task of making sense of big data possible. Here, in this thesis, I will work on a specific branch of parallel computing focused on CPU based parallelism.

1.1.3 Multi-view Learning Using Multi-omics Data

In a setting where multiple views of the same individuals are available, multi-view learning methods aim to integrate them in a manner which maximizes the outcome of the learning problem of interest. As multi-view data is commonly observed in various situations, and analyzing them using multi-view learning approaches has been found to be effective, various multi-view learning methods have been developed. Based on the taxonomy provided by Zhao

et al. [7] these methods fall into three groups: 1) co-training type algorithms, 2) co-regularization type algorithms, and 3) margin-consistency type algorithms.

Originally proposed by Blum and Mitchell [8], co-training is one of the first methods of multi-view learning for semi-supervised settings. This method, at first, focuses on training multiple classifiers on multiple views separately using labeled data. After that, it tunes the classifiers by making predictions on the unlabeled data using all the classifiers separately with a goal to maximize the consensus of predictions among them. Examples of co-training type algorithms include co-EM [9], co-EMT [10], co-testing [11], Bayesian co-training [12], co-clustering [13] etc.

Co-regularization-based algorithms use different approaches to augment regularization to the objective function to make multiple views consistent with each other [7]. The approaches to augment regularization may be categorized as early and late regularizations. In the early regularization approach, the original data is usually projected in a new space in an unsupervised or supervised manner, and then the projection is used for downstream analyses (classification, clustering etc.). The methods for unsupervised projection mainly include correlation-based algorithms where the features in the newly transformed space are constrained to be as similar as possible. Examples of this type include canonical correlation analysis (CCA) [14–16], kernel CCA [17, 18], Bayesian CCA [19], Tensor CCA [20] etc. Recently, several deep neural network-based models have been developed to achieve unsupervised projection. Deep canonical correlation analysis (DCCA) [21], split autoencoders (SplitAE) [22] are two classic examples of this type. Whereas the above correlation-based approaches only consider the similarity among the transformed features, by exploiting the original class information, the supervised method of projection considers the dissimilarity among the views as well. Discriminative CCA [23–25], multi-view fisher

discriminant analysis [26], and multi-view linear discriminate analysis (LDA) [27] are examples of this type. On the other hand, in the late regularization approach, instead of adding regularization to the features, the classifiers or regressors are regularized to make their outputs as similar as possible. SVM-2K (a combination of kernel CCA and Support Vector Machine (SVM)) [28], multi-view twin SVMs [29] are some examples of the late regularization approach.

Finally, the margin consistency type algorithms aim to find the hyperplanes which have a consistent margin with the sample data points among views. Examples include multi-view maximum entropy discrimination (MVMED) [30], soft margin consistency-based multi-view MED [31], and consensus and complementarity-based MED (MED-2C) [32] etc.

1.2 Motivation

DNA sequence data consisting of nucleotide base pairs (adenine, thymine, guanine, cytosine) is a kind of omics data which represents the sequence of base pairs in a genome of a particular organism. The sequence data is the blueprint of life as it contains the information of genes which determines biological features of an organism. Studying the differences or similarities among DNA sequences of different organisms allows us to understand the relationships among those organisms, and may answer many important biological questions. However, for most of the organisms (e.g., human) the sequence data is very large in size. For instance, the human genome consists of 3 billion nucleotide base pairs. This large amount of data incurs challenges in their analysis. Parallel computing of this type of omics data can provide significant ease in the computation. Hence, in the first part of this thesis, I have focused on developing parallel computing solutions for a special type of comparative analysis of DNA sequences.

In the second part of this thesis, I developed new computational techniques for integrating and making sense of multi-view (e.g., multi-omics) data. Datasets comprised of multiple feature sets from multiple sources (also known as "views") measured on the same individuals or subjects are known as multi-view data. Examples may include publication data comprised of texts and citation information, video data comprised of images and audio signals, multi-omics data comprising of genomes, metabolomes, microbiomes etc. Learning objective function of interest (e.g., clustering, classification etc.) from these types of datasets using only single view will not be able to capture all the available information. A straightforward solution is to directly concatenate features from multiple views into a single view and then learn from that, which may provide somewhat better performance than the former approach. However, this approach has several drawbacks. First, multiple views of the datasets may contain different statistical properties, and concatenating them in a straightforward manner may prevent recognizing and exploiting the individual properties of each view [7, 33, 34]. Second, features from different views could be of disparate data structures (e.g., vectors, graphs, trees etc.), which adds challenges to the straightforward merging of them [33]. Finally, straightforward concatenation will lead to a higher number of dimensions in the resulting feature space, which may induce over-fitting given a limited number of training samples [7]. To address these issues, a new paradigm named multi-view learning has been developed, which aims to exploit relationships among and within the views to provide better learning performance than single view methods while discovering patterns from the multi-view data [35, 36].

In my thesis, I am interested in multi-view data originating from bioinformatics field. To be specific, my work is focused on multi-view learning using multi-omics data comprised of gene expressions and microbiome abundance profiles. Multi-omics data have some fundamental differences in characteristics compared to other multi-view data (e.g., image,

audio, web pages etc.), as the number of features in multi-omics data is very large with respect to the number of samples. For instance, in one of our used datasets, both the gene expression and the microbiome views consist of hundreds to thousands of features making the datasets very high dimensional in nature but commonly there is a relatively small number of samples. In most of the cases, combining host gene expression data with microbiome abundance data results in a unique representation where the number of features (e.g., tens of thousands) are very large compared to the number of available samples (e.g., hundreds). This increased number of features incur the curse of dimensionality problem while analysing the data [37]. These enormous number of features provide new challenges for using most of the mathematical and statistical evaluation methods [38]. With such high dimensional data, it is often the case that a subset of the dimensions represent irrelevant information. Therefore, prior to learning any objective functions of interest, these datasets need to be reduced to a lower dimensional subspace using useful feature selection and/or dimension reduction methods. However, one should keep in mind the multi-view nature of the datasets too, as the reduced latent or feature space should retain the original information from individual views as well as the relationship among the views. There exist a number of dimension reduction techniques: principal component analysis (PCA), correlation-based approaches (e.g., canonical correlation analysis, regularized canonical correlation analysis etc.) [16], multi-dimensional scaling, deep learning approaches (e.g., Autoencoders) [39, 40], tensor decomposition-based methods [41, 42] etc.

The role of human microbiome in human health has been found to be significant. Human microbiome has influence in different important functions such as: digestion, immune system etc. The alterations in the microbiome community composition has been found to be associated with several diseases such as: psoriasis, acne, inflammatory bowel disease, Crohn disease etc. [4, 43, 44]. Analysis of microbiome abundance data has some special challenges

due to its sparsity and compositional nature [4]. As this is count data one has to deal with skewed distribution, and over-dispersion. Besides, the counts of different operational taxonomic units (OTU) are highly variable which requires some preprocessing steps before analyzing. The total count per sample is usually constrained by the maximum read count limit of the specific DNA sequencer used. This makes the count information highly relative to each other which needs to be considered while analyzing this type of data.

1.3 Research Objectives

The objective of my thesis is two-fold. **First**, to investigate and develop novel and time-efficient ways of analyzing high-dimensional omics datasets using parallel computing techniques. **Second**, to develop novel framework for multi-view learning of multi-omics data. Correlation-based approaches for feature learning and dimension reduction are mainly considered. Furthermore, I also explore answers of some novel biological questions.

Chapter 2: CPU-based Parallel Algorithms for Finding LCS Using Omics Data

Finding the longest common subsequence (LCS) among sequences in a time-efficient way is a non-trivial problem. It has significant demand in many sectors specifically in the field of bioinformatics for finding DNA sequence alignment and pattern discovery. Here, I propose new CPU-based parallel implementations, which can provide significant advantages in terms of execution times, monetary cost, and pervasiveness in finding LCS of DNA sequences in a setting where Graphics Processing Units (GPUs) are not available. For general use, I also made the OpenMP-based tool publicly available for the end-users. In this study, I developed novel parallel versions of the LCS algorithm on message passing interface (MPI), OpenMP, and hybrid MPI-OpenMP platforms. The experimental results with both simulated and real DNA sequence data show that the OpenMP implementation provides at least two-times absolute speedup than the best sequential version of the algorithm and a relative speedup of almost 7. I provided a detailed comparison among the implementations on different platforms and different versions of the algorithm in terms of execution times. I also showed that removing branch conditions negatively affects the performance of the CPU-based parallel algorithm on OpenMP platform.

2.1 Introduction

Finding the Longest Common Subsequence (LCS) is a classic problem in the field of computer algorithms and has diversified application domains. A subsequence of a string is another string which can be derived from the original string by deleting none or few characters (contiguous or non-contiguous) from the original string. A longest common subsequence of two given strings is a string which is the longest string that is a subsequence of both the strings. The sequential version of the LCS algorithm using "equal-unequal" comparisons takes $\Omega(mn)$ time, where m and n represent the length of the two sequences being compared [45, 46]. It is necessary to mention that the problem of finding the LCS of more than two strings is NP-Complete [47, 48].

LCS has various applications in multiple fields including DNA sequence alignment in bioinformatics [49–51], speech and image recognition [52, 53], file comparison, optimization of database query, etc. [54]. In the field of bioinformatics, pattern discovery helps to discover common patterns among DNA sequences of interest which might suggest that they have biological relation among themselves (e.g., similar biological functions) [55]. In discovering patterns between sequences, LCS plays an important role to find the longest common region between two sequences. Although a praiseworthy amount of efforts have been made in the task of pattern discovery, with the increase of sequence lengths, algorithms seemingly face performance bottlenecks [56]. Furthermore, with the advent of next-generation sequencing technologies, sequence data is increasing rapidly [57], which demands algorithms with minimum possible execution time. Parallel algorithms can play a vital role in this regard.

Out of the parallel solutions of the LCS problem, anti-diagonal [58] and bit-parallel [59] algorithms are few of the firsts and noteworthy attempts. Recently, with the rise of Graphics Processing Unit (GPU)- based accelerators, several Compute Unified Device Architecture (CUDA)-based GPU targeted solutions to the LCS problem have been proposed. Yang et al. [60] are one of the firsts to propose an improved row-wise independent parallel version of the LCS algorithm by changing the data dependency used by a dynamic programming approach and using unique memory-access properties of GPUs. More recently, Li et al. [61] have proposed a parallel formulation of the anti-diagonal approach to the LCS algorithm using a GPU-based model. Although these GPU-based models offer faster execution times, not many computers (compared to CPUs) are equipped with GPUs. In such cases, to achieve performance improvement, CPU-based (e.g., MPI, OpenMP) parallel LCS algorithms are still greatly demanded. However, to the best of our knowledge, there is no such publicly available CPU-based tool for the end-users. I addressed this gap by developing a new OpenMP-based tool for the end-users by improving the row-wise independent version [60] of the LCS algorithm. Moreover, I also developed two other CPU-based parallel implementations (MPI, hybrid MPI-OpenMP) of the algorithm and provided a detailed benchmarking of all these implementations on simulated and real DNA sequence data, which was absent for this version of the LCS algorithm. The main contributions of this study are listed below.

1. A new OpenMP-based publicly available tool for finding the length of LCS of DNA sequences for the end-users.

2. A detailed benchmarking of the newly developed CPU-based parallel algorithms using different performance metrics on both simulated and real DNA sequence data, where I found that our OpenMP-based algorithm provides at-least 2 times absolute speedup (compared to the best sequential version) and 7 times relative speedup (compared to using only 1 thread).
3. A comparison of the newly developed OpenMP-based LCS algorithm with and without branch conditions.

2.2 Preliminaries

Given two sequence strings $A[1,2,\dots,m]$ and $B[1,2,\dots,n]$, the LCS of the two strings can be found by calculating the LCS of all possible prefix strings of A and B. The LCS of a prefix pair $A[1,2,\dots,i]$ and $B[1,2,\dots,j]$ can be calculated using the previously calculated prefix pairs with the following recurrence relation:

$$R[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ R[i-1,j-1] + 1 & if\ A[i] = B[j] \\ \max(R[i-1,j], R[i,j-1]) & otherwise \end{cases} \qquad (2.1)$$

Here, R is a score table consisting of the lengths of the LCS of all the possible prefixes of the two strings. The LCS of A and B can be found in the cell $R[m,n]$ of table R. While calculating the length of LCS, the case with $A[i] = B[j]$ resembles a *dominant match* which is an essential part in solving the problem [45]. From equation 2.1, we can see that the value of a cell $R[i,j]$ in the scoring table R depends on $R[i-1,j-1]$, $R[i,j-1]$ and $R[i-1,j]$.

Algorithm 1: Classic Algorithm of LCS

Input: Sequence Strings $A[1,2,\dots,m]$ and $B[1,2,\dots,n]$

Output: Length of LCS

1: **for** i=1 to m **do**

2: **for** j=1 to n **do**

3: calculate R[i,j] using equation 2.1

2.2.1 Row-wise Independent Algorithm (version 1)

Yang et al. [46] has devised a row-wise independent parallel algorithm by removing dependency among the cells of the same row. They have modified equation 2.1 so that the value of a cell in a particular row depends only on the previous row. The modified equation is as follows:

$$R[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ R[i-1,j-1] + 1 & if\ A[i] = B[j] \\ \max(R[i-1,j], R[i-1,j-k-1] + 1) & if\ A = B[j-k] \\ \max(R[i-1,j], 0) & if\ j-k = 0 \end{cases} \quad (2.2)$$

Here, k denotes the number of steps required to find either a match such as $A[i] = B[j-k]$ or $j-k = 0$.Yang et al. [46] has divided their algorithm into two steps. First, they calculated the values of $j-k$ for every i and stored these values in another table named P. The equation to calculate the value of P is given below.

$$P[i,j] = \begin{cases} 0 & if\ j = 0 \\ j-1 & if\ B[j-1] = C[i] \\ \mathrm{P}[i,j-1] & otherwise \end{cases} \quad (2.3)$$

Here, C is the string comprised of the unique characters of string A and string B. After that the value of score table R were calculated using the following updated equation.

$$R[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ R[i-1,j-1]+1 & if\ A[i] = B[j] \\ \max(R[i-1,j],0) & if\ P[c,j] = 0 \\ \max(R[i-1,j], R[i-1,P[c,j]-1]+1) & otherwise \end{cases} \quad (2.4)$$

Here, c denotes the index of character A[i-1] in string C.

Algorithm 2: Row-wise independent Algorithm (version 1)

Input: Sequence Strings $A[1,2,\dots,m]$ and $B[1,2,\dots,n]$, unique character string $C[1,2,\dots,l]$

Output: Length of LCS

1: **for** i=1 to l **do**

2: **for** j=1 to n **do**

3: calculate P[i,j] using equation (2.3)

4:

5: **for** i=1 to m **do**

6: **for** j=1 to n **do**

7: calculate R[i,j] using equation (2.4)

2.2.2 Row-wise Independent Algorithm (Version 2)

Looking into equation 2.4, we can see that there are few branching conditions associated with it. As branching can hamper the performance of parallel algorithms, Yang et al. [46] further modified the calculation of P matrix using the following equation to reduce branching conditions.

$$P[i,j] = \begin{cases} 0 & if\ j = 0 \\ j & if\ B[j-1] = C[i] \\ \mathrm{P}[i,j-1] & otherwise \end{cases} \quad (2.5)$$

Then equation 2.4 can be rewritten as follows with one branching condition reduced.

$$R[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ \max(R[i-1,j], 0) & if\ P[c,j] = 0 \\ \max(\mathrm{R[i-1,j]}, \mathrm{R}[i-1, P[c,j]-1]+1) & otherwise \end{cases} \quad (2.6)$$

From the two versions of row-wise independent algorithms, we can see that the calculation of values of table P only depends on the same row. In contrast, the calculation of the values of score table R depends on the previous row only. Upon observing the equations of the row-wise independent algorithms. we can see that the rows of the P table can be calculated in parallel. Furthermore, for a given row of the R score table, all the elements can be calculated in parallel.

Algorithm 3: Row-wise independent Algorithm (version 2)

Input: Sequence Strings $A[1,2,\ldots,m]$ and $B[1,2,\ldots,n]$, unique character string $C[1,2,\ldots,l]$

Output: Length of LCS

1: **for** i=1 to l **do**

2: **for** j=1 to n **do**

3: calculate P[i,j] using equation (2.5)

4:

5: **for** i=1 to m **do**

6: **for** j=1 to n **do**

7: t = the sign bit of (0 - P[c,j])

8: s = the sign bit of (0 – (R[i-1,j] – t . R[i-1, P[c,j] – 1))

9: R[i,j] = R[i-1,j] + t . (s $\oplus$ 1)

2.3 Methodology

I have developed the row-wise independent algorithms using MPI, OpenMP, and hybrid MPI-OpenMP platforms. Here, I will talk about the PCAM (Partition, Communication, Agglomeration, and Mapping) formulation of our solution for distributed memory machines. Then I will provide our solution approaches for all the three platforms subsequently.

2.3.1 PCAM Formulation

The algorithm involves two steps. First, it calculates the table of P. Finally, it calculates the score table R using the values of P, and previous row values of R. Here, I will discuss parallel formulation of the second step involving calculation of the score table R. The first step can also be parallelized in a similar fashion with some minor adjustments. The PCAM formulation of the second step of the implementation is discussed below.

2.3.1.1 Partitioning

Observing equation 2.4, we see that a task can be defined as the calculation of a single cell of the score table R. While calculating a single cell ($R[i,j]$) of table R, the value of $R[i-1,j-1]$, $R[i-1,j]$, and $P[i,j]$ are required. However, I plan to calculate the rows one after another. I will calculate the first row at first, then move on to the second row and so on. In addition, the value of table P will be already calculated in the first step of the algorithm. Hence, one task will consist of calculation of $R[i,j]$, and three values ($R[i-1,j-1], R[i-1,j]$, and $P[i,j]$. Therefore, we can define a 2D task matrix T, where task T[i,j] will calculate the value of R[i,j].

2.3.1.2 Communication

Here, the values of R[i-1,j-1] and R[i-1,j] will be calculated by tasks T[i-1,j-1] and T[i-1,j]. Hence, task T[i,j] will have to communicate with tasks T[i-1,j-1] and T[i-1,j] for calculating the value of R[i,j].

2.3.1.3 Agglomeration & Mapping

Cells of the same row of the score table don't have any dependency between them. Therefore, we can calculate their values in a parallel fashion. If we have p processes and n be the

number of columns in the score table, then we can assign n/p tasks into a single process. We can map the processes with processors in a one-to-one fashion.

2.3.2 MPI-based Approach

For the calculation of the P table, each row can be calculated in a parallel way. Therefore, in our implementation, I have scattered the P table to all the processes in the beginning. After calculating the corresponding chunk values, process number zero gathers the partial results from all the other processes.

For the calculation of score table R, elements in each row can be calculated in parallel. In the beginning, process number zero scatters the values of a row to all the processes. After that process number zero gathers and broadcasts these results. This scatter and gather operations need to be done for every row. Hence, the communication and synchronization overheads are expected to be higher for the MPI implementation approach (code provided in Additional file 1).

Instead of scattering the row values, I have tried broadcasting at first. However, it took extra time due to the increase in communication data size. Therefore, to reduce communication bandwidth I have used scatter. I have also used 16-bit **MPI_SHORT** instead of 32-bit **MPI_INT** to further reduce the communication data size.

2.3.3 OpenMP-based Approach

In the distributed memory implementation, we saw that the number of scatter and gather is too high (equal to the length of sequence string A). A shared memory implementation can mitigate these communication and synchronization overheads. Therefore, I have also developed the algorithm using OpenMP platform.

In case of the OpenMP implementation (provided in Additional file 2) of the score table R, the outer loop can't be calculated in parallel, as every row depends on its' previous rows. Therefore, I have parallelized the inner loop only. Work-sharing construct **#pragma omp parallel for** (an OpenMP directive for sharing iterations of a loop among the available threads) was used for this purpose. In addition, the values of *t* and *s* needs to be private for all the threads. Otherwise, the result will be erroneous. I have tried different scheduling strategies for sharing works among the threads. The comparison among these scheduling strategies will be discussed in the Results section.

The calculation of the P table was also shared among threads. This time, the outer loop was parallelized using **#pragma omp parallel for** construct, as every row is independent of each other.

2.3.4 Hybrid MPI-OpenMP-based Approach

After experimenting with a different number of processes from MPI version, I have selected the optimum number of processes that provides better execution time. Similarly, from the observation of experimental results of OpenMP implementation approach, I have selected the optimum number of the threads. Then I implemented the algorithm using hybrid MPI-OpenMP platform with the selected parameters.

In the MPI implementation, every row was first scattered among processes. Then each process calculated their chunks. In the hybrid implementation, this chunk calculation was further shared among threads using **#pragma omp parallel for**.

2.4 Results & Discussion

2.4.1 Data Sets and Specifications of the Computer

I used two different data sets for the experiments. First one is a simulated DNA sequence data, collected from University of California Riverside's (UCR) random DNA sequence generator [47]. The lengths of the different pairs of sequences are between 128 base pairs to 32,768 base pairs. The second data set consists of 8 virus genome sequence pairs and two entire chromosome genome sequence pairs of two eukaryotes, collected from the website of National Center for Biotechnology Information (NCBI) [48]. The selected sequence lengths vary from 359 base pairs to 32,276 base pairs for the viruses, and from 15,05,371 base pairs to 1,61,99,981 base pairs for the eukaryotes. **Table 2.1** represents the selected virus and eukaryote pairs and their sequence lengths.

Table 2.1: Information of real DNA sequence data set collected from NCBI [48]**.** "bp" stands for the number base pairs.

#	Virus A	Virus B
1	Potato spindle tuber viroid (360 bp)	Tomato apical stunt viroid (359 bp)
2	Rottboellia yellow mottle virus (4194 bp)	Carrot mottle virus (4193 bp)
3	Rehmannia mosaic virus (6395 bp)	Tobacco mosaic virus (6395 bp)
4	Potato virus A (9588 bp)	Soybean mosaic virus N (9585 bp)
5	Chicken megrivirus (9566 bp)	Chicken picornavirus 4 (9564 bp)
6	Microbacterium phage VitulaEligans (17534 bp)	Rhizoctonia cerealis alphaendornavirus 1 (17486 bp)
7	Lucheng Rn rat coronavirus (28763 bp)	Helicobacter phage Pt1918U (28760 bp)
8	Lactococcus phage ASCC368 (32276 bp)	Uncultured Mediterranean phage uvMED (32133 bp)
9	Athene Cunicularia (Chromosome 25, 1505370 bp)	Bombus Terrestris (Chromosome LG B18, 3078061 bp)
10	Athene Cunicularia (Chromosome 25, 1505370 bp)	Bombus Terrestris (Chromosome LG B01, 16199981 bp)

All the experiments were run on University of Manitoba's on-campus cluster computing system (Mercury machine). The cluster consists of four fully connected computing nodes with 2-gigabit ethernet lines between every pair of nodes. Each node consists of two 14-core Intel Xeon E5-2680 v4 2.40GHz CPUs with 128GB of RAM. Having a total of 28 cores inside, with the help of hyper-threading, each node is capable of running twice as many hardware threads (56 threads) at a time.

2.4.2 Comparison among Different Approaches

For the MPI approach, I tuned for the number of processes and found that using 4 process gives better relative speedup. For the OpenMP approach, I tuned for the number of threads and the scheduling strategy (static, dynamic, and guided). I found that using 16 threads and a static scheduling of work sharing among the threads provided 7 times relative speedup (see **Figure 2.1 (a)** and **Figure 2.1 (b)**. Finally, for the hybrid MPI-OpenMP approach, I used 4 processes (or nodes) and 16 threads.

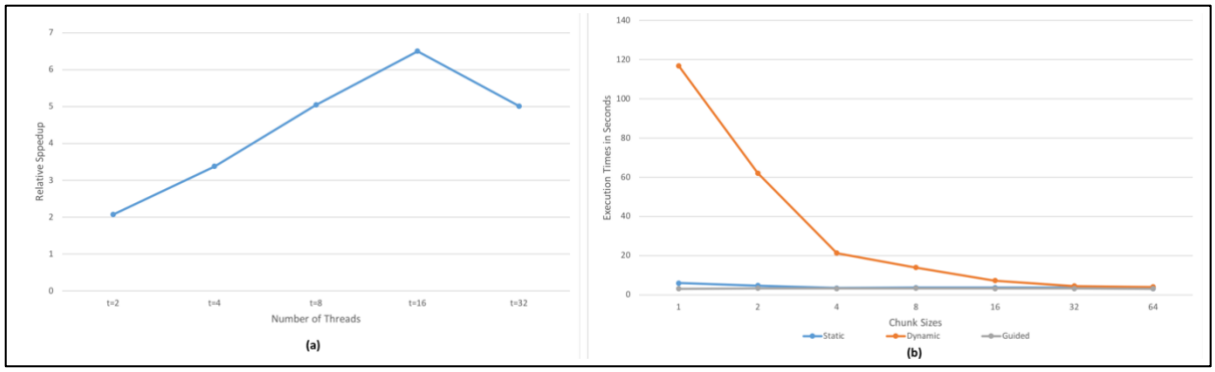

Figure 2.1: Relative speedup and execution times of different scheduling strategies. (a) Relative speedup with different number of threads. (b) Execution times (in seconds) for different scheduling strategies and chunk sizes. Number of threads was 16. Sequence lengths were set to 32768 for both cases.

For the comparison purpose, I experimented with a varying number of sequence lengths. **Figure 2.2(a)**, illustrates the execution times for different implementations where we can see that our OpenMP implementation outperforms all the other approaches and is almost 2 times faster than the best sequential version. However, the MPI approach provides poor results due to the increased amount of communication and synchronization overhead caused by m scatter and gather operations (blocking in nature). The hybrid MPI-OpenMP approach performs the worst. As in the hybrid approach, the number of scatter and gather operations is the same as the MPI approach, and it also adds synchronization overheads of the OpenMP,

this implementation provides the worst result. Therefore, distributed memory implementation is discouraged for the LCS algorithm. In order to validate our results, I also experimented with the real-world DNA sequence data (see **Table 2.1**). From **Figure 2.2 (a)**, we can see that even for the real world data the OpenMP implementation is having at-least 2 times speedup from the best sequential version. For longer DNA sequences (SP 9, SP 10 in **Figure 2.2 (b)**, the OpenMP speedups are even higher, whereas the MPI and the hybrid implementations took more than a week to complete.

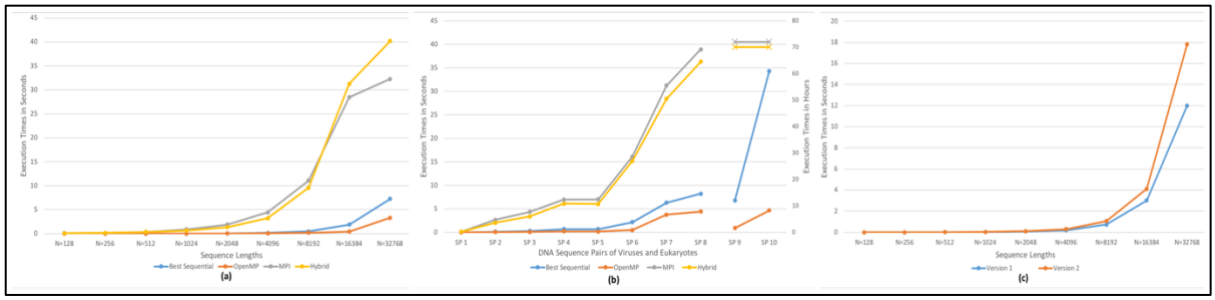

Figure 2.2: Execution times for different implementations and versions of the LCS algorithm with real and simulated data. (a) Execution times for different implementations with varying sequence lengths for the simulated dataset. (b) Execution times for different implementations with different DNA sequences of real world DNA sequence pairs. Here "SP" stands for sequence pairs from **Table** 1. The primary y-axis (execution times in seconds) describes the timing of sequence pairs SP 1 to SP 8, the secondary y-axis (execution times in hours) describes the timings of SP 9 and SP 10. Points marked by cross signs denote that those experiments took more than 7 days to complete. (c) Execution times for different lengths of sequence strings from sequential implementation of the two versions of the row-wise independent algorithm.

2.4.3 Comparison between the Two Versions of the Algorithm in OpenMP Approach

In the above experiments, I used version 2 of the row-wise independent algorithm. In order to compare the execution times of the two versions (version 1 and version 2), I also developed the version 1. **Figure 2.2 (c)** illustrates the execution times for the two versions with varying

sequence sizes and 1 thread only where we can see that version 1 performs relatively better than version 2 of the algorithm. Although version 2 has removed branching conditions, it has added more computations which might be the reason for its relatively bad execution times. Furthermore, CPU architectures are much better at branch predictions than GPUs. Therefore, the second version of the row-wise independent parallel algorithm performed well on GPUs [46] but not on CPUs.

2.5 Availability and Documentation of Tool

The CPU-based parallel LCS tool has been made public and is available in github. **Table 2.2** contains a brief summary of the tool with version requirements, license, and online links.

Table 2.2: Summary of our parallel LCS tool.

Project Name	LCS Row Parallel (CPU)
Project Home Page	https://github.com/RayhanShikder/lcs_parallel/
Operating Systems	Platform Independent
Programming Language	**C**
Other Requirements	gcc 4.8.5 or later, OpenMPI version 1.10.7 or later, OpenMP version 3.1 or later.
License	MIT License
Any restrictions to use by non-academics	None

2.5.1 Requirements of the Installation

The tool is developed on OpenMP. Therefore, a working version of OpenMP is required for the tool to work. The gcc compilers usually come with built-in support for OpenMP. However, for Mac OS you may need to change some configurations to make OpenMP working.

2.5.1.1 Mac OS X

While running the make command of this tool in your Mac, if you face fatal error: 'omp.h' file not found error, that means the version of clang in your PC doesn't support OpenMP. If you do not see any error, you are good to go. If you get the error, you may have to reinstall the **gcc** using **brew reinstall gcc**. Then you have to link to the installed path by modifying your **$PATH** variable. Or you can forcefully create the symbolic links (if asked) while reinstalling gcc. To do so, run **brew link --overwrite gcc**.

Now to use OpenMP you have to specify the proper **gcc** compiler (e.g., gcc, or gcc-7, or gcc-8), which supports OpenMP. In the makefile inside the **Tool** directory, please change the **CC** value (name of the compiler) to the specific name of the compiler (e.g., gcc, or gcc-7, or gcc-8). One can find the details from [49].

2.5.1.2 Linux, Windows

The gcc compilers in **Linux** come with built-in support for OpenMP. So, you do not need to do anything there. For **Windows** you may need to install a suitable version of **gcc** to use OpenMP. You can look at this reference [50].

2.5.2 How to Run?

At first, clone or download this repository. Then go to the **Tool** directory and run **make** command from **Terminal/Command Prompt**. This will create an executable file named **find_lcs**.

2.5.2.1 Prepare the Input

At first get the sequence files (e.g., FASTA) to compare from a convenient source (e.g., [47, 48]). Make sure the files consist of only nucleotide bases (**A**, **T**, **C**, or **G**). Therefore, you may

need to remove the description lines (lines started with a '>'), and any other characters (newline character, whitespaces etc.) from the file.
After collecting the sequence files and removing unnecessary characters from it, create the input text file in the following manner.

1. Put the sequence lengths (as integer value) of the two sequences in two lines.
2. Then put the sequence length of unique characters (4) in the third line.
3. After that, put the sequence strings in three consecutive lines. The first two lines are for the sequences to compare, and the third lines will be for the unique characters (in this case **ATCG**).

Figure 2.3 is an example of a small input file where the first sequence consists of 16 base pairs, and the second sequence consists of 15 base pairs.

Figure 2.3: Snapshot of a sample input file.

2.5.2.2 Execution

Now to get the LCS of two sequences, find the path to your desired data file. Then, run the following command

./find_lcs dumm_input.txt > output.txt.

This will use the **dummy_input.txt** file as your input and write the output in **output.txt** file. Note that this will use the maximum number of threads available in your PC.

2.5.2.3 Understanding the Output

After the completion of the execution, one can find the output of in the **output.txt** file (see **Figure 2.4**). The output consists of three parts. The first part provides a summary of the input sequences. The second part provides the number of threads, LCS length, percentage of match between the two sequences, and the total time taken (in seconds) for the program. Following is the output file by using the **../data/simulated/11.txt** file as input and using 4 threads for parallelization.

```
Your input file: ../data/simulated/11.txt
Length of sequence 1: 131072 bp
Length of sequence 2: 131071 bp

######## Results ########
Number of threads used: 4
Length of the LCS is: 127963
97.63% of the first sequence matches with second one
Total time taken: 112.497263 seconds
```

Figure 2.4: Snapshot of a sample output file.

Please note that, with the increase of sequence lengths and number of threads, the timings of the parallel program improve significantly. A video tutorial on how to use this tool can be found in [51].

2.6 Conclusion & Future Directions

As I found that the version of the row-wise independent algorithm with branching performs better than the other version, I will investigate this version in more detail. My study investigated parallelization of the row-wise independent version of the LCS algorithm only, as it provided ease in parallelization using the MPI, and OpenMP frameworks. However, in future I will also investigate other versions of the algorithm with the goal of finding better parallelization.

Chapter 3: Correlation Analysis, Classification, and Visualization of Disease-based Multi-omics Data

3.1 Introduction

In a real-world scenario, an individual sample or entity can be represented from different viewpoints. For instance, a single object can have multiple representations consisting of images captured from different angles using multiple cameras. Or, a video can be thought as a combination of image frames, and audio signals. The different representations are termed as views and this type of datasets are commonly referred to as multi-view datasets. Analyzing such datasets in an aggregated way reveals more information than treating the individual views separately. This special type of treating multi-view datasets is known as multi-view learning. CCA-based approaches are one of the most popular and effective ways of multi-view learning methods. CCA-based approaches learn a shared representation from multiple views by maximizing the correlation among them. Most of the CCA-based approaches are unsupervised (e.g., regularized canonical correlation analysis (RCC) [68–70], sparse canonical correlation analysis (SCCA) [71–73], etc.). A DNN-based CCA approach, deep canonical correlation analysis (DCCA) [21], has recently received significant attention in the research community for its ability to capture non-linear behaviors and better correlation scores. However, in these unsupervised approaches, the label information (if any) remains unused. Considering the label information in correlation-based analyses have the potential to provide correlated features with more predictive power.

In the field of bioinformatics, multi-view datasets mainly consist of omics data (e.g., genomics, proteomics, microbiomes, etc.), and are known as multi-omics datasets. Multi-omics datasets have some unique properties of themselves and sometimes require special type of considerations while doing analysis. Here, first I have explored and benchmarked the

existing CCA-based approaches for multi-view learning on a special type of multi-omics dataset consisting of gene expression and microbiome views from the patients with related inflammatory bowel diseases (IBD). Then, I focused on a special type of unsupervised CCA approach known as Deep Canonical Correlation Analysis (DCCA) [21] and developed new supervised versions of this approach. I have also proposed a new general framework for multi-view analysis of high-dimensional multi-omics datasets using this new supervised version of the DCCA. Along with the correlation analysis, I have also investigated better ways of classification of the multi-omics datasets and developed new DNN-based classification models. Finally, in order to get more insights, I have visualized the datasets at different steps of the analyses. In section 3.2, I present the benchmarking analyses. In section 3.3, my new approaches for supervised DCCA are presented. Then in sections 3.4, and 3.5, I discuss the DNN-based new classification approaches and related visualization. section 3.6 provides concluding remarks and potential future research directions.

3.2 Review of CCA Approaches

Multi-omics datasets are very high-dimensional and have a relatively fewer number of samples compared to the number of features. CCA-based methods are commonly used for reducing the dimensions of such high-dimensional multi-view (multi-omics) datasets to analyze the associations among the features from different views and to make them suitable for downstream analyses (classification, clustering, etc.). However, most of the CCA approaches suffer from lack of interpretability and result in poor performance in the downstream analyses. Besides, there is no well-explored comparison study for CCA methods with application to multi-omics datasets (especially microbiome and gene expression datasets). In this study, I address this gap by providing a detail comparison study of four popular CCA approaches: regularized canonical correlation analysis (RCC), deep canonical

correlation analysis (DCCA), sparse canonical correlation analysis (SCCA), and supervised SCCA using a multi-omics dataset consisting of microbiome and gene expression profiles of IBD. I evaluated the methods in terms of the total correlation score, and the classification performance. From the experimental results, I found that the SCCA method provides reasonable correlation scores in the reduced space, enables interpretability, and also provides the best classification performance among the four methods.

3.2.1 Introduction

Multi-omics data (or multi-view) consists of data from various omics sources (e.g., genomics, transcriptomics, proteomics, etc.) but measured on the same individuals. While analyzing a single omics data at a time may provide associations of the omics features (e.g., genes, genetic variants, etc.) with the disease of interest, it has several limitations. The identified features, in most of the cases, are only capable of explaining a small amount of the heritable component of the disease [74]. Furthermore, complex diseases generally involve a multitude of factors originating from multiple omics sources and environments and the interactions among the factors as well. Integrated study of the omics data has the potential to reveal more information about the diseases as it may tell us about the individual associations, interactions among the factors and the flow of information from the cause of diseases to consequences [74, 75].

Most of the omics datasets are very high dimensional. For instance, microbiome datasets generally consist of a few hundreds to thousands of operational taxonomic units. Likewise, genomics datasets comprise of thousands of genes. In these cases, combining the omics datasets usually results in an exclusive representation with a very large number of features (e.g., tens of thousands). However the number of available samples is relatively very small (e.g., hundreds). These bigger number of features cause the curse of dimensionality problem in the data analysis step [37]. Again, this large number of features also create

challenges in applying most of the mathematical and statistical methods [38]. Moreover, a large subset of these features may represent redundant and irrelevant information. Therefore, before learning any objective functions, the feature sets need to be reduced to lower-dimensional subspace (may consist of either the original features or their projections).

Most often, researchers want to investigate the relationships between two or more omics datasets of interest. CCA-based approaches, which find the linear combinations of features from two datasets and try to maximize the correlation between them, are common ways to find such relationships [14]. Besides, canonical correlations also reduce the dimensionality of the original high-dimensional omics datasets making it suitable for fusion [76] and downstream predictive analysis [36]. The original version of the canonical correlation analysis has been extended in several ways to make it suitable for different types of applications. In a setting, where the numbers of features are larger than the number of samples, the basic version of the CCA is not effective. To deal with this situation, regularized versions of the canonical correlation analysis (regularized canonical correlation analysis or RCCA) have been developed [68–70]. To incorporate non-linear combinations of the features while calculating the correlations, kernel canonical correlation analysis (KCCA) methods have been developed too [17, 18, 77]. Recently, a deep neural network-based parametric non-linear version of the CCA (named as deep canonical correlation analysis (DCCA)) has been proposed whose mapping function is not restricted to reproducing kernel Hilbert space as KCCA [21, 40]. In the case of high-dimensional datasets, interpreting the results from the above-mentioned methods is difficult and in some cases unachievable. However, in biological applications, along with finding correlations between datasets, researchers seek to trace the original features that correspond to the resulting correlations. Therefore, to deal with this issue, sparse versions of the canonical correlation analysis (SCCA) methods have been developed [71, 73, 78]. Supervised version of the SCCA has also been developed which

considers the class label information while calculating the correlated features [79]. Along with the above methods, tensor canonical correlation analysis [20], Bayesian canonical correlation analysis [19], and some autoencoder based methods [22, 40] also exist. However, there exists no study which highlighted the comparison of the approaches with application to multi-omics datasets especially datasets consisting of microbiome and gene expression profiles.

Here I perform a detailed comparison of the canonical correlation methods (RCC, SCCA, supervised SCCA, and DCCA) in terms of applications with a multi-omics dataset consisting of microbiome and gene expression profiles. To the best of our knowledge, this study is the first to investigate the CCA approaches for microbiome and gene expression data together. I will perform the comparison based on the total correlation scores between the omics datasets, and performance of learning methods using the learned representations from different CCA methods. Here, first I will provide the fundamentals of the respective canonical correlation analysis methods. Then, I will provide the details of the experiments and discuss the results. Finally, I will discuss several pros and cons of the methods.

3.2.2 Preliminaries

Originally proposed by [14], the CCA methods have been modified in various ways for different application domains. In this section, I will discuss the fundamentals of the original version of the CCA and its four variants: regularized canonical correlation analysis (RCC), deep canonical correlation analysis (DCCA), sparse canonical correlation analysis, and supervised sparse canonical correlation analysis (SCCA (S)).

3.2.2.1 Canonical Correlation Analysis (CCA)

Having two datasets X_1 and X_2 with $(n \times p_1)$ and $(n \times p_2)$ dimensions measured on the same subject $i = 1,2 \ldots, n$, CCA finds linear combinations of the features from the two datasets which are maximally correlated [14]. The objective of the CCA method is to maximize the following:

$$corr({w_1}^T X_1, {w_2}^T X_2) \ or, \left(\frac{{w_1}^T \Sigma_{12} w_2}{\sqrt{{w_1}^T \Sigma_{11} w_1 {w_2}^T \Sigma_{22} w_2}} \right) \quad (3.1)$$

CCA finds the linear projections ${w_1}^T X_1$ and ${w_2}^T X_2$ which have a maximum correlation between them, where w_1 and w_2 are the canonical coefficients. Here, Σ_{11} and Σ_{22} are the covariances of X_1 and X_2, and Σ_{12} is the cross-covariance between the features of the datasets.

3.2.2.2 Regularized Canonical Correlation (RCC) Analysis

When the number of features (p_1 or p_2) become larger than the total number of samples (n), the basic version of the CCA doesn't work as the first n canonical variates possess larger values while the rest of the canonical covariates become zero [52]. To deal with this, regularization parameters ($\lambda_1$and λ_2) can be added with the covariance matrices in the following manner (I_{p_1} and I_{p_2} are identity matrices) [53, 54].

$${\Sigma_{11}}' = \Sigma_{11} + \lambda_1 I_{p_1} \quad (3.2)$$

$${\Sigma_{22}}' = \Sigma_{22} + \lambda_2 I_{p_2} \quad (3.3)$$

Here, $\lambda_1$and $\lambda_2$are the regularization parameters, and I_{p_1} and I_{p_2} are identity matrices. The covariance matrices Σ_{11}, and Σ_{22} of equation 3.1 can be replaced with Σ_{11}', and Σ_{22}' for the regularized version.

3.2.2.3 Deep Canonical Correlation Analysis (DCCA)

While CCA and RCC look for linear combinations of the features, deep canonical correlation analysis (DCCA) searches for complex nonlinear projections of the input features which are maximally correlated [21]. DCCA is a DNN-based approach, where two densely connected networks (Network 1 and Network 2 in **Figure 3.1**) are separately trained on two views of the datasets. These two networks learn nonlinear feature combinations and use a correlation maximization objective function to adjust the weight values.

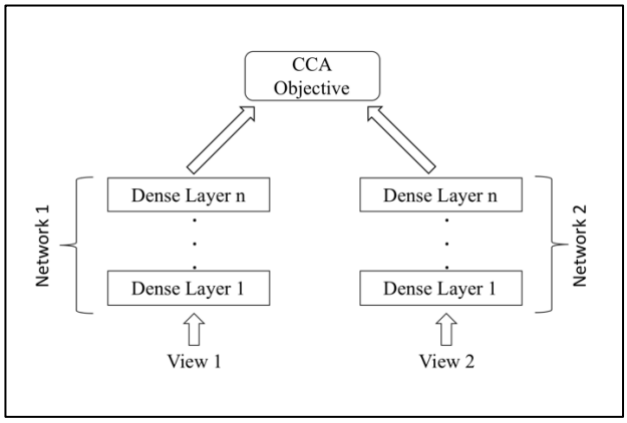

Figure 3.1: Schematic diagram of the DCCA method.

Let's consider Network 1 and Network 2 as f and g and the learned features from these two networks as *f(X)* and *g(Y)*. Then the CCA objective function takes the following form:

$$\max \mathrm{tr}\,(U^T f(X) g(Y)^T V) \qquad (3.4)$$

$$such\ that \begin{cases} U^T f(X) f(X)^T U = I \\ V^T g(Y) g(Y)^T V = I \end{cases}$$

Here, *tr* denotes the trace function which calculates the trace of a given matrix. *U*, *V* are the projection matrices for projecting the output of *f(X)* and *g(Y)*.

3.2.2.4 Sparse Canonical Correlation Analysis (SCCA)

For datasets with a large number of features, the interpretation of linear combinations becomes impracticable. Hence, considering a sparse subset of the features is a viable approach [55, 56]. In this case, the objective function to be maximized takes the following form:

$$corr({w_1}^T X_1, {w_2}^T X_2)$$

$$where \left|\left|w_1\right|\right|^2 \leq 1, \left|\left|w_2\right|\right|^2 \leq 1, \boldsymbol{P}_1(w_1) \leq c_1, and\ \ \boldsymbol{P}_2(w_2) \leq c_2 \quad (3.5)$$

Here, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are called penalty functions or sparse CCA criterion. These penalty functions are chosen in a way to provide sparse feature combinations and also to make the CCA deal with situations where the feature sets are large compared to the number of samples. $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ can be lasso or fused lasso penalty functions. The parameters c_1, c_2 are used to control the level of penalization.

3.2.2.5 Supervised Sparse Canonical Correlation Analysis (SCCA (S))

In the above approaches, the class label information remains unused. If the class label information can be used while calculating the learned correlated features they can provide discriminative powers in downstream analyses (e.g., classification, clustering). Inspired by this rationale, a supervised version of the SCCA method was developed [57]. This method modifies equation 3.5 in the following way:

$$corr({w_1}^T X_1, {w_2}^T X_2)\ where\ \left|\left|w_1\right|\right|^2 \leq 1, \left|\left|w_2\right|\right|^2 \leq 1,$$

$$\boldsymbol{P}_1(w_1) \leq c_1, \boldsymbol{P}_2(w_2) \leq c_2, w_{1j} = 0 \forall j \notin Q_1, and\ w_{2j} = 0 \forall j \notin Q_2 \qquad (3.6)$$

Here, Q_1, Q_2 denote the set of features in X_1, and X_2, which are largely correlated with the class labels.

3.2.3 Experiments and Results

3.2.3.1 Dataset

I considered a multi-omics dataset consisting of two views: gene expression and microbiome profile [58]. All the patients in this cohort had undergone ileal-pouch-anal anastomosis (IPAA) surgery and were recruited from Mount Sinai Hospital, Toronto, Canada. The cohort represents a wide range of variation in terms of clinical and molecular data. The gene expression dataset consists of 184 patients and there are 20,253 gene expression features (representing the level of expression for 20,253 genes) available for each patient. These gene expression data were generated using microarray technology and are continuous. The microbiome dataset represents the microbiome community abundance from the same cohort. This data-set consists of 7,000 microbiome features (representing the count of 7,000 bacteria groups or OTUs) for every individual. These microbiome data were generated using 16s rRNA technology and have zero-inflated discrete count values. There are four pouchitis sub-types recorded in the dataset. These disease sub-types are Familial Adenomatous Polyposis (FAP), No Pouchitis (NP), Acute Pouchitis (AP), and Crohn's Disease-Like Inflammation (CDL). Among the 184 patients, 63 are classified as AP, 28 are classified as CDL, 35 are classified as FAP, and the remaining 58 of the patients are classified as NP. Along with the above information, the dataset also contains some meta information including biopsy location (pouch or pre-pouch ileum), inflammation score (0 to 11). None of the patients were taking any antibiotic at the time of the biopsy. **Table 3.1** shows a summary of the dataset.

Table 3.1: Summary of the dataset.

Type	Count
Patient	184
Genes	20,253
Operational Taxonomic Units	7,000
Classes	4

3.2.3.2 Preprocessing and Hyperparameter Tuning

At first, all the zero and constant valued features were removed from the dataset. The remaining dataset contained 20,251 features in the gene expression view and 5,443 features in the microbiome view. Then the datasets were normalized to have zero mean and unit variance. The 184 samples were randomly divided into train (147) and test (37) groups in a stratified manner.

For the RCC analysis, we need to find the values of λ_1 and λ_2 . To do so, I searched them in a 5×5 grid where λ_1 was in one axis and λ_2 was in another. The value of λ_1 and λ_2 were varied from 0.1 to 0.9. The pair of λ_1 and λ_2 which provided the best train correlation (total correlation of 119.32) was 0.1 and 0.1. This pair was selected for finding the final canonical projections for both the train and test data. I used the R package: CCA for the RCC analysis [80].

For the DCCA, I have used the python implementation from [82] which was based on the original DCCA article [21]. RMSProp optimizer and sigmoid activation function were used in the model. However, for our purpose, I tuned the hyperparameters (learning rate, number of layers, regularization value, batch size, etc.). For the DCCA, I split the train data (147) into train (110) and validation (37) groups. Learning rate of 10KL provided the best validation correlation. As our dataset consists of very few numbers of samples, a shallow

network with only one hidden layer (1024 units) performed better than deeper networks (with two or more hidden layers). For the regularization step, I varied the values from 10^(-1) to 10^(-11) and found 10^(-9) to provide the best validation correlation. Finally, as suggested by [40], I have set a larger value (100) for the mini batch size as it enabled more information for estimating the covariances accurately.

For the SCCA, I have used the PMA package in R [83]. The CCA method in the PMA package uses a lasso penalty when the features of the datasets are unordered. In the case of ordered features, a fused lasso penalty is used. As the features of our dataset are unordered, I have used the lasso penalty. The levels of penalization were set using the $penaltyX$, and $penaltyZ$ parameters whose value should be in the range (0, 1) . To find the optimal values of $penaltyX$, and $penaltyZ$ I have searched in a 10 × 10 grid and found $penaltyX$ = 0.8 and $penaltyZ$ = 0.8 provide the best train correlation (total correlation of 87.9) when the output dimension (K) was set to 100. The PMA package also provides a supervised version of the sparse canonical correlation analysis, where the output labels are used to ensure that the learned feature projections are also correlated with the output labels. I will call this version supervised SCCA or SCCA (S). After tuning for SCCA (S), $penaltyX$ = 0.8 and $penaltyZ$ = 0.9 provided the best training correlations which I selected for the subsequent analyses.

3.2.3.3 Total Correlation Scores

After tuning the hyperparameters to their appropriate values, I have performed the canonical correlation analyses (RCC, SCCA, SCCA (S), DCCA). I have conducted the experiments for different number of output dimensions (10, 20, 30, 40, 50) and learned the canonical coefficients ($w_1$and w_2) for each of the output dimensions for every method. After learning the coefficients, I have multiplied it with the original dataset (train and test) to generate the

projections. I have calculated total correlations (using the *linear_cca* method provided in [59]) from the projections of the test data to evaluate different CCA methods. The results are illustrated in **Figure 3.2**.

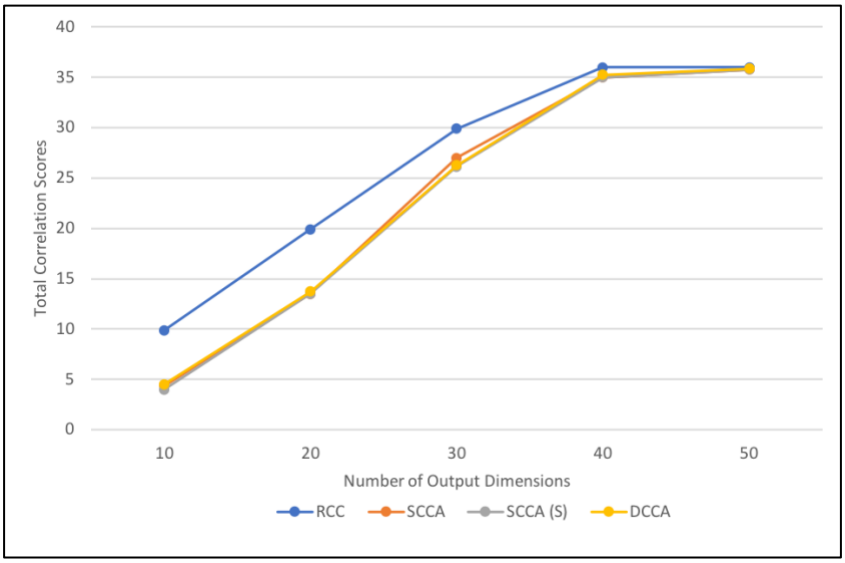

Figure 3.2: Total correlation scores for different canonical correlation approaches. The x-axis represents the number of output dimensions and the y-axis represents corresponding total correlation scores.

From the above figure (Figure 3.2), we can see that RCC provides better correlation scores. On the other hand, SCCA, SCCA(S), and DCCA provide almost similar correlation scores. However, with the increasing number of output dimensions (when the output dimension surpasses the number of test samples), the correlation scores become almost the same for all of the approaches. For the SCCA methods, the sparsity nature may correspond to the compromise in the total correlation score. As DNN-based approaches are always data-hungry, the fewer number of samples is the main reason behind the relatively lower correlation scores of DCCA method.

3.2.3.4 Classification Performance

CCA is often used to reduce the dimensionality of high-dimensional datasets to make them suitable for downstream analyses (classification, clustering, etc.). To evaluate the classification performances of different CCA methods, I have performed binary classifications using the projected data from the CCA methods. The original dataset contains four disease classes (FAP, NP, AP, and CDL) which I converted into binary by taking the NP (No Pouchitis) class in one group and all the other classes in another group. I have used the support vector machine (SVM) method for the classification. I have tried several kernel functions (linear, radial basis function, polynomial, and sigmoid) for the SVM method and adjusted the hyperparameters (C, sigma, gamma, degree, etc.) accordingly. For the evaluation, I have used accuracy and area under the receiver operating characteristics (ROC) curve (AUC) metrics. Table 3.2 illustrates the results.

Table 3.2: Binary class classification results using SVM on the output projections from different CCA methods. Evaluation metrics are accuracy and area under the ROC curve (AUC).

Dimensions	Metrics	RCC	SCCA	SCCA(S)	DCCA
10	Accuracy	67.56%	72.97%	72.97%	67.56%
	AUC	0.5	0.6	0.6	0.5
20	Accuracy	67.56%	75.67%	75.67%	67.56%
	AUC	0.5	0.71	0.67	0.5
30	Accuracy	70.27%	**75.67%**	**75.67%**	67.56%
	AUC	0.54	**0.756**	**0.67**	0.5
40	Accuracy	70.27%	**78.37%**	**78.37%**	67.56%
	AUC	0.54	**0.69**	**0.69**	0.5
50	Accuracy	67.56%	70.27%	70.27%	67.56%
	AUC	0.5	0.63	0.6	0.5

From **Table 3.2**, we can see that DCCA provides the worst classification performance both in terms of accuracy and AUC value for all the output dimensions. The nonlinear nature of the DCCA method and the smaller size of the dataset may be the reason behind this performance loss. The RCC method's performance is also poor which is easily observed with the low AUC values. Although multi-omics datasets consist of very high-dimensional features, only a handful of these features are responsible for a particular phenotype. Therefore, incorporating all the input features for finding the projections may be responsible for the poor classification performance of RCC. Finally, it is visible that the SCCA methods (SCCA and SCCA (S)) provide relatively better classification performances than the other two methods. The sparse nature of these methods is the main reason behind this performance. However, it is surprising that the supervised version of the SCCA didn't provide any better results than the unsupervised version.

3.2.4 Conclusion

In this study, I found that SCCA provides interpretable correlation scores and better performance in downstream analysis while projecting high-dimensional multi-omics datasets in a low-dimensional space. The regularized version of the canonical correlation analysis (RCC), although provides good correlation scores, lacks interpretability and provides poor classification performance. On the other hand, the DCCA provides moderate correlation scores but lacks interpretability and suffers from poor performance in classification. Therefore, it is advised to not use DCCA with high-dimensional multi-omics datasets having fewer number of samples.

3.3 Classification of Multi-omics Data Using Supervised Deep Canonical Correlation Analysis

3.3.1 Introduction

CCA-based approaches are the widely used multi-view learning methods. The basic versions of the CCA (e.g., CCA, Regularized CCA, etc.) consider linear combinations of the input features for correlated projections, hence failing to capture the non-linear characteristics of the multi-view datasets. In the previous section, we saw that the sparse version of the CCA is better suitable for the analysis of high-dimensional multi-omics datasets. Besides, the deep version of the CCA (DCCA) is capable of extracting non-linear feature combinations. All these methods are unsupervised. However, in a real world scenario, many of the multi-omics datasets have the sample labels or disease information. Therefore, using this type of unsupervised approaches for multi-view learning keeps the label information unexploited.

To leverage the class label information, various supervised versions of the canonical correlation analysis (CCA) method have been developed. Generalized version of the CCA (GCCA) is one of the first such methods which learns maximally correlated features and makes sure that these learned features minimize within class scatters as well [60]. Supervised sparse canonical correlation analysis is another one which modified the original sparse canonical correlation analysis (SCCA) and is specifically designed for high-dimensional multi-omics datasets [57]. Some other notable approaches are the group sparse canonical correlation analysis (GSCCA), supervised multi-view canonical correlation analysis ensemble (SMVCCAE) etc. [61, 62]. Recently, a supervised version of the deep canonical correlation analysis (SDCCA) method have also been proposed which modified the original DCCA objective function by incorporating class labels [63]. However, the implementation is not tested on very high-dimensional dataset like us.

Here, in this thesis, I propose a novel framework for multi-view analysis of the multi-omics datasets. The framework considers sparse feature extraction from high-dimensional datasets and learns correlated features from different views in a supervised manner. For the correlation step I propose two new versions of the supervised deep canonical correlation analysis (DCCA S1, and DCCA S2). Whereas the supervised DCCA method modified the original DCCA objective function, I combined the cross entropy loss [64] with the DCCA objective function for the proposed two new models. Besides, in these new models the effect of these two losses (CCA loss and cross entropy loss) can be configured to have an overall customized behavior.

3.3.2 Methodology

Based on the fact that sparse methods work better for very high-dimensional datasets, our proposed framework includes a sparse feature selection step. **Figure 3.3** illustrates the overall process of the proposed framework, which includes three major steps: Step 1 – Lasso-based feature selection for each view of the data; Step 2 – supervised DCCA to select correlated features and Step 3 – supervised DNN for the classification using the learned features For the Steps 2 and 3, new computational approaches are proposed. Lasso-based methods assign zero weights to features with less importance. The number of features can be varied by varying the regularization parameter (e.g., C parameter in logistic regression method in sklearn software [65]).

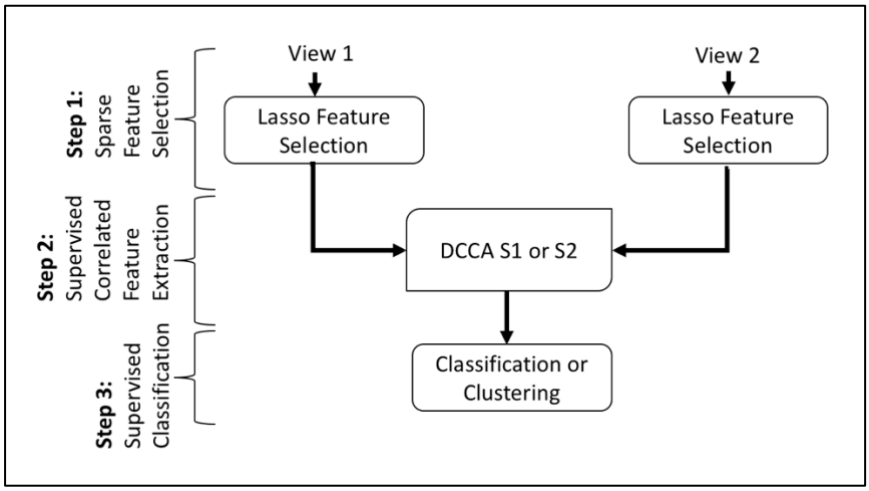

Figure 3.3: Proposed supervised deep learning framework for multi-view learning.

Followed by the feature selection step, our framework includes the correlated feature extraction step. This step extracts the shared correlated features with possibly smaller dimension which are representative of the original data as well as discriminative in terms of the class label (e.g., disease phenotype). I have proposed two new models for this step, which I will describe in the following subsections. The final step involves the downstream analyses including classification or clustering on the extracted features.

From **Table 3.2**, we can observe that the learned features from DCCA provide no discriminative advantage (e.g., AUC of 0.5). Hence, I incorporated the class label information to make the extracted features more discriminative in nature. To do so, I have passed the learned features to a new network off fully connected layers. This network at the tail is optimized based on the cross entropy loss which maximizes the classification accuracy. However, due to the backpropagation of the whole network, this loss affects all the layers of the model. Hence, the first two networks and the merged layer's weights are updated based on both the CCA loss and the cross entropy loss. I call this model version 1 of the supervised DCCA or DCCA S1 in short. **Figure 3.4** depicts the schematic diagram of this model.

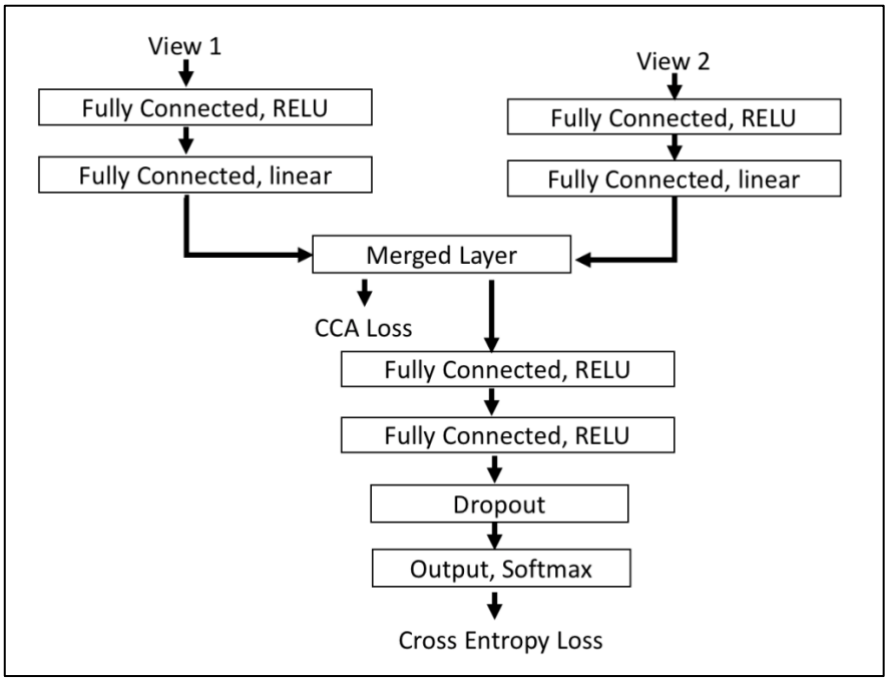

Figure 3.4: Schematic diagram of the version 1 of the supervised DCCA model (DCCA S1).

In the DCCA S1, the output of the merged layer was fed to the fully connected network of the tail. It is worthy of mentioning that the network up to the merged layer is identical to the original DCCA model [21]. Hence, the output of this merged layer consists of the learned correlated features from the original views. In the second version of the supervised DCCA model (DCCA S2), I made a short circuit connection of the output of the first two networks with the next fully connected layers at the tail. This was done based on the thought that only feeding the correlated features could compromise the discriminative power of the model. This new model is illustrated in **Figure 3.5**.

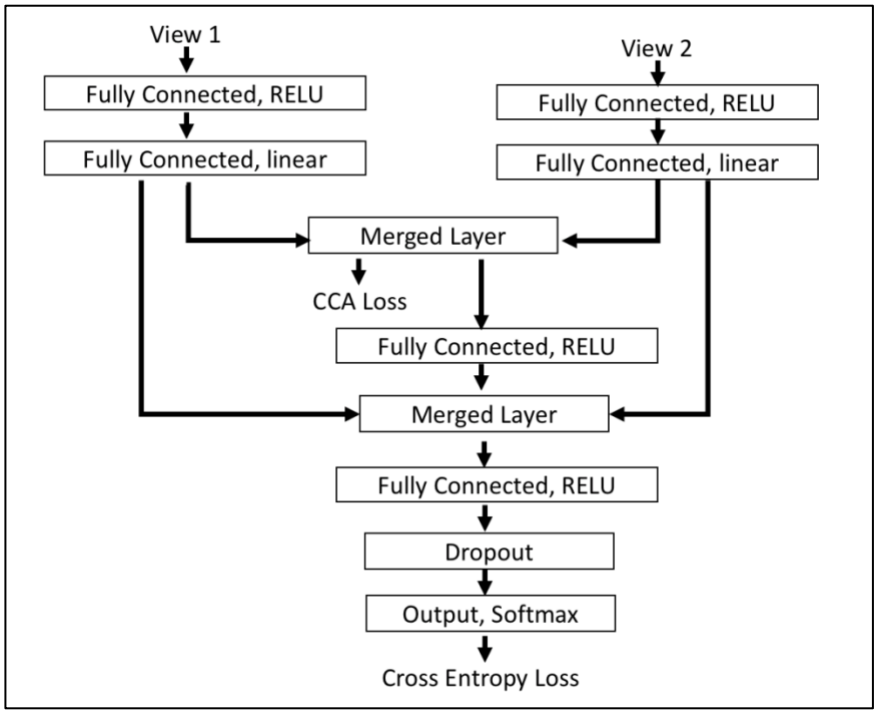

Figure 3.5: Schematic Diagram version 2 of the supervised DCCA model (DCCA S2).

3.3.3 Results and Discussion

Correlation-based multi-view learning approaches aim to learn shared representations of multiple views. This representation should also be capable of retaining the original class separability information. Hence, I have evaluated the newly proposed methods based on total correlation scores, and classification performances. In this study, I have used the same dataset as used in section 3.2. Therefore, I will only give a very short description of the dataset here. The dataset is based on gene expression and microbiome profiles of 184 patients with 4 pouchitis (a special disease of the ileal pouch) states (Familial Adenomatous Polyposis (FAP), No Pouchitis (NP), Acute Pouchitis (AP), and Crohn's Disease-Like Inflammation (CDL)) [81]. The gene expression view of the dataset has 20,253 genes, and the microbiome view has 5,443 operational taxonomic units (OUT). The gene expression data are mainly continuous values and represent the level of expression for a given gene in the corresponding sample. The microbiome data are mostly sparse and represent the abundance of the OTUs in

a sample. In all the analyses, I have considered both multi-class and binary class supervised cases. In the binary case, I have kept the NP (no pouchitis) class in one group and other types in another group. This was done to divide the data into a case (others) and a control (NP) group.

Table 3.3 shows the number of features selected by the Lasso model. The number of features from this feature selection was varied by varying the regularization parameter of the lasso method. A smaller value of the regularization adds more regularization (selects fewer features) than larger ones. For feature selection in the multi-class case, multinomial loss function was used in the Lasso model. The new models were built using "keras" framework with "theano" backend [90].

Table 3.3: Different number of selected features from Lasso models. Regularization values were varied from 0.5 to 3.0. View 1 is microbiome data, View 2 is gene expression data.

#	Regularization	Multi-class			Binary		
		View 1	View 2	Both	View 1	View 2	Both
f1	0.5	10	93	39	9	145	82
f2	0.8	39	292	240	57	432	436
f3	1.0	97	667	427	97	667	718
f4	1.5	138	789	849	205	1269	1431
f5	2.0	221	1150	1270	342	1872	2183
f6	3.0	406	1843	2059	599	3107	3619
f7	N/A	All	All	All	All	All	All

3.3.3.1 Correlation Scores

Here, I have compared the two new models (DCCA S1 and DCCA S2) with the original DCCA model using the features selected by the Lasso models. I compared the total

correlation scores for multi-view and binary cases. For both these cases, the number of selected features is varied. The total correlation scores were computed using the output from the merged layer (see **Figure 3.4** and **Figure 3.5**) of the new models and the final layers of the DCCA model (see **Figure 3.1**). The output dimensions of the models were set to 50 for all the cases.

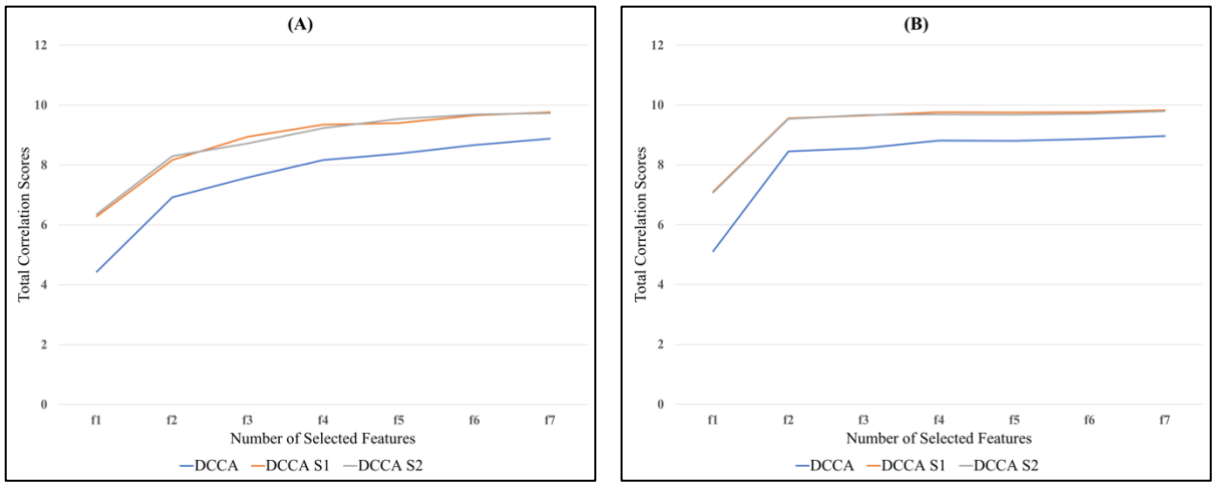

Figure 3.6: Total correlation scores from supervised version of the DCCA. **f1** to **f7** correspond to different number of selected features from Lasso. (A) Multi-class supervised analyses (B) Binary class supervised analyses.

Figure 3.6 illustrates the results from these correlation analyses. From this figure, we can see that for both multi-class and binary class cases, DCCA S1 and DCCA S2 are providing better total correlation scores than the original DCCA model. Initially, the total correlation scores are relatively smaller due to the small number of selected features (see f1 and f2 in **Table 3.3**) from the lasso model. In the supervised models (DCCA S1, and DCCA S2) the within class scatters are minimized. This could be the reason behind these supervised models with better correlation scores than the original unsupervised DCCA model. Likewise, it is also noticeable that the total correlation scores for the binary class are marginally greater than the multi-class case.

3.3.3.2 Classification Performances

After the correlation score analysis, I did the classification analysis using the learned correlated features. SVM was used to do the classification. Kernel, gamma value, degree, regularization hyperparameters were tuned appropriately. Classification performance was also measured from the softmax layer of the supervised models (DCCA S1, and DCCA S2). 10 fold cross-validation was done for all the cases.

Accuracy, and area under the ROC (receiver operating characteristics) curve (AUC) measures were used for evaluation purpose. For multi-class classification, two types of AUC measures can be taken. One is macro averaging and another is micro averaging technique. The macro averaging technique considers each class equally. It calculates AUCs for each class in a one versus all manner and then averages them for the final AUC. On the other hand, micro averaging aggregates all the results together and calculates a single metric. It is capable of dealing with class imbalance. Therefore, in this study, I have used micro averaging of the AUC value for the multi-class evaluation.

In Table 3.4, I present the results from the multi-class analyses. The SVM results from the DCCA, DCCA S1, and DCCA S2 are almost similar and none of them are good enough (with the best AUC 0.62 and accuracy of 35.09%. The classification performances of the softmax layer of the DCCA S1 and DCCA S2 is slightly better than the others. Here, the best AUC is around 0.68 and accuracy of 40.56%.

Table 3.4: Multi-class classification performance comparisons. Here, f1 to f7 represent the features from view 1 and view 2 of the first column (multi-class) of **Table 3.3**.

Features	Metrics	DCCA + SVM	DCCA S1 + SVM	DCCA S2 + SVM	DCCA S1	DCCA S2
f1	Accuracy	30.6%	33.46%	32.17%	35.52%	37%
	AUC	0.65	0.62	0.6	0.66	0.64
f2	Accuracy	35.48%	31.13%	27.55%	41.47%	34.28%
	AUC	0.58	0.61	0.61	0.65	0.62
f3	Accuracy	33.03%	33.44%	34.1%	37.5%	39%
	AUC	0.58	0.6	0.58	0.64	0.63
f4	Accuracy	34.4%	31.2%	31.1%	33.63%	37.67%
	AUC	0.5	0.58	0.61	0.59	0.63
f5	Accuracy	35.09%	30%	35%	36.7%	39.2%
	AUC	0.53	0.57	0.63	0.65	0.63
f6	Accuracy	34.85%	31.04%	34.31%	40.56%	41.5%
	AUC	0.58	0.58	0.61	0.68	0.63
f7	Accuracy	32.06%	34.23%	27.87%	61.82%	39.46%
	AUC	0.53	0.59	0.59	0.35	0.63

From the multi-class analyses, the performances were not up to the mark. I then grouped the 4 class into 2 groups. One group had the NP class (control) and all the other classes were in another group (case). Similar to the multi-class case, SVM and softmax layer of the supervised models were used for classification. In this case, we can observe that the SVM AUC and accuracies are almost similar for all the DCCA models (**Table 3.5**). However, the softmax layers AUC and accuracy is much better compared to the unsupervised DCCA model with SVM classification. One possible explanation of this behavior is that the

correlated features may have a very complex non-linear class separability which is learned by the fully connected network at the tail of the supervised models.

Table 3.5: Binary class classification performance comparisons. Here, f1 to f7 represent the features from view 1 and view 2 of the second column (Binary) of **Table 3.3**.

Features	Metrics	DCCA + SVM	DCCA S1 + SVM	DCCA S2 + SVM	DCCA S1	DCCA S2
f1	Accuracy	68.5%	61.4%	64.75%	67.45%	67.42%
	AUC	0.71	0.5	0.5	0.65	0.67
f2	Accuracy	69%	59.75%	61.95%	70%	63.18%
	AUC	0.62	0.5	0.56	0.63	0.58
f3	Accuracy	68.5%	58.63%	58.63%	64.23%	67.98%
	AUC	0.59	0.5	0.5	0.65	0.66
f4	Accuracy	68.5%	62.9%	59.75%	66.95%	70.61%
	AUC	0.59	0.53	0.51	0.69	0.71
f5	Accuracy	68.5%	60.3%	58.63%	74.47%	74.07%
	AUC	0.65	0.5	0.51	0.77	0.74
f6	Accuracy	68.5%	58.11%	58.6%	73.91%	72.2%
	AUC	0.53	0.51	0.51	0.73	0.71
f7	Accuracy	68.5%	57.9%	58.63%	64.73%	60.24%
	AUC	0.47	0.5	0.53	0.52	0.5

3.3.4 Conclusion

In this study, I propose a simple framework for multi-view learning of high-dimensional multi-omics data. I also propose two new supervised Deep Canonical Correlation Analysis (DCCA) models (DCCA S1, and DCCA S2). From the experimental results, we can observe a significant gain in the total correlation scores in the new supervised models than the

original unsupervised DCCA model. However, the classification performances didn't provide any noteworthy improvements. In future, I want to test the new models on larger data. I will look for modifying the models to improve the classification performance as well.

3.4 Classification of Multi-omics Data Using Classic Machine Learning and Deep Neural Network Models

After doing the correlation-based multi-view learning experiments, I focused on studying classical and DNN-based classification approaches. At first, I have conducted the baseline classification experiments on the pouchitis dataset (see section 3.1). I ran classical methods: SVM, Random Forest (RF) on the cleaned dataset to perform classifications. I also developed two DNN-based models for integrating the two views of the data for classification purpose. As the dataset is high-dimensional, most of the features may represent unnecessary and redundant information. Therefore, feature selection was done before the classification steps.

3.4.1 Methodology

Classification of high-dimensional datasets is always a hard problem due to the irrelevance and redundancy of the features. It becomes even harder when we have multiple high-dimensional views of the data with different number of dimensions and statistical properties. In this part of my study, I experimented with different feature selection strategies and combined them with classification step. I used RF and lasso based feature selection techniques. After the feature selection step, I ran two classical classification algorithms: SVM, and RF.

DNN is capable of dealing with high-dimensional data. Further, DNNs can capture the non-linear structure of datasets easily. This inspired me to develop DNN-based integration models for classification of the Pouchitis dataset. **Figure 3.7** illustrates the

structure of this model (I named this model as DNN Y model). The two views of the data have varying number of dimensions (one has around 20k features, another has around 5k features) and different statistical properties (one is continuous-valued, and another is discrete and sparse). That's why the gene expression and the microbiome views are fed into two different networks with different structures (different number of hidden units) and concatenated in a hidden layer which is followed by another fully connected network. The dropout layers were added for regularization purpose. Before the concatenation, the activation layers are linear, whereas after the concatenation rectified linear unit (RELU) activation was used. The final output layer was a Softmax activation layer. In this model, all the hyper-parameters (e.g., learning rate, loss functions, optimizers, initializer, batch size, number of epochs, etc.) were appropriately tuned.

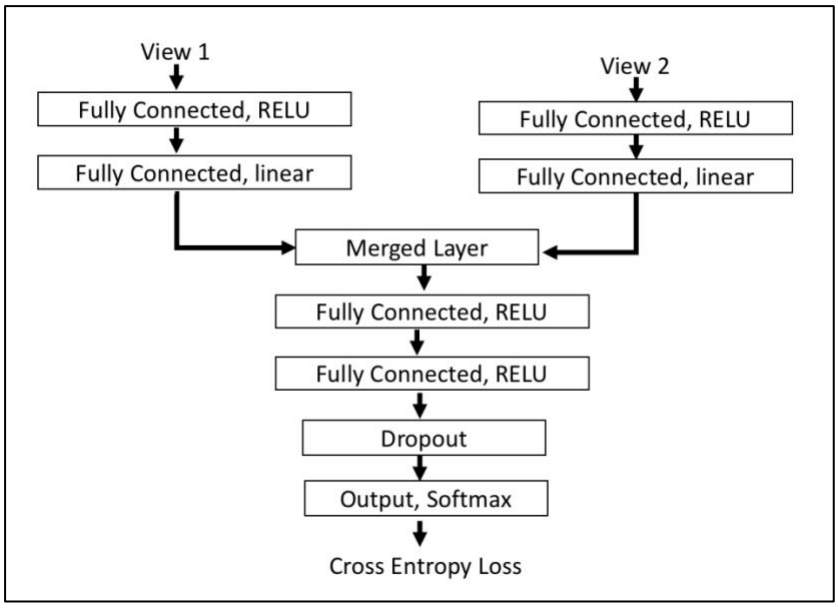

Figure 3.7: Integration of the two views of data using a deep neural network-based architecture. This version of the DNN model is called as DNN Y Model.

I have also built a simple DNN model by concatenating the input features from both the views and feeding it to a fully connected network. Dropout was used to avoid overfitting

on the train data. I named this model as the DNN flat model (see **Figure 3.8** for the schematic diagram).

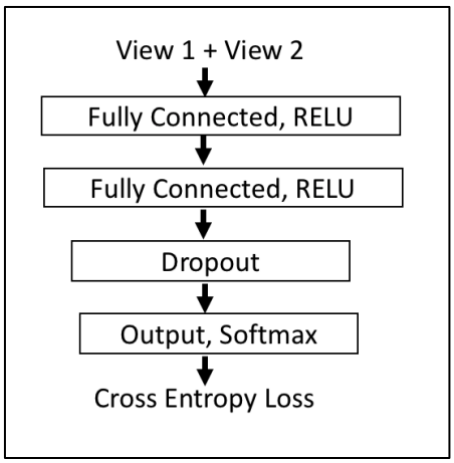

Figure 3.8: DNN flat model

3.4.2 Results and Discussion

I have done both multi-class and binary class classifications using the classical and the newly developed DNN-based models. At first, I investigated differences between treating the individual views separately and treating them together. Then I moved on to the performance evaluation of the DNN-based multi-view models.

3.4.2.1 Single View Approach vs. Multi-view Approach

I used individual views (gene expression, and microbiome views) separately as well as together by concatenating them in a straightforward manner. I used the SVM and RF algorithms from python module scikit-learn [66]. While using SVM, I tried several kernels including linear, polynomial (with degree 2 and 3), radial basis function (rbf). However, in most of the cases, linear and polynomial kernel provided better results. In case of RF algorithm, I have set the number of trees to 50 (*n_estimators = 50*), and the minimum samples split to 30 (*min_samples_split = 30*).

Along with using all the features from the cleaned data, I have also done supervised feature selection using the *feature_importances_* attribute of the RF algorithm. I have selected the top 20, 50, 100, 500, and 1000 features to build the training models.

Table 3.6 illustrates the overall accuracies of multi-class classification using the SVM algorithm. From the table, I can see that the best result is around 61.93% when both the genes and OTUs were used together with the top 500 features.

Table 3.6: Overall accuracies of multi-class classifications using SVM. Different rows represent accuracies using different number of features. There are four classes: NP, CDL, AP, FAP. The bold cell of the table represents the best result.

Number of Features	Accuracy (Genes)	Accuracy (OTUs)	Accuracy (Both Genes and OTUs)
All	48.43%	47.54%	46.03%
20	43.28%	45.02%	43.64%
50	44.99%	43.83%	49.0%
100	48.22%	43.9%	52.31%
500	53.93%	53.08%	**61.93%**
1000	51.97%	48.65%	55.73%

In **Table 3.7**, the accuracies of multi-class classification from the RF algorithm is shown. Again, in this case, we can see that the best result (55.83%) is achieved when both the genes and OTUs were used together. However, RF worked best with only 20 selected features. As RF cannot perform well when the features have dependencies among them, with increased number of features (which may increase the dependencies among the features) RF's classification performance degrades.

Table 3.7: Overall accuracies of multi-class classifications using RF. Different rows represent accuracies using different number of features. There are four classes: NP, CDL, AP, FAP. The bold cell of the table represents the best result.

Number of Features	Accuracy (Genes)	Accuracy (OTUs)	Accuracy (Both Genes and OTUs)
All	42.02%	41.31%	43.36%
20	50.94%	51.24%	**55.83%**
50	49.84%	50.48%	52.65%
100	50.83%	48.24%	52.23%
500	50.2%	47.49%	49%
1000	48.81%	44.82%	47.31%

Table 3.8: Binary classification (NP vs. all others) using SVM. Different rows represent accuracies using different number of features selected from RF importance score.

Number of Features	Accuracy (Genes)		Accuracy (OTUs)		Accuracy (Both Genes and OTUs)	
	ACC	AUC	ACC	AUC	ACC	AUC
20	72.15%	0.73	73.39%	0.73	69.15%	0.68
50	72.31%	0.77	73.48%	0.65	72.8%	0.72
100	71.78%	0.75	74.01%	0.72	70.87%	0.69
500	78.15%	0.81	72.89%	0.69	70.17%	0.75
1000	73.39%	0.78	71.93%	0.6	**78.3%**	**0.79**
All	68.5%	0.5	70.29%	0.6	68.5%	0.5

Table 3.8, and **Table 3.9** represent the binary class classification results using SVM and RF. In this case, NP (No Pouchitis) was kept in one class, and all the other clases (AP, FAP, CDL) were in another. Here, we can also see the same scenario that using both genes and OTUs information together provides better classification performances both in terms of

accuracy (78.3% from SVM, and 75.56% from RF) and AUC (0.79 from SVM, and 0.86 from RF).

Table 3.9: Binary classification (NP vs. all others) using RF. Different rows represent accuracies using different number of features selected from RF importance score.

Number of Features	**Accuracy (Genes)**		**Accuracy (OTUs)**		**Accuracy (Both Genes and OTUs)**	
	ACC	AUC	ACC	AUC	ACC	AUC
20	74.47%	0.83	73.48%	0.73	74.47%	0.83
50	69.61%	0.81	71.17%	0.72	**75.56%**	**0.86**
100	72.3%	0.82	71.8%	0.76	73.9%	0.86
500	72.27%	0.82	68.65%	0.69	72.2%	0.85
1000	69.58%	0.82	66.52%	0.66	70%	0.82
All	66.89%	0.63	65.31%	0.56	65.9%	0.65

I have also run the DNN Y model using all the features from the two views. The best test accuracy of multi-class classification from this model is 50%. For binary class case (NP in one class and all other in another class), the model provided 67.85% accuracy and 0.65 AUC. From all the above results, we can conclude that the multi-view approach of learning provides better learning outcomes than comparing the views individually.

3.4.2.2 Evaluation of the DNN-based multi-view models

Inspired by the advantages of using the multi-view approaches, I have built two new DNN-based multi-view classification models. From section 3.2 we have learned that sparse approaches works better for high-dimensional omics datasets. Therefore, I have used a sparse lasso based feature selection step before feeding the data in the DNN models. The number of selected features are termed as f1, f2, …, f7 (see **Table 3.3**). For DNN flat model, SVM, and

RF, the features from the both views were concatenated in the beginning, whereas for the DNN Y model, the features of the two views were fed into the two input networks. The performances were evaluated using accuracy and area under the ROC curve (AUC). For multi-class case, micro averaging of the AUC was done for evaluation. 10 fold cross validation (in a stratified manner) was used for all the cases.

From **Table 3.10**, we can see the classification results of the multi-class analyses. We can see that, the DNN Flat model, SVM, and RF provides the best results with relatively small number of selected features (f1 and f2). It's also noticeable that if we use all the features, all the models performances degrade.

Table 3.10: Performance of multi-class classifications. Here, f1 to f7 represent the features from view 1 and view 2 of the first column (multi-class) of **Table 3.3** for DNN Y model. For all the other models they represent the features from both views of the first column (multi-class) of **Table 3.3 3.3**.

Features	Metrics	SVM	RF	DNN Y	DNN Flat
f1	Accuracy	43.68%	**48.51%**	34.24%	41.4%
	AUC	0.73	**0.75**	0.65	0.74
f2	Accuracy	**45.8%**	41.39%	39.09%	**45.98%**
	AUC	**0.77**	0.72	0.63	**0.76**
f3	Accuracy	44.08%	38.75%	29.87%	39.6%
	AUC	0.78	0.7	0.58	0.74
f4	Accuracy	42.6%	37.92%	37.06%	40.3%
	AUC	0.77	0.68	0.64	0.72
f5	Accuracy	44.38%	34.36%	**40.52%**	42.78%
	AUC	0.76	0.67	**0.67**	0.71
f6	Accuracy	42.04%	32.2%	39%	41.36%
	AUC	0.76	0.66	0.67	0.7
f7	Accuracy	34.31%	30.45%	62.73%	33.83%
	AUC	0.59	0.62	0.39	0.59

Finally, I also performed the binary classification analyses (NP vs. all others) using all the developed models. In this case, we can observe (see **Table 3.11**) that the accuracy and AUC of the DNN flat model is very good and mostly consistent regardless of the number of selected features. The performance of the DNN Y model is also good in most of the cases.

Table 3.11: Performance of binary classification results. Here, f1 to f7 represent the features from view 1 and view 2 of the second column (Binary) of **Table 3.3** for DNN Y model. For all the other models they represent the features from both views of the second column (Binary) of **Table 3.3**.

Features	Metrics	SVM	RF	DNN Y	DNN Flat
f1	Accuracy	68.5%	**68.47%**	65.9%	80.38%
	AUC	0.83	**0.77**	0.62	0.84
f2	Accuracy	68.5%	69.5%	67.54%	**87.37%**
	AUC	0.9	0.73	0.71	**0.9**
f3	Accuracy	68.5%	68.5%	68.59%	85.73%
	AUC	0.9	0.7	0.74	0.89
f4	Accuracy	70.73%	68.5%	79.97%	84.25%
	AUC	0.9	0.68	0.85	0.89
f5	Accuracy	73.95%	68.5%	76.7%	83.14%
	AUC	0.89	0.66	0.83	0.88
f6	Accuracy	**79.33%**	68.56%	**77.69%**	80.94%
	AUC	**0.9**	0.62	**0.88**	0.85
f7	Accuracy	65.9%	66.9%	62.96%	66.86%
	AUC	0.55	0.42	0.58	0.47

3.4.3 Conclusion

In this part of my thesis, I investigated the differences between single view and multi-view learning approaches. I have found that the multi-view approaches have more advantages in terms of classification performances (accuracy and AUC). I have also developed two new DNN-based models for mulit-view classification. From the experimental results, I have observed that the new models provide better or comparable results than the classical approaches (SVM, RF).

3.5 Visualization of the multi-view omics data

Visual exploration of data is an integral part of any bioinformatics and machine learning study. This step provides valuable insights of the data and guides the researchers to the right directions. Depending on the insights from particular visualizations, one may take important decisions on the downstream analyses (classification, clustering etc.). However, presenting data at a suitable level of detail without bombarding the user with too much complexity has always been a challenge [67]. Dimension reduction techniques with an output dimension of two or three are generally used for this visualization task. There are many dimension reduction methods, and visualization results may differ based on the type of dimension reduction methods one uses. Therefore, to get unique and meaningful insights from the visualization of the data, selecting the appropriate methods for reducing the dimension is essential [68].

In my thesis, I have performed visual exploration of the data to get an idea about the underlying distributions at different steps of the analyses. Principal component analysis (PCA) is a widely used approach for dimensionality reduction and visualization. However, PCA can not capture the non-linear behavior of the data. Recently, t-stochastic neighbor embedding (t-SNE), a deep neural network based visualization approach has gained attractions in the research community for its ability to perform dimensionality reduction and capture non-linear behaviors [69]. In my thesis, I have used t-SNE for visualizing the pouchitis data. For all the visualizations, I have considered both multi-class and binary class (NP vs. all others) cases.

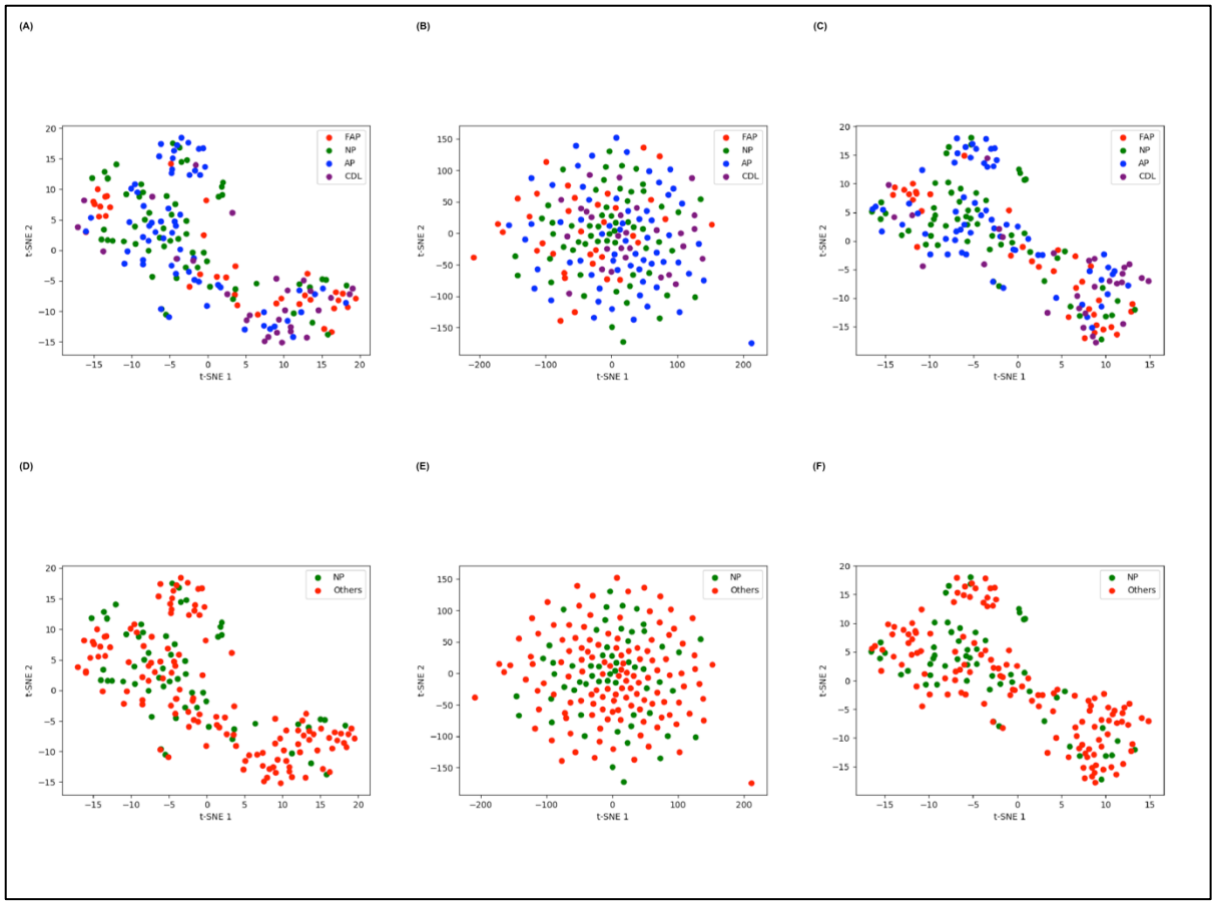

Figure 3.9: t-SNE visualization using all the features from both microbiome and gene expression views. (A) gene expression view multi-class; (B) microbiome view multi-class; (C) both views multi-class; (D) gene expression view binary class (NP vs. others); (E) microbiome view binary class; (F) both views binary class.

At first, I performed 2D t-SNE visualization on all the features from both the views (see **Figure 3.9**). The perplexity of the t-SNE method was tuned for better visualizations. The data points were colored based on the class labels. From this visualization we can see that there is no observable patterns in the data in terms of class labels (both multi-class and binary).

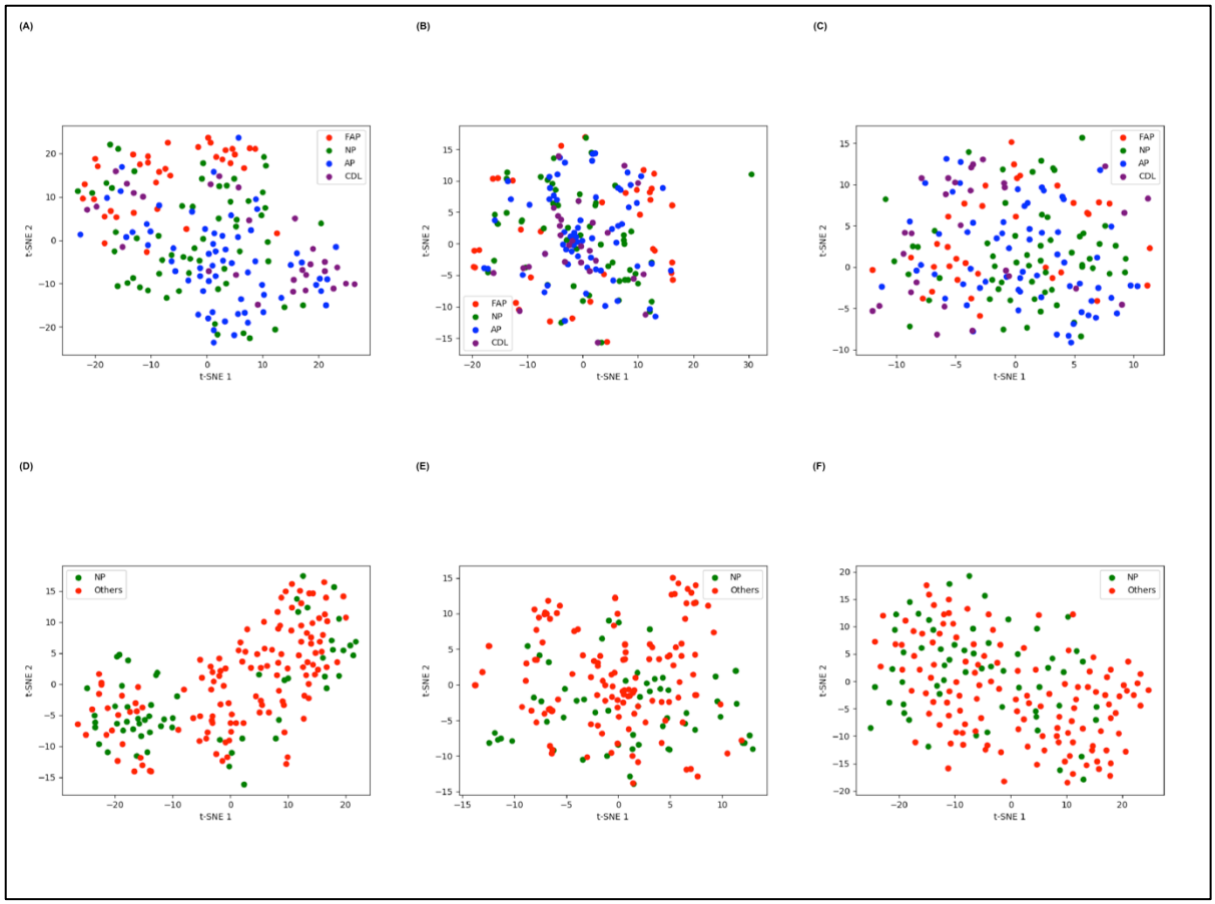

Figure 3.10: t-SNE visualizations using f2 (see Table 3.3) features selected by lasso. (A) gene expression view multi-class; (B) microbiome view multi-class; (C) both views multi-class; (D) gene expression view binary class (NP vs. others); (E) microbiome view binary class; (F) both views binary class.

From **Table 3.10** and **Table 3.11**we can see that with **f2** selected features (240 features for multi-class, and 436 features for binary class), almost all the methods provide good classification performance. That's why I performed t-SNE visualization using these selected features to understand the reason behind this improved performance (**Figure 3.10**). The visualization of the gene expression data for multi-class case reveals that the FAP class clusters separately from all the others. However, for all the other cases no noticeable pattern can be found.

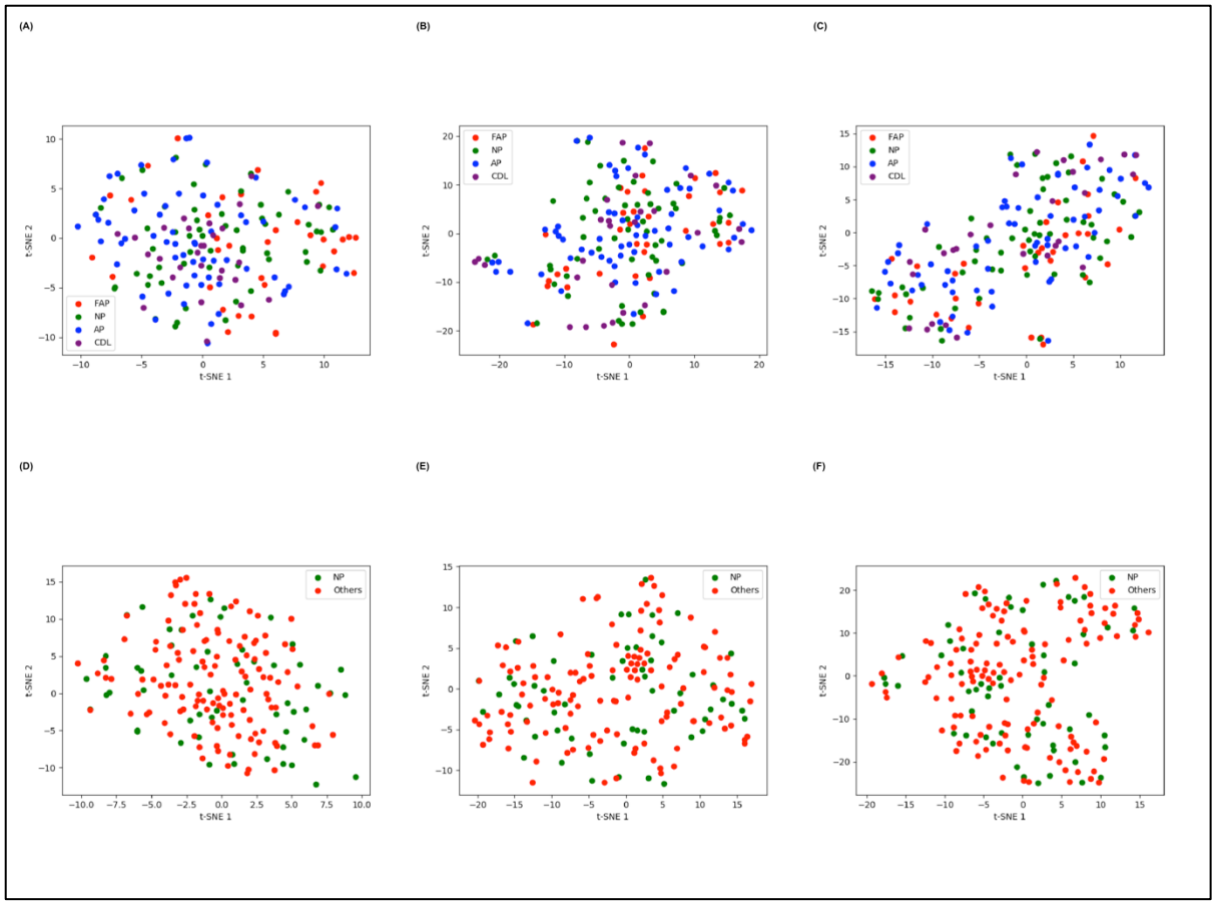

Figure 3.11: t-SNE visualizations using correlated features from DCCA, DCCA S1, and DCCA S2. (A) correlated features of DCCA (multi-class); (B) correlated features from DCCA S1 (multi-class); (C) correlated features from DCCA S2 (multi-class); (D) correlated features of DCCA (binary class); (E) correlated features from DCCA S1 (binary class); (F) correlated features from DCCA S2 (binary class).

In section 3.3, I studied new models of deep canonical correlation analysis. One of our goals was to extract correlated features which will be capable of discriminating the disease subtypes as well. In this part of the thesis, I took the learned correlated features from the supervised new DCCA models (DCCA S1, DCCA S2), and from the original unsupervised DCCA model and visualized them using t-SNE (see **Figure 3.11**). From the visualization we can observe the similar behavior where there data points from all the classes are almost uniformly mixed with each other.

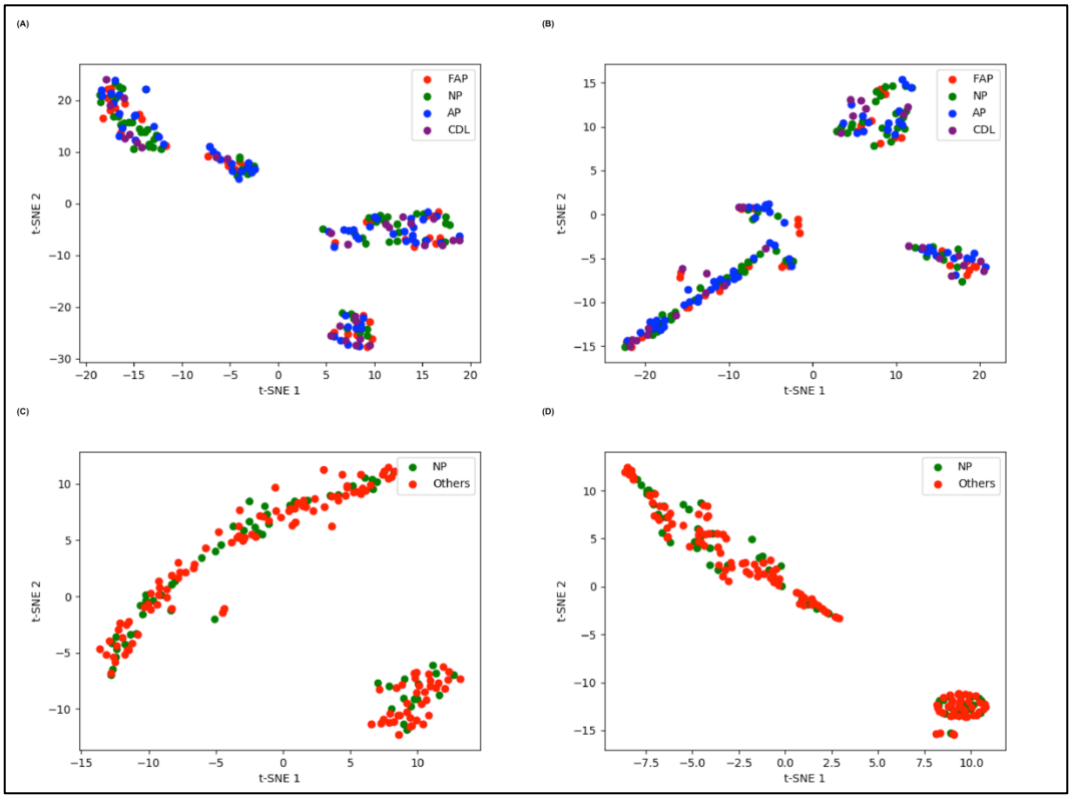

Figure 3.12: t-SNE visualizations of the features of DCCA S1 and DCCA S2 from the last fully connected layer. (A) output from final fully connected layer of DCCA S1 (multi-class); (B) output from the final fully connected layer of DCCA S2 (multi-class); (C) output from the final fully connected layer of DCCA S1 (binary class); (D) output from the final fully connected layer of DCCA S2 (binary class).

In section 3.3 we observed that the performance of the final output layer of the supervised versions of the DCCA models is better than using a SVM over the learned correlated features. Hence, I visualized the final dense layer of the DCCA S1, and DCCA S2 models to understand the reason behind this. **Figure 3.12** shows these visualizations. Although some clusters are visible in the figures, there is no grouping based on the class labels.

3.6 Conclusion and Future Directions

In this part of the thesis, I investigated a few correlation-based multi-view learning methods for multi-omics datasets. I found that the sparse versions of the CCA methods perform better than others in terms of both correlation scores and classification performance. Two new supervised versions of the deep canonical correlation analysis (DCCA S1, and DCCA S2) have been developed along with a general framework for the overall workflow. The new models are found to outperform the original DCCA method in terms of total correlation scores. I have also developed new DNN-based classification models for classifying high-dimensional multi-omics data. Finally, explorative visualization of different steps was done for getting a clearer insight into the underlying mechanisms. This study answered some important questions in the field of multi-omics data analyses. First, existing CCA methods are not better suited for high-dimensional multi-view learning. Second, new supervised DCCA methods have improved correlation scores. Third, new DNN-based multi-view classification models have state-of-the-art or better classification performances. However, several important questions remained unanswered and several new questions were also raised from this research. For instance, although the new supervised DCCA models provided better total correlation scores, they didn't offer better classification performance. Also, it is yet to be tested whether the methods presented in this paper are readily transferrable to all the other types of multi-omics data (e.g., proteomics, transcriptomics, metabolomics, etc.). The future directions of this research will mainly focus on these issues.

Chapter 4: Conclusion, Limitations, and Future Directions

This thesis focused on studying novel CPU-based time-efficient ways of finding LCS of DNA sequence data in the field of bioinformatics. Here, I also performed a thorough investigation of multi-view learning techniques for disease-based multi-omics data. The multi-view learning techniques included correlation-based integration methods, classification algorithms, and visualizations. Although I found a few promising findings and developed novel techniques, there exist several limitations which need to be addressed in the future works.

In the first part of this thesis, I developed CPU-based parallel algorithms for finding LCS using MPI, OpenMP, and hybrid MPI-OpenMP frameworks. I found that the OpenMP based parallel algorithm provides 2 times absolute speedup than the sequential version of the LCS algorithm. This version also provides a 7 times relative speedup. This version of LCS is a special one where each row of the scoring table of the LCS was made independent for the sake of parallel calculation. However, there exist some other versions of the LCS algorithm such as anti-diagonal, bit-wise parallel algorithm, etc. which I didn't investigate in this study. Therefore, I intend to study other versions of the LCS algorithm with a view to finding even better parallel algorithms. Besides, in this thesis, I have only focused on CPU-based techniques. In the future, I also plan to study GPU-based parallel algorithms as they are capable of offering highly parallel solutions.

In the second part of this thesis, my focus was on studying multi-view learning techniques for disease based multi-omics data. Here, I extended an existing DNN-based multi-view learning method to new supervised versions (supervised deep canonical correlation analysis (DCCA S1, and DCCA S2)). The experimental results show that the new methods are capable of offering more correlated features than the original deep canonical correlation analysis (DCCA) method. However, the classification performance of the new

methods remains almost the same as the original DCCA method. In the future, I plan to investigate the reason behind this issue in detail and will try to design models with improved classification performance. We know that the deep learning models are generally data hungry. Working with a larger data or augmented data could have provided a clearer picture of the advantage of this type of methods. Besides, this thesis only focused on correlation-based multi-view learning techniques. In the future, I will also study other methods (e.g., autoencoder-based, co-training-based, etc.) of multi-view learning. In this part of the thesis, I also developed new DNN-based models for classifying the disease-based multi-omics data. The new models provided better or similar classification performances than the classical SVM and RF. Finally, t-SNE visualizations were carried out to have a better understanding of the process. While the existing methods are good at projecting the high-dimensional data into lower dimensions (e.g., 2D or 3D), I felt that they lack the flexibility to explore in the data. I also felt the need for an interactive visualization tool to explore the canonical correlations of such high-dimensional multi-omics data. Developing such interactive tools of visualization could be a promising field of research and may significantly benefit the community.

References

1. Manzoni C, Kia DA, Vandrovcova J, Hardy J, Wood NW, Lewis PA, et al. Genome, transcriptome and proteome: The rise of omics data and their integration in biomedical sciences. Brief Bioinform. 2018.

2. Collins FS, Lander ES, Rogers J, Waterson RH. Finishing the euchromatic sequence of the human genome. Nature. 2004.

3. Venter JC, Smith HO, Adams MD. The sequence of the human genome. Clin Chem. 2015;61:1207–8.

4. Calle ML. Statistical analysis of metagenomics data. Genomics Inform. 2019;17.

5. Moore GE. Cramming More Components Onto Integrated Circuits, Electronics, April 19, 1965. Electronics. 1965.

6. Gottlieb A, Almasi G. Highly parallel computing. Benjamin/Cummings Redwood City, CA; 1989.

7. Zhao J, Xie X, Xu X, Sun S. Multi-view learning overview: Recent progress and new challenges. Inf Fusion. 2017.

8. Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98. 1998.

9. Nigam K, Ghani R. Analyzing the effectiveness and applicability of co-training. In: Proceedings of the ninth international conference on Information and knowledge management - CIKM '00. 2000.

10. Ion M, Minton S, Knoblock CA. Active + Semi-Supervised Learning = Robust Multi-View Learning. In: Proceedings of the Nineteenth International Conference on Machine

Learning. 2002.

11. Muslea I, Minton S, Knoblock CA. Active learning with multiple views. J Artif Intell Res. 2006.

12. Yu S, Krishnapuram B, Rosales R, Rao R. Bayesian co-training. J Mach Learn Res. 2011.

13. Yang P, Gao W. Information-theoretic multi-view domain adaptation: A theoretical and empirical study. J Artif Intell Res. 2014.

14. Hotelling H. Relations Between Two Sets of Variates. Biometrika. 1936.

15. Kettenring JR. Canonical analysis of several sets of variables. Biometrika. 1971.

16. Chaudhuri K, Kakade SM, Livescu K, Sridharan K. Multi-view clustering via canonical correlation analysis. In: Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. 2009.

17. Lai PL, Fyfe C. Kernel and nonlinear canonical correlation analysis. Int J Neural Syst. 2000;10:365–77.

18. Akaho S. A kernel method for canonical correlation analysis. arXiv Prepr cs/0609071. 2006.

19. Klami A, Seppo J Virtanen, Kaski S. Bayesian Canonical Correlation Analysis. J Mach Learn Res. 2013.

20. Luo Y, Tao D, Ramamohanarao K, Xu C, Wen Y. Tensor canonical correlation analysis for multi-view dimension reduction. In: 2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016. 2016.

21. Andrew G, Arora R, Bilmes JA, Livescu K. Deep Canonical Correlation Analysis. Icml. 2013.

22. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY. Multimodal Deep Learning. In: Proceedings of the 28th Annual International Conference on Machine Learning (ICML'11). 2011.

23. Kim T, Kittler J, Cipolla R. Learning Discriminative Canonical Correlations for Object Recognition with Image Sets. Eccv. 2006.

24. Diethe T, Hardoon DR, Shawe-Taylor J. Constructing nonlinear discriminants from multiple data views. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2010.

25. Sun T, Chen S, Yang J, Shi P. A novel method of combined feature extraction for recognition. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. 2008. p. 1043–8.

26. Diethe T, Hardoon DR, Shawe-taylor J. Multiview Fisher Discriminant Analysis. In: NIPS workshop on learning from multiple sources. 2008.

27. Sharma A, Kumar A, Daume H, Jacobs DW. Generalized Multiview Analysis: A discriminative latent space. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2012.

28. Farquhar J, Hardoon D, Meng H, Shawe-taylor JS, Szedmak S. Two view learning: SVM-2K, theory and practice. Adv Neural Inf Process Syst. 2005.

29. Xie X, Sun S. Multi-view twin support vector machines. Intell Data Anal. 2015;19:701–12.

30. Sun S, Chao G. Multi-view maximum entropy discrimination. In: IJCAI International Joint Conference on Artificial Intelligence. 2013.

31. Chao G, Sun S. Alternative Multiview Maximum Entropy Discrimination. IEEE Trans Neural Networks Learn Syst. 2016.

32. Chao G, Sun S. Consensus and complementarity based maximum entropy discrimination for multi-view classification. Inf Sci (Ny). 2016.

33. Long B, Yu PS, Zhang Z (Mark). A General Model for Multiple View Unsupervised Learning. In: Proceedings of the 2008 SIAM International Conference on Data Mining. 2008.

34. Xu C, Tao D, Xu C. A survey on multi-view learning. arXiv Prepr arXiv13045634. 2013.

35. Sun S. A survey of multi-view machine learning. Neural Computing and Applications. 2013.

36. Kakade S, Foster D. Multi-view regression via canonical correlation analysis. Conf Learn Theory. 2007.

37. Verleysen M, François D. The curse of dimensionality in data mining and time series prediction. In: International Work-Conference on Artificial Neural Networks. 2005. p. 758–70.

38. Fodor IK. A survey of dimension reduction techniques. Cent Appl Sci Comput Lawrence Livermore Natl Lab. 2002;9:1–18.

39. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science (80-). 2006.

40. Wang W, Arora R, Livescu K, Bilmes J. On deep multi-view representation learning. In: International Conference on Machine Learning. 2015. p. 1083–92.

41. Phan AH, Cichocki A. Tensor decompositions for feature extraction and classification of high dimensional datasets. Nonlinear Theory Its Appl IEICE. 2010.

42. Rabanser S, Shchur O, Günnemann S. Introduction to Tensor Decompositions and their Applications in Machine Learning. arXiv Prepr arXiv171110781. 2017.

43. Young VB. The role of the microbiome in human health and disease: An introduction for clinicians. BMJ (Online). 2017.

44. Cho I, Blaser MJ. The human microbiome: at the interface of health and disease. Nat Rev Genet. 2012.

45. Bergroth L, Hakonen H, Raita T. A survey of longest common subsequence algorithms. In: String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on. 2000. p. 39–48.

46. Yang J, Xu Y, Shang Y. An efficient parallel algorithm for longest common subsequence problem on gpus. In: Proceedings of the World Congress on Engineering. 2010. p. 499–504.

47. Random DNA Sequence Generator. http://www.faculty.ucr.edu/~mmaduro/random.htm. Accessed 2 Apr 2018.

48. National Center for Biotechnology Information (NCBI). https://www.ncbi.nlm.nih.gov/. Accessed 20 Sep 2018.

49. Installing OpenMP on Mac OS X 10.11. Stack Overflow. https://stackoverflow.com/questions/35134681/installing-openmp-on-mac-os-x-10-11. Accessed 30 Oct 2018.

50. Getting started with openMP. install on windows. Stack Overflow. https://stackoverflow.com/questions/11079586/getting-started-with-openmp-install-on-windows. Accessed 15 Dec 2018.

51. Shikder R. Tutorial on How to Use an OpenMP-based LCS Tool. Youtube. https://www.youtube.com/watch?v=2CBsNiu0i1w&feature=youtu.be.

52. Gonzalez I, Déjean S, Martin P, Baccini A. **CCA** : An *R* Package to Extend Canonical Correlation Analysis. J Stat Softw. 2008.

53. Vinod HD. Canonical ridge and econometrics of joint production. J Econom. 1976.

54. Leurgans, S. E., R. A. Moyeed and BWS. Canonical Correlation Analysis when the Data are Curves. J R Stat Soc Ser B. 1993.

55. Parkhomenko E, Tritchler D, Beyene J. Sparse canonical correlation analysis with application to genomic data integration. Stat Appl Genet Mol Biol. 2009.

56. Witten DM, Tibshirani R, Hastie T. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics. 2009.

57. Witten DM, Tibshirani RJ. Extensions of sparse canonical correlation analysis with applications to genomic data. Stat Appl Genet Mol Biol. 2009.

58. Morgan XC, Kabakchiev B, Waldron L, Tyler AD, Tickle TL, Milgrom R, et al. Associations between host gene expression, the mucosal microbiome, and clinical outcome in the pelvic pouch of patients with inflammatory bowel disease. Genome Biol. 2015.

59. Noroozi V. VahidooX/DeepCCA. GitHub. https://github.com/VahidooX/DeepCCA. Accessed 30 Aug 2018.

60. Sun Q Sen, Liu ZD, Heng PA, Xia D Sen. A theorem on the generalized canonical projective vectors. Pattern Recognit. 2005.

61. Zhang Z, Zhao M, Chow TWS. Binary-and multi-class group sparse canonical correlation analysis for feature extraction and classification. IEEE Trans Knowl Data Eng. 2013.

62. Samat A, Persello C, Gamba P, Liu S, Abuduwaili J, Li E. Supervised and semi-supervised multi-view canonical correlation analysis ensemble for heterogeneous domain adaptation in remote sensing image classification. Remote Sens. 2017.

63. Liu Y, Li Y, Yuan Y-H, Qiang J-P, Ruan M, Zhang Z. Supervised deep canonical correlation analysis for multiview feature learning. In: International Conference on Neural Information Processing. 2017. p. 575–82.

64. De Boer PT, Kroese DP, Mannor S, Rubinstein RY. A tutorial on the cross-entropy method. Ann Oper Res. 2005.

65. Pedregosa F, Michel V, Grisel OLIVIERGRISEL O, Blondel M, Prettenhofer P, Weiss R, et al. Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. J Mach Learn Res. 2011.

66. McCoy F, Eckard L, Nutt LK. Janus-Faced PIDD: A Sensor for DNA Damage-Induced Cell Death or Survival? Mol Cell. 2012;47:667–8.

67. Gehlenborg N, O'Donoghue SI, Baliga NS, Goesmann A, Hibbs MA, Kitano H, et al. Visualization of omics data for systems biology. Nature Methods. 2010.

68. Fanaee-t H, Thoresen M. Multi-insight visualization of multi-omics data via ensemble dimension reduction and tensor factorization. Bioinformatics. 2018.

69. Maaten L van der, Hinton G. Visualizing Data using t-SNE. J Mach Learn Res. 2008.