Deep IDA: A Deep Learning Method for Integrative Discriminant Analysis of Multi-View Data with Feature Ranking–An Application to COVID-19 severity

Jiuzhou Wang, Sandra E. Safo*
Division of Biostatistics
University of Minnesota, MN

Abstract

COVID-19 severity is due to complications from SARS-Cov-2 but the clinical course of the infection varies for individuals, emphasizing the need to better understand the disease at the molecular level. We use clinical and multiple molecular data (or views) obtained from patients with and without COVID-19 who were (or not) admitted to the intensive care unit to shed light on COVID-19 severity. Methods for jointly associating the views and separating the COVID-19 groups (i.e., one-step methods) have focused on linear relationships. The relationships between the views and COVID-19 patient groups, however, are too complex to be understood solely by linear methods. Existing nonlinear one-step methods cannot be used to identify signatures to aid in our understanding of the complexity of the disease. We propose Deep IDA (Integrative Discriminant Analysis) to address analytical challenges in our problem of interest. Deep IDA learns nonlinear projections of two or more views that maximally associate the views and separate the classes in each view, and permits feature ranking for interpretable findings. Our applications demonstrate that Deep IDA has competitive classification rates compared to other state-of-the-art methods and is able to identify molecular signatures that facilitate an understanding of COVID-19 severity.

Contact: ssafo@umn.edu
Supplementary Material: Supplementary data are available online.

Keywords: COVID-19; Multi-view learning; Nonlinearity; Deep learning; Variable selection; One-step methods.

1 Introduction

COVID-19 severity is due to complications from the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) but the clinical course of the infection varies for individuals. Research suggests that patients with and without severe COVID-19 have different genetic, pathological, and clinical signatures (Severe-Covid-19-GWAS-Group, 2020; Overmyer et al., 2021). Further, beyond viral factors, COVID-19 severity depends on host factors, emphasizing the need to use molecular data to better understand the individual response of the disease (Overmyer et al., 2021). In Overmyer et al. (2021), blood samples from patients admitted to the Albany Medical Center, NY from 6 April 2020 to 1 May 2020 for moderate to severe respiratory issues who had COVID-19 or exhibited COVID-19-like symptoms were collected and quantified for transcripts, proteins, metabolomic features and lipids. In addition to the molecular (or omics) data, several clinical and demographic data were collected at the time of enrollment. The authors analyzed each omics data separately, correlated the biomolecules with several clinical outcomes including disease status and severity, and also considered pairwise associations of the omics data to better understand COVID-19 mechanisms. Their findings suggested that COVID-19 severity is likely due to dysregulation in lipid transport system. In this paper, we take a holistic approach to integrate the omics data and the outcome (i.e., COVID-19 patient groups). In particular, instead of assessing pairwise associations and using unsupervised statistical methods as was done in (Overmyer et al., 2021) to correlate the omics data, we model the overall dependency structure among the omics data while simultaneously modeling the separation of the COVID-19 patient groups. Ultimately, our goal is to elucidate the molecular architecture of COVID-19 by identifying molecular signatures with potential to discriminate patients with and without COVID who were or were not admitted to the the intensive care unit (ICU).

There exists many linear (e.g., canonical correlation analysis, CCA [Hotelling (1936); Carroll (1968); Safo, Li and Long (2018); Safo, Ahn, Jeon and Jung (2018)], co-inertia analysis (Min et al., 2019)) and nonlinear (e.g.,Akaho (2001); Andrew et al. (2013); Kan et al. (2016); Benton et al. (2019)) methods that could be used to associate the multiple views. Canonical Correlation Analysis with deep neural network (Deep CCA) (Andrew et al., 2013), and its variations (e.g.Wang et al. (2015),(Benton et al., 2019)), for instance, have been proposed to learn nonlinear projections of two or more views that are maximally correlated. These association-based methods are all unsupervised and they do not use the outcome data (i.e., class labels) when learning the low-dimensional representations. A naive way of using the class labels and all the views simultaneously is to stack the different views and then to perform classification on the stacked data, but this approach does not appropriately model the dependency structure among the views.

To overcome the aforementioned limitations, one-step linear methods (e.g., Safo et al. (2021); Zhang and Gaynanova (2021); Luo et al. (2016)) have been proposed that could be used to jointly associate the multiple views and to separate the COVID-19 patient groups. For instance, in Safo et al. (2021), we proposed a method that combined linear discriminant analysis (LDA) and CCA to learn linear representations that associate the views and separate the classes in each view. However, the relationships among the multiple views and the COVID-19 patient groups are too complex to be understood solely by linear methods. Nonlinear methods including kernel and deep learning methods could be used to model nonlinear structure among the views and between a view and the outcome.

The literature is scarce on nonlinear joint association and separation methods. In Hu et al. (2019), a deep neural network method, multi-view linear discriminant analysis network (MvLDAN), was proposed to learn nonlinear projections of multiple views that maximally correlate the views and separate the classes in each view but the convergence of MvLDAN is not guaranteed. Further, MvLDAN and the nonlinear association-based methods for multiple views mentioned above have one major limitation: they do not rank or select features, as such it is difficult to interpret the models and this limits their ability to produce clinically meaningful findings. If we apply MvLDAN or any of the nonlinear association methods to the COVID-19 omics data, we will be limited in our ability to identify molecules contributing most to the association of the views and the separation of the COVID-19 patient groups.

The problem of selecting or ranking features is well-studied in the statistical learning literature but less-studied in the deep learning literature, especially in deep learning methods for associating multiple views. In Li et al. (2016), a deep feature selection method that adds a sparse one-to-one linear layer between the input layer and the first hidden layer was proposed for feature selection. In another article (Chang et al., 2017), a feature ranking method based on variational dropout was proposed. These methods were developed for data from one view and are not directly applicable to data from multiple views. In Mirzaei et al. (2019), a two-step approach for feature selection using a teacher-student (TS) network was proposed. The "teacher" step obtains the best low-dimensional representation of the data using any dimension reduction method (e.g., deep CCA). The "student" step performs feature selection based on these low-dimensional representations. In particular, a single-layer network with sparse weights is trained to reconstruct the low-dimensional representations obtained from the "teacher" step, and the features are ranked based on the weights. This approach is limiting because the model training (i.e., the identification of the low-dimensional representation of the data) and the feature ranking steps are separated as such, one cannot ensure that the top-ranked features identified are meaningful.

Property/ Methods	Linear One-step Methods	Deep IDA (Proposed)	Randomized KCCA*	Deep CCA*, Deep GCCA+
Nonlinear Relationships		✓	✓	✓
Classification	✓	✓		
Variable ranking/selection	✓	✓		
Covariates	✓	✓		✓
One-step	✓	✓		

Table 1: Unique features of Deep IDA compared to other methods. *Only applicable to two views. +Covariates could be added as additional view in Deep GCCA.

We propose Deep IDA, (short for Deep Integrative Discriminant Analysis), a deep learning method for integrative discriminant analysis, to learn complex nonlinear relationships among the multiple molecular data, and between the molecular data and the COVID-19 patient groups. Deep IDA uses deep neural networks (DNN) to nonlinearly transform each view, constructs an optimization problem that takes as input the output from our DNN (i.e., the nonlinearly transformed views), and learns view-specific projections that result in maximum linear correlation of the transformed views and maximum linear separation within each view. Further, we propose a homogeneous ensemble approach for feature ranking where we implement Deep IDA on different training data subsets to yield low-dimensional representations (that are correlated among the views and separate the classes in each view), we aggregate the classification performance from these low-dimensional representations, we rank features based on the aggregates, and we obtain low-dimensional representations of the data based on the top-ranked variables. As a result, Deep IDA permits feature ranking of the views and enhances our ability to identify features from each view that contribute most to the association of the views and the separation of the classes within each view. We note that our framework for feature ranking is general and adaptable to many deep learning methods and has potential to yield explainable deep learning models for associating multiple views. Table 1 highlights the key features of Deep IDA in comparison with some linear and nonlinear methods for multi-view data. Results from our real data application and simulations with small sample sizes suggest that Deep IDA may be a useful method for small sample size problems compared to other deep learning methods for associating multiple views.

The rest of the paper is organized as follows. In Section 2, we introduce the proposed method and algorithms for implementing the method. In Section 3, we use simulations to evaluate the proposed method. In Section 4, we use two real data applications to showcase the performance of the proposed method. We end with a conclusion remark in Section 5.

2 Method

Let $\mathbf{X}^d \in \mathbf{R}^{n\times p_d}$ be the data matrix for view d, $d = 1, \ldots, D$ (e.g., proeotimcs, metabolomics, image, clinical data). Each view, $\mathbf{X}^d$, has p_d variables, all measured on the same set of n individuals or units. Suppose that each unit belongs to one of two or more classes, K. Let $y_i, i = 1, \ldots, n$ be the class membership for unit i. For each view, let $\mathbf{X}^d$ be a concatenation of data from each class, i.e., $\mathbf{X}^d = [\mathbf{X}^d_1, \mathbf{X}^d_2, ..., \mathbf{X}^d_K]^T$, where $\mathbf{X}^d_k \in \mathbf{R}^{n_k\times p_d}, k = 1, ..., K$ and $n = \sum_{k=1}^K n_k$. For the k-th class in the d-th view, $\mathbf{X}^d_k = [\mathbf{x}^d_{k,1}, \mathbf{x}^d_{k,2}, ..., \mathbf{x}^d_{k,n_k}]^T$, where $\mathbf{x}^d_{k,i} \in \mathbf{R}^{p_d}$ is a column vector denoting the view d data values for the i-th unit in the k-th class. Given the views and data on class membership, we wish to explore the association among the views and the separation of the classes, and also to predict the class membership of a new unit using the unit's data from all views or from some of the views. Additionally, we wish to identify the features that contribute most to the overall association among the views and the separation of classes within each view. Several existing linear (e.g., CCA, generalized CCA) and nonlinear (e.g., deep CCA, deep generalized CCA, random Kernel CCA) methods could be used to first associate the different views to identify low dimensional representations of the views that maximize the correlation among the views or that explain the dependency structure among the views. These low-dimensional representations and the data on class membership could then be used for classification. In this two-step approach, the classification step is independent of the association step and does not take into consideration the effect of class separation on the dependency structure. Alternatively, classification algorithms (e.g., linear or nonlinear methods) could be implemented on the stacked views, however, this approach ignores the association among the views. Recently, Safo et al. (2021) and Zhang and Gaynanova (2021) proposed one-step methods that couple the association step with the separation step and showed that these one-step methods often times result in better classification accuracy when compared to classification on stacked views or the two step methods: association followed by classification. We briefly review the one-step linear method (Safo et al., 2021) for joint association and classification since it is relevant to our method.

2.1 Integrative Discriminant Analysis (IDA) for joint association and classification

Safo et al. (2021) proposed an integrative discriminant analysis (IDA) method that combines linear discriminant analysis (LDA) and canonical correlation analysis (CCA) to explore linear associations among multiple views and linear separation between classes in each view. Let $\mathbf{S}_b^d$ and $\mathbf{S}_w^d$ be the between-class and within-class covariances for the d-th view, respectively. That is, $\mathbf{S}_b^d = \frac{1}{n-1}\sum_{k=1}^K n_k(\boldsymbol{\mu}_k^d - \boldsymbol{\mu}^d)(\boldsymbol{\mu}_k^d - \boldsymbol{\mu}^d)^{\mathrm{T}}$; $\mathbf{S}_w^d = \frac{1}{n-1}\sum_{k=1}^K\sum_{i=1}^n(\mathbf{x}_{ik}^d - \boldsymbol{\mu}_k)(\mathbf{x}_{ik}^d - \boldsymbol{\mu}_k^d)^{\mathrm{T}}$, and $\boldsymbol{\mu}_k^d = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_{ik}^d$ is the mean for class k in view d. Let $\mathbf{S}_{dj}, j < d$ be the cross-covariance between the d-th and j-th views. Let $\mathcal{M}^d = \mathbf{S}_w^{d^{-1/2}}\mathbf{S}_b^d\mathbf{S}_w^{d^{-1/2}}$ and $\mathcal{N}_{dj} = \mathbf{S}_w^{d^{-1/2}}\mathbf{S}_{dj}\mathbf{S}_w^{j^{-1/2}}$. The IDA method finds linear discriminant vectors $(\widehat{\boldsymbol{\Gamma}}^1, \ldots, \widehat{\boldsymbol{\Gamma}}^d)$ that maximally associate the multiple views and separate the classes within each view by solving the optimization problem:

$$\max_{\boldsymbol{\Gamma}^1,\cdots,\boldsymbol{\Gamma}^D} \rho\sum_{d=1}^D \mathrm{tr}(\boldsymbol{\Gamma}^{d\mathrm{T}}\mathcal{M}^d\boldsymbol{\Gamma}^d) + \frac{2(1-\rho)}{D(D-1)}\sum_{d=1,d\neq j}^D \mathrm{tr}(\boldsymbol{\Gamma}^{d\mathrm{T}}\mathcal{N}_{dj}\boldsymbol{\Gamma}^j\boldsymbol{\Gamma}^{j\mathrm{T}}\mathcal{N}_{jd}\boldsymbol{\Gamma}^d) \text{ s.t. } \mathrm{tr}(\boldsymbol{\Gamma}^{d\mathrm{T}}\boldsymbol{\Gamma}^d) = K-1. \quad (1)$$

The first term in equation (1) maximizes the sum of the separation of classes in each view, and the second term maximizes the sum of the pairwise squared correlations between two views. Here, ρ was used to control the influence of separation or association in the optimization problem. The optimum solution for equation (1) was shown to solve D systems of eigenvalue problem (Safo et al., 2021). The discriminant loadings $(\widehat{\boldsymbol{\Gamma}}^1, \ldots, \widehat{\boldsymbol{\Gamma}}^d)$ correspond to the eigenvectors that maximally associate the views and separate the classes within each view. Once the discriminant loadings were obtained, the views were projected onto these loadings to yield the discriminant scores and samples were classified using nearest centroid. In order to obtain features contributing most to the association and separation, the authors imposed convex penalties on $\boldsymbol{\Gamma}^d$ subject to modified eigensystem constraints. In the following sections, we will modify the IDA optimization problem, cast it as an objective function for deep neural networks to study nonlinear associations among the views and separation of classes within a view. We will also implement a feature ranking approach to identify features contributing most to the association of the views and the separation of the classes in a view.

2.2 Deep Integrative Discriminant Analysis (Deep IDA)

We consider a deep learning formulation of the joint association and classification method(Safo et al., 2021) to learn nonlinear relationships among multi-view data and between a view and a binary or multi-class outcome. We follow notations in Andrew et al. (2013) to define our deep learning network. Assume that the deep neural network has $m = 1, \ldots, M$ layers for view d (each view can have its own number of layers), and each layer has c_m^d nodes, for $m = 1, \ldots, M-1$. Let $o_1, o_2, ..., o_D$ be the dimensions of the final(Mth) layer for the D views. Let $h_1^d = s(W_1^d\mathbf{x}^d + b_1^d) \in \Re^{c_1^d}$ be the output of the first layer for view d. Here, $\mathbf{x}^d$ is a length-p^d vector representing a row in $\mathbf{X}^d$, $W_1^d \in \Re^{c_1^d \times p^d}$ is a matrix of weights for view d, $b_1^d \in \Re^{c_1^d}$ is a vector of biases for view d in the first layer, and $s \in \Re \longrightarrow \Re$ is a nonlinear activation function. Using h_1^d as an input for the second layer, let the output of the second layer be denoted as $h_2^d = s(W_2^d h_1^d + b_2^d) \in \Re^{c_2^d}$, $W_2^d \in \Re^{c_2^d \times c_1^d}$ and $b_2^d \in \Re^{c_2^d}$. Continuing in this fashion, let the output of the $(m-1)$th layer be $h_{m-1}^d = s(W_{m-1}^d h_{m-2}^d + b_{m-1}^d) \in \Re^{c_{m-1}^d}$, $W_{m-1}^d \in \Re^{c_{m-1}^d \times c_{m-2}^d}$ and $b_{m-1}^d \in \Re^{c_{m-1}^d}$. Denote the output of the final layer as $f^d(\mathbf{x}^d, \theta^d) = s(W_M^d h_{M-1}^d + b_M^d) \in \Re^{o_d}$, where θ^d is a collection of all weights, W_m^d, and biases, b_m^d for $m = 1, \ldots, M$ and $d = 1, \ldots, D$. In matrix notation, the output of the final layer of the d-th view is denoted as $\mathbf{H}^d = f^d(\mathbf{X}^d) \in \Re^{n \times o_d}$, where it is clear that f^d depends on the network parameters. On this final layer, we propose to solve a modified IDA optimization problem to obtain projection matrices that maximally associate the views and separate the classes. Specifically, we propose to find a set of linear transformations $\mathbf{A}_d = [\boldsymbol{\alpha}_{d,1}, \boldsymbol{\alpha}_{d,2}, ..., \boldsymbol{\alpha}_{d,l}] \in \mathbf{R}^{o_d \times l}$, $l \leq \min\{K-1, o_1, ..., o_D\}$ such that when the nonlinearly-transformed data are projected onto these linear spaces, the views will have maximum linear association and the classes within each view will be linearly separated. Figure 1 is a visual representation of Deep IDA. For a specific view d, $\mathbf{H}^d = [\mathbf{H}_1^d, \mathbf{H}_2^d, ..., \mathbf{H}_K^d]^T, \mathbf{H}_k^d \in \mathbf{R}^{n_k \times o_d}, k = 1, ..., K$ and $n = \sum_{k=1}^K n_k$. For the k-th class in the d-th final output, $\mathbf{H}_k^d = [\mathbf{h}_{k,1}^d, \mathbf{h}_{k,2}^d, ..., \mathbf{h}_{k,n_k}^d]^T$, where $\mathbf{h}_{k,i}^d \in \mathbf{R}^{o_d}$ is a column vector representing the output for subject i in the kth class for view d. Using $\mathbf{H}^d$ as the data matrix for view d, we define the between-class covariance (i.e., $\mathbf{S}_b^d \in \mathbf{R}^{o_d \times o_d}$), the total covariance (i.e., $\mathbf{S}_t^d \in \mathbf{R}^{o_d \times o_d}$), and the cross-covariance between view d and j ($\mathbf{S}_{dj} \in \mathbf{R}^{o_d \times o_j}$) respectively as : $\mathbf{S}_b^d = \frac{1}{n-1}\sum_{k=1}^K n_k(\boldsymbol{\mu}_k^d - \boldsymbol{\mu}^d)(\boldsymbol{\mu}_k^d - \boldsymbol{\mu}^d)^{\mathrm{T}}$; $\mathbf{S}_t^d = \frac{1}{n-1}\sum_{i=1}^n(\mathbf{h}_{k,i}^d - \boldsymbol{\mu}^d)(\mathbf{h}_{k,i}^d - \boldsymbol{\mu}^d)^{\mathrm{T}} = \frac{1}{n-1}(\mathbf{H}^{d^{\mathrm{T}}} - \boldsymbol{\mu}^d \cdot \mathbf{1})(\mathbf{H}^{d^{\mathrm{T}}} - \boldsymbol{\mu}^d \cdot \mathbf{1})^{\mathrm{T}}$, and $S_{dj} = \frac{1}{n-1}(\mathbf{H}^{d^{\mathrm{T}}} - \boldsymbol{\mu}^d \cdot \mathbf{1})(\mathbf{H}^{j^{\mathrm{T}}} - \boldsymbol{\mu}^j \cdot \mathbf{1})^{\mathrm{T}}$. Here, $\mathbf{1}$ is an all-ones row vector of dimension n, $\boldsymbol{\mu}_k^d = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{h}_{k,i}^d \in \mathbf{R}^{o_d}$ is the k-th class mean, and $\boldsymbol{\mu}^d = \frac{1}{K}\sum_{i=1}^K \boldsymbol{\mu}_k^d \in \mathbf{R}^{o_d}$ is the mean for projected view d. To obtain the linear transformations $\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_D$ and the parameters of Deep IDA defining the functions f^d, (i.e., the weights and biases), we propose to solve

the optimization problem:

$$\underset{\mathbf{A}_1,\ldots,\mathbf{A}_D,f^1,\ldots,f^D}{\operatorname{argmax}} \left\{ \rho \frac{1}{D} \sum_{d=1}^{D} tr[\mathbf{A}_d^T \mathbf{S}_b^d \mathbf{A}_d] + (1-\rho) \frac{2}{D(D-1)} \sum_{d=1}^{D} \sum_{j,j\neq d}^{D} tr[\mathbf{A}_d^T \mathbf{S}_{dj} \mathbf{A}_j \mathbf{A}_j^T \mathbf{S}_{dj}^T \mathbf{A}_d] \right\}$$
$$\text{subject to } tr[\mathbf{A}_d^T \mathbf{S}_t^d \mathbf{A}_d] = l, \forall d = 1, ..., D, \tag{2}$$

where $tr[]$ is the trace of some matrix and ρ is a hyper-parameter that controls the relative contribution of the separation and the association to the optimization problem. Here, the first term is an average of the separation for the D views, and the second term is an average of the pairwise squared correlation between two different views ($\frac{D(D-1)}{2}$ correlation measures in total).

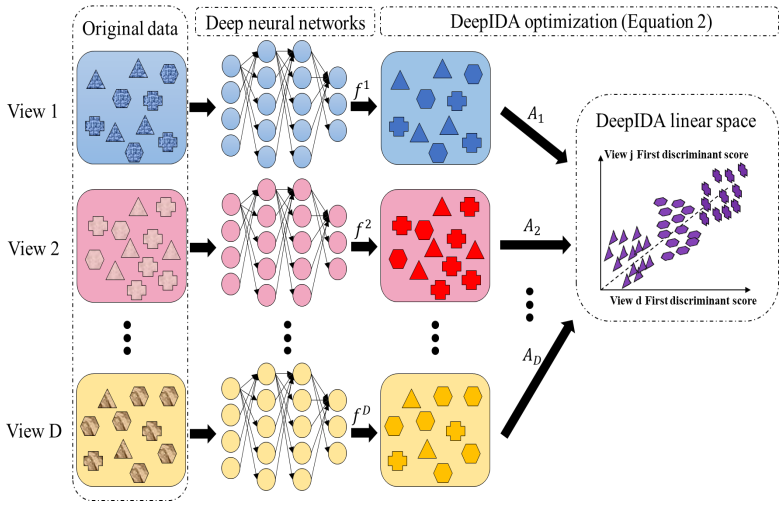

Figure 1: The framework of Deep IDA. Classes are represented by shapes and views are represented by colors. The deep neural networks (DNN) are used to learn nonlinear transformations of the D views, the outputs of the DNN for the views (f^d) are used as inputs in the optimization problem, and we learn linear projections $\mathbf{A}_d, d = 1, \ldots, D$ that maximally correlate the nonlinearly transformed views and separate the classes within each view.

For fixed Deep IDA parameters, (i.e., weights and biases), equation (2) reduces to solving the optimization problem:

$$\underset{\mathbf{A}_1,\ldots,\mathbf{A}_D}{\operatorname{argmax}} \left\{ \rho \frac{1}{D} \sum_{d=1}^{D} tr[\mathbf{A}_d^T \mathbf{S}_b^d \mathbf{A}_d] + (1-\rho) \frac{2}{D(D-1)} \sum_{d=1}^{D} \sum_{j,j\neq d}^{D} tr[\mathbf{A}_d^T \mathbf{S}_{dj} \mathbf{A}_j \mathbf{A}_j^T \mathbf{S}_{dj}^T \mathbf{A}_d] \right\}$$
$$\text{subject to } tr[\mathbf{A}_d^T \mathbf{S}_t^d \mathbf{A}_d] = l, \forall d = 1, ..., D. \tag{3}$$

Denote $\mathbf{S}_t^{d^{-\frac{1}{2}}}$ as the square root of the inverse of $\mathbf{S}_t^d$. With the assumption that $o_d < n$, $\mathbf{S}_t^d$ is non-singular, as such we can take the inverse. Let $\mathcal{M}^d = \mathbf{S}_t^{d^{-\frac{1}{2}}} \mathbf{S}_b^d \mathbf{S}_t^{d^{-\frac{1}{2}}}$, $\mathcal{N}_{dj} = \mathbf{S}_t^{d^{-\frac{1}{2}}} \mathbf{S}_{dj} \mathbf{S}_t^{j^{-\frac{1}{2}}}$ and $\mathbf{\Gamma}_d = \mathbf{S}_t^{d^{\frac{1}{2}}} \mathbf{A}_d$. Then, the optimization problem in equation (3) is equivalently

$$\underset{\mathbf{\Gamma}_1,\mathbf{\Gamma}_2,\ldots,\mathbf{\Gamma}_D}{\operatorname{argmax}} \left\{ \rho \frac{1}{D} \sum_{d=1}^{D} tr[\mathbf{\Gamma}_d^T \mathcal{M}^d \mathbf{\Gamma}_d] + (1-\rho) \frac{2}{D(D-1)} \sum_{d=1}^{D} \sum_{j,j\neq d}^{D} tr[\mathbf{\Gamma}_d^T \mathcal{N}_{dj} \mathbf{\Gamma}_j \mathbf{\Gamma}_j^{\mathrm{T}} \mathcal{N}_{dj}^{\mathrm{T}} \mathbf{\Gamma}_d] \right\}$$
$$\text{subject to } tr[\mathbf{\Gamma}_d^{\mathrm{T}} \mathbf{\Gamma}_d] = l, \forall d = 1, ..., D, \tag{4}$$

and the solution reduces to solving a system of eigenvalue problems. More formally, we have the following theorem.

Theorem 1. *Let $\mathbf{S}_t^d$ and $\mathbf{S}_b^d$ respectively be the total covariance and the between-class covariance for the top-level representations $\mathbf{H}^d, d = 1, \ldots, D$. Let $\mathbf{S}_{dj}$ be the cross-covariance between top-level representations d and j. Assume $\mathbf{S}_t^d \succ 0$. Define $\mathcal{M}^d = \mathbf{S}_t^{d^{-\frac{1}{2}}} \mathbf{S}_b^d \mathbf{S}_t^{d^{-\frac{1}{2}}}$ and $\mathcal{N}_{dj} = \mathbf{S}_t^{d^{-\frac{1}{2}}} \mathbf{S}_{dj} \mathbf{S}_t^{j^{-\frac{1}{2}}}$. Then $\mathbf{\Gamma}^d \in \Re^{o_d \times l}$, $l \leq \min\{K - 1, o_1, \ldots, o_D\}$ in equation (4) are eigenvectors corresponding respectively to eigenvalues $\mathbf{\Lambda}_d = diag(\lambda_{d_k}, \ldots, \lambda_{d_l})$,*

$\lambda_{d_k} > \cdots > \lambda_{d_l} > 0$ *that iteratively solve the eigensystem problems:*

$$\left(c_1 \mathcal{M}^d + c_2 \sum_{j \neq d}^{D} \mathcal{N}_{dj} \mathbf{\Gamma}_j \mathbf{\Gamma}_j^T \mathcal{N}_{dj}^T \right) \mathbf{\Gamma}_d = \mathbf{\Lambda}_d \mathbf{\Gamma}_d, \forall d = 1, ..., D$$

where $c_1 = \frac{\rho}{D}$ *and* $c_2 = \frac{2(1-\rho)}{D(D-1)}$.

The proof of Theorem 1 is in the supplementary material. We can initialize the algorithm using any arbitrary normalized nonzero matrix. After iteratively solving D eigensystem problems until convergence, we obtain the optimized linear transformations $\widetilde{\mathbf{\Gamma}}_1, ..., \widetilde{\mathbf{\Gamma}}_D$ which maximize both separation of classes in the top-level representations, $\mathbf{H}^d$, and the association among the top-level representations. Since we find the eigenvector-eigenvalue pairs of $(c_1 \mathcal{M}^d + c_2 \sum_{j=1, j \neq d}^{D} \mathcal{N}_{dj} \mathbf{\Gamma}_j \mathbf{\Gamma}_j^T \mathbf{N}_{dj}^T)$, the columns of $\widetilde{\mathbf{\Gamma}}_d$, $d = 1, ..., D$ are orthogonal and provide unique information that contributes to the association and separation in the top-level representations. Given the optimized linear transformations $\widetilde{\mathbf{\Gamma}}_1, ..., \widetilde{\mathbf{\Gamma}}_D$, we construct the objective function for the D deep neural networks as:

$$\underset{f^1, f^2, ..., f^D}{\operatorname{argmax}} \; c_1 \sum_{d=1}^{D} tr[\widetilde{\mathbf{\Gamma}}_d^{\mathrm{T}} \mathcal{M}^d \widetilde{\mathbf{\Gamma}}_d] + c_2 \sum_{d=1}^{D} \sum_{j, j \neq d}^{D} tr[\widetilde{\mathbf{\Gamma}}_d^{\mathrm{T}} \mathcal{N}_{dj} \widetilde{\mathbf{\Gamma}}_j \widetilde{\mathbf{\Gamma}}_j^{\mathrm{T}} \mathcal{N}_{dj}^T \widetilde{\mathbf{\Gamma}}_d]. \tag{5}$$

Theorem 2. *For d fixed, let* $\eta_{d,1}, \ldots, \eta_{d,l}$, $l \leq \min\{K-1, o_1, \ldots, o_D\}$ *be the largest l eigenvalues of* $c_1 \mathcal{M}^d + c_2 \sum_{j \neq d}^{D} \mathcal{N}_{dj} \mathbf{\Gamma}_j \mathbf{\Gamma}_j^T \mathcal{N}_{dj}^T$. *Then the solution* $\widetilde{f}^d$ *to the optimization problem in equation (5) for view d maximizes*

$$\sum_{r=1}^{l} \eta_{d,r}. \tag{6}$$

The objective function in Theorem 2 aims to maximize the sum of the l largest eigenvalues for each view. In obtaining the view-specific eigenvalues, we use the cross-covariances between that view and each of the other views, and the total and between-class covariances for that view. Thus, by maximizing the sum of the eigenvalues, we are estimating corresponding eigenvectors that maximize both the association of the views and the separation of the classes within each view. By Theorem 2, the solution $\widetilde{f}^1, \ldots, \widetilde{f}^D$, i.e. weights and biases for the D neural networks of the optimization problem (5) is also given by the following:

$$\underset{f^1, f^2, ..., f^D}{\operatorname{argmax}} \sum_{d=1}^{D} \sum_{r=1}^{l} \eta_{d,r}. \tag{7}$$

The objectives (4) and (7) are naturally bounded because the characteristic roots of every $\mathbf{S}_t^{d^{-1}} \mathbf{S}_b^d$ (and hence $\mathbf{S}_t^{d^{-\frac{1}{2}}} \mathbf{S}_b^d \mathbf{S}_t^{d^{-\frac{1}{2}}}$) is bounded and every squared correlation is also bounded. This guarantees convergent solutions of the loss function in equation (7) compared to the method in (Dorfer et al., 2015) that constrain the within-group covariance and has unbounded loss function. We optimize the objective in (7) with respect to the weights and biases for each layer and each view to obtain $\widetilde{f^1}, ..., \widetilde{f^D}$. The estimates $\widetilde{f^1(\mathbf{X}^1)}, ..., \widetilde{f^D(\mathbf{X}^D)}$ are used as the low-dimensional representation for classification. For classification, we follow the approach in Safo et al. (2021) and we use nearest centroid to assign future events to the class with the closest mean. For this purpose, we have the option to use the pooled low-dimensional representations $\widehat{f} = (\widetilde{f^1(\mathbf{X}^1)}, ..., \widetilde{f^D(\mathbf{X}^D)})$ or the individual estimates $\widetilde{f^d(\mathbf{X}^d)}, d = 1, \ldots, D$.

2.3 Optimization and Algorithm

- Feed forward and calculate the loss. The output for D deep neural networks are $\mathbf{H}^1, ..., \mathbf{H}^D$, which includes the neural network parameters (i.e., the weights and biases). Based on the objective in equation (7), the final loss is calculated and denoted as $\mathcal{L} = -\sum_{d=1}^{D} \sum_{r=1}^{l} \eta_{d,r}$.

- Gradient of the loss function. The loss function $\mathcal{L}$ depends on the estimated linear projections $\widetilde{\mathbf{\Gamma}}^d, d = 1, \cdots, D$ and since these linear projections use the outputs of the last layer of the network, there are no parameters involved. Therefore we calculate gradient of the loss function with respect to the view-specific output, i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{H}^d}, d = 1, \ldots, D$.

- Gradient within each sub-network. Since each view-specific sub-network is propagated separately, we can calculate the gradient of each sub-network independently. As the neural network parameters (i.e., weights and biases) of view d network is denoted as θ^d, we can calculate the partial derivative of last layer with respect to sub-network parameters as $\frac{\partial \mathbf{H}^d}{\partial \theta^d}$. These networks include shallow or multiple layers and nonlinear activation functions, such as Leaky-ReLu (Maas et al., 2013).

- Deep IDA update via gradient descent. By the chain rule, we can calculate $\nabla_{\theta^d}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{H}^d} \cdot \frac{\partial \mathbf{H}^d}{\partial \theta^d}$. We use the *autograd* function in the PyTorch (Paszke et al., 2019) package to compute this gradient. Therefore, for every optimization step, a stochastic gradient descent-based optimizer, such as Adam (Kingma and Ba, 2014), is used to update the network parameters.

We describe the Deep IDA algorithm in Algorithm 1. We also describe in Algorithm 2 the approach for obtaining the linear projections using the output of the final layer.

Algorithm 1 Algorithm of Deep IDA

Input: Training data $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^D\}$ and class labels $\mathbf{y}$; number of nodes of each layer in D neural networks (including dimensions of linear sub-spaces to project onto, $o_1, o_2, ..., o_D$); learning rate α

Output: Optimized weights and biases for D neural networks and corresponding estimates $(\widetilde{f^1(\mathbf{X}^1)}, ..., \widetilde{f^D(\mathbf{X}^D)})$

1: **Initialization** Assign random numbers to weights and biases for D neural networks
2: **while** loss not converge **do**
3: Feed forward the network of each view with latest weights and biases to obtain the final layer $\mathbf{H}^d = f^d(\mathbf{X}^d) \in \mathbf{R}^{n \times o_d}, d = 1, 2, .., D$
4: Apply Algorithm 2 to obtain $\widetilde{\mathbf{\Gamma}}_1, ..., \widetilde{\mathbf{\Gamma}}_D$
5: Compute eigenvalues of $c_1\mathcal{M}^d + c_2\sum_{j,j\neq d}^{D}\mathcal{N}_{dj}\widetilde{\mathbf{\Gamma}}_j\widetilde{\mathbf{\Gamma}}_j^{\mathrm{T}}\mathcal{N}_{dj}^{\mathrm{T}}$ to obtain the loss function $-\sum_{d=1}^{D}\sum_{i=1}^{l}\eta_{d,i}$
6: Compute the gradient of weights and biases for each network by the PyTorch *Autograd* function
7: Update the weights and biases with the specified learning rate α
8: **end while**

Algorithm 2 Algorithm for iteratively solving Eigenvalue Problem

Input: Training data $\mathbf{H} = \{\mathbf{H}^1, \mathbf{H}^2, ..., \mathbf{H}^D\}$ and corresponding class labels $\mathbf{y}$; convergence criteria ϵ

Output: Optimized discriminant loadings $\widetilde{\mathbf{\Gamma}}_1, ..., \widetilde{\mathbf{\Gamma}}_D$

1: Compute $\mathcal{M}^d$, $\mathcal{N}_{dj}$ for $d, j = 1, 2, .., D$
2: **while** $\max_{d=1,..,D} \frac{\|\widehat{\mathbf{\Gamma}}_{d,new} - \widehat{\mathbf{\Gamma}}_{d,old}\|_F^2}{\|\widehat{\mathbf{\Gamma}}_{d,old}\|_F^2} > \epsilon$ **do**
3: **for** $d = 1, .., D$ **do**: fix $\widehat{\mathbf{\Gamma}}_j, \forall j \neq d$, compute $\widehat{\mathbf{\Gamma}}_d$ by **Theorem 1**
4: **end while**
5: Set $\widetilde{\mathbf{\Gamma}}_d = \widehat{\mathbf{\Gamma}}_d, \forall d = 1, ..., D$

2.4 Comparison of Deep IDA with Multi-view Linear Discriminant Analysis Network, MvLDAN

Our proposed method is related to the method in Hu et al. (2019) since we find linear projections of nonlinearly transformed views that separate the classes within each view and maximize the correlation among the views. The authors in Hu et al. (2019) proposed to solve the following optimization problem for linear projection matrices $\mathbf{A}_1, \cdots, \mathbf{A}_D$ and neural network parameters (weights and biases):

$$\underset{f^1,\cdots,f^D,\mathbf{A}_1,\cdots,\mathbf{A}_D}{\operatorname{argmax}} \; tr\left((\mathbf{S}_w + \beta\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}(\mathbf{S}_b + \lambda\mathbf{S}_c)\right), \tag{8}$$

where $\mathbf{A} = [\mathbf{A}_1^{\mathrm{T}} \cdots \mathbf{A}^{D^{\mathrm{T}}}]^{\mathrm{T}}$ is a concatenation of projection matrices from all views, $\mathbf{S}_b$ and $\mathbf{S}_w$ are the between-class and within-class covariances for all views, respectively, and $\mathbf{S}_c$ is the cross-covariance matrix for all views. Further, λ and β are regularization parameters. The authors then considered to learn the parameters of the deep neural network by maximizing the smallest eigenvalues of the generalized eigenvalue problem arising from equation (8) that do not exceed some threshold that is specified in advance. Although we have the same goal as the authors in Hu et al. (2019), our optimization formulation in equation (2), and our loss function are different. We constrain the total covariance matrix $\mathbf{S}_t$ instead of $\mathbf{S}_w$ and as noted above, our loss function is bounded. As such, we do not have convergence issues when training our deep learning parameters. Second, in associating the multiple views, we square the pairwise correlations. As noted in Carroll (1968), the canonical variates estimated from the sum of correlations (SUMCOR) formulation for generalizing CCA to multiple views, which is used in Hu et al. (2019), is affected by sign changes. For example, if the canonical variates for two views are almost the same (in terms of absolute value entries) but the signs are opposite, the sum of the pairwise correlations will be close to zero and this will mask the true relationship between the different views. The sum of the squared correlations avoids this problem. A major drawback of MvLDAN (and existing nonlinear

association-based methods for multi-view data) is that they cannot be used to identify variables contributing most to the association among the views and/or separation in the classes. In the next section, we propose an approach that bridges this gap.

2.5 Feature Ranking via Bootstrap

A main limitation of existing nonlinear methods for integrating data from multiple views is that it is difficult to interpret the models and this limits their ability to produce clinically meaningful findings. We propose a general framework for ranking features in deep learning models for data from multiple views that is based on ensemble learning. Specifically, we propose a homogeneous ensemble approach for feature ranking where we implement Deep IDA on different training data subsets to yield low-dimensional representations (that are correlated among the views while separating the classes in each view), we aggregate the classification performance from these low-dimensional representations, we rank features based on the aggregates, and we obtain a final low-dimensional representations of the data based on the top-ranked variables. We emphasize that while we embed Deep IDA in this feature ranking procedure, in principle, any method for associating multiple views can be embedded in this process. This makes the proposed approach general. We outline our feature ranking steps below. Figure 2 is a visual representation of the feature ranking procedure.

1. Generate M bootstrap sets of sample indices of the same sample size as the original data by random sampling the indices with replacement. Denote the bootstrap sets of indices as $B_1, B_2, ..., B_M$. Let the out-of-bag sets of indices be $B_1^c, B_2^c, ..., B_M^c$. In generating the bootstrap training sets of indices, we use stratified random sampling to ensure that the proportions of samples in each class in the bootstrap sets of indices are similar to the original data.

2. Draw q bootstrap sets of feature indices for each view. For view j, $j = 1, \cdots, D$, draw $0.8p_j$ samples from the index set and denote as $V_{m,j}$. $V_m = \{V_{m,1}, V_{m,2}, ..., V_{m,D}\}$ is the m-th bootstrap feature index for all D views.

3. Pair sample and feature index sets randomly. Denote the pairing results as $(B_1, V_1), (B_2, V_2), ..., (B_M, V_M)$. For each pair (B_m, V_m) and $(B_m^c, V_m), (m = 1, 2, ..., M)$, extract corresponding subsets of data.

4. For the mth pair, denote the bootstrap data as $\mathbf{X}_{m,1}, ..., \mathbf{X}_{m,D}$ and the out-of-bag data as $\mathbf{X}_{m,1}^c, ..., \mathbf{X}_{m,D}^c$. Train Deep IDA based on $\mathbf{X}_{m,1}, ..., \mathbf{X}_{m,D}$, and calculate the test classification rate based on $\mathbf{X}_{m,1}^c, ..., \mathbf{X}_{m,D}^c$. Record this rate as baseline classification rate for pair $m, m = 1, 2, ..., M$.

5. For the dth view in the mth pair, permute the kth variable in $\mathbf{X}_{m,d}^c$ and keep all other variables unchanged. Denote the permuted view d data as $\mathbf{X}_{m,d,k-permuted}^c$. Use the learned model from Step 4 and the permuted data $(\mathbf{X}_{m,1}^c, ..., \mathbf{X}_{m,d,k-permuted}^c, ..., \mathbf{X}_{m,D}^c)$ to obtain the classification rate for the permuted data.

6. Repeat Step 5 for $m = 1, ..., M$, $d = 1, ..., D$ and $k = 1, ..., p_d$. Record the variables that lead to a decrease in classification rate when using the permuted data.

7. For the dth view, calculate the occurrence proportion of variable k, $k = 1, 2, ..., p_d$ (i.e., the proportion of times a variable leads to a decrease in classification accuracy) as $\frac{n_k}{N_k}$, where n_k denotes the number of times that permuting variable k leads to a decrease in out-of-bag classification rate, and N_k denotes the number of times that variable k is permuted (i.e. the total number of times variable k is selected in the bootstrap feature index sets). Repeat this process for all views.

8. For each view, rank the variables based on the occurrence proportions and select the top-ranked variables as the important variables. The top-ranked variables could be the top r variables or top $r\%$ variables.

Once we have obtained the top-ranked variables for each view (number of variables need to be in advance), we train Deep IDA on the original data but with these top-ranked variables. If testing data are available, say $\mathbf{X}_{test}^d$, we use the learned neural network parameters to construct the output of the top-level representations for each view, i.e., $\mathbf{H}_{test}^d = \widetilde{f^d(\mathbf{X}_{test}^d)}, d = 1, \ldots, D$. These are then used in a nearest neighbor algorithm to predict the test classes. Thus, our final low-dimensional representations of the data are based on features from each view that contribute most to the association of the views and the separation of the classes within each view. We implement the proposed algorithm as a Python 3.0 package with dependencies on NumPy (Oliphant, 2006) and PyTorch (Paszke et al., 2019) for numerical computations, and Matlib for model visualization.

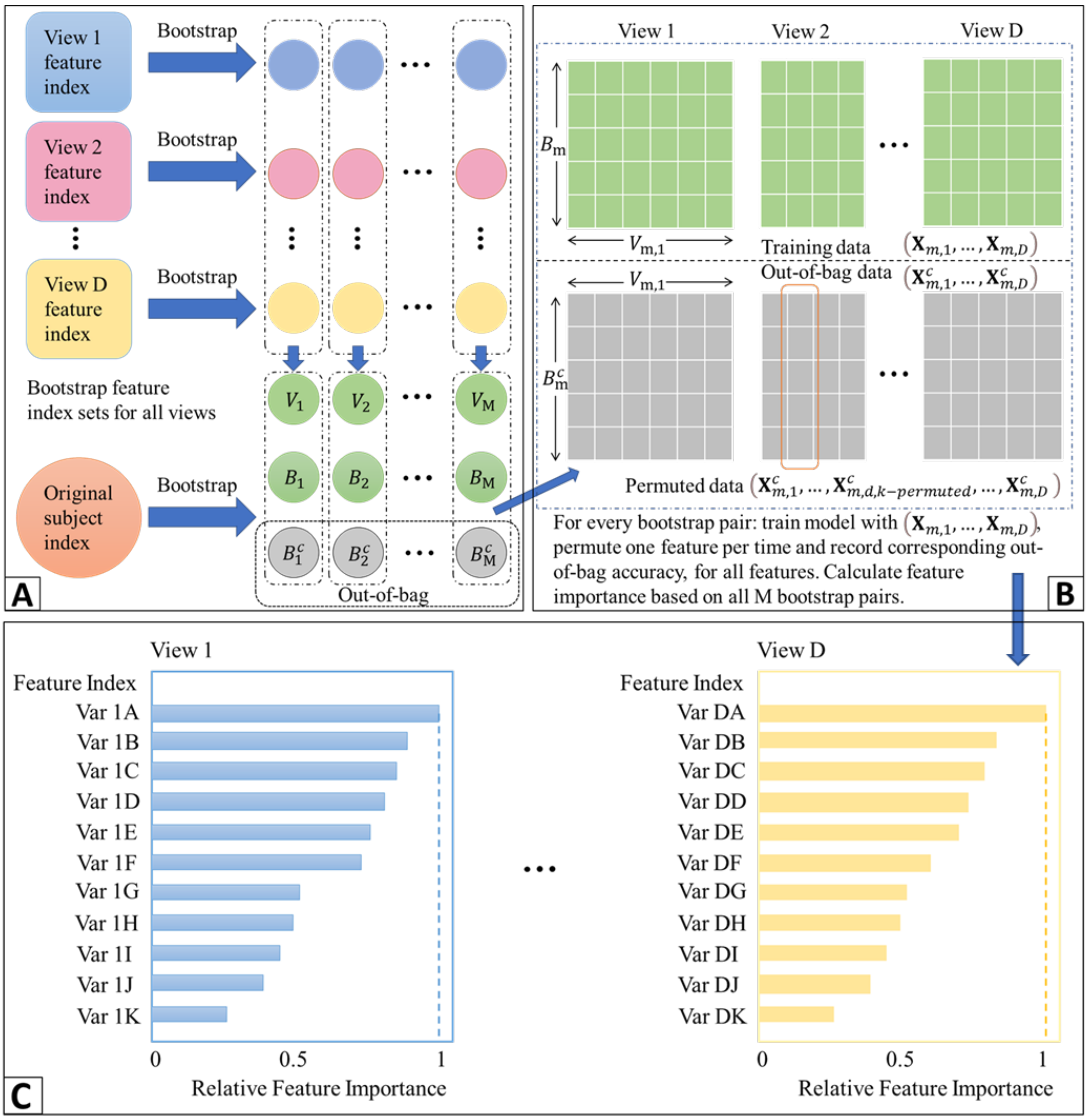

Figure 2: The framework of feature ranking process. A) Bootstrapping samples and features. It includes Steps 1 and 2. V_m: the m-th bootstrap feature index; B_m: the m-th bootstrap sample index; B_m^c: the m-th bootstrap out-of-bag sample index. B) Pairing data, training the reference model, permuting and recording the decrease in classification performance. This includes Steps 3-6. C) Ranking features based on how often the baseline classification accuracy is reduced when permuted. This includes Steps 7 and 8.

3 Simulations

3.1 Set-up

We conduct simulation studies to assess the performance of Deep IDA for varying data dimensions, and as the relationship between the views and within a view become more complex. We demonstrate that Deep IDA is capable of i) simultaneous association of data from multiple views and discrimination of sample classes, and ii) identifying signal variables.

We consider two different simulation Scenarios. In Scenario One, we simulate data to have linear relationships between views and linear decision boundaries between classes. In Scenario Two, we simulate data to have nonlinear relationships between views and nonlinear decision boundaries between classes. There are $K = 3$ classes and $D = 2$ and $D = 3$ views in Scenario One. In Scenario Two, there are $K = 2$ classes and $D = 2$ views. In all Scenarios, we generate 20 Monte Carlo training, validation, and testing sets. We evaluate the proposed and existing methods using the following criteria: i) test accuracy, and ii) feature selection. For feature selection, we evaluate the methods ability to select the true signals. In Scenario One, the first 20 variables are important, and in Scenario Two, the Top 10% of variables in view 1 are signals. Since Deep IDA and the teacher-student (TS) framework rank features, and SIDA assigns zero weights to unimportant variables, for fair comparison, we only assess the number of signal variables that are in the Top 20 (for Scenario One) and the Top 10% (for Scenario Two) variables selected by the methods. We compare test accuracy for Deep IDA with and without the variable ranking approach proposed in this manuscript.

3.2 Comparison Methods

We compare Deep IDA with classification-, association-, and joint association and classification-based methods. For classification-based methods, we consider the support vector machine (Hastie et al., 2009) on stacked views. For association-based methods, we consider nonlinear methods such as deep canonical correlation analysis (Deep CCA) (Andrew et al., 2013), deep generalized CCA (DGCCA) (Benton et al., 2019) when there are three or more views, and randomized kernel CCA (RKCCA) (Lopez-Paz et al., 2014). The association-based methods only consider nonlinear associations between views, as such we follow with SVM to perform classification using the learned low-dimensional representations from the methods. We also compare Deep IDA to SIDA (Safo et al., 2021), a joint association and classification method that models linear relationships between views and among classes in each view. We perform SIDA and RKCCA using the Matlab codes the authors provide. We set the number of random features in RKCCA as 300 and we select the bandwidth of the radial basis kernel using median heuristic (Garreau et al., 2017). We perform Deep CCA and Deep generalized CCA using the PyTorch codes the authors provide. We couple Deep CCA and Deep GCCA with the teacher-student framework (TS) (Mirzaei et al., 2019) to rank variables. We also investigate the performance of these methods when we use variables selected from Deep IDA.

3.3 Linear Simulations

We consider two simulation settings in this Scenario and we simulate data similar to simulations in Safo et al. (2021). In Setting One, there are $D = 2$ views $\mathbf{X}^1$ and $\mathbf{X}^2$, with $p_1 = 1,000$ and $p_2 = 1,000$ variables respectively. There are $K = 3$ classes each with size $n_k = 180$, $k = 1, 2, 3$ giving a total sample size of $n = 540$. Let $\mathbf{X}^d = [\mathbf{X}_1^d, \mathbf{X}_2^d, \mathbf{X}_3^d], d = 1, 2$ be a concatenation of data from the three classes. The combined data $\left(\mathbf{X}_k^1, \mathbf{X}_k^2\right)$ for each class are simulated from $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$, $\boldsymbol{\mu}_k = (\boldsymbol{\mu}_k^1, \boldsymbol{\mu}_k^2)^{\mathrm{T}} \in \Re^{p_1+p_2}, k = 1, 2, 3$ is the combined mean vector for class k; $\boldsymbol{\mu}_k^1 \in \Re_1^p, \boldsymbol{\mu}_k^2 \in \Re_2^p$ are the mean vectors for $\mathbf{X}_k^1$ and $\mathbf{X}_k^2$ respectively. The covariance matrix $\boldsymbol{\Sigma}$ for the combined data for each class is partitioned as

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}^1 & \boldsymbol{\Sigma}^{12} \\ \boldsymbol{\Sigma}^{21} & \boldsymbol{\Sigma}^2 \end{pmatrix}, \boldsymbol{\Sigma}^1 = \begin{pmatrix} \widetilde{\boldsymbol{\Sigma}}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{p_1-20} \end{pmatrix}, \boldsymbol{\Sigma}^2 = \begin{pmatrix} \widetilde{\boldsymbol{\Sigma}}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{p_2-20} \end{pmatrix}$$

where $\boldsymbol{\Sigma}^1$, $\boldsymbol{\Sigma}^2$ are respectively the covariance of $\mathbf{X}^1$ and $\mathbf{X}^2$, and $\boldsymbol{\Sigma}^{12}$ is the cross-covariance between the two views. We generate $\widetilde{\boldsymbol{\Sigma}}^1$ and $\widetilde{\boldsymbol{\Sigma}}^2$ as block diagonal with 2 blocks of size 10, between-block correlation 0, and each block is a compound symmetric matrix with correlation 0.8. We generate the cross-covariance matrix $\boldsymbol{\Sigma}^{12}$ as follows. Let $\mathbf{V}^1 = [\mathbf{V}_1^1,\ \mathbf{0}_{(p_1-20)\times 2}]^{\mathrm{T}} \in \Re^{p_1\times 2}$ and the entries of $\mathbf{V}_1^1 \in \Re^{20\times 2}$ are *i.i.d* samples from U(0.5,1). We similarly define $\mathbf{V}^2$ for the second view, and we normalize such that ${\mathbf{V}^1}^T\boldsymbol{\Sigma}^1\mathbf{V}^1 = \mathbf{I}$ and ${\mathbf{V}^2}^{\mathrm{T}}\boldsymbol{\Sigma}^2\mathbf{V}^2 = \mathbf{I}$. We then set $\boldsymbol{\Sigma}^{12} = \boldsymbol{\Sigma}^1\mathbf{V}^1\mathbf{D}{\mathbf{V}^2}^{\mathrm{T}}\boldsymbol{\Sigma}^2$, $\mathbf{D} = \mathrm{diag}(0.4, 0.2)$ to depict moderate association between the views. For class separation, define the matrix $[\boldsymbol{\Sigma}\mathbf{A}, \mathbf{0}_{(p_1+p_2)}] \in \Re^{(p_1+p_2)\times 3}$; $\mathbf{A} = [\mathbf{A}^1, \mathbf{A}^2]^{\mathrm{T}} \in \Re^{(p_1+p_2)\times 2}$, and set the first, second, and third columns as the mean vector for class 1, 2, and 3, respectively. Here, the first column of $\mathbf{A}^1 \in \Re^{p_1\times 2}$ is set to $(c\mathbf{1}_{10}, \mathbf{0}_{p_1-10})$ and the second column is set to $(\mathbf{0}_{10}, -c\mathbf{1}_{10}, \mathbf{0}_{p-20})$; we fix c at 0.2. We set $\mathbf{A}^2 \in \Re^{p_2\times 2}$ similarly, but we fix c at 0.1 to allow for different class separation in each view.

In Setting Two, we simulate $D = 3$ views, $\mathbf{X}^d, d = 1, 2, 3$, and each view is a concatenation of data from three classes as before. The combined data $\left(\mathbf{X}_k^1, \mathbf{X}_k^2, \mathbf{X}_k^3\right)$ for each class are simulated from $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}_k = (\boldsymbol{\mu}_k^1, \boldsymbol{\mu}_k^2, \boldsymbol{\mu}_k^3)^{\mathrm{T}} \in \Re^{p_1+p_2+p_3}, k = 1, 2, 3$ is the combined mean vector for class k; $\boldsymbol{\mu}_k^d \in \Re^{p_d}, j = 1, 2, 3$ are the mean vectors for $\mathbf{X}_k^d, d = 1, 2, 3$. The true covariance matrix $\boldsymbol{\Sigma}$ is defined similar to Setting One but with the following modifications. We include $\boldsymbol{\Sigma}_3$, $\boldsymbol{\Sigma}_{13}$, and $\boldsymbol{\Sigma}_{23}$, and we set $\boldsymbol{\Sigma}_{13} = \boldsymbol{\Sigma}_{23} = \boldsymbol{\Sigma}_{12}$. Like $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, $\boldsymbol{\Sigma}_3$ is partitioned into signals and noise, and the covariance for the signal variables, $\widetilde{\boldsymbol{\Sigma}}^3$, is also block diagonal with 2 blocks of size 10, between-block correlation 0, and each block is compound symmetric matrix with correlation 0.8. We take $\boldsymbol{\mu}_k$ to be the columns of $[\boldsymbol{\Sigma}\mathbf{A}, \mathbf{0}_{(p_1+p_2+p_3)}] \in \Re^{(p_1+p_2+p_3)\times 2}$, and $\mathbf{A} = [\mathbf{A}^1, \mathbf{A}^2, \mathbf{A}^3]^{\mathrm{T}} \in \Re^{(p_1+p_2+p_3)\times 2}$. The first column of $\mathbf{A}^j \in \Re^{p_j \times 2}$ is set to $(c_j \mathbf{1}_{10}, \mathbf{0}_{p_1-10})$ and the second column is set to $(\mathbf{0}_{10}, -c_j \mathbf{1}_{10}, \mathbf{0}_{p-20})$ for $j = 1, 2, 3$. We set $(c_1, c_2, c_3) = (0.2, 0.1, 0.05)$ to allow for different class separation in each view.

3.3.1 Results for Linear Simulations

Table 2 gives classification accuracy for the methods and the true positive rates for the top 20 variables selected. We implemented a three-hidden layer network with dimensions 512, 256, and 64 for both Deep IDA and Deep CCA. The dimension of the output layer was set as 10. Table 3 in the supplementary material lists the network structure used for each setting. For Deep IDA + Bootstrap, the bootstrap algorithm proposed in the Methods Section was implemented on the training data to choose the top 20 ranked variables. We then implemented Deep IDA on the training data but with just the variables ranked in the top 20 in each view. The learned model and the testing data were used to obtain test error. To compare our feature ranking process with the teacher-student (TS) network approach for feature ranking, we also implemented Deep IDA without the bootstrap approach for feature ranking, and we used the learned model from Deep IDA in the TS framework for feature ranking. We also performed feature ranking using the learned model from Deep CCA (Setting One) and Deep GCCA (Setting Two). The average error rates for the nonlinear methods are higher than the error rate for SIDA, a linear method for joint association and classification analysis. This is not surprising as the true relationships among the views, and the classes within a view are linear. Nevertheless, the average test error rate for Deep IDA that is based on the top 20 variables in each view from the bootstrap method (i.e., Deep IDA + Bootstrap) is lower than the average test error rates from Deep CCA, RKCCA, and SVM (on stacked views). When we implemented Deep CCA, RKCCA, SVM, and DGCCA on the top 20 variables that were selected by our proposed method, we observed a decrease in the error rate across most of the methods, except for RKCCA. For instance, the error rates for Deep CCA using all variables compared to using the top 20 variables identified by our method were 33.17% and 22.95%, respectively. Further, compared to Deep IDA on all variables (i.e., Deep IDA + TS), Deep IDA + Bootstrap has a lower average test error, demonstrating the advantage of variable selection. In Setting Two, the classification accuracy for Deep GCCA was poor. In terms of variable selection, compared to SIDA, the proposed method was competitive at identifying the top-ranked 20 variables. The TS framework for ranking variables was sub-optimal as evident from the true positive rates for Deep IDA + TS, Deep CCA + TS, and Deep GCCA + TS.

3.4 Nonlinear Simulations

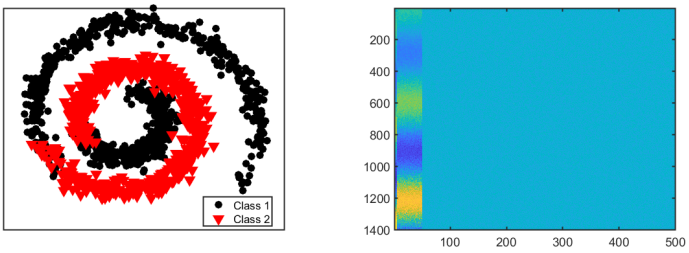

Figure 3: Setting One. (Left panel) Structure of nonlinear relationships between signal variables in view 1. (Right panel) Image plot of view 1 showing the first 50 variables as signals.

We consider four different settings for this scenario. Each setting has $K = 2$ classes but they vary in their dimensions. In each setting, 10% of the variables in the first view are signals and the first five signal variables in the first view are related to the remaining signal variables in a nonlinear way (See Figure 3). We generate data for View 1 as: $\mathbf{X}_1 = \widetilde{\mathbf{X}}_1 \cdot \mathbf{W} + 0.2\mathbf{E_1}$ where $(\cdot)$ is element-wise multiplication, $\mathbf{W} \in \Re^{n\times p_1} = [\mathbf{1}_{0.1\times p_1}, \mathbf{0}_{0.9\times p_1}]$ is a matrix of ones and zeros, $\mathbf{1}$ is a matrix of ones, $\mathbf{0}$ is matrix of zeros, and $\mathbf{E}_1 \sim N(0, 1)$. The first five columns (or variables) of $\widetilde{\mathbf{X}}_1 \in \Re^{n\times p^1}$ are simulated from $\exp(0.15\boldsymbol{\theta}) \cdot \sin(1.5\boldsymbol{\theta})$. The next $0.1p^1 - 5$ variables

Table 2: Linear Setting: RS; randomly select tuning parameters space to search. TPR-1; true positive rate for $\mathbf{X}^1$. Similar for TPR-2. TS: Teacher student network. − indicates not applicable. Deep IDA + Bootstrap is when we use the bootstrap algorithm to choose the top 20 ranked variables, train Deep IDA with the top 20-ranked variables, and then use the learned model and the test data to obtain test errors.

Method	Error (%)	TPR-1	TPR-2	TPR-3
Setting One				
Deep IDA + Bootstrap	24.69	100.00	95.25	-
Deep IDA+ TS	33.87	33.25	21.75	-
SIDA	20.81	99.50	93.50	-
Deep CCA + TS	33.17	4.25	3.25	-
Deep CCA on selected top 20 variables	22.95	-	-	-
RKCCA	40.1	-	-	-
RKCCA on selected top 20 variables	42.07	-	-	-
SVM (Stacked views)	31.53	-	-	-
SVM on selected top 20 variables (Stacked views)	22.03	-	-	-
Setting Two				
Deep IDA + Bootstrap	23.16	100.00	94.75	78.75
Deep IDA + TS	31.22	72.00	57.75	47.75
SIDA	19.79	99.75	99.50	97.25
DGCCA + TS	60.01	2.0	2.0	2.25
DGCCA on selected top 20 variables	57.40	-	-	-
SVM (Stacked views)	29.06	-	-	-
SVM on selected top 20 variables (Stacked views)	19.56	-	-	-

are simulated from $\exp(0.15\boldsymbol{\theta}) \cdot \cos(1.5\boldsymbol{\theta})$. Here, $\boldsymbol{\theta} = \tilde{\boldsymbol{\theta}} + 0.5U(0,1)$, and $\tilde{\boldsymbol{\theta}}$ is a vector of n evenly spaced points between 0 and 3π. The remaining $0.9p^1$ variables (or columns) in $\widetilde{\mathbf{X}}_1$ are generated from the standard normal distribution. View 2 has no signal variables and the variables do not have nonlinear relationships. Data for View 2 are generated as follows. We set each variable in $\mathbf{X}_1$ with negative entries to zero, normalized each variable to have unit norm and added a random number generated from the standard uniform distribution.

3.4.1 Results for Nonlinear Simulations

Table 3 gives the classification and variable selection accuracy. We chose the number of layers that gave the minimum validation error (based on our approach without bootstrap) or better variable selection. Table 4 in the supplementary material lists the network structure used for each setting. We compare Deep IDA to the nonlinear methods. Similar to the linear setting, for Deep IDA + Bootstrap, we implemented the bootstrap approach for variable ranking on the training data to choose the top 10% ranked variables. We then implemented Deep IDA on the training data but with just the selected variables. The learned model and the testing data were used to obtain test classification accuracy. We also implemented Deep CCA, RKCCA, and SVM with the variables that were selected by Deep IDA + Bootstrap to facilitate comparisons. For Deep IDA + TS and Deep CCA + TS, we implemented the teacher-student algorithm to rank the variables. Since in this Scenario, only view 1 had informative features, we expected the classification accuracy from view 1 to be better than the classification accuracy from both views and this is what we observed across most methods. We note that when training the models, we used both views. The classification accuracy from Deep IDA was generally higher than the other methods, except in Setting Three where it was lower than Deep CCA on the whole data (i.e., Deep CCA + TS). We compared the classification accuracy of the proposed method with (i.e., Deep IDA + Bootstrap) and without feature ranking by our method (i.e., Deep IDA + TS) to assess the effect variable selection has on classification estimates from our deep learning models. Deep IDA + Bootstrap had competitive or better classification accuracy (especially when using view 1 only for classification) compared to Deep IDA + TS. Further, the classification accuracy for Deep IDA + Bootstrap was generally higher than the other methods applied to data with variables selected by Deep IDA + Bootstrap (e.g., Deep CCA on top 50 selected features, Setting One). SVM applied on both views stacked together and on just view 1, either using the whole data or using data with variables selected by Deep IDA, resulted in similar classification performance, albeit lower than the proposed method. Thus, in this example, although only view 1 had signal variables, the classification performance from using both views was better than using only view 1 (e.g., SVM on view 1), attesting to the benefit of multi-view analyses. In terms of variable selection, the TS framework applied on Deep IDA and Deep CCA yielded sub-optimal results.

Taken together, the classification and variable selection accuracy from both the linear and nonlinear simulations suggest that the proposed method is capable of ranking the signal variables higher, and is also able to yield competitive or better classification performance, even in situations where the sample size is less than the

number of variables.

4 Real Data Analyses

We consider two real datasets: a) handwriting image data, and b) COVID-19 omics data. The image data will be used to primarily assess the classification performance of the proposed method without feature ranking while the COVID-19 data will be used to assess classification performance and to also demonstrate that Deep IDA is capable of identifying biologically relevant features.

4.1 Evaluation of the Noisy MNIST digits data

The original MNIST handwritten image dataset (LeCun et al., 1998) consists of 70,000 grayscale images of handwritten digits split into training, validation and testing sets of 50,000, 10,000 and 10,000 images, respectively. The validation set was used to select network parameters from the best epoch (lowest validation loss). Each image is 28×28 pixels and has associated with it a label that denotes which digit the image represents (0-9). In Wang et al. (2015), a more complex and challenging noisy version of the original data was generated and used as a second view. First, all pixel values were scaled to 0 and 1. The images were randomly rotated at angles uniformly sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$, and the resulting images were used as view 1. Each rotated image was paired with an image of the same label from the original MNIST data, independent random noise generated from U(0,1) was added, and the pixel values were truncated to [0,1]. The transformed data is view 2. Figure 4 shows two image plots of a digit for views 1 and 2. Of note, view 1 is informative for classification and view 2 is noisy. Therefore, an ideal multi-view classification method should be able to extract the useful information from view 1 while disregarding the noise in view 2.

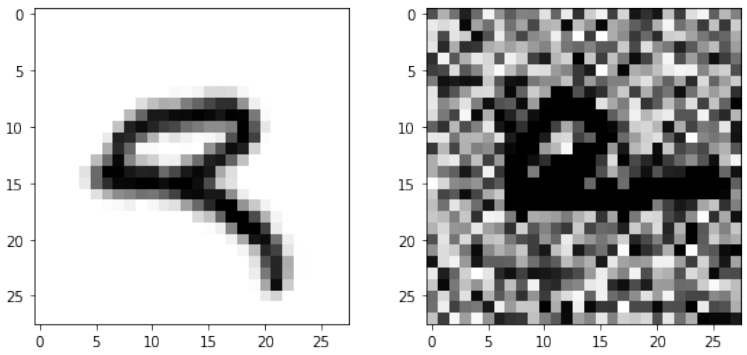

Figure 4: An example of Noisy MNIST data. For the subject with label "9", view 1 observation is on the left and view 2 observation is on the right.

The goal of this application is to evaluate how well the proposed method without feature ranking can classify the digits using the two views. Thus, we applied Deep IDA without feature ranking and the competing methods to the training data and we used the learned models and the testing data to obtain test classification accuracy. The validation data was used in Deep IDA and Deep CCA to choose the best model among all epochs. Table 5 in the supplementary material lists the network structure used in this analysis. Table 4 gives the test classification results of the methods. We observe that the test classification accuracy of the proposed method with nearest centroid classification is better than SVM on the stacked data, and is comparable to Deep CCA. We observe a slight advantage of the proposed method when we implement SVM on the final layer of Deep IDA.

4.2 Evaluation of the COVID-19 Omics Data

4.2.1 Study Design and Goal

The molecular and clinical data we used are described in Overmyer et al. (2021). Briefly, blood samples from 128 patients admitted to the Albany Medical Center, NY from 6 April 2020 to 1 May 2020 for moderate to severe respiratory issues collected. These samples were quantified for metabolomics, RNA-seq, proteomics, and lipidomics data. In addition to the molecular data, various demographic and clinical data were obtained at the time of enrollment. For eligibility, subjects had to be at least 18 years and admitted to the hospital for COVID-19-like symptoms. Of those eligible, 102 had COVID-19, and 26 were without COVID-19. Of those with COVID-19, 51 were admitted to the Intensive Care Unit (ICU) and 51 were not admitted to the ICU (i.e., Non-ICU). Of those without COVID-19, 10 were Non-ICU patients and 16 were ICU patients. Our goal

Table 3: Mean (std.error) accuracy and true positive rates. View 1 data have signal variables with nonlinear relationships. TPR-1; true positive rate for $\mathbf{X}^1$. Deep IDA/Deep CCA/RKCCA view 1 means using the discriminant scores of view 1 only for classification. SVM view 1 uses view 1 data to train and test the model. – indicates not applicable

Method	Mean (%) Accuracy	TPR-1
Setting One		
$(p_1 = 500, p_2 = 500, n_1 = 200, n_2 = 150)$		
Deep IDA + Bootstrap	60.87 (1.28)	100.0
Deep IDA + Bootstrap view 1	81.17 (2.89)	100.0
Deep IDA + TS	62.60 (2.02)	10.20
Deep IDA + TS View 1	81.49 (3.06)	10.20
Deep CCA + TS	58.20(0.59)	8.30
Deep CCA + TS view 1	61.26(0.77)	8.30
Deep CCA on top 50 selected features	58.91(0.82)	-
Deep CCA on top 50 selected features view 1	59.87(0.87)	-
RKCCA	61.06 (0.47)	-
RKCCA View 1	64.93 (0.63)	-
RKCCA on top 50 selected features	56.21 (0.77)	-
RKCCA View 1 on top 50 selected features	58.94 (0.78)	-
SVM	54.20(0.30)	-
SVM view 1	50.26(0.21)	-
SVM on top 50 selected features	50.37(0.27)	-
SVM on top 50 selected features view 1	50.07(0.26)	-
Setting Two		
$(p_1 = 500, p_2 = 500, n_1 = 3,000, n_2 = 2,250)$		
Deep IDA + Bootstrap	89.45(2.16)	63.70
Deep IDA + Bootstrap View 1	90.49(2.25)	63.70
Deep IDA + TS	91.78 (1.73)	10.50
Deep IDA + TS View 1	84.37 (1.49)	10.50
Deep CCA + TS	59.84(0.40)	33.70
Deep CCA + TS view 1	60.35(0.34)	33.70
Deep CCA on top 50 selected features	58.50(0.52)	-
Deep CCA on top 50 selected features view 1	58.32(0.52)	-
RKCCA	57.14 (0.00)	-
RKCCA View 1	57.14 (0.00)	-
RKCCA on top 50 selected features	66.30 (0.96)	-
RKCCA View 1 on top 50 selected features	66.32 (0.98)	-
SVM	54.42(0.11)	-
SVM view 1	52.81(0.13)	
SVM on top 50 selected features	50.56(0.07)	-
SVM on top 50 selected features view 1	50.49(0.04)	-
Setting Three		
$(p_1 = 2000, p_2 = 2000, n_1 = 200, n_2 = 150)$		
Deep IDA + Bootstrap	54.77(0.91)	96.05
Deep IDA + Bootstrap View 1	70.40(2.17)	96.05
Deep IDA + TS	61.67 (1.74)	30.55
Deep IDA + TS View 1	60.73 (1.76)	30.55
Deep CCA + TS	62.27(0.46)	10.28
Deep CCA + TS view 1	70.83(0.36)	10.28
Deep CCA on top 200 selected features	61.43(0.62)	-
Deep CCA on top 200 selected features view 1	68.67(0.83)	-
RKCCA	60.24 (0.63)	-
RKCCA View 1	63.54 (0.57)	-
RKCCA on top 200 selected features	58.70 (0.52)	-
RKCCA View 1 on top 200 selected features	61.90 (0.91)	-
SVM	54.69(0.45)	-
SVM view 1	53.97(0.41)	-
SVM on top 200 selected features	51.14(0.42)	-
SVM on top 200 selected features view 1	50.19(0.35)	-
Setting Four		
$(p_1 = 2000, p_2 = 2000, n_1 = 3,000, n_2 = 2,250)$		
Deep IDA + Bootstrap	69.38(1.44)	83.20
Deep IDA + Bootstrap View 1	71.34(1.66)	83.20
Deep IDA + TS	64.52 (1.32)	10.78
Deep IDA + TS View 1	69.99 (2.88)	10.78
Deep CCA + TS	60.33(0.76)	33.48
Deep CCA + TS view 1	58.54(1.00)	33.48
Deep CCA on top 200 selected features	59.58(1.66)	-
Deep CCA on top 200 selected features view 1	59.28(1.71)	-
RKCCA	57.14 (0.00)	-
RKCCA View 1	57.14 (0.00)	-
RKCCA on top 200 selected features	55.30 (0.79)	-
RKCCA View 1 on top 200 selected features	61.21 (0.81)	-
SVM	53.13(0.16)	-
SVM view 1	53.37(0.13)	-
SVM on top 200 selected features	52.72(0.13)	-
SVM on top 200 selected features view 1	50.59(0.12)	-

Table 4: Noisy MNIST data: SVM was implemented on the stacked data. For Deep CCA + SVM, we trained SVM on the combined outputs (from view 1 and view 2) of the last layer of Deep CCA. For Deep IDA + NCC, we implemented the Nearest Centroid Classification on the combined outputs (from view 1 and view 2) of the last layer of Deep IDA. For Deep IDA + SVM, we trained SVM on the combined outputs (from view 1 and view 2) of the last layer of Deep IDA.

Method	Accuracy (%)
SVM (combined view 1 and 2)	93.75
Deep CCA + SVM	97.01
Deep IDA + NCC	97.74
Deep IDA + SVM	99.15
RKCCA + SVM	91.79

is to elucidate the molecular architecture of COVID-19 severity by identifying molecular signatures that are associated with each other and have potential to discriminate patients with and without COVID-19 who were or were not admitted to the ICU.

4.2.2 Data pre-processing and application of Deep IDA and competing methods

Of the 128 patients, 125 had both omics and clinical data. We focused on proteomics, RNA-seq, and metabolomics data in our analyses since many lipids were not annotated. We formed a four-class classification problem using COVID-19 and ICU status. Our four groups were: with COVID-19 and not admitted to the ICU (COVID Non-ICU), with COVID-19 and admitted to the ICU (COVID ICU), no COVID-19 and admited to the ICU (Non-COVID ICU), and no COVID-19 and not admitted to the ICU (Non-COVID Non-ICU). The frequency distribution of samples in these four groups were: 40% COVID ICU, 40% COVID Non-ICU, 8% Non-COVID Non-ICU, and 12% Non-COVID ICU. The initial dataset contained $18,212$ genes, 517 proteins, and 111 metabolomics features. Prior to applying our method, we pre-processed the data (see Supplementary Material) to obtain a final dataset of $\mathbf{X}^1 \in \Re^{125\times 2,734}$ for the RNA-sequencing data, $\mathbf{X}^2 \in \Re^{125\times 269}$ for the protoemics data, and $\mathbf{X}^3 \in \Re^{125\times 66}$ for the metabolomics.

We randomly split the data into training ($n = 74$) and testing ($n = 51$) sets while keeping the proportions in each group similar to the original data, we applied the methods on the training data and we assessed error rate using the test data. We evaluated Deep IDA with and without feature selection. For Deep IDA with feature selection, we obtained the top 50 and top 10% molecules after implementing Algorithm 1, learned the model on the training data with only the molecules that were selected, and estimated the test error with the testing data. We also assessed the performance of the other methods using variables that were selected by Deep IDA. This allowed us to investigate the importance of feature selection for these methods.

4.2.3 Test Accuracy and Molecules Selected

Table 5 gives the test accuracy for Deep IDA in comparison with deep generalized CCA (Deep GCCA), SIDA, and SVM. Deep IDA on selected features and Deep IDA refer to applying the proposed method with and without variable selection, respectively. Deep IDA with feature selection based on the top 10% variables yield the same test classification accuracy as Deep IDA without feature selection, and these estimates are higher than the test accuracy from the other methods. Further, we observed a slight increase in classification performance when we implemented Deep IDA with feature selection based on the top 50 ranked molecules for each omics. Naively stacking the data and applying support vector machine results in the worse classification accuracy. When we applied SVM on the stacked data obtained from the variables that were selected by Deep IDA, the classification accuracy increased to 62.74% (based on the top 10% features selected by Deep IDA), representing a 13.73% increase from applying SVM on the stacked data obtained from all the variables. We also observed an increase in classification accuracy when we implemented Deep GCCA on the top 10% selected features from Deep IDA compared to Deep GCCA on the whole data. Compared to SIDA, the joint association and classification method that assesses linear relationships in the views and among the groups, the proposed method has a higher test accuracy. These findings suggest that the proposed method is capable of modeling nonlinear relationships among the different views and groups, and has potential to identify features that can lead to better classification results. Figure 5 gives the top 50 genes, proteins, and metabolomics features that were highly-ranked by Deep IDA. Feature importance for each variable was normalized to the feature ranked highest for each omics.

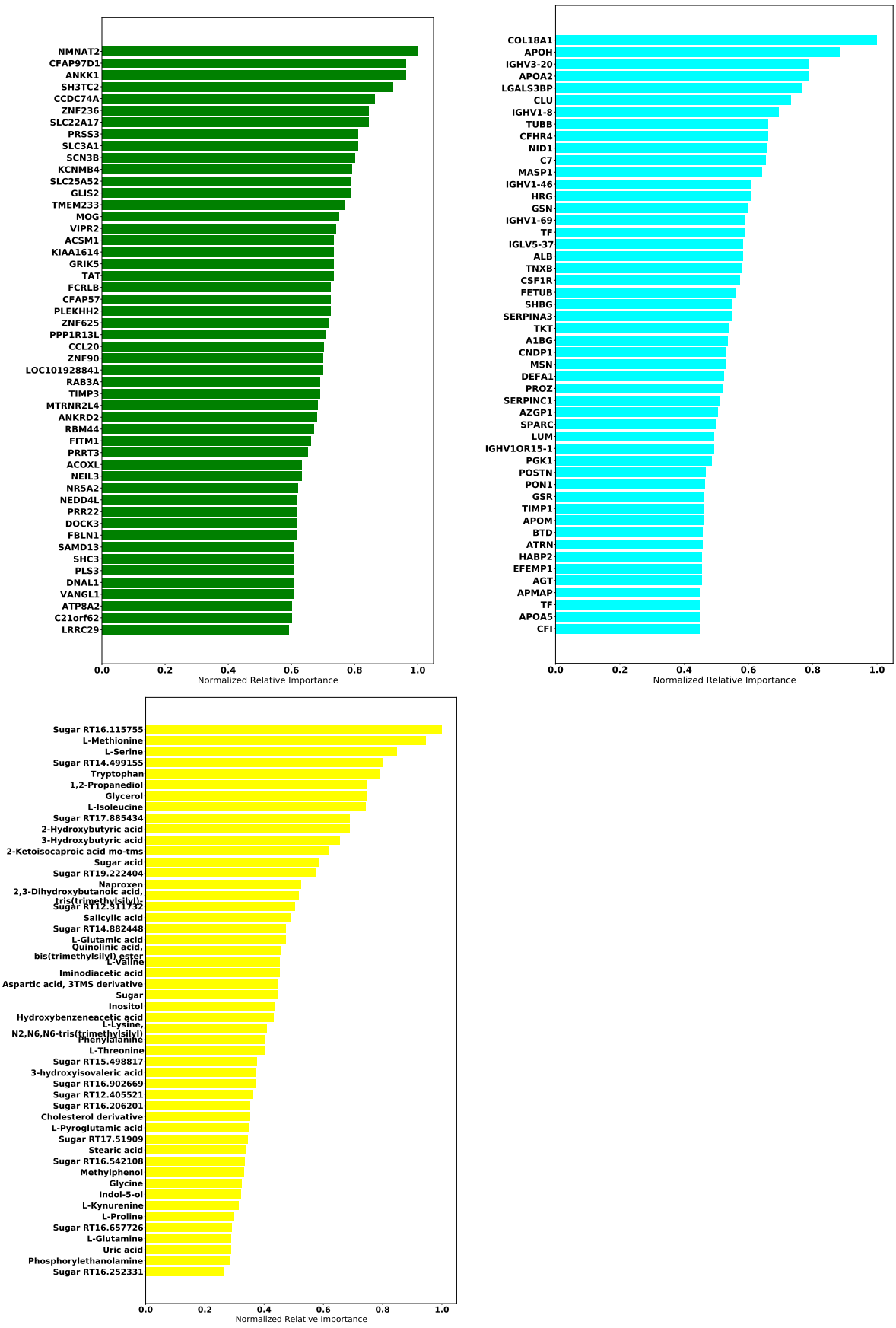

Figure 5: Feature importance plots of the omics data used in the COVID-19 application. Upper left: RNA-Seq; upper right: Proteomics ; lower left: Metabolomics. Feature importance for each variable was normalized to the feature ranked highest for each omics.

Table 5: COVID-19 Omics data: SVM is based on stacked three-view raw data. DeepCCA+SVM is training SVM based on the last layer of DeepCCA. DeepSIDA applies nearest centroid on the last layer for classification. The reported classification accuracy for Deep IDA are based on optimized network structure. DeepIDA + top 10% is based on input-512-20 network structure. DeepIDA + top 50 is based on input-512-256-20 network structure.

Method	Accuracy (%)
SVM	49.01
SVM on selected top 10% features	62.74
SVM on selected top 50 features	64.71
Deep GCCA + SVM	64.71
Deep GCCA + SVM on selected top 10% features	68.63
Deep GCCA + SVM on selected top 50 features	64.71
SIDA	60.78
Deep IDA	76.47
Deep IDA on selected top 10% features	76.47
Deep IDA on selected top 50 features	78.43

4.2.4 Pathway analysis of highly-ranked molecules

We used the Ingenuity Pathway Analysis (IPA) software to investigate the molecular and cellular functions, pathways, and diseases enriched in the proteins, genes, and metabolites that were ranked in the top 50 by our variable selection method. IPA searches the ingenuity pathway knowledge base, which is manually curated from scientific literature and over 30 databases, for gene interaction. We observed strong pathways, molecular and cellular functions, and disease enrichment (Supplementary Tables 1 and 2). The top disease and disorders significantly enriched in our list of genes are found in Supplementary Table 2. We note that 36 of the biomolecules in our gene list were determined to be associated with neurological diseases. This finding aligns with studies that suggest that persons with COVID-19 are likely to have neurological manifestations such as reduced consciousness and stroke Berlit et al. (2020); Taquet et al. (2021). Further, 48 genes from our list were determined to be associated with cancer. Again, this supports studies that suggest that individuals with immuno-compromised system from cancer or individuals who recently recovered from cancer, for instance, are at higher risk for severe outcomes. Compared to the general population, individuals with cancer have a two-fold increased risk of contracting SARS-CoV-2Al-Shamsi et al. (2020). The top 2 networks determined to be enriched in our gene list was "hereditary disorder, neurological disease, organismal injury and abnormalities", and "cell signaling, cellular function and maintenance, and small molecule biochemistry".

As in our gene list, 34 proteins were determined to be associated with neurological disease. Other significantly enriched diseases in our protein list included infectious diseases (such as infection by SARS coronavirus), inflammatory response (such as inflammation of organ), and metabolic disease (including Alzheimer disease and diabetes mellitus). A recent review (Steenblock et al., 2021) found that up to 50% of those who have died from COVID-19 had metabolic and vascular disorders. In particular, patients with metabolic dysfunction (e.g., obesity, and diabetes) have an increased risk for developing severe COVID-19. Further, getting infected with SARS-CoV-2 can likely lead to new onset of diabetes. The top 2 networks determined to be enriched in our protein list was "infectious diseases, cellular compromise, inflammatory response", and "tissue development, cardiovascular disease, hematological disease". The top enriched canonical pathways in our protein list include the LXR/RXR activation FXR/RXR activation, acute phase response signaling, and atherosclerosis signaling (Table 6). These pathways are involved in metabolic processes such as cholesterol metabolism. The molecular and cellular functions enriched in our protein list include cellular movement and lipid metabolism (Supplementary Table 2). Overlapping canonical pathways (Figure 6) in IPA was used to visualize the shared biology in pathways through the common molecules participating in the pathways. The two pathways "FXR/RXR Activation" and "LXR/RXR Activation" share a large number (eight) of molecules: AGT, ALB, APOA2, APOH, APOM, CLU, PON1 and TF. The LXR/RXR pathway is involved in the regulation of lipid metabolism, inflammation, and cholesterol to bile acid catabolism. The farnesoid X receptor (FXR) is a member of the nuclear family of receptors and plays a key role in metabolic pathways and regulating lipid metabolism, cell growth and malignancy (Wang et al., 2008). We observed lower levels of ALB, APOM, and TF in patients with COVID-19 (and more so in patients with COVID-19 who were admitted to the ICU) relative to patients without COVID-19 (Figure 7). We also observed higher levels of AGT and CLU in patients with COVID-19 admitted to the ICU compared to the other groups. The fact that the top enriched pathways, and molecular and cellular functions are involved in metabolic processes such as lipid metabolism seem to corroborate the findings in (Overmyer et al., 2021) that a key signature for COVID-19 is likely a dysregulated lipid transport system.

For the top-ranked metabolomics features, we first used the MetaboAnalyst 5.0 (Pang et al., 2021) software

Table 6: Top 5 Canonical Pathways from Ingenuity Pathway Analysis (IPA).

Omics Data	**Top Canonical Pathway**	**P-value**	**Molecules Selected**
RNA Sequencing	4-hydroxybensoate Biosynthesis	2.07E-03	TAT
	4-hydroxyphneylpyruvate Biosythesis	2.07E-03	TAT
	Tyrosine Degradation 1	1.03E-02	TAT
	Role of IL-17A in Psoriasis	2.86E-02	CCL20
	Fatty Acid Activation	2.86E-02	ACSM1
Proteomics	LXR/RXR Activation	4.14E-11	AGT, ALB, APOA2, APOH, APOM, CLU, PON1, TF
	FXR/RXR Activation	5.02E-11	AGT, ALB, APOA2, APOH, APOM, CLU, PON1, TF
	Acute Phase Response Signaling	3.30E-08	AGT, ALB, APOA2, APOH, HRG, SERPINA3, TF
	Atherosclerosis Signaling	1.06E-07	ALB, APOA2, APOM, CLU, COL18A1, PON1
	Clathrin-mediated Endocytosis Signaling	1.09E-06	ALB, APOA2, APOM, CLU, PON1, TF
Metabolomics	tRNA Charging	2.25E-13	L-glutamic acid, L-Phenylalanine, L-Glutamine, Glycine, L-Serine, L-Methionine; L-Valine, L-Isoleucine, L-Threonine, L-Tryptophan, L-Proline
	Glutamate Receptor Signaling	5.43E-05	L-glutamic acid, glycine, L-Glutamine
	Phenylalanine Degradation IV (Mammalian, via Side Chain)	7.24E-05	L-glutamic acid, glycine, L-Glutamine, L-Phenylalanine
	Superpathway of Serine and Clycine Biosynthesis I	3.28E-04	L-glutamic acid, glycine, L-serine
	y-glutamyl Cycle	4.23E-04	L-glutamic acid, glycine, pyrrolidonecarboxylic acid

to obtain their human metabolome database reference ID and then used IPA on the mapped data for pathway, diseases, molecular and cellular function enrichment analysis. Of the top 50 ranked features, we were able to map 25 features. The top disease and disorders significantly enriched in our list of metabolites (Table 1 of the supplementary material) included cancer and gastrointestinal disease (such as digestive system, and hepatocellular cancer). Molecular and cellular functions enriched included amino acid metabolism, cell cycle, and cellular growth and proliferation. Figure 1 (supplementary material) shows the overlapping pathway network for the metabolites.

Taken together, these findings suggest that COVID-19 disrupts many biological systems. The relationships found with diseases such as cancer, gastrointestinal, neurological conditions, and metabolic diseases (e.g., Alzheimers and diabetes mellitus) heighten the need to study the post sequelae effects of this disease in order to better understand the mechanisms and to develop effective treatments.

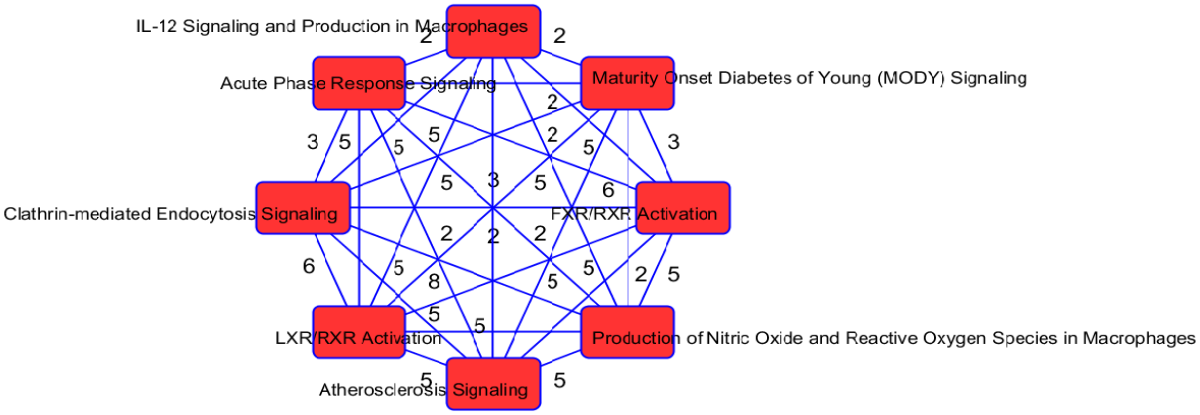

Figure 6: Network of overlapping canonical pathways from highly ranked proteins. Nodes refer to pathways and a line connects any two pathways when there is at least two molecules in common between them.

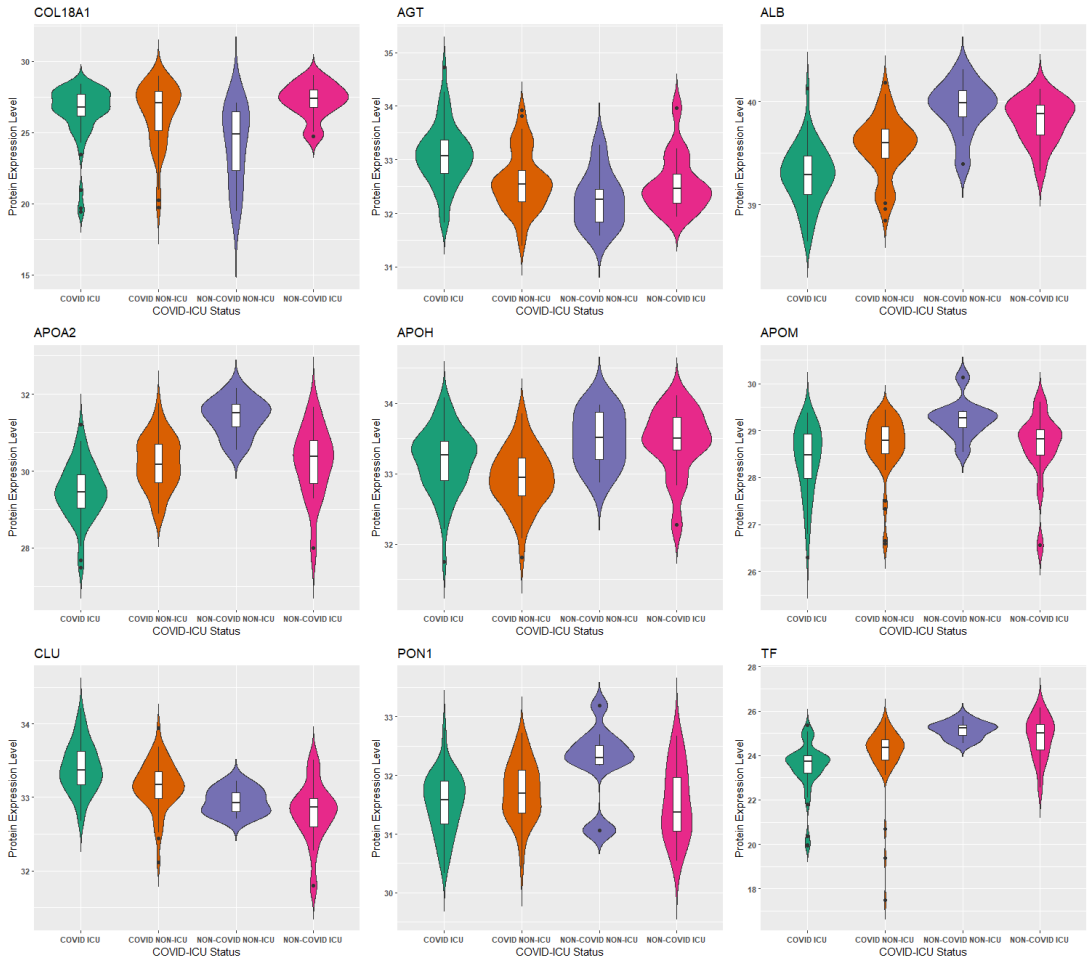

Figure 7: Comparison of protein levels among COVID-19 patient groups (p-value < 0.05, Kruskal-Wallis test). COL18A1 was highly ranked by Deep IDA, and the other 8 proteins are shared by the "FXR/RXR Activation" and "LXR/RXR Activation" pathways. Protein expression levels for ALB, APOM, and TF are lower in patients with COVID-19 (especially in patients with COVID-19 who were admitted to the ICU).

5 Conclusion

We have proposed a deep learning method, Deep IDA, for joint integrative analysis and classification studies of multi-view data. Our framework extends the joint association and classification method proposed in Safo et al. (2021) to model nonlinear relationships among multiple views and between classes in a view. Specifically, we use deep neural networks (DNN) to nonlinearly transform each view and we construct an optimization problem that takes as input the output from our DNN and learns view-specific projections that result in maximum linear correlation of the transformed views and maximum linear separation within each view. Further, unlike most existing nonlinear methods for integrating data from multiple views, we have proposed a feature ranking approach based on resampling to identify features contributing most to the dependency structure among the views and the separation of classes within a view. Our framework for feature ranking is general and applicable to other nonlinear methods for multi-view data integration. The proposed algorithm, developed in Python 3, is user-friendly and will be useful in many data integration applications. Through simulation studies, we showed that the proposed method outperforms several other linear and nonlinear methods for integrating data from multiple views, even in high-dimensional scenarios where the sample size is typically smaller than the number of variables.

When Deep IDA was applied to proteomics, RNA sequencing, and metabolomics data obtained from individuals with and without COVID-19 who were or were not admitted to the ICU, we identified several molecules that better discriminated the COVID-19 patient groups. We also performed enrichment analysis of the molecules that were highly ranked and we observed strong pathways, molecular and cellular functions, and disease enrichment. The top disease and disorders significantly enriched in our list of genes, proteins, and metabolomics data included cancer, neurological disorders, infectious diseases, and metabolic diseases. While some of these findings corroborate earlier results, the top-ranked molecules could be further investigated to delineate their impact on COVID-19 status and severity.

Our work has some limitations. First, the bootstrap technique proposed is computationally tasking. In our algorithm, we use parallelization to mitigate against the computational burden, however, more is needed to make the approach less expensive. Second, the proposed method has focused on binary or categorical outcomes. Future work could consider other outcome types (e.g., continuous and survival). Third, the number (or proportion) of top-ranked features need to be specified in advance. In our proposed bootstrap method, once we have identified the top-ranked variables, we fit another deep learning model to obtain low-dimensional representations of the data that result in maximum association among the views and separation of classes based on the top-ranked variables, and we use these to obtain test classification accuracy if testing data are available. Alternatively, instead of learning a new model with the top-ranked variables, we could consider using the learned neural network parameters from the M bootstrap implementations to construct M $\mathbf{H}_{test}^{d}$s, and then aggregate these (over M) to obtain an estimate of the view-specific top-level representations for classification. Future work could compare this alternative with the current approach.

In conclusion, we have developed a deep learning method to jointly model nonlinear relationships between data from multiple views and a binary or categorical outcome, while also producing highly-ranked features contributing most to the overall association of the views and separation of the classes within a view. The encouraging simulations and real data findings, even for scenarios with small to moderate sample sizes, motivate further applications.

Funding and Acknowledgements

The project described was supported by the Award Numbers 5KL2TR002492-04 from the National Center For Advancing Translational Science and 1R35GM142695-01 from the National Institute Of General Medical Sciences of the National Institutes of Health. The content is solely the responsibility of the authors and does not represent the official views of the National Institutes of Health.

Declaration of Conflicting Interests: The authors declare that there is no conflict of interest.

Data Availability and Software

The data used were obtained from Overmyer et al. (2021). We provide a Python package, *Deep IDA*, to facilitate the use of our method. Its source codes, along with a README file would be made available via `https://github.com/lasandrall/DeepIDA`.

Supplementary Data

In the online Supplementary Materials, we provide proof of Theorems 1 and 2, and also give more results from real data analyses.

References

Akaho, S. (2001), 'A kernel method for canonical correlation analysis', Int'l Meeting on Psychometric Society .

Al-Shamsi, H. O., Alhazzani, W., Alhuraiji, A., Coomes, E. A., Chemaly, R. F., Almuhanna, M., Wolff, R. A., Ibrahim, N. K., Chua, M. L., Hotte, S. J. et al. (2020), 'A practical approach to the management of cancer patients during the novel coronavirus disease 2019 (covid-19) pandemic: an international collaborative group', The oncologist **25**(6), e936.

Andrew, G., Arora, R., Bilmes, J. and Livescu, K. (2013), 'Deep canonical correlation analysis', Journal of Machine Learning Research: Workshop and Conference Proceedings .

Benton, A., Khayrallah, H., Gujral, B., Reisinger, D. A., Zhang, S. and Arora, R. (2019), 'Deep generalized canonical correlation analysis', Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019) p. 1–6.

Berlit, P., Bösel, J., Gahn, G., Isenmann, S., Meuth, S. G., Nolte, C. H., Pawlitzki, M., Rosenow, F., Schoser, B., Thomalla, G. et al. (2020), '"neurological manifestations of covid-19"-guideline of the german society of neurology', Neurological Research and Practice **2**(1), 1–14.

Carroll, D. (1968), 'Generalization of canonical correlation analysis to three or more sets of variables', In Convention of the American Psychological Association pp. 227—-228.

Chang, C.-H., Rampasek, L. and Goldenberg, A. (2017), 'Dropout feature ranking for deep learning models', arXiv preprint arXiv:1712.08645 .

Dorfer, M., Kelz, R. and Widmer, G. (2015), 'Deep linear discriminant analysis', arXiv preprint arXiv:1511.04707 .

Garreau, D., Jitkrittum, W. and Kanagawa, M. (2017), 'Large sample analysis of the median heuristic', arXiv preprint arXiv:1707.07269 .

Hastie, T., Tibshirani, R. and Friedman, J. (2009), The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition), Springer.

Hotelling, H. (1936), 'Relations between two sets of variables', Biometrika pp. 312–377.

Hu, P., Peng, D., Sang, Y. and Xiang, Y. (2019), 'Multi-view linear discriminant analysis network', IEEE Transactions on Image Processing **28**(11), 5352–5365.

Kan, M., Shan, S. and Chen, X. (2016), 'Multi-view deep network for cross-view classification', IEEE Conference on Computer Vision and Pattern Recognition (CVPR) .

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', arXiv preprint arXiv:1412.6980 .

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', Proc. IEEE (11), 2278–2324.

Li, Y., Chen, C.-Y. and Wasserman, W. W. (2016), 'Deep feature selection: theory and application to identify enhancers and promoters', Journal of Computational Biology **23**(5), 322–336.

Lopez-Paz, D., Sra, S., Smola, A., Ghahramani, Z. and Schölkopf, B. (2014), Randomized nonlinear component analysis, in 'International conference on machine learning', PMLR, pp. 1359–1367.

Luo, C., Liu, J., Dey, D. K. and Chen, K. (2016), 'Canonical variate regression', Biostatistics **17**(3), 468–483. **URL:** *+ http://dx.doi.org/10.1093/biostatistics/kxw001*

Maas, A. L., Hannun, A. Y. and Ng, A. Y. (2013), 'Rectifier nonlinearities improve neural network acoustic models', ICML .

Min, E. J., Safo, S. E. and Long, Q. (2019), 'Penalized co-inertia analysis with applications to-omics data', Bioinformatics **35**(6), 1018–1025.

Mirzaei, A., Pourahmadi, V., Soltani, M. and Sheikhzadeh, H. (2019), 'Deep feature selection using a teacher-student network', arXiv:1903.07045 .

Oliphant, T. E. (2006), A guide to NumPy, Vol. 1, Trelgol Publishing USA.

Overmyer, K. A., Shishkova, E., Miller, I. J., Balnis, J., Bernstein, M. N., Peters-Clarke, T. M., Meyer, J. G., Quan, Q., Muehlbauer, L. K., Trujillo, E. A. et al. (2021), 'Large-scale multi-omic analysis of covid-19 severity', Cell systems **12**(1), 23–40.

Pang, Z., Chong, J., Zhou, G., de Lima Morais, D. A., Chang, L., Barrette, M., Gauthier, C., Jacques, P.-t., Li, S. and Xia, J. (2021), 'MetaboAnalyst 5.0: narrowing the gap between raw spectra and functional insights', Nucleic Acids Research **49**(W1), W388–W396.
URL: *https://doi.org/10.1093/nar/gkab382*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, in 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.
URL: *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*

Safo, S. E., Ahn, J., Jeon, Y. and Jung, S. (2018), 'Sparse generalized eigenvalue problem with application to canonical correlation analysis for integrative analysis of methylation and gene expression data', Biometrics **74**(4), 1362–1371.

Safo, S. E., Li, S. and Long, Q. (2018), 'Integrative analysis of transcriptomic and metabolomic data via sparse canonical correlation analysis with incorporation of biological information', Biometrics **74**(1), 300–312.

Safo, S. E., Min, E. J. and Haine, L. (2021), 'Sparse linear discriminant analysis for multiview structured data', Biometrics .
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.13458*

Severe-Covid-19-GWAS-Group (2020), 'Genomewide association study of severe covid-19 with respiratory failure', New England Journal of Medicine **383**(16), 1522–1534.

Steenblock, C., Schwarz, P. E., Ludwig, B., Linkermann, A., Zimmet, P., Kulebyakin, K., Tkachuk, V. A., Markov, A. G., Lehnert, H., de Angelis, M. H. et al. (2021), 'Covid-19 and metabolic disease: mechanisms and clinical management', The Lancet Diabetes & Endocrinology .

Taquet, M., Geddes, J. R., Husain, M., Luciano, S. and Harrison, P. J. (2021), '6-month neurological and psychiatric outcomes in 236 379 survivors of covid-19: a retrospective cohort study using electronic health records', The Lancet Psychiatry **8**(5), 416–427.

Wang, W., Arora, R., Livescu, K. and Bilmes, J. (2015), 'On deep multi-view representation learning', Journal of Machine Learning Research: Workshop and Conference Proceedings .

Wang, Y.-D., Chen, W.-D., Moore, D. D. and Huang, W. (2008), 'Fxr: a metabolic regulator and cell protector', Cell research **18**(11), 1087–1095.

Zhang, Y. and Gaynanova, I. (2021), 'Joint association and classification analysis of multi-view data', Biometrics **n/a**(n/a).
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.13536*

Supplementary Material for "Deep IDA: A Deep Learning Method for Integrative Discriminant Analysis of Multi-View Data with Feature Ranking–An Applicaition to COVID-19 severity"

Jiuzhou Wang, Sandra E. Safo
Division of Biostatistics
University of Minnesota, MN

1 Theorems and Proofs

Theorem 1. *Let $\mathbf{S}_t^d$ and $\mathbf{S}_b^d$ respectively be the total covariance and the between-class covariance for the top-level representations $\mathbf{H}^d, d = 1, \ldots, D$. Let $\mathbf{S}_{dj}$ be the cross-covariance between top-level representations d and j. Assume $\mathbf{S}_t^d \succ 0$. Define $\mathcal{M}^d = \mathbf{S}_t^{d^{-\frac{1}{2}}}\mathbf{S}_b^d\mathbf{S}_t^{d^{-\frac{1}{2}}}$ and $\mathcal{N}_{dj} = \mathbf{S}_t^{d^{-\frac{1}{2}}}\mathbf{S}_{dj}\mathbf{S}_t^{j^{-\frac{1}{2}}}$. Then $\mathbf{\Gamma}^d \in \Re^{o_d \times l}$, $l \leq \min\{K-1, o_1, \ldots, o_D\}$ in equation (4) of main text are eigenvectors corresponding respectively to eigenvalues $\mathbf{\Lambda}_d = diag(\lambda_{d_k}, \ldots, \lambda_{d_l})$, $\lambda_{d_k} > \cdots > \lambda_{d_l} > 0$ that iteratively solve the eigensystem problems:*

$$\left(c_1\mathcal{M}^d + c_2\sum_{j\neq d}^{D}\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T\right)\mathbf{\Gamma}_d = \mathbf{\Lambda}_d\mathbf{\Gamma}_d, \forall d = 1, ..., D$$

where $c_1 = \frac{\rho}{D}$ and $c_2 = \frac{2(1-\rho)}{D(D-1)}$.

Prove. Solving the optimization problem is equivalent to iteratively solving the following generalized eigenvalue systems:

$$\left(c_1\mathcal{M}^1 + c_2\sum_{j=2}^{D}\mathcal{N}_{1j}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{1j}^T\right)\mathbf{\Gamma}_1 = \mathbf{\Lambda}_1\mathbf{\Gamma}_1$$

$$\vdots$$

$$\left(c_1\mathcal{M}^d + c_2\sum_{j=1,j\neq d}^{D}\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T\right)\mathbf{\Gamma}_d = \mathbf{\Lambda}_d\mathbf{\Gamma}_d$$

$$\vdots$$

$$\left(c_1\mathcal{M}^D + c_2\sum_{j=1}^{D-1}\mathcal{N}_{Dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{Dj}^T\right)\mathbf{\Gamma}_D = \mathbf{\Lambda}_D\mathbf{\Gamma}_D$$

where $c1 = \frac{\rho}{D}$ and $c2 = \frac{2(1-\rho)}{D(D-1)}$.

Proof. The Lagrangian is

$$\begin{aligned} L(\mathbf{\Gamma}_1, ..., \mathbf{\Gamma}_D, \lambda_1, ..., \lambda_D) &= \rho\frac{1}{D}\sum_{d=1}^{D}tr[\mathbf{\Gamma}_d^T\mathcal{M}^d\mathbf{\Gamma}_d] + (1-\rho)\frac{2}{D(D-1)}\sum_{d=1}^{D}\sum_{j,j\neq d}^{D}tr[\mathbf{\Gamma}_d^T\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T\mathbf{\Gamma}_d] - \sum_{d=1}^{D}\eta_d(tr[\mathbf{\Gamma}_d^T\mathbf{\Gamma}_d] - l) \\ &= c_1\sum_{d=1}^{D}tr[\mathbf{\Gamma}_d^T\mathcal{M}^d\mathbf{\Gamma}_d] + c_2\sum_{d=1}^{D}\sum_{j,j\neq d}^{D}tr[\mathbf{\Gamma}_d^T\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T\mathbf{\Gamma}_d] - \sum_{d=1}^{D}\lambda_d(tr[\mathbf{\Gamma}_d^T\mathbf{\Gamma}_d] - l) \end{aligned} \quad (1)$$

The first order stationary solution for $\mathbf{\Gamma}_d(\forall d = 1, ..., D)$ is

$$\frac{\partial L(\mathbf{\Gamma}_1, ..., \mathbf{\Gamma}_D, \lambda_1, ..., \lambda_D)}{\partial \mathbf{\Gamma}_d^T} = 2c_1\mathcal{M}^d\mathbf{\Gamma}_d + 2c_2\sum_{j,j\neq d}^{D}(\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T)\mathbf{\Gamma}_d - 2\lambda_d\mathbf{\Gamma}_d = \mathbf{0} \quad (2)$$

Rearranging the equation 2 we have

$$\left(c_1\mathcal{M}^d + c_2\sum_{j,j\neq d}^{D}\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T\right)\mathbf{\Gamma}_d = \lambda_d\mathbf{\Gamma}_d$$

For $\mathbf{\Gamma}_j, \forall j \neq d$ fixed, the above can be solved for the eigenvalues of $(c_1\mathcal{M}^d + c_2\sum_{j,j\neq d}^{D}\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T)$. Arrange the eigenvalues from large to small and denote $\mathbf{\Lambda}_d \in \mathbf{R}^{o_d\times o_d}$ as the diagonal matrix of those values. For the top l largest eigenvalues, denote the corresponding eigenvectors as $\widehat{\mathbf{\Gamma}}_d = [\gamma_{d,1}, ..., \gamma_{d,l}]$. Therefore, starting from $d = 1$, following this process, $\widehat{\mathbf{\Gamma}}_1$ is updated; then, update $\widehat{\mathbf{\Gamma}}_2$ and so on; finally, update $\widehat{\mathbf{\Gamma}}_D$. We iterate until convergence, which is defined as, $\frac{\|\widehat{\mathbf{\Gamma}}_{d,new} - \widehat{\mathbf{\Gamma}}_{d,old}\|_F^2}{\|\widehat{\mathbf{\Gamma}}_{d,old}\|_F^2} < \epsilon$. When convergence is achieved, set $\widetilde{\mathbf{\Gamma}}_d = \widehat{\mathbf{\Gamma}}_d, \forall d = 1, ..., D$. ■

Theorem 2. *For d fixed, let $\eta_{d,1}, \ldots, \eta_{d,l}$, $l \leq \min\{K-1, o_1, \ldots, o_D\}$ be the largest l eigenvalues of $c_1\mathcal{M}^d + c_2\sum_{j\neq d}^{D}\mathcal{N}_{dj}\mathbf{\Gamma}_j\mathbf{\Gamma}_j^T\mathcal{N}_{dj}^T$. Then the solution $\widetilde{f}^d$ to the optimization problem in equation (5) [main text] for view d maximizes*

$$\sum_{r=1}^{l}\eta_{d,r}. \tag{3}$$

Proof. Fix d and let $\eta_{d,1}, \eta_{d,2}, ..., \eta_{d,l}$ be the top l eigenvalues of

$$c_1\mathcal{M}^d + c_2\sum_{j\neq d}^{D}\mathcal{N}_{dj}\widetilde{\mathbf{\Gamma}}_j\widetilde{\mathbf{\Gamma}}_j^T\mathcal{N}_{dj}^T.$$

Then,

$$\sum_{r=1}^{l}\eta_{d,r} = c_1 tr[\widetilde{\mathbf{\Gamma}}_d^T\mathcal{M}^d\widetilde{\mathbf{\Gamma}}_d] + c_2\sum_{j,j\neq d}^{D} tr[\widetilde{\mathbf{\Gamma}}_d^T\mathcal{N}_{dj}\widetilde{\mathbf{\Gamma}}_j\widetilde{\mathbf{\Gamma}}_j^T\mathcal{N}_{dj}^T\widetilde{\mathbf{\Gamma}}_d]$$

Proof.

$$\begin{aligned}
&c_1 tr[\widetilde{\mathbf{\Gamma}}_d^T\mathcal{M}^d\widetilde{\mathbf{\Gamma}}_d] + c_2\sum_{j,j\neq d}^{D} tr[\widetilde{\mathbf{\Gamma}}_d^T\mathcal{N}_{dj}\widetilde{\mathbf{\Gamma}}_j\widetilde{\mathbf{\Gamma}}_j^T\mathcal{N}_{dj}^T\widetilde{\mathbf{\Gamma}}_d] \\
&= tr(\widetilde{\mathbf{\Gamma}}_d^T(c_1\mathcal{M}^d + c_2\sum_{j,j\neq d}^{D}\mathcal{N}_{dj}\widetilde{\mathbf{\Gamma}}_j\widetilde{\mathbf{\Gamma}}_j^T\mathcal{N}_{dj}^T)\widetilde{\mathbf{\Gamma}}_d) \\
&= tr(\widetilde{\mathbf{\Gamma}}_d^T\mathbf{\Lambda}_d\widetilde{\mathbf{\Gamma}}_d) \\
&= \sum_{r=1}^{l}\eta_{d,r}
\end{aligned}$$

■

2 More Results From Real Data Analysis

2.1 Data preprocessing and application of Deep IDA and competing methods

Of the 128 patients, 125 had both omics and clinical data. We focused on proteomics, RNA-seq, and metabolomics data in our analyses since many lipids were not annotated. We formed a four-class classification problem using COVID-19 and ICU status. Our four groups were: with COVID-19 and not admitted to the ICU (COVID Non-ICU), with COVID-19 and admitted to the ICU (COVID ICU), no COVID-19 and admited to the ICU (Non-COVID ICU), and no COVID-19 and not admitted to the ICU (Non-COVID Non-ICU). The frequency distribution of samples in these four groups were: 40% COVID ICU, 40% COVID Non-ICU, 8% Non-COVID Non-ICU, and 12% Non-COVID ICU. The initial dataset contains 18,212 genes, 517 proteins, and 111 metabolomics features. Prior to applying our method, we pre-processed the data as follows. All genes which were missing in our samples were removed from the dataset and 15,106 genes remained. We selected genes that more than half of their variables are non-zero, and we applied box-cox transformation on each gene as the gene data were highly

skewed. The transformed data were standardized to have mean zero and variance one. We kept genes with variance less than the 25th percentile. We then used ANOVA on the standardized data to filter out (p-values > 0.05) genes with low potential to discriminate among the four groups. For the proteomics and metabolomics data, we standardized each molecule to have mean zero and variance one, pre-screened with ANOVA and filtered out molecules with p-values > 0.05. Our final data were $\mathbf{X}^1 \in \Re^{125\times 2,734}$ for the gene data, $\mathbf{X}^2 \in \Re^{125\times 269}$ for the protoemics data, and $\mathbf{X}^3 \in \Re^{125\times 66}$ for the metabolomics data.

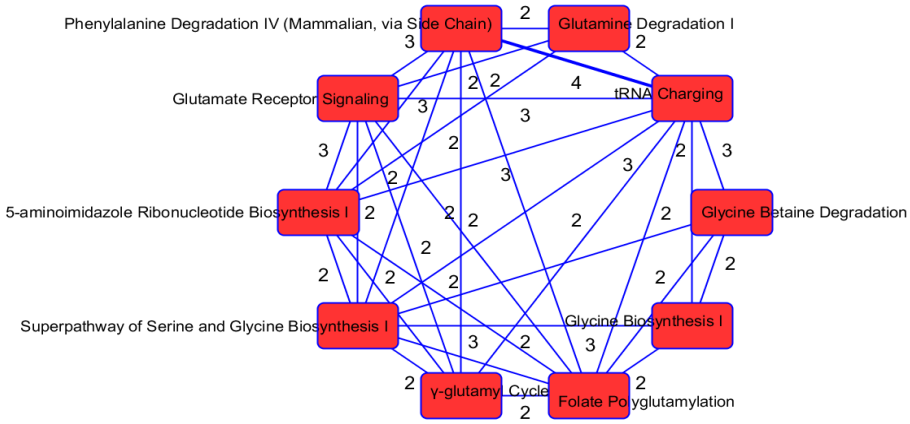

Figure 1: Network of overlapping canonical pathways from highly ranked metabolites. Nodes refer to pathways and a line connects any two pathways when there is at least two molecules in common between them.

Table 1: Top Diseases and Biological Functions from Ingenuity Pathway Analysis (IPA).

	Top Diseases and Bio Functions	P-value range	Molecules Selected
RNA Sequencing	Cancer (such as non-melanoma solid tumor, head and neck tumor)	4.96E-02 – 2.74E-05	48
	Organismal Injury and Abnormalities	4.96E-02 – 2.74E-05	48
	Neurological Disease (such as glioma cancer, brain lesion, neurological deficiency)	4.86E-02 – 5.84E-05	36
	Developmental Disorder (such as intellectual diability with ataxia)	4.46E-02 – 3.76E-05	16
	Hereditary Disorder (such as familial midline effect)	4.86E-02 – 2.02E-05	16
Proteomics	Infectious Diseases (such as Severe COVID-19, COVID-19, infection by SARS coronavirus)	1.75E-03 – 8.31E-13	19
	Inflammatory Response (such as inflammation of organ, degranulation of blood platelets)	1.29E-03 – 1.34E-12	32
	Metabolic Disease (such as amyloidosis, Alzheimer disease, diabetes mellitus)	1.47E-03 – 2.48E-12	20
	Organismal Injury and Abnormalities (such as amyloidosis, tauopathy)	1.72E-03 – 2.48E-12	39
	Neurological Disease (such as tauopathy, progressive encephalopathy, progressive neurological disorder)	1.57E-03 – 3.41E-11	34
Metabolomics	Cancer	3.63E-02 – 5.20E-14	18
	Gastrointestinal Disease (such as digestive system cancer, hepatocellular carcinoma)	3.64E-02 – 5.20E-14	20
	Organismal Injury and Abnormalities (such as digestive system cancer, abdominal cancer)	3.79E-02 – 5.20E-14	22
	Hepatic System Disease (such as hepatocellular carcinoma, liver lesion)	2.91E-02 – 1.66E-11	15
	Developmental Disorder (such as mucopolysaccharidosis type I, spina bifida)	2.44E-02 – 1.83E-09	11

Table 2: Top Molecular and Cellular Functions Functions from Ingenuity Pathway Analysis (IPA).

	Molecular and Cellular Functions	P-value range	Molecules Selected
RNA Sequencing	Cell Death and Survival	4.46E-02 – 2.00E-03	8
	Amino Acid Metabolism	3.47E-02 – 2.07E-05	2
	Cell-to-cell Signaling and Interaction	4.86E-02 – 2.07E-03	10
	Cellular Assembly and Organization	4.46E-02 – 2.07E-03	9
	Cellular Function and Maintenance	4.86E-02 – 2.07E-03	10
Proteomics	Cellular Compromise	1.29E-03 – 1.34E-12	13
	Cellular Movement	1.65E-03 – 2.19E-09	24
	Lipid Metabolism	1.28E-03 – 2.95E-09	15
	Molecular Transport	1.28E-03 – 2.95E-09	15
	Small molecule Biochemistry	1.61E-03 – 2.95E-09	19
Metabolomics	Amino Acid Metabolism	3.64E-02 – 3.99E-08	9
	Molecular Transport	3.82E-02 – 3.99E-08	17
	Small Molecule Biochemistry	3.64E-02 – 3.99E-08	18
	Cellular Growth and Proliferation	3.79E-02 – 5.12E-08	16
	Cell Cycle	3.63E-02 – 5.81E-07	10

Table 3: **Linear Simulations** Network structures for all deep learning based methods. In order to make fair comparisons, for each dataset, the network structure for Deep CCA/Deep GCCA is the same as the proposed Deep IDA method. The activation function is Leakly Relu with parameter 0.1 by default. After activation, batch normalization is also implemented. − indicates not applicable

Data	Sample size (Train,Valid,Test)	Feature size (p^1, p^2, p^3)	Method	Network structure	Epochs per run	Batch size
Setting 1	540,540,1080	1000,1000,-	Deep IDA (+Bootstrap)	Input-512-256-64-10	50	540
Setting 1	540,540,1080	1000,1000,-	Deep CCA	Input-512-256-64-10	50	180
Setting 2	540,540,1080	1000,1000,1000	Deep IDA (+Bootstrap)	Input-512-256-20	50	540
Setting 2	540,540,1080	1000,1000,1000	Deep GCCA	Input-512-256-64-20	200	540

Table 4: **Nonlinear Simulations** Network structures for all deep learning based methods. In order to make fair comparisons, for each dataset, the network structure for Deep CCA/Deep GCCA is the same as the proposed Deep IDA method. The activation function is Leakly Relu with parameter 0.1 by default. After activation, batch normalization is also implemented.

Data	Sample size (Train,Valid,Test)	Feature size (p^1, p^2, p^3)	Method	Network structure	Epochs per run	Batch size
Setting 1	350,350,350	500,500	Deep IDA (+Bootstrap)	Input-256*10-64-20	50	350
Setting 1	350,350,350	500,500	Deep CCA	Input-256*10-64-20	50	350
Setting 3b	5250,5250,5250	500,500	Deep IDA (+Bootstrap)	Input-256-256-256-256-256-256-64-20	50	500
Setting 3b	5250,5250,5250	500,500	Deep CCA	Input-256-256-256-256-256-256-64-20	50	500
Setting 4	350,350,350	2000,2000	Deep IDA (+Bootstrap)	input-256-256-256-256-256-256-256-64-20	50	350
Setting 4	350,350,350	2000,2000	Deep CCA	input-256-256-256-256-256-256-256-64-20	50	350
Setting 5b	5250,5250,5250	2000,2000	Deep IDA (+Bootstrap)	Input-256-256-256-256-256-256-64-20	50	500
Setting 5b	5250,5250,5250	2000,2000	Deep CCA	Input-256-256-256-256-256-256-64-20	50	500

Table 5: **Real Data Analysis** Network structures for all deep learning based methods. In order to make fair comparisons, for each dataset, the network structure for Deep CCA/Deep GCCA is the same as the proposed Deep IDA method. The activation function is Leakly Relu with parameter 0.1 by default. After activation, batch normalization is also implemented. For Covid-19 data, we select the top 50 features for each view from Bootstrap Deep IDA with input-512-20. − indicates not applicable

Data	Sample size (Train,Valid,Test)	Feature size (p^1, p^2, p^3)	Method	Network structure	Epochs per run	Batch size
Noisy MNIST	50000,10000, 10000	784,784,-	Deep CCA non-Bootstrap Deep IDA	Input-512-256-64-20	50	50000
Covid-19	74,0,21	2734,269,66	non-Bootstrap Deep IDA	Input-512-20	20	74
Covid-19	74,0,21	2734,269,66	Deep IDA on selected top 50 features	Input-512-256-20	20	74
Covid-19	74,0,21	2734,269,66	Deep IDA on selected top 10 percent features	Input-256-64-20	20	74
Covid-19	74,0,21	2734,269,66	Deep GCCA on selected top 50 features	Input-256-20	150	74