1 Details on Variational IB Loss

1.1 Derivation of Variational IB Loss in (1) and (2)

Following \cite{1}, the variational information bottleneck (IB) loss for the joint latent representation \(Z\) can be given as

\[
L_{\theta, \psi}^{IB-J}(\bar{X}, Y) = -I(Z; Y) + \beta I(\bar{X}; Z) \tag{S.1}
\]

where \(\beta \geq 0\) is a coefficient chosen to balance between the two information quantities. Here, \(I(Z; Y)\) and \(I(\bar{X}; Z)\) can be derived using variational approximations, i.e., \(q_\theta(Z|\bar{X})\) and \(q_\psi(Y|Z)\), as follows:

\[
I(Z; Y) = \int dzdy p(z, y) \log \frac{p(y|z)}{p(y)} = \int d\bar{x}dzdy p(\bar{x}, y, z) \log p(y|z) + H(Y)
\]

\[
= \int d\bar{x}dzdy p(\bar{x})p(y|\bar{x})p(z|\bar{x}) \log p(y|z) + H(Y)
\]

\[
\approx \int d\bar{x}dzdy p(\bar{x})p(y|\bar{x})q_\theta(z|y) \log q_\psi(y|z) + H(Y)
\]

\[
= \mathbb{E}_{\bar{x}, y \sim p(\bar{x}, y)} \mathbb{E}_{z \sim q_\theta(z|\bar{x})} \left[ \log q_\psi(y|z) \right] + H(Y)
\]

and

\[
I(\bar{X}; Z) = \int d\bar{x}dz p(\bar{x}, z) \log \frac{p(z|\bar{x})}{p(z)} = \int d\bar{x}dz p(\bar{x})p(z|\bar{x}) \log \frac{p(z|\bar{x})}{p(z)}
\]

\[
\approx \int d\bar{x}dz p(\bar{x})q_\theta(z|\bar{x}) \log \frac{q_\theta(z|\bar{x})}{q(z)}
\]

\[
= \mathbb{E}_{\bar{x} \sim p(\bar{x})} \left[ KL(q_\theta(Z|\bar{x}) || q(Z)) \right]
\]

where \(KL(q_\theta(Z|\bar{x}) || q(Z))\) denotes the Kullback-Leibler (KL) divergence between the two distributions \(q_\theta(Z|\bar{x})\) and \(q(Z)\). Since the entropy of the labels \(H(Y)\) is independent of our optimization procedure, we can simply ignore it and approximate the IB loss by plugging (S.2) and (S.3) into (S.1):\[
L_{\theta, \psi}^{IB-J}(\bar{X}, Y) \approx \mathbb{E}_{\bar{x}, y \sim p(\bar{x}, y)} \mathbb{E}_{z \sim q_\theta(z|\bar{x})} \left[ -\log q_\psi(y|z) \right] + \beta \mathbb{E}_{\bar{x} \sim p(\bar{x})} \left[ KL(q_\theta(Z|\bar{x}) || q(Z)) \right]
\]

We can similarly derive the IB loss for the marginal representation \(Z_v\) for \(v \in \mathcal{V}\).
1.2 Training DeepIMV via IB Losses

Different network components are trained based on the joint and marginal IB losses, respectively. More specifically, the parameters of the view-specific encoders and the multi-view predictor – i.e., \((\theta, \psi)\) – are updated based on the joint IB loss while those of the view-specific encoders and the view-specific predictors – i.e., \((\theta, \phi)\) – are updated based on the marginal IB losses. Figure 1 depicts the network components trained via the joint and the marginal IB losses.

2 Implementation Details

Among the 4 network components of DeepIMV, we use multi-layer perceptrons (MLP) as the baseline architecture for the view-specific encoders, view-specific predictors, and multi-view predictor. (Note that the PoE module does not have parameters to be trained.) The number of hidden units and layers in each component are optimized by cross-validation, and we choose the ones with the minimum total loss in (4), i.e., \(L_{\theta, \psi}^{IB-J}\), on the validation set. The number of hidden units is selected among \{50, 100, 300\} and the number of layers is selected among \{1, 2, 3\}. We choose the dimension of latent representations among \{50, 100\} and use ReLu as the activation function at each hidden layer. The parameters \((\theta, \phi, \psi)\) are initialized by Xavier initialization \[2\] and optimized via Adam optimizer \[3\] with learning rate of \(1 \times 10^{-4}\). We utilize dropout \[4\] with dropout probability 0.7 to regularize the network. The network is further regularized via \(L_1\)-regularization with \(1 \times 10^{-4}\) for the CCLE dataset and no additional regularization is used for the TCGA dataset.

Figure 1: An illustration of the proposed network architecture with \(V = 3\) views. For an illustration purpose, we assume that the current sample has the second view missing, i.e., \(x_2 = \emptyset\).
For the balancing coefficients, we use $\alpha = 1.0$ and $\beta = 0.01$ where we assume that $\beta_v = \beta$ for $v \in \{V\}$ for convenience. Please refer to our sensitivity analysis in Section S.5.2 for the selection of balancing coefficients $\alpha$ and $\beta$.

3 Details of the Benchmarks

We compare DeepIMV with 2 baseline methods (i.e., Base1 and Base2) and 6 state-of-the-art multi-view learning methods (i.e., GCCA [5], DCCA [6], DCCAE [7], MVAE [8], CPM-Nets [9], and MOFA [10]).

For the baseline methods (i.e., Base1 and Base2), we directly use the observations from multiple views and the corresponding labels for training a MLP. For unsupervised multi-view learning methods (i.e., GCCA, DCCA, DCCAE, MVAE, and MOFA), we use a two-step approach to provide predictions on the target labels: First, we train each method to find the representations of multi-view observations in a common (latent) space. Then, we use the learned representations and the corresponding labels to train a MLP as a downstream task. For the network architecture of MLPs for the downstream task in each benchmark, we use ReLU as the activation function at each hidden layer and use dropout with dropout probability 0.7 to regularize the network. The details of the benchmarks are described as follows:

- **Base1**: To handle multi-view observations, we concatenate features from multiple views as a pre-integration step and train a baseline network with the concatenated features as an input. The number of hidden units is selected among $\{50V, 100V, 300V\}$ and the number of layer is selected among $\{1, 2, 3\}$.

- **Base2**: We separately train a MLP for each individual view, and then make an ensemble by averaging the predictions from observed views as a post-integration step. We use a MLP for each view where the number of hidden units is selected among $\{50, 100, 300\}$ and the number of layer is selected among $\{1, 2, 3\}$.

- **GCCA [5]**: GCCA generalizes the CCA framework when there are more than two views. To provide predictions on the target label, we first train GCCA to find the representations in a common space, and then train a MLP based on the concatenated representations. The dimension of the common space is selected among $\{50, 100\}$ which provides the best prediction performance on the validation set. For the downstream task, we use a MLP where the number of hidden units is selected among $\{50, 100, 300\}$ and the number of layer is selected among $\{1, 2, 3\}$.

- **DCCA [6]** and **DCCAE [7]**: DCCA extracts low-dimensional representations for observations from two views in a common space by training neural networks to maximize the canonical correlation between the extracted representations. Similarly, DCCAE extracts low-dimensional representations by training auto-encoders to optimize a combination of reconstruction loss and the canonical correlation. To make predictions on the target task, we first train each method to find latent representations in a latent space, and then train a baseline network based on the concatenated representations. For implementation, we utilize MLPs for DCCA and DCCAE. And, we select the number of hidden units among $\{50, 100, 300\}$ and the number of layers among $\{1, 2, 3\}$ based on the validation loss. We use ReLU as the activation function at each hidden layer. The dimension of the latent representation among $\{50, 100\}$ is selected based on the prediction performance on the validation set.

It is worth highlighting that we select the two best performing views when the available views are more than two, i.e., $V > 2$, since DCCA and DCCAE can utilize only two. For the downstream task, we use a MLP where the number of hidden units is selected among $\{50, 100, 300\}$ and the number of layer is selected among $\{1, 2, 3\}$.

- **MVAE [8]**: MVAE learns latent representations for incomplete multi-view observations that can generate the original views under the VAE framework. We modify the publicly available code since it only supports observations from two views. We implement the VAE components using MLPs where the number of hidden units, the number of layers, and the dimension of the latent representations are selected among $\{50, 100, 300\}$, $\{1, 2, 3\}$, and $\{50, 100\}$, respectively, based on its validation loss. We use ReLU as the activation function at each hidden layer. To make predictions on the target task, we first train MVAE to find latent representations

---

*https://github.com/rupy/GCCA*

*https://ttic.uchicago.edu/~wwang5/dccae.html*

*https://github.com/mhw32/multimodal-vae-public*
of incomplete multi-view observations in the latent space, and then train a MLP based on the learned representations. For the downstream task, the number of hidden units is selected among \{50, 100, 300\} and the number of layer is selected among \{1, 2, 3\}.

- **CPM-Nets**: CPM-Nets learns representations in a common space to provide predictions on the target classification task based on the incomplete multi-view observations. We implement each component of CPM-Nets using MLPs where the number of hidden units, the number of layers, and the dimension of the latent representations are selected among \{50, 100, 300\}, \{1, 2, 3\}, and \{50, 100\}, respectively, based on the prediction performance on the validation set. We use ReLu as the activation function at each hidden layer.

- **MOFA**: MOFA infers a low-dimensional representation of the data in terms of a small number of (latent) factors that capture the joint aspects across different views. For training MOFA, we used the original views (i.e., views without conducting kernel PCA) as it is well-known for capturing sparse factors across multiple views. We set the initial number of factors as 50. For the downstream task, the number of hidden units is selected among \{50, 100, 300\} and the number of layer is selected among \{1, 2, 3\}.

It is worth highlighting that among the benchmarks, MVAE, CPM-Nets, and MOFA can flexibly handle incomplete multi-view observations during training. Hence, for training the other benchmarks (except for Base2), we use mean imputation for missing views. For Base2, we train the baseline network for each individual view using samples that have observations for the corresponding view.

4 Obtaining Multi-Omics Datasets

4.1 TCGA Dataset

For constructing multiple views and the labels, the following datasets were downloaded from [http://gdac.broadinstitute.org](http://gdac.broadinstitute.org):

- DNA methylation (epigenomics): *Methylation_Preprocess.Level3.2016012800.0.0.tar.gz*
- microRNA expression (transcriptomics): *miRseq_Preprocess.Level3.2016012800.0.0.tar.gz*
- mRNA expression (transcriptomics): *mRNAseq_Preprocess.Level3.2016012800.0.0.tar.gz*
- RPPA (proteomics): *RPPA_AnnotateWithGene.Level3.2016012800.0.0.tar.gz*
- clinical labels: *Clinical_Pick_Tier1.Level4.2016012800.0.0.tar.gz*

Time to death or censoring in clinical labels was converted to a binary label for 1-year mortality.

4.2 CCLE Dataset

For constructing multiple views and the labels, the following datasets were downloaded from [https://portals.broadinstitute.org/ccle/data](https://portals.broadinstitute.org/ccle/data):

- DNA copy number (genomics): *CCLE_copynumber_byGene_2013-12-03.txt*
- DNA methylation (epigenomics): *CCLE_RRBS_enh_CpG_clusters_20181119.txt*
- microRNA expression (transcriptomics): *CCLE_miRNA_20181103.gct.txt*
- mRNA expression (transcriptomics): *CCLE_RNASEq_genes_counts_20180929.gct*
- RPPA (proteomics): *CCLE_RPPA_20181003.csv*
- metabolites (Metabolomics): *CCLE_metabolomics_20190502.csv*
In this section, we provide sensitivity analysis using the TCGA dataset to see effects of $\alpha$ and $\beta$ on the prediction performance of DeepIMV. Figure 2 shows the AUROC performance of our method with respect to different values of $\alpha$ and $\beta$, respectively. For training the variants of the proposed method, we used both complete-view and incomplete-view samples.
The Effect of $\alpha$. As shown in Figure 2a, the discriminative performance drops at $\alpha = 10.0$ since too high value of $\alpha$ makes DeepIMV to focus on view-specific aspects which may end up with sacrificing joint aspects of observed views for predicting the target. Contrarily, too small value of $\alpha$ makes the learning of task-relevant information from the observed views difficult since each marginal representations do not capture the important information for predicting the target from the corresponding view.

The Effect of $\beta$. Similar to the findings via extensive experiments in [1], $\beta$, which balances between having a representation that is concise and one that provides good prediction power, plays an important role in DeepIMV. As shown in Figure 2b, the classification performance drops at $\beta = 0.1$ since too high value of $\beta$ blocks information from the input that is required to provide good predictions on the target task. For small values of $\beta$, we witness DeepIMV becomes overfitted since the view-specific encoder block learns to be more deterministic and thereby reducing the benefits of regularization. (Note that, in such cases, early-stopping is used to prevent from overfitting.)

Throughout our experiments, we set $(\alpha, \beta) = (1.0, 0.01)$ for the TCGA dataset and $(\alpha, \beta) = (0.1, 0.01)$ for the CCLE dataset.
References


