Supplementary Material for A Variational Information Bottleneck Approach to Multi-Omics Data Integration

Changhee Lee
University of California, Los Angeles

Mihaela van der Schaar
University of Cambridge
University of California, Los Angeles
Alan Turing Institute

1 Details on Variational IB Loss

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

Following [1], the variational information bottleneck (IB) loss for the joint latent representation Z can be given as

$$\mathcal{L}^{\theta,\psi}_{\text{IB-J}}(\bar{\mathbf{X}}, Y) = -I(Z;Y) + \beta I(\bar{\mathbf{X}}; Z) \tag{S.1}$$

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

$$\begin{aligned} I(Z;Y) = \int dzdy\ p(z,y) \log \frac{p(y|z)}{p(y)} &= \int d\bar{\mathbf{x}}dzdy\ p(\bar{\mathbf{x}},y,z) \log p(y|z) + H(Y) \\ &= \int d\bar{\mathbf{x}}dzdy\ p(\bar{\mathbf{x}})p(y|\bar{\mathbf{x}})p(z|\bar{\mathbf{x}}) \log p(y|z) + H(Y) \\ &\approx \int d\bar{\mathbf{x}}dzdy\ p(\bar{\mathbf{x}})p(y|\bar{\mathbf{x}})q_\theta(z|\bar{\mathbf{x}}) \log q_\psi(y|z) + H(Y) \\ &= \mathbb{E}_{\bar{\mathbf{x}},y\sim p(\bar{\mathbf{x}},y)}\mathbb{E}_{z\sim q_\theta(z|\bar{\mathbf{x}})}\Big[\log q_\psi(y|z)\Big] + H(Y) \end{aligned} \tag{S.2}$$

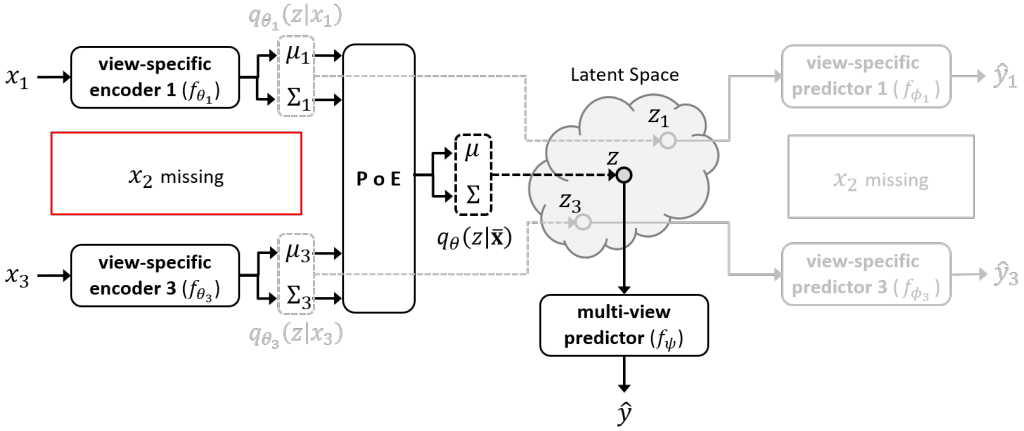
and

$$\begin{aligned} I(\bar{\mathbf{X}};Z) = \int d\bar{\mathbf{x}}dz\ p(\bar{\mathbf{x}},z) \log \frac{p(z|\bar{\mathbf{x}})}{p(z)} &= \int d\bar{\mathbf{x}}dz\ p(\bar{\mathbf{x}})p(z|\bar{\mathbf{x}}) \log \frac{p(z|\bar{\mathbf{x}})}{p(z)} \\ &\approx \int d\bar{\mathbf{x}}dz\ p(\bar{\mathbf{x}})q_\theta(z|\bar{\mathbf{x}}) \log \frac{q_\theta(z|\bar{\mathbf{x}})}{q(z)} \\ &= \mathbb{E}_{\bar{\mathbf{x}}\sim p(\bar{\mathbf{x}})}\Big[KL\big(q_\theta(Z|\bar{\mathbf{x}})\big\|q(Z)\big)\Big] \end{aligned} \tag{S.3}$$

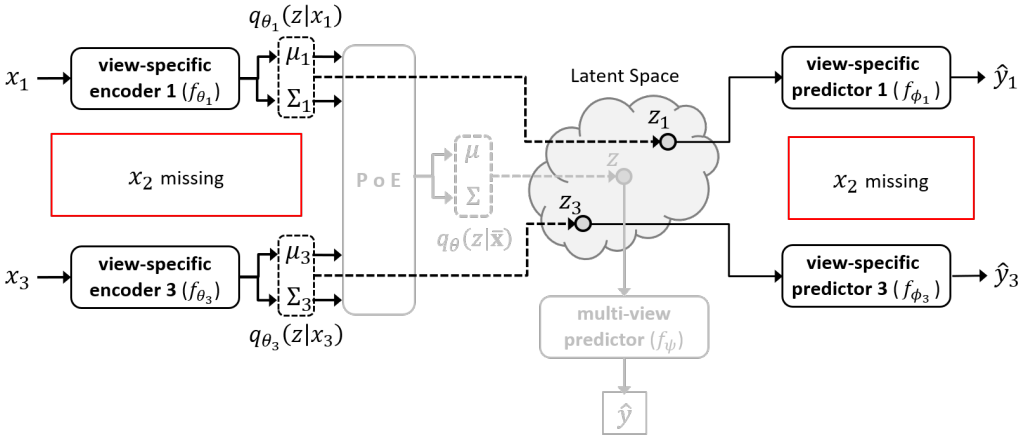
where $KL\big(q_\theta(Z|\bar{\mathbf{x}})\big\|q(Z)\big)$ denotes the Kullback-Leibler (KL) divergence between the two distributions $q_\theta(Z|\bar{\mathbf{x}})$ and $q(Z)$. Since the entropy of the labels $H(Y)$ is independent of our optimization procedure, we can simply ignore it and approximate the IB loss by plugging (S.2) and (S.3) into (S.1):

$$\mathcal{L}^{\theta,\psi}_{\text{IB-J}}(\bar{\mathbf{X}}, Y) \approx \mathbb{E}_{\bar{\mathbf{x}},y\sim p(\bar{\mathbf{x}},y)}\mathbb{E}_{z\sim q_\theta(z|\bar{\mathbf{x}})}\big[-\log q_\psi(y|z)\big] + \beta\mathbb{E}_{\bar{\mathbf{x}}\sim p(\bar{\mathbf{x}})}\big[KL\big(q_\theta(Z|\bar{\mathbf{x}})\big\|q(Z)\big)\big]$$

We can similarly derive the IB loss for the marginal representation Z_v for $v \in \mathcal{V}$.



(a) The network components updated via the joint IB losses in (2)



(b) The network components updated via the marginal IB losses in (1)

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., $x_2 = \varnothing$.

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., (θ, ψ) – are updated based on the joint IB loss while those of the view-specific encoders and the view-specific predictors – i.e., (θ, ϕ) – 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., $\mathcal{L}_{\text{IB-J}}^{\theta,\psi}$, 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 (θ, ϕ, ψ) are initialized by Xavier initialization [2] and optimized via Adam optimizer [3] with learning rate of 1×10^{-4}. We utilize dropout [4] with dropout probability 0.7 to regularize the network. The network is further regularized via L_1-regularization with 1×10^{-4} for the CCLE dataset and no additional regularization is used for the TCGA dataset.

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 α and β.

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**[1] [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**[2] [6] and **DCCAE**[2] [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**[3] [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

[1] `https://github.com/rupy/GCCA`
[2] `https://ttic.uchicago.edu/~wwang5/dccae.html`
[3] `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**[4] [9]: 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**[5] [10]: 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`:

- DNA methylation (epigenomics): *Methylation_Preprocess.Level_3.2016012800.0.0.tar.gz*
- microRNA expression (transcriptomics): *miRseq_Preprocess.Level_3.2016012800.0.0.tar.gz*
- mRNA expression (transcriptomics): *mRNAseq_Preprocess.Level_3.2016012800.0.0.tar.gz*
- RPPA (proteomics): *RPPA_AnnotateWithGene.Level_3.2016012800.0.0.tar.gz*
- clinical labels: *Clinical_Pick_Tier1.Level_4.2016012800.0.0.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`:

- 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*

[4] `https://github.com/hanmenghan/CPM_Nets`
[5] `https://pypi.org/project/mofapy/`

Table 1: Comparison of the AUROC performance (mean ± 95%-CI) with different view completion methods on the TCGA dataset. All methods are trained with both complete and incomplete multi-view samples. The values are reported by varying the number of observed views from $|\mathcal{V}^n| = 1$ to $|\mathcal{V}^n| = 4$ during testing.

Methods	*1 View*		*2 Views*		*3 Views*		*4 Views*	
	mean impt.	*MVAE impt.*	*mean impt.*	*MVAE impt.*	*mean impt.*	*MVAE impt.*	*mean impt.*	*MVAE impt.*
Base1	0.675±0.02	0.679±0.02	0.739±0.02	0.744±0.01	0.765±0.02	0.771±0.01	0.781±0.01	0.780±0.02
Base2	0.717±0.02	0.717±0.02	0.766±0.00	0.765±0.02	0.775±0.01	0.784±0.01	0.790±0.01	0.790±0.01
GCCA	0.650±0.03	0.660±0.01	0.737±0.03	0.737±0.03	0.769±0.02	0.774±0.01	0.792±0.01	0.794±0.00
DCCA	0.638±0.03	0.671±0.02	0.761±0.02	0.763±0.02	0.775±0.01	0.784±0.02	0.784±0.01	0.794±0.01
DCCAE	0.605±0.04	0.626±0.03	0.763±0.01	0.763±0.03	0.775±0.01	0.773±0.02	0.778±0.02	0.779±0.01
MVAE	0.589±0.04		0.674±0.02		0.730±0.01		0.781±0.01	
CPM-Nets	0.709±0.01		0.761±0.02		0.771±0.01		0.788±0.01	
DeepIMV	**0.724±0.02**		**0.772±0.01**		**0.791±0.01**		**0.801±0.01**	

Table 2: Comparison of the AUROC performance (mean ± 95%-CI) with different view completion methods on the CCLE dataset with $M = 6$ and $R = 0.6$.

Methods	*Irinotecan*		*Panobinostat*		*Lapatinib*		*PLX4720*	
	mean impt.	*MVAE impt.*	*mean impt.*	*MVAE impt.*	*mean impt.*	*MVAE impt.*	*mean impt.*	*MVAE impt.*
Base1	0.736±0.01	0.726±0.02	0.751±0.01	0.758±0.01	0.600±0.01	0.632±0.01	0.633±0.01	0.630±0.01
Base2	0.738±0.02	0.730±0.02	0.728±0.02	0.752±0.01	0.641±0.01	0.627±0.01	0.632±0.02	0.631±0.02
GCCA	0.694±0.01	0.698±0.02	0.709±0.01	0.715±0.01	0.620±0.01	0.619±0.01	0.615±0.02	0.617±0.01
DCCA	0.647±0.02	0.662±0.02	0.714±0.01	0.717±0.01	0.610±0.01	0.608±0.02	0.568±0.02	0.567±0.01
DCCAE	0.661±0.02	0.660±0.02	0.688±0.01	0.697±0.01	0.565±0.01	0.559±0.01	0.554±0.01	0.563±0.01
MVAE	0.672±0.01		0.678±0.01		0.603±0.01		0.593±0.01	
CPM-Nets	0.675±0.02		0.702±0.01		**0.648±0.01**		0.635±0.01	
MOFA	0.708±0.02		0.727±0.02		0.585±0.02		0.559±0.02	
DeepIMV	**0.752±0.01**		**0.768±0.01**		0.641±0.01		**0.640±0.01**	

- drug sensitivities: *CCLE_NP24.2009_Drug_data_2015.02.24.csv*

Drug response was converted to a binary label by dividing cell lines into quartiles ranked by ActArea; the top 25% were assigned to the "sensitive" class and the rest were assigned to the"non-sensitiv" class.

For both datasets, we imputed missing values within the observed views with mean values. To focus our experiments on the integrative analysis and to avoid "curse-of-dimensionality" in the high-dimensional multi-omics data, we extracted low-dimensional representations (i.e., 100 features) using the kernel-PCA (with polynomial kernels) on each view [11].

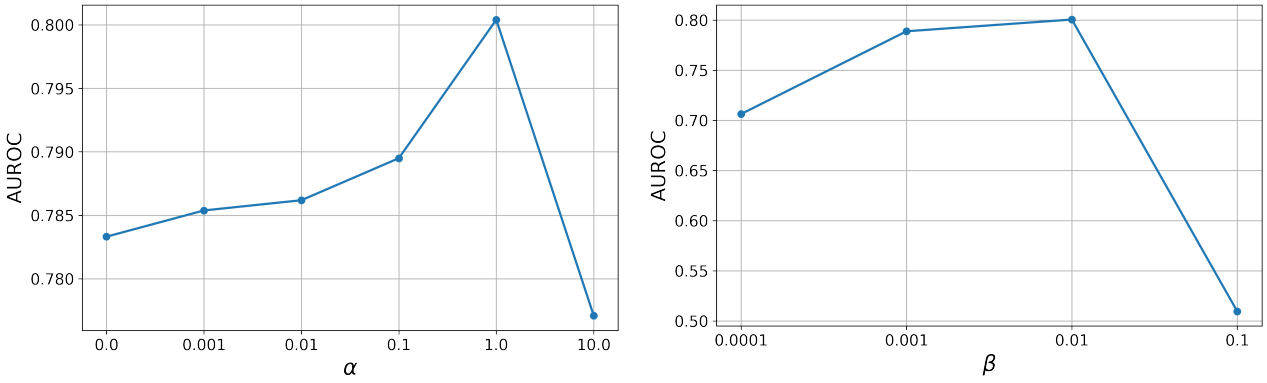
5 Additional Experiments

5.1 Additional Experiments with Multi-View Imputations

We also imputed observations from missing views by utilizing the reconstructed inputs of MVAE, which can flexibly integrate incomplete multi-view observations regardless of the view-missing patterns. Table 1 and 2 shows the AUROC performance when two different imputation methods are used for the multi-view learning methods (except for MVAE, CPM-Nets, MOFA, and DeepIMV that do not depend on the imputation methods), for the TCGA dataset and the CCLE dataset, respectively. For the TCGA dataset, all the methods are trained with both complete-view and incomplete-view samples. And, for the CCLE dataset, we set $M = 6$ and $R = 0.6$ for constructing missing views. The benchmarks trained with imputed observations based on MVAE did not always provide performance gain over those trained with mean imputed observations since reconstructing the inputs can fail to maintain information that is relevant for predicting the target. Even when the imputation based on MVAE improves the discriminative performance, our method still outperforms the benchmark for all the datasets except for *Lapatinib* of the CCLE dataset.

5.2 Sensitivity Analysis – Effects of α and β

In this section, we provide sensitivity analysis using the TCGA dataset to see effects of α and β on the prediction performance of DeepIMV. Figure 2 shows the AUROC performance of our method with respect to different values of α and β, respectively. For training the variants of the proposed method, we used both complete-view and incomplete-view samples.



(a) AUROC vs α

(b) AUROC vs β

Figure 2: The AUROC performance for the TCGA dataset with $|\mathcal{V}^n| = 4$ in terms of different values for (a) α (while setting $\beta = 0.01$) and (b) β (while setting $\alpha = 1.0$).

The Effect of α. As shown in Figure 2a, the discriminative performance drops at $\alpha = 10.0$ since too high value of α 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 α 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 β. Similar to the findings via extensive experiments in [1], β, 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 β blocks information from the input that is required to provide good predictions on the target task. For small values of β, 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

[1] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 2017.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, 2010.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 14(1), January 2014.

[5] J. R. Kettenring. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451, 1971.

[6] G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. *In Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[7] W. Wang, R. Arora, K. Livescu, and J. Bilmes. On deep multi-view representation learning. *In Proceedings of the 32th International Conference on Machine Learning (ICML 2015)*, 2015.

[8] M. Wu and N. Goodman. Multimodal generative models for scalable weakly-supervised learning. *In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

[9] Changqing Zhang, Zongbo Han, Yajie Cui, Huazhu Fu, Joey T. Zhou, and Qinghua Hu. CPM-nets: Cross partial multi-view networks. *In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[10] R. Argelaguet, B. Velten, D. Arnol, S. Dietrich, T. Zenz, J. Marioni, F. Buettner, W. Huber, and O. Stegle. Multi-omics factor analysis–a framework for unsupervised integration of multi-omics data sets. *Mol Syst Biol*, 14(6):3348-3356, 2018.

[11] Y. Shiokawa, Y. Date, and J. Kikuchi. Application of kernel principal component analysis and computational machine learning to exploration of metabolites strongly associated with diet. *Scientific Reports*, 8(3426), February 2018.