A APPENDIX

A.1 PROOF OF PROPOSITION 1

Let $B_{i,j}$ and $C_{i,j}$ denote the (i,j)-th entry of B and C, respectively. Then we have:

$$(BC)_{i,j} = \sum_{r=1} B_{i,r} C_{r,j}$$

Since each row of C has a mean of 0, we have $\sum_{j=1}^{n} C_{r,j} = 0, \forall r$. For the mean value of i-th row of BC, we can write:

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n}(BC)_{i,j} &= \frac{1}{n}\sum_{j=1}^{n}\sum_{r=1} B_{i,r}C_{r,j} \\
&= \frac{1}{n}\sum_{r=1}\sum_{j=1}^{n} B_{i,r}C_{r,j} \\
&= \frac{1}{n}\sum_{r=1} B_{i,r}\left(\sum_{j=1}^{n} C_{r,j}\right) \\
&= \frac{1}{n}\sum_{r=1} B_{i,r}\cdot 0 \\
&= 0
\end{aligned} \tag{8}$$

A.2 PROOF OF PROPOSITION 2

$$\begin{aligned}
\text{Corr}(X_k, A_k) &= \text{tr}((\Sigma_{11}^{-1/2}\Sigma_{12}\Sigma_{22}^{-1/2})'\Sigma_{11}^{-1/2}\Sigma_{12}\Sigma_{22}^{-1/2})^{1/2} \\
&= \text{tr}(\Sigma_{22}^{-1/2}\Sigma_{12}'\Sigma_{11}^{-1/2}\Sigma_{11}^{-1/2}\Sigma_{12}\Sigma_{22}^{-1/2})^{1/2} \\
&= \text{tr}(\Sigma_{22}^{-1/2}\Sigma_{22}^{-1/2}\Sigma_{12}'\Sigma_{11}^{-1/2}\Sigma_{11}^{-1/2}\Sigma_{12})^{1/2} \\
&= \text{tr}(\Sigma_{22}^{-1}\Sigma_{12}'\Sigma_{11}^{-1}\Sigma_{12})^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}((A_kA_k')^{-1}(X_kA_k')'(X_kX_k')^{-1}(X_kA_k'))^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}((A_kX_k')^{-1}(A_kX_k')(X_kX_k')^{-1}(X_kA_k'))^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}((A_kA_k')^{+}(A_kX_k')(X_kX_k')^{+}(X_kA_k'))^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}(A_k'(A_kA_k')^{+}A_kX_k'(X_kX_k')^{+}X_k)^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}(A_k^{+}A_kX_k^{+}X_k)^{1/2}
\end{aligned} \tag{9}$$

The first row is based on the definition of Corr, the second row is because trace is invariant under cyclic permutation, the fifth row is to replace matrix inverse by MPI and the ninth row is due to $Y^{+} = Y'(YY^{+})$ (Petersen et al., 2008).

A.3 PROOF OF PROPOSITION 3

Firstly, we have the k-th view data X_k to be full-rank, as we can always delete the redundant data, and the random noise A_k is full-rank as each column is generated independently. Then by utilizing Proposition 2, we derive that the correlation between X_k and A_k remains unchanged before and after the transformation:

$$\begin{aligned}
\text{Corr}(W_kX_k, W_kA_k) &= \frac{1}{(n-1)^2}\text{tr}((W_kA_k)^+W_kA_k(W_kX_k)^+W_kX_k)^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}((W_k^+W_kA_k)^+(W_kA_kA_k^+)^+W_kA_k(W_k^+W_kX_k)^+(W_kX_kX_k^+)^+W_kX_k)^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}((W_k^+W_kA_k)^+W_k^+W_kA_k(W_k^+W_kX_k)^+W_k^+W_kX_k)^{1/2} \\
&= \frac{1}{(n-1)^2}\text{tr}(A_k^+A_kX_k^+X_k)^{1/2} \\
&= \text{Corr}(X_k, A_k)
\end{aligned} \tag{10}$$

The first row is based on Proposition 2, the second row is because given two matrices B and C, $(BC)^+ = (B^+BC)^+(BC^+C)^+$ always holds (Petersen et al., 2008), and the third row utilizes the properties of full-rank and square matrix W_k: $W_k^+ = W_k^-$, which means $W_k^+W_k = W_kW_k^+ = I_{d_k}$ (Petersen et al., 2008).

A.4 Proof of Proposition 4

This proposition is equivalent to its contra-positive proposition: if W_k is not a full-rank matrix, there exists random noise data A_k such that $\eta_k = |Corr(W_kX_k, W_k(A_k)) - Corr(X_k, A_k)|$ is not 0. And we find that when W_k is not full-rank, there exists $A_k = X_k$ such that $\eta_k \neq 0$. We have the following derivation:

$$\begin{aligned}
\eta_k &= |Corr(W_kX_k, W_kA_k) - Corr(X_k, A_k)| \\
&= \left|\frac{1}{(n-1)^2}\text{tr}((W_kA_k)^+(W_kA_k)(W_kX_k)^+(WX_k))^{1/2} - \frac{1}{(n-1)^2}\text{tr}(A^+AX^+X)^{1/2}\right| \\
&= \left|\frac{1}{(n-1)^2}\text{tr}((W_k^+W_kA_k)^+(W_kA_kA_k^+)^+W_kA_k(W_k^+W_kX_k)^+(W_kX_kX_k^+)^+W_kX_k)^{1/2} - \frac{1}{(n-1)^2}\text{tr}(A_k^+A_kX_k^+X_k)^{1/2}\right| \\
&= \left|\frac{1}{(n-1)^2}\text{tr}((W_k^+W_kA_k)^+W_k^+W_kA_k(W_k^+W_kX_k)^+W_k^+W_kX_k)^{1/2} - \frac{1}{(n-1)^2}\text{tr}(A_k^+AX_k^+X_k)^{1/2}\right|
\end{aligned} \tag{11}$$

The first row is the definition of NR loss with respect to W_k, the second row is based on the new form of CCA, the third row is because given two specific matrices B and C, it holds the equality $(BC)^+ = (B^+BC)^+(BC^+C)^+$ (Petersen et al., 2008), and the fourth row utilizes the properties of full-rank matrix: for full-rank matrices X_k and A_k, whose sample size is larger than dimension size, they fulfill: $X_k{X_k}^+ = I_{d_k}, A_k{A_k}^+ = I_{d_k}$ (given a specific full-rank matrix Y, if its number of rows is smaller than that of cols, it holds that $Y^+ = Y'(YY')^-$, which means that $YY^+ = I$) (Petersen et al., 2008).

Let us analyze the case when $A_k = X_k$:

$$\begin{aligned}
\eta_k &= \left|\frac{1}{(n-1)^2}\text{tr}((W_k^+W_kX_k)^+W_k^+W_kX_k(W_k^+W_kX_k)^+W_k^+W_kX_k)^{1/2} - \frac{1}{(n-1)^2}\text{tr}(X_k^+X_kX_k^+X_k)^{1/2}\right| \\
&= \left|\frac{1}{(n-1)^2}\text{tr}((W_k^+W_kX_k)^+W_k^+W_kX_k)^{1/2} - \frac{1}{(n-1)^2}\text{tr}(X_k^+X_k)^{1/2}\right|.
\end{aligned} \tag{12}$$

The first row is to replace A_k with X_k, the second row is because $X_k^+X_kX_k^+ = X_k^+$ and $(W_k^+W_kX_k)^+W_k^+W_kX_k(W_k^+W_kX_k)^+ = W_k^+W_kX_k$, which are based on the definition of MPI that given a specific matrix Y, $Y^+YY^+ = Y^+$ (Petersen et al., 2008).

Next, we need to prove the following two lemmas:

Lemma 1 *Given a specific matrix Y and its MPI Y^+, let Rank(Y) and Rank(Y^+Y) be the ranks of Y and Y^+Y, respectively. It is true that:*

$$Rank(Y) = Rank(Y^+Y)$$

$$Rank(Y^+Y) = tr(Y^+Y)$$

Proof *Firstly, the column space of Y^+Y is a subspace of the column space of Y. Therefore, Rank$(Y^+Y) \leq$ Rank(Y). On the other hand, according to the definition of MPI (Petersen et al.,*

2008), we know that $Y = Y(Y^+Y)$. Since the rank of a product of matrices is at most the minimum of the ranks of the individual matrices, we have Rank$(Y) \leq$ Rank(Y^+Y). Combining the two inequalities, we have Rank$(Y) =$ Rank(Y^+Y).

Furthermore, since $(Y^+Y)(Y^+Y) = Y^+Y$ (it holds that $Y^+ = Y^+YY^+$ according to the definition of MPI (Petersen et al., 2008)), Y^+Y is an idempotent and symmetric matrix, and thus its eigenvalues must be 0 or 1. So the sum of its eigenvalues is exactly its rank. Considering matrix trace is the sum of eigenvalues of matrices, we have Rank$(Y^+Y) = tr(Y^+Y)$.

Lemma 2 *Rank$(W_kX_k) <$ Rank(X_k) , when W_k is not a full-rank matrix and X_k is a full-rank matrix.*

Proof *Since the rank of a product of matrices is at most the minimum of the ranks of the individual matrices, we have Rank$(W_kX_k) \leq min($Rank$(W_k),$ Rank$(X_k))$. Considering X_k is full-rank, Rank$(X_k) = min(d_k, n)$ and then Rank$(W_kX_k) \leq min($Rank$(W_k),$ Rank$(X_k)) = min($Rank$(W_k), min(d_k, n))$. Since W_k is not full-rank, we have Rank$(W_k) < d_k$. In conclusion, Rank$(W_kX_k) < min(d_k, min(d_k, n))$ and then Rank$(W_kX_k) < d_k \leq$ Rank(X_k).*

As a result, we can know that when the random noise data A_k is exactly X_k and W_k is not full-rank, η_k can not be zero:

$$\begin{aligned}
\eta_k &= \left| \frac{1}{(n-1)^2} \text{tr}((W_k^+ W_k X_k)^+ W_k^+ W_k X_k)^{1/2} - \frac{1}{(n-1)^2} \text{tr}(X_k^+ X_k)^{1/2} \right| \\
&= \left| \frac{1}{(n-1)^2} \text{Rank}(W_k^+ W_k X_k)^{1/2} - \frac{1}{(n-1)^2} \text{Rank}(X)^{1/2} \right| \\
&\neq \left| \frac{1}{(n-1)^2} \text{Rank}(X_k)^{1/2} - \frac{1}{(n-1)^2} \text{Rank}(X_k)^{1/2} \right| \\
&\neq 0
\end{aligned} \tag{13}$$

The first row is due to Equation 12, the second row is based on Lemma 1 that $\text{tr}((W_k^+W_kX_k)^+W_k^+W_kX_k) = \text{Rank}(W_k^+W_kX_k)$ and $\text{tr}(X_k^+X_k) = \text{Rank}(X)$, and the third row is because of Lemma 2.

Finally, we have if η_k is always constrained to 0 for any A_k, then W_k must be a full-rank matrix.

A.5 Details of Datasets and Baselines

Synthetic datasets:

We make 6 groups of multi-view data originating from the same $G \in \mathbb{R}^{d \times n}$ (we set $n = 4000, d = 100$). Each group consists of tuples with 2 views (2000 tuples for training and 2000 tuples for testing) and a distinct common rate. Common rates of these sets are from $\{0\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$ and there are 50 downstream regression tasks. We report the mean and standard deviation of R2 score across all the tasks.

Real-world datasets:

PolyMnist (Sutter et al., 2021): A dataset consists of tuples with 5 different MNIST images ($60,000$ tuples for training and $10,000$ tuples for testing). Each image within a tuple possesses distinct backgrounds and writing styles, yet they share the same digit label. The background of each view is randomly cropped from an image and is not used in other views. Thus, the digit identity represents the common information, while the background and writing style serve as view-specific factors. The downstream task is the digit classification task. **CUB** (Wah et al., 2011): A dataset consists of tuples with deep visual features (1024-d) extracted by GoogLeNet and text features (300-d) obtained through Doc2vec (Le & Mikolov, 2014) (480 tuples for training and 600 tuples for testing). This MVRL task utilizes the first 10 categories of birds in the original dataset and the downstream task is the bird classification task. **Caltech** (Deng et al., 2018): A dataset consists of tuples with traditional visual features extracted from images that belong to 101 object categories, including an additional background category (6400 tuples for training and 9144 tuples for testing). Following Yang et al. (2021), three features are used as views: a $1,984$-d HOG feature, a 512-d GIST feature, and a 928-d SIFT feature.

Baselines:

Direct method:

- **CONCAT** straightforwardly concatenates original features from different views.

CCA methods:

- **CCA** (Hotelling, 1992) maps multiple views' data into a common space that maximizes their correlation and concatenates the new representations of different views.
- **PRCCA** Tuzhilina et al. (2023) preserves the internal data structure by grouping high-dimensional data features while applying an l2 penalty to CCA,.

Kernel CCA Methods:

- **KCCA** (Akaho, 2006) employs CCA methods through positive-definite kernels.

DCCA-based methods:

- **DCCA** (Andrew et al., 2013) employs neural networks to individually project multiple sets of views, obtaining new representations that maximize the correlation between each pair of views.
- **DGCCA** (Benton et al., 2017) constructs a shared representation and maximizes the correlation between each view and the shared representation.
- **DCCAE/DGCCAE** (Wang et al., 2015) introduces reconstruction objectives to DCCA, which simultaneously optimize the canonical correlation between the learned representations and the reconstruction errors of the autoencoders.
- **DCCA_PRIVATE/DGCCA_PRIVATE** (Wang et al., 2016) incorporates dropout and private autoencoders, thus preserving both shared and view-specific information.

Information theory-based methods:

- **MVTCAE** (Hwang et al., 2021) maximizes the reduction in Total Correlation to capture both shared and view-specific factors of variations.

All CCA-based methods leverage the implementation of CCA-Zoo (Chapman & Wang, 2021). To ensure fairness, we use the official implementation of MVTCAE while replacing the strong CNN backbone with MLP.

A.6 Hyper-parameter Settings

To ensure a fair comparison, we tune the hyper-parameters of all baselines within the ranges suggested in the original papers, including hyper-parameter r of ridge regularization, except for the following fixed settings:

The embedding size for the real-world datasets is set as 200, while the size for synthetic datasets is set as 100. Batch size is $\min(2000, \text{full-size})$. The same MLP architectures are used for DCCA-based methods.

In the synthetic datasets, DCCA, DGCCA, DCCAE, and DGCCAE methods utilize a minimum learning rate of $5e-3$. DCCA_PRIVATE/DGCCA_PRIVATE employ a slightly higher learning rate of $1e-2$. In contrast, our proposed methods, NR-DCCA/NR-DGCCA, utilize the maximum learning rate of $1.5e-2$. And the regularization weight α is set as 200.

In the real-world datasets, both the learning rates in PolyMnist and CUB are set to $1e-4$. For Caltech101, a slightly lower learning rate of $5e-5$ is used. To expedite the computation of Corr(X_k, A_k), on the PolyMnist dataset, we utilize the initialized f_k to reduce the feature dimensions of X_k and A_k separately. Subsequently, we calculate their correlation. For the extracted features in the CUB and Caltech101 datasets, we simply employ $X_k[: outdim, :]$ and $A_k[: outdim, :]$ to compute of Corr. The hyper-parameter r of ridge regularization is set as 0 in our NR-DCCA and NR-DGCCA. The optimal α values of NR-DCCA for the CUB, PolyMnist, and Caltech datasets are found to be 1.5, 5, and 15, respectively.

A.6.1 HYPER-PARAMETER r IN RIDGE REGULARIZATION

In this section, we discuss the effects of hyper-parameter r in ridge regularization. Ridge regularization is commonly used across almost all (D)CCA methods, whichimproves numerical stability. It works by adding an identity matrix I to the estimated covariance matrix. However, ridge regularization mainly regularizes the features, rather than the transformation (i.e., W_k in CCA and f_k in DCCA) and it cannot prevent the neural networks from degenerating (i.e., model collapse). To further support our arguments, we provide the experimental results with different ridge parameters on a real-world dataset CUB as shown in Figure 7. One can see that the ridge regularization even damages the performance of DCCA. In our NR-DCCA, we actually set the ridge parameter to zero. We conjecture the reason is that the large ridge parameter could make the neural network even "lazier" to actively project the data into a better feature space, as the full-rank property of features and covariance matrix are already guaranteed, and this is also evidenced by the "Square sum of feature covariance" shown in the figure.

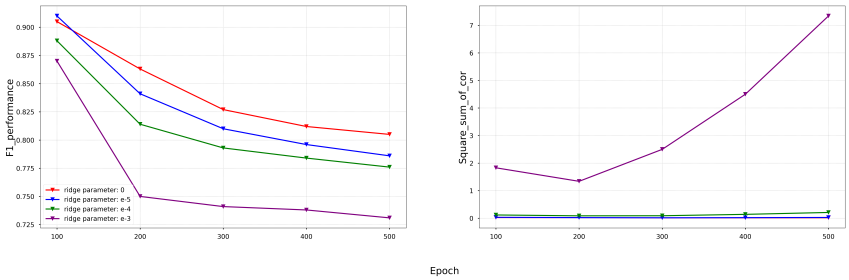


Figure 7: The effects of hyper-parameter r of DCCA in the CUB dataset.

A.6.2 HYPER-PARAMETER α OF NR-DCCA

The choice of the hyper-parameter α is essential in NR-DCCA. Different from the conventional hyper-parameter tuning procedures, the determination of α is simpler, as we can search for the smallest α that can prevent the model collapse, and the model collapse can be directly observed on the validation data. Specifically, we increase the α adaptively until the model collapse issue is tackled, i.e., the correlation with noise will not increase or the performance of DCCA will not drop with increasing training epochs, then the optimal α is found. To further illustrate the influence of α in NR-DCCA, we present performance curves of NR-DCCA in CUB under different α. As shown in Figure 8, if α is too large, the convergence of the training becomes slow; if α is too small, model collapse still remains. Additionally, one can see the NR-DCCA outperforms DCCA robustly with a wide range of α.

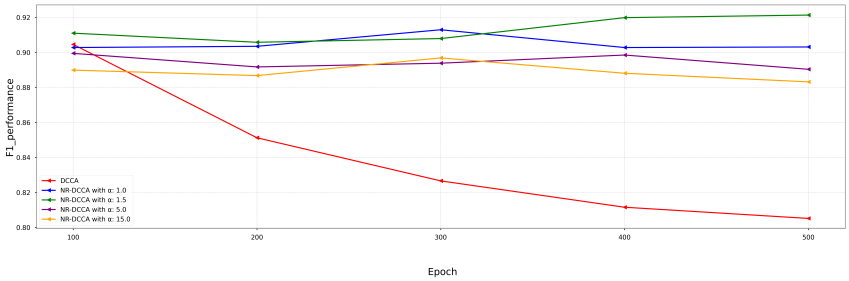


Figure 8: The effects of hyper-parameter α of NR-DCCA in the CUB dataset.

A.7 IMPLEMENTATION DETAILS OF SYNTHETIC DATASETS

We draw n random samples with dim d from a Gaussian distribution as $G \in \mathbb{R}^{d\times n}$ to represent complete representations of n objects. We define the non-linear transformation ϕ_k as the addition of noise to the data, followed by passing it through a randomly generated MLP. To generate the data

for the k-th view, we select specific feature dimensions from G based on a given common rate 3 and then apply ϕ_k to those selected dimensions. And we define ψ_j as a linear layer, and task T_j is generated by directly passing G through ψ_j.

A.8 Complexity Analysis

In this section, we compare the computational complexity of different DCCA-based methods. Assuming that we have data from K views, with each view containing N samples and D feature dimensions, then we have the computational complexity of each method in Table 1.

Table 1: Comparisons of computational complexity against baselines

	DCCA	DCCAE	DCCA_PRIVATE	NR-DCCA
Generation of Noise	-	-	-	$O(K*N*D)$
MLP Encoder	$O(K*N*L*H^2)$	$O(K*N*L*H^2)$	$O(2*K*N*L*H^2)$	$O(2*K*N*L*H^2)$
MLP Decoder	-	$O(K*N*L*H^2)$	$O(K*N*L*H^2)$	-
Reconstruction Loss	-	$O(K*N*D)$	$O(K*N*D)$	-
Correlation Maximization	$O((M*K)^3)$	$O((M*K)^3)$	$O((M*K)^3)$	$O((M*K)^3)$
Noise Regularization	-	-	-	$O(2*K*(M*K)^3)$

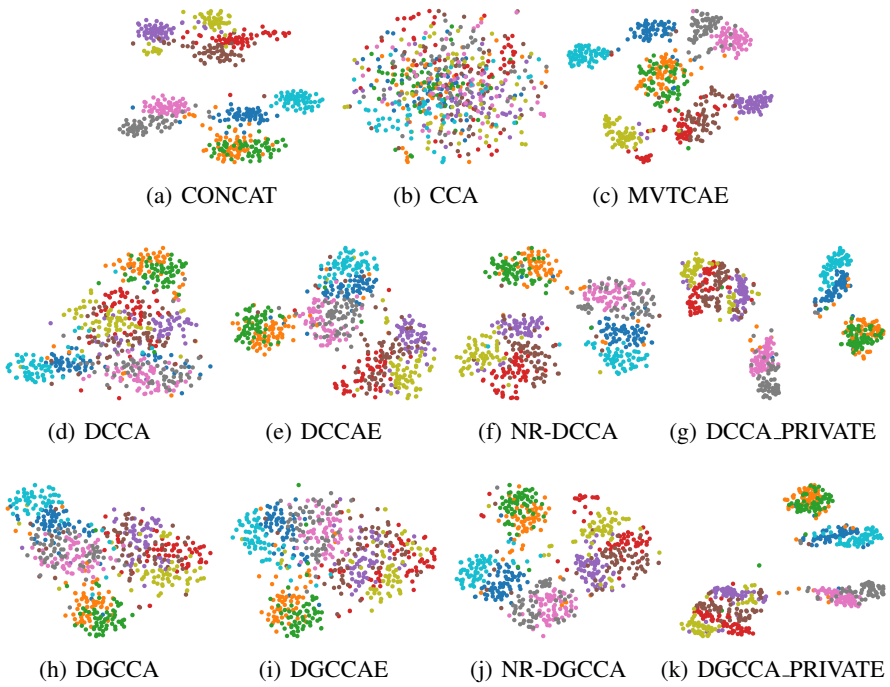
- **Complexity of MLP:** We will use DNNs with the same MLP structure, consisting of L hidden layers, each with H neurons. Therefore, the computational complexity of one pass of the data through the DNNs can be expressed as $O(N*(D*H+D*M+L*H^2))$. To simplify, we use $O(N*L*H^2)$.
- **Complexity of Corr:** During the process of calculating Cor among K views, three main computations are involved. The calculation complexity of the covariance is $O(N*(M*K)^2$. Second, the complexity of the inverse matrix and the eigenvalues are $O((M*K)^3$. As a result, the computational complexity of calculating Cor can be considered as $O((M*K)^3)$.
- **Complexity of reconstruction loss:** The reconstruction loss, also known as the mean squared error (MSE) loss, has a complexity of $O(N*D)$.

A.9 Visualization of the learned representations

To further demonstrate the effectiveness of our method, we employ 2D-tSNE visualization to depict the learned representations of the CUB dataset (test set) under different methods. Each data point is colored based on its corresponding class, as illustrated in Figure 9. There are a total of 10 categories, with 60 data points in each category. A reasonable distribution of learned representations entails that data points belonging to the same class are grouped together in the same cluster, which is distinguishable from clusters representing other classes. Additionally, within each cluster, the data points should exhibit an appropriate level of dispersion, indicating that the data points within the same class can be differentiated rather than collapsing into a single point. This dispersion is indicative of the preservation of as many distinctive features of the data as possible.

From Figure. 9, we can observe that CCA, DCCA / DGCCA have all confused the data from different categories. Specifically, CCA completely scatter the data points as it cannot handle nonlinear relationships. By incorporating autoencoders, DCCAE / DGCCAE and DCCA_PRIVATE / DGCCA_PRIVATE have partially separated the data; however, they have not fully separated the green and orange categories. NR-DCCA / NR-DGCCA is the only method that successfully separates all categories.

It is worth noting that our approach not only separates the data into different clusters but also maintains dispersion within each cluster. Unlike DCCA_PRIVATE / DGCCA_PRIVATE, where the data points within a cluster form a strip-like distribution, our method ensures that the data points within each cluster remain appropriately scattered.



(a) CONCAT (b) CCA (c) MVTCAE

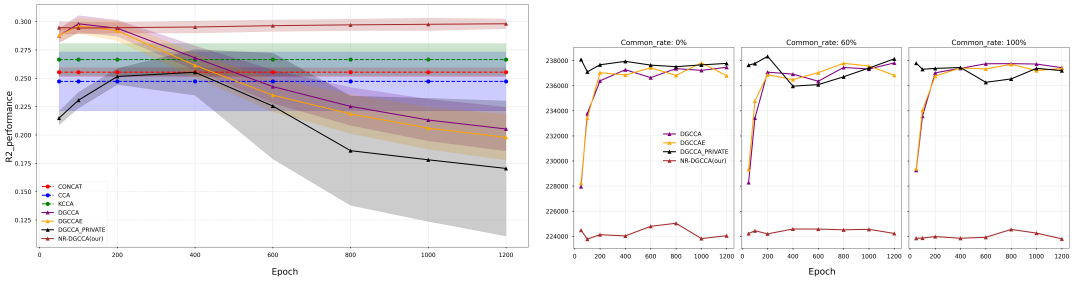
(d) DCCA (e) DCCAE (f) NR-DCCA (g) DCCA_PRIVATE

(h) DGCCA (i) DGCCAE (j) NR-DGCCA (k) DGCCA_PRIVATE

Figure 9: Visualization of the learned representations with t-SNE in the **CUB** dataset.

A.10 DGCCA AND NR-DGCCA

This section presents the experimental results for DGCCA and NR-DGCCA, which supplement the results of GCCA and NR-DCCA presented in the main paper. In general, DGCCA is a variant of DCCA, and hence the proposed noise regularization approach can also be applied. We repeat the experiments in Figures 4, 5, and 6, and hence we have the results for DGCCA in Figure 10, 11, and 12. One can see that the proposed noise regularization approach can also help DGCCA prevent model collapse, proving its generalizability.



(a) Performance (b) Correlation

Figure 10: (a) Mean and standard deviation of the GCCA-based method performance across synthetic datasets in different training epochs. (b) The correlation between noise and real data after transformation varies with epochs in different common rate settings for GCCA-based methods.

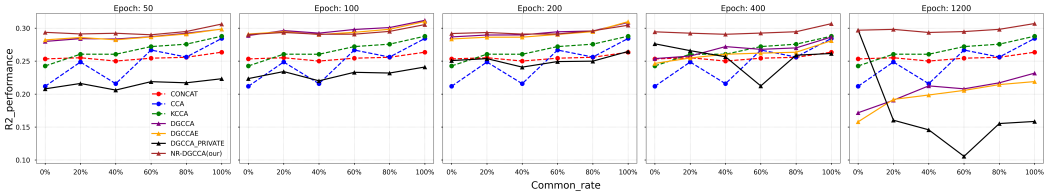


Figure 11: Performance of DGCCA-based methods with respect to different common rates during the training. Each column represents the testing accuracy of the method at a specific training epoch.

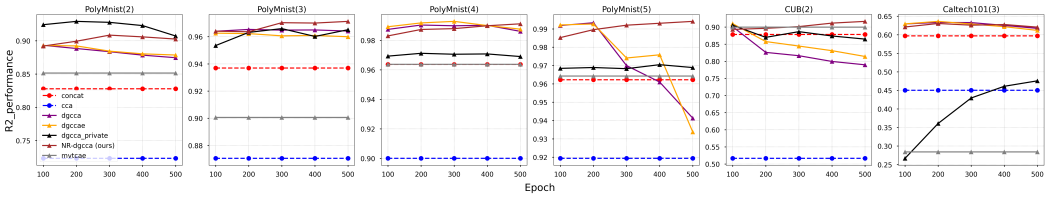


Figure 12: Performance of DGCCA-based methods in real-world datasets. Each column represents the performance on a specific dataset. The number of views in the dataset is denoted in the parentheses next to the dataset name.

A.11 ADDITIONAL EXPERIMENTAL RESULTS

Table 2 and 3 present the model performance of various MVRL methods in synthetic and real-world datasets, and both tables correspond to the final epoch of the results presented in Figure 5 and 6. It should be noted that the values in Table 2 represent the mean and standard deviation of the methods across different tasks, indicating their performance and variability.

Table 2: Performance in synthetic datasets.

R2/Common Rate	0%	20%	40%	60%	80%	100%
CONCAT	0.253±0.038	0.255±0.039	0.250±0.040	0.254±0.040	0.256 ±0.042	0.264 ± 0.033
CCA	0.212±0.053	0.249±0.046	0.216±0.055	0.267±0.046	0.256±0.052	0.284±0.039
KCCA	0.243±0.047	0.261±0.046	0.260±0.043	0.272±0.045	0.276±0.049	0.288±0.038
PRCCA	0.212±0.053	0.249±0.046	0.216±0.055	0.267±0.046	0.256±0.052	0.284±0.039
MVTCAE	0.065±0.015	0.071±0.016	0.067±0.016	0.069±0.016	0.071±0.016	0.069±0.015
DCCA	0.136±0.036	0.188±0.040	0.194±0.044	0.192±0.049	0.215±0.044	0.221±0.036
DCCAE	0.134±0.056	0.200±0.043	0.211±0.040	0.224±0.042	0.228±0.043	0.230 ±0.040
DCCA_PRIVATE	0.279±0.044	0.143±0.043	0.14±0.042	0.114±0.04	0.139±0.041	0.144±0.042
NR-DCCA (ours)	**0.296**±0.042	**0.295**±0.045	**0.293**±0.042	**0.295**±0.045	**0.300**±0.048	**0.308**±0.038
DGCCA	0.172±0.039	0.191±0.044	0.212±0.039	0.208±0.042	0.217±0.042	0.232±0.04
DGCCAE	0.158±0.04	0.192±0.041	0.199±0.04	0.206±0.041	0.214±0.041	0.219±0.038
DGCCA_PRIVATE	0.297±0.043	0.16±0.045	0.146±0.038	0.106±0.044	0.155±0.041	0.159±0.035
NR-DGCCA (ours)	**0.297**±0.043	**0.298**±0.046	**0.293**±0.043	**0.295**±0.043	**0.298**±0.048	**0.307**±0.039

Table 3: Performance in real-world datasets

F1 Score/Data	PolyMnist (2)	PolyMnist (3)	PolyMnist (4)	PolyMnist (5)	CUB	Caltech101
CONCAT	0.828	0.937	0.964	0.962	0.878	0.597
CCA	0.723	0.871	0.900	0.920	0.517	0.450
KCCA	-	-	-	-	-	-
PRCCA	0.712	0.849	0.899	0.918	-	-
MVTCAE	0.852	0.901	0.964	0.964	0.900	0.284
DCCA	0.870	0.959	0.975	0.934	0.805	0.604
DCCAE	0.871	0.958	0.983	0.965	0.850	0.605
DCCA_PRIVATE	**0.923**	0.963	0.972	0.969	0.853	0.480
NR-DCCA (ours)	0.913	**0.969**	**0.991**	**0.993**	**0.921**	**0.625**
DGCCA	0.875	0.964	0.986	0.941	0.790	0.617
DGCCAE	0.879	0.960	0.988	0.934	0.814	0.612
DGCCA_PRIVATE	**0.907**	0.965	0.969	0.969	0.864	0.476
NR-DGCCA(ours)	0.903	**0.971**	**0.991**	**0.994**	**0.917**	**0.621**