Learning Disentangled Latent Factors from Paired Data in Cross-Modal Retrieval: An Implicit Identifiable VAE Approach

Minyoung Kim*[1], Ricardo Guerrero†[1], and Vladimir Pavlovic‡[1,2]

[1]Samsung AI Center Cambridge, UK
[2]Dept. of Computer Science, Rutgers University, NJ, USA

December 2, 2020

Abstract

We deal with the problem of learning the underlying disentangled latent factors that are shared between the paired bi-modal data in cross-modal retrieval. Our assumption is that the data in both modalities are complex, structured, and high dimensional (e.g., image and text), for which the conventional deep auto-encoding latent variable models such as the Variational Autoencoder (VAE) often suffer from difficulty of accurate decoder training or realistic synthesis. A suboptimally trained decoder can potentially harm the model's capability of identifying the true factors. In this paper we propose a novel idea of the implicit decoder, which completely removes the ambient data decoding module from a latent variable model, via implicit encoder inversion that is achieved by Jacobian regularization of the low-dimensional embedding function. Motivated from the recent Identifiable-VAE (IVAE) model, we modify it to incorporate the query modality data as conditioning auxiliary input, which allows us to prove that the true parameters of the model can be identifiable under some regularity conditions. Tested on various datasets where the true factors are fully/partially available, our model is shown to identify the factors accurately, significantly outperforming conventional encoder-decoder latent variable models. We also test our model on the Recipe1M, the large-scale food image/recipe dataset, where the learned factors by our approach highly coincide with the most pronounced food factors including savoriness, wateriness, and greenness.

1 Introduction

Accurately learning the underlying factors that explain the source of variability in the complex structured data (e.g., images), is one of the core problems in representation learning in computer vision. Whereas there has been considerable research work and significant success in the unsupervised scenario recently [9, 22, 11, 3, 19, 15, 8, 16], there remains fundamental difficulty in identifying the true factors due to some known challenges [21, 24]. Having the factor labels can significantly boost the performance, but it is inherently difficult and costly to annotate data with true factors. As a remedy for this, we deal with the paired bi-modal data setup as means of partial supervision. Formally, we consider paired data $(\mathbf{x}_1, \mathbf{x}_2)$, where the goal is to learn latent factors that are shared between the two. Collecting paired data requires least amount of supervision, and can often be automated (e.g., web scrapping to pair image ($\mathbf{x}_1$) and the text ($\mathbf{x}_2$) that reside in the same web page) [28, 23].

While the CCA and its recent deep nonlinear extension, Deep CCA [1, 35], are the most popular models to extract shared features (subspace) for such paired data, the dependency of CCA's cross-covariance matrix on the entire embedded data features hinders the mini-batch stochastic gradient methods from being applied in a principled manner. Moreover, applying the latent variable generative models like the Variational Autoencoder (VAE) [18] and its bi-modal extension (Bi-VAE) [36], requires the latent-to-data synthesis/decoder modules, and learning the decoders for the complex high-dim ambient data is often very difficult. A suboptimally trained decoder can potentially harm the model's capability of identifying the true factors.

In the retrieval setup, where we let w.l.o.g., $\mathbf{x}_1$ be the query and $\mathbf{x}_2$ the search item, it is well known that the simple idea of aligning the pairs in the low-dim embedded space via cosine similarity, dubbed Cos-Sim, is shown to yield very good retrieval performance [28, 23]. Despite this success, Cos-Sim has no capability of identifying latent variables that can be controlled explicitly by users. Although the model can be extended to incorporate an extra bottleneck auto-encoding model on top of the embedded space, there is no theoretical underpinning that such extension can help discovering the true factors.

*mikim21@gmail.com
†ricardo.guerrero09@alumni.imperial.ac.uk
‡vladimir@cs.rutgers.edu

Our main goal in this paper is to learn the latent factors from paired data in the retrieval setup. More specifically, we would like to identify all the factors, each as a single latent variable in the model, and it would be ideal if the learned latent variables are *disentangled* (i.e., avoid a single latent variable impacting on the variation of two or more factors at the same time), and *complete* or minimally redundant (i.e., each of the underlying true factors has to be explained by only one of the latent variables, not by multiple ones).

To accomplish these desiderata and address the issues of the existing approaches, we introduce two strategies. First, we adopt the recent Identifiable-VAE (IVAE) model [14] that can provably identify the true model parameters by having additional auxiliary input on which the latent variables are conditioned. We modify IVAE by incorporating the query data $\mathbf{x}_1$ as conditioning auxiliary input, which enhances the model identifiability under some regularity conditions. Secondly, to remove the difficult-to-learn ambient data decoding module, we propose a novel idea of the *implicit decoder*. The modified IVAE model is built on the embedded space, and the relevance score required in the retrieval is defined by implicit inversion of the embedding function, which is accomplished by the Jacobian-based regularization of the embedding function.

Tested on various datasets where the true factors are fully/partially available, our model is shown to identify the factors accurately, significantly outperforming conventional encoder-decoder latent variable models. We also test our model on the Recipe1M, the large-scale food image/recipe dataset, where the learned factors by our approach highly coincide with the most pronounced food factors including *savoriness*, *wateriness*, and *greenness*.

Notations and background on embedding. Many approaches in cross-modal retrieval [4, 13, 37, 31, 33, 27, 6, 5, 28, 23, 38], including our model, adopt the so called *shared space embedding* strategy, introducing mappings from ambient data to the shared space $\mathcal{V}$. More specifically, $\mathbf{v}_i = \mathbf{e}_i(\mathbf{x}_i)$ for $i = 1, 2$, are the low-dim embeddings induced by the deep nonlinear (neural net) functions $\mathbf{e}_i(\cdot)$. The Cos-Sim model, for instance, learns the embedding functions by enforcing the embedded vectors of the paired data to be aligned well, with the cosine-angle of two embeddings $\cos(\mathbf{v}_1, \mathbf{v}_2)$ as the alignment score. Its latent variable model extension discussed previously, introduces additional bottleneck encoding/decoding layers, $\mathbf{v}_1 \to \mathbf{z}$ and $\mathbf{z} \to \mathbf{v}_1'$, where the relevance score is measured by $\cos(\mathbf{v}_1', \mathbf{v}_2)$. Then the latent variables $\mathbf{z}$ can be controlled directly (e.g., latent traversal). Instead of this ad-hoc model, in the next section we build the identifiable probabilistic latent variable model (IVAE) on the embedded space, while we avoid the ambient data synthesis module via Jacobian-based regularization of the embedding function.

2 Implicit Identifiable Retrieval VAE

Our model is motivated from the recent work of the Identifiable VAE (IVAE for short) [14], in which the true model can be provably identified by having a latent variable model with an auxiliary inputs on which the latent variables are conditioned. We modify this model to circumvent difficult synthesis model learning in the cross-modal retrieval setup (dubbed **Retrieval IVAE** or **RIVAE** for short). Considering that our goal is to learn the underlying disentangled latent representation that explains the sources of data variability, the provable model identifiability allows us to recover true disentangled latent factors accurately, provided that the data generating process admits disentangled factors.

Unlike conventional VAE models, the IVAE adopts the so-called auxiliary input $\mathbf{u}$ on which the latent $\mathbf{z}$ is conditioned a priori, i.e., $P_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$. In our retrieval setup, we regard the query $\mathbf{v}_1$ as the auxiliary input, and treat the target embedding $\mathbf{v}_2$ as the observed data. There are several modeling assumptions to make the IVAE model fully identifiable from the augmented data $\{(\mathbf{v}_1, \mathbf{v}_2)\}$. Among others, the most important are: 1) the conditional prior $P_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{v}_1)$ needs to be a (conditionally) factorized exponential family distribution with natural parameters $\boldsymbol{\lambda}$, and 2) the output decoding process $P_{\boldsymbol{\theta}}(\mathbf{v}_2|\mathbf{z})$ is homo-scedastic (i.e., $\mathbf{v}_2 = \mathbf{f}(\mathbf{z}) + \boldsymbol{\xi}$ where $\mathbf{f}(\mathbf{z}) = \mathbb{E}[\mathbf{v}_2|\mathbf{z}]$ and $\boldsymbol{\xi}$ is 0-mean random and independent of $\mathbf{z}$). Then we define the following components to meet the IVAE's modeling assumptions:

$$P_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{v}_1) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{v}_1), \text{Diag}(\boldsymbol{\sigma}(\mathbf{v}_1))) \tag{1}$$

$$P_{\boldsymbol{\theta}}(\mathbf{v}_2|\mathbf{z}) = \mathcal{N}(\mathbf{f}(\mathbf{z}), \eta^2\mathbf{I}) \tag{2}$$

$$Q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2) = \mathcal{N}(\mathbf{m}(\mathbf{v}_1, \mathbf{v}_2), \text{Diag}(\mathbf{s}(\mathbf{v}_1, \mathbf{v}_2))) \tag{3}$$

where $\boldsymbol{\mu}(\cdot)$, $\boldsymbol{\sigma}(\cdot)$, $\mathbf{f}(\cdot)$, $\mathbf{m}(\cdot,\cdot)$, and $\mathbf{s}(\cdot,\cdot)$ are all deep neural networks (whose parameters are denoted by $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, resp.) and η^2 is small fixed noise variance (e.g., $\eta = 10^{-3}$). Hence, if the query embedding (auxiliary input) $\mathbf{v}_1$ retains all salient information about the underlying shared factors[1], then the true process of $\mathbf{v}_2|\mathbf{v}_1$, presumably following the model structure of (1–3), can be identified accurately (due to the Theorem 1 in [14]), and so are the true factors $\mathbf{z}$.

To train the RIVAE, we first consider the lower bound (ELBO or $\mathcal{L}_{\text{LB}}$) of the data likelihood $\log P(\mathbf{v}_2|\mathbf{v}_1)$, where

$$\mathcal{L}_{\text{LB}} = \text{KL}(Q(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2)||P(\mathbf{z}|\mathbf{v}_1)) - \mathbb{E}_Q[\log P(\mathbf{v}_2|\mathbf{z})] \tag{4}$$

[1] This will be achieved by the cross-modal retrieval loss (10).

Although we maximize the ELBO wrt the IVAE parameters $\mathbf{\Lambda} := (\boldsymbol{\lambda}, \boldsymbol{\theta}, \boldsymbol{\phi})$, it cannot be used to optimize the embedding networks. This is because the embedding networks can determine the auxiliary input $\mathbf{v}_1$ and the target observed $\mathbf{v}_2$ in an arbitrary way just to make the IVAE model fit to $(\mathbf{v}_1, \mathbf{v}_2)$ very well. As an extreme case, the constant mappings $\mathbf{e}_1(\cdot) = \mathbf{e}_2(\cdot) = \mathbf{v}^{\text{const}}$, can lead to a perfect-fit model that is degenerate. To learn the embedders, we introduce another learning criterion related to cross-modal prediction.

Let $(\mathbf{x}_1, \mathbf{x}_2)$ be a matching pair from the training data, while $(\mathbf{x}_1, \mathbf{x}_2')$ a mismatch one. It is desired for the model to score them as:

$$\log P(\mathbf{x}_2|\mathbf{x}_1) \gg \log P(\mathbf{x}_2'|\mathbf{x}_1). \tag{5}$$

To avoid difficult synthesis modeling for ambient data $\mathbf{x}_2$ in (5), we utilize the *rule of the change of random variables*. Assuming that the embedding function $\mathbf{e}_2(\cdot)$ is *injective*[2],

$$\begin{aligned} \log P(\mathbf{x}_2|\mathbf{x}_1) &= \log P(\mathbf{e}_2^{-1}(\mathbf{v}_2)|\mathbf{x}_1) && (6)\\ &= \log P(\mathbf{v}_2|\mathbf{x}_1) + \log \text{vol}\nabla\mathbf{e}_2(\mathbf{x}_2) && (7)\\ &\approx \log P(\mathbf{v}_2|\mathbf{x}_1) + \text{const}. && (8) \end{aligned}$$

To establish (8), we will enforce the *Jacobian volume*, $\text{vol}\nabla\mathbf{e}_2(\mathbf{x}_2)$ to be constant over different points $\mathbf{x}_2$ as regularization (See below for details). Using

$$P(\mathbf{v}_2|\mathbf{x}_1) = P(\mathbf{v}_2|\mathbf{v}_1) = \mathbb{E}_{P(\mathbf{z}|\mathbf{v}_1)}[P(\mathbf{v}_2|\mathbf{z})], \tag{9}$$

the desiderata (5) can be encoded as the following loss:

$$\mathcal{L}_{\text{Retr}} = \big(1 + \mathbb{E}_Q[\log P(\mathbf{v}_2'|\mathbf{z}) - \log P(\mathbf{v}_2|\mathbf{z})]\big)_+ \tag{10}$$

where $\mathbf{v}_2' = \mathbf{e}_2(\mathbf{x}_2')$ is the embedding of the mismatch sample $\mathbf{x}_2'$, and $(a)_+ = \max(0, a)$. That is, we implicitly invert the embedding function without a synthesis model for $\mathbf{x}_2$.

Regularizing the embedding network. Derivation (8) is valid only if $\text{vol}\nabla\mathbf{e}_2(\mathbf{x}_2)$ remains (approximately) constant across all plausible data samples $\mathbf{x}_2$. Directly regularizing this by minimizing a loss like $(\text{vol}\nabla\mathbf{e}_2(\mathbf{x}_2) - c)^2$ for some constant c, is computationally prohibitive. However, since the Jacobian volume essentially measures the change in the function output due to small perturbations in the input, we can attain similar effect by enforcing the change originating from a random input perturbation[3] to remain constant regardless of the input point. Specifically, we impose the following regularization for $\mathbf{e}_2(\cdot)$:

$$\mathcal{L}_{\text{Reg}} = \mathbb{E}_{\mathbf{x}_2, \boldsymbol{\epsilon}}\Big[\big(||\mathbf{e}_2(\mathbf{x}_2) - \mathbf{e}_2(\mathbf{x}_2 + \boldsymbol{\epsilon})|| - c\big)^2\Big] \tag{11}$$

where $\boldsymbol{\epsilon} \sim P(\boldsymbol{\epsilon})$ is a random sample from a noise distribution $P(\boldsymbol{\epsilon})$ with small magnitude (e.g., $||\boldsymbol{\epsilon}|| = 0.001$). We can also optimize c.

Summary. The full training can be written as the following optimization (λ_{Retr} and λ_{Reg} are the trade-off parameters). Note that the arguments in each loss $\mathcal{L}(\cdot)$ indicates which parameters should be updated regarding the loss.

$$\min\ \mathcal{L}_{\text{LB}}(\mathbf{\Lambda}) + \lambda_{\text{Retr}}\mathcal{L}_{\text{Retr}}(\mathbf{\Lambda}, \mathbf{W}) + \lambda_{\text{Reg}}\mathcal{L}_{\text{Reg}}(\mathbf{W}) \tag{12}$$

Cross-modal Retrieval. Basically we need to solve: $\arg\max_{\mathbf{x}_2 \in \mathcal{D}_2} \log P(\mathbf{x}_2|\mathbf{x}_1)$, where $\mathbf{x}_1$ is the query data point, and $\mathcal{D}_2$ is the search database. Using the derivation of (7–8), $\log P(\mathbf{x}_2|\mathbf{x}_1)$ can be approximated as $\log P(\mathbf{v}_2|\mathbf{z})$ with $\mathbf{z} \sim P(\mathbf{z}|\mathbf{v}_1)$, which results in the following three-step algorithm:

1. Embed the query point: $\mathbf{v}_1 = \mathbf{e}_1(\mathbf{x}_1)$.
2. Sample $\mathbf{z} \sim P(\mathbf{z}|\mathbf{v}_1)$.
3. Solve: $\arg\max_{\mathbf{x}_2 \in \mathcal{D}_2} \log P(\mathbf{e}_2(\mathbf{x}_2)|\mathbf{z})$.

Latent Traversal. While varying the value of one particular latent dimension with the rest being fixed, we inspect the change in the retrieved data. This helps us understand what type of aspect each latent dimension corresponds to, that is, the source of variability. We retrieve the output for each traversed point $\mathbf{z}$, namely the mode of $P(\mathbf{x}_2|\mathbf{z})$. We regard the conditional distribution $P(\mathbf{v}_2|\mathbf{z})$ as a proxy of $P(\mathbf{x}_2|\mathbf{z})$ using the derivation similar to (7–8) with the regularized embedding networks. The latent traversal algorithm is summarized as follows:

[2]Formally, $\mathbf{e}_2(\mathbf{x}_2) \neq \mathbf{e}_2(\hat{\mathbf{x}}_2)$ if $\mathbf{x}_2 \neq \hat{\mathbf{x}}_2$. It allows the inverse $\mathbf{e}_2^{-1}(\cdot)$ to be defined. It is a reasonable assumption that is also considered in [14].

[3]The input space perturbation can be done pixel-wisely for images, and on the word vectors for text data.

1. Traverse $\mathbf{z}$ along the j-th dim ($j = 1, \ldots, d$).
2. Prepare the conditional distribution: $P(\mathbf{v}_2|\mathbf{z})$.
3. Solve: $\arg\max_{\mathbf{x}_2 \in \mathcal{D}_2} \log P(\mathbf{e}_2(\mathbf{x}_2)|\mathbf{z})$.

Comparison to Conditional VAE [30]. The idea of conditioning the latent variables $\mathbf{z}$ is similar to the Conditional VAE (CVAE) [30]. However, there are two main differences: i) The conditioning variables in the CVAE are typically class labels, and its main goal is to enrich the representational capacity of VAE to cover different regions/clusters of the data domain indexed by the class label. ii) CVAE explicitly models dependency between the observed and the conditioning variables. On the other hand, in our RIVAE, there is no direct link between the output $\mathbf{v}_2$ and the auxiliary $\mathbf{v}_1$, thus enforcing core information from $\mathbf{v}_1$ to $\mathbf{v}_2$ to flow through the bottleneck $\mathbf{z}$. This allows the latents to learn the salient factors more effectively.

3 Experimental Results

We test our Retrieval-IVAE model on both controlled datasets where the ground-truth factors are fully/partially available for quantitative comparison, and the large-scale Recipe1M dataset [28, 23] in the context of food image to recipe retrieval. We especially focus on demonstrating our model's capability of identifying the underlying true factors with high degree of disentanglement, compared to the existing approaches and baselines (See Sec. 3.1).

For the cross-modal retrieval at test time, we construct the search database $\mathcal{D}_2$, of size $|\mathcal{D}_2| = 1000$ or 2000, randomly selected from the test dataset. We repeat this procedure randomly 10 times, and run the models to report average performance. For the retrieval metrics, we consider the median rank (Med-R) and the recall-at-K ($R@K$) with $K = 1, 5, 10$, where $R@K$ stands for the fraction (out of $|\mathcal{D}_2|$ queries) where the true item is found by the model in its top-K scored items. Although these retrieval scores are indicative of how well the models extract shared information, the main focus in this paper is to judge the goodness of the learned latent factors. For this purpose, see the quantitative metrics in Sec. 3.2.

3.1 Competing Methods and Datasets

Our RIVAE is compared with the following methods:

- **Cos-Sim-LVM**: As described previously, we extend the Cos-Sim embedded space alignment method to identify/control the latent variables. That is, this extended model has encoder ($\mathbf{v}_1 \to \mathbf{z}$) and decoder ($\mathbf{z} \to \mathbf{v}_1'$), both of which roughly correspond to $P(\mathbf{z}|\mathbf{v}_1)$ and $P(\mathbf{v}_2|\mathbf{z})$ in our RIVAE, respectively, and we use neural nets with similar architectures for fair comparison.
- **Bi-VAE**: This is the bi-modal extension of the VAE via the product-of-experts approximation [36]. It requires difficult synthesis (decoder) model learning for the ambient data (i.e., $\mathbf{z} \to \mathbf{x}_1$ and $\mathbf{z} \to \mathbf{x}_2$).
- **Bi-VAE-on-$\mathcal{V}$**: As a reasonable workaround to circumvent the difficult synthesis model learning in Bi-VAE, we can think of building the bi-modal VAEon the embedded space ($\mathcal{V}$) instead. That is, the embedding networks are fixed (e.g., simply borrowed from the trained Cos-Sim model). The intuition is to regard the embeddings $\mathbf{v}_{1/2}$ as proxy for ambient data $\mathbf{x}_{1/2}$, and a similar idea was previously explored in [29].
- **DCCA**[4]: The Deep CCA model [1] that learns the nonlinear mapping from inputs to the embedding vectors.
- **RBi-VAE**: We consider a *joint* model $P(\mathbf{v}_1, \mathbf{v}_2, \mathbf{z})$ in place of the *conditional* $P(\mathbf{v}_2, \mathbf{z}|\mathbf{v}_1)$ in our RIVAE. (Supplement for details) Dubbed Retrieval-Bi-VAE (or RBi-VAE for short), we adopt the same Jacobian embedder regularization for the implicit embedder inversion, and it is compared to RIVAE to see how effective the identifiable model learning in RIVAE.

For fair comparison, we make the experimental setup as equal as possible for all competing methods. E.g., the number of latent variables adopted in the competing models is set to be the same. The details of the model architectures and optimization strategies are described in the Supplement.

And we test the above models on four datasets:

- **Synth** (Sec. 3.3): The synthetic data generated from a nonlinear function with disentangled latent variables partitioned into shared and private factors.
- **Sprites** (Sec. 3.4): Binary images of sprites. The shape is regarded as modality, where the locations and size of the sprite are considered as shared factors.

[4] Other variants including DCCAE (DCCA with the auto-encoding loss) [35], are often on a par with DCCA, and not considered here.

Table 1: (Synth) Retrieval performance with 1000 test samples as the search set. (averaged over 10 random runs).

Method	$R@1 \uparrow$	$R@5 \uparrow$	$R@10 \uparrow$	Med-R $\downarrow$
Cos-Sim-LVM	0.02	0.14	0.27	24.00
Bi-VAE [36]	0.01	0.05	0.10	101.00
Bi-VAE on $\mathcal{V}$	0.20	0.58	0.76	4.00
DCCA [1]	0.29	0.62	0.81	3.00
RBi-VAE	0.33	0.81	0.92	2.00
RIVAE	0.35	0.80	0.92	2.00

- **Split-MNIST** (Sec. 3.5): The bi-modal data created from the MNIST dataset by splitting images into left (modality-1) and right (modality-2) halves.
- **Recipe1M** (Sec. 3.6): The large scale food dataset [28, 23] that consists of pairs of food image and recipe text including title, ingredient list, and instructions.

3.2 Metrics for Goodness of Learned Latents

We define metrics for the goodness of the learned latents, including the degree of disentanglement and completeness. Our quantitative measures are similar to the D/C/I metrics [10], but whereas in [10] they assume an offline data pool for which all factor labels are available, but in our case, not all factor labels are available for entire data instances for some datasets. To this end, we modify the original metrics, adapted to the latent traversal results as follows.

For each reference/query $\mathbf{x}_1^{\text{Ref}}$, we embed/encode it to obtain the latent vector $\mathbf{z}^{\text{Ref}}$. Then from this anchor point $\mathbf{z}^{\text{Ref}}$, we traverse along the z_j axis for each latent dimension $j = 1, \ldots, d$, i.e., varying the value of z_j while freezing the rest dimensions. We then collect a set of retrieved items $\mathbf{x}_2$, each of which corresponds to each of the traversed points along z_j axis. And for each retrieved item, its true factor values $[f_1, \ldots, f_K]$ are looked up. This way, we collect the paired (z_j, f_k) data, and estimate the Pearson's correlation coefficient. In particular, we deal with $c_{jk} = |\text{Corr}(z_j, f_k)|$, the absolute correlation score, where the larger c_{jk} implies that the latent variable z_j is more related to the true factor f_k. We repeat this procedure for many reference/query points, and take the average scores c_{jk}, which helps marginalizing out the instance-specific impact on the correlations between latents and true factors.

Then with the $(d \times K)$ correlation table $\mathbf{c}$ (whose (j,k)-entry is c_{jk}), we measure the three metrics similarly as [10]. First, the *Disentanglement* metric (D) measures the degree of dedication of each latent variable z_j in predicting f_k against others f_{-k}. Intuitively, we have perfect disentanglement if each latent variable z_j is correlated with only a single true factor f_k. On the other hand, if z_j is correlated with multiple true factors at the same time, it is deemed entangled. To capture this idea, each j-th row of the table $\mathbf{c}$ is normalized to a probability distribution, specifically,

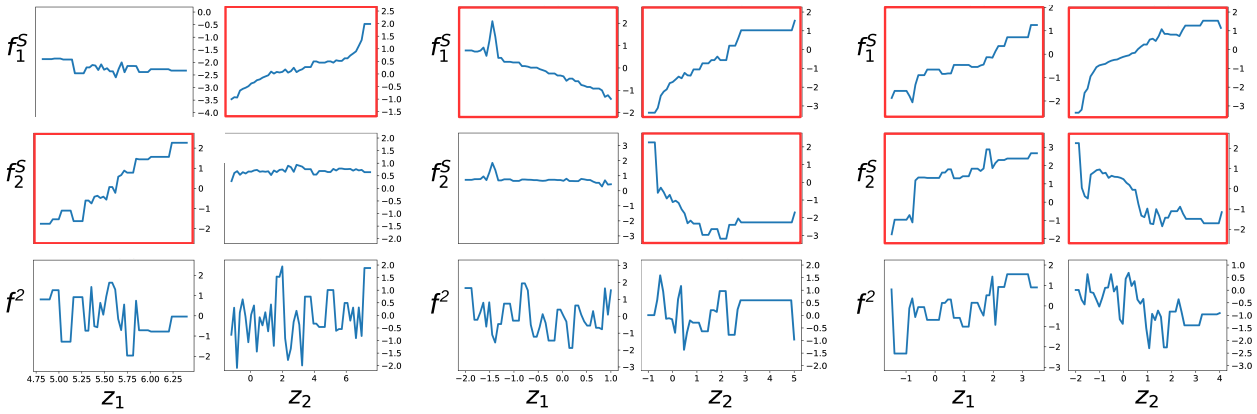
$$p_{jk} = \frac{\rho(c_{jk})}{\sum_{k'=1}^{d} \rho(c_{jk'})}, \tag{13}$$

where $\rho()$ is positive monotonic increasing (e.g., $\rho(c) = e^{\alpha c}$ for some $\alpha > 0$). Then we compute the normalized ($[0,1]$-scaled) entropy, $H_j = -(1/\log K) \sum_k p_{jk} \log p_{jk}$, and we define $D = 1 - (1/d) \sum_{j=1}^{d} H_j$ (the higher, the better disentangled). The *Completeness* metric (C) captures the degree of exclusive contribution of z_j in predicting f_k against others z_{-j}. As it essentially aims for minimal redundancy, we would achieve perfect completeness if variability of each true factor f_k is explained by only a single latent variable z_j, instead of multiple latents. The metric C can be computed similarly as D, by replacing all row-wise operations with column-wise. Likewise, the higher C is, the better. Lastly, the *Informativeness* metric (I) measures how informative each latent variable is in predicting a true factor. To this end, for each f_k, we find the best predictor z_j (i.e., $c_k^* = \max_{1 \le j \le d} c_{jk}$), and define I as the average of c_k^* over $k = 1, \ldots, K$.

3.3 Synthetic Data (Synth)

To demonstrate our model's capability of identifying the shared latent factors from bi-modal data, we devise a synthetic data setup as follows. First we generate 4-dim factors, $\mathbf{f} = [\mathbf{f}^S, f^1, f^2]$ which are all iid samples from $\mathcal{N}(0,1)$ with $\dim(\mathbf{f}^S) = 2$ and $\dim(f^1) = \dim(f^2) = 1$. The ambient data points are then generated by the nonlinear functions, $\mathbf{x}_1 = \mathbf{G}_1(\mathbf{f}^S, f^1)$ and $\mathbf{x}_2 = \mathbf{G}_2(\mathbf{f}^S, f^2)$ where $\mathbf{G}_1(\cdot)$ and $\mathbf{G}_2(\cdot)$ are two-layer neural networks that output 50-dim vectors. So our intention is that $\mathbf{f}^S$ serves as the *shared factors* that govern the data variability in the two modalities, while f^1 and f^2 are the *private factors* that only affect individual modalities, being independent on the other. Additionally, it would be desirable if the model can further disentangle the two individual factors in $\mathbf{f}^S$.

For the embedding networks $\mathbf{e}_{1/2}(\cdot)$, we use two-layer neural networks, and the embedding dimension is set as $\dim(\mathbf{v}_{1/2}) = 3$. We set $\dim(\mathbf{z}) = 2$ for the latent spaces, which matches the number of true shared factors. The retrieval performance of



(a) RIVAE (b) RBi-VAE (c) DCCA

Figure 1: (Synth) True factors vs. latent variables. Each column shows traversal of one latent z_j with the other fixed. The Y axes are true factors obtained from retrieved items $\mathbf{x}_2$. Red boxes indicate significant changes in the true *shared factors*, i.e., high correlation.

Table 2: (Synth) Goodness of the learned latents.

	Disent. ↑	Comple. ↑	Inform. ↑
Cos-Sim-LVM	0.6169	0.7613	0.5486
Bi-VAE	0.4429	0.6822	0.6902
Bi-VAE on $\mathcal{V}$	0.0520	0.8725	0.5630
DCCA	0.0017	0.2766	0.1371
RBi-VAE	0.0063	0.6900	0.3923
RIVAE	**0.9186**	**0.8782**	**0.8995**

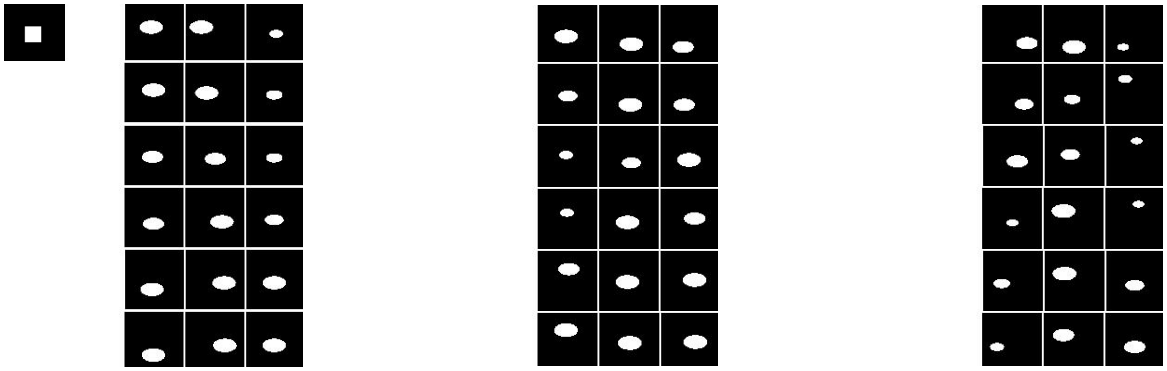
the competing models is summarized in Table 1. Our RIVAE outperforms Cos-Sim-LVM and DCCA. The poor performance of the Bi-VAE model implies that suboptimally trained decoders for high-dimensional noisy ambient data can degrade the retrieval performance significantly. Even the Bi-VAE-on-$\mathcal{V}$ trained on the fixed embedded space exhibits performance comparable to Cos-Sim-LVM, although it still underperforms our model.

Next we inspect the learned latent representations. We are especially interested in the correspondence between the learned latent variables $\mathbf{z}$ and the true shared factors $\mathbf{f}^S$. A desirable result would be exclusive one-to-one correspondence, where a single latent variable z_j affects only one shared factor, independent from the other. To this end, we perform latent traversal. For the retrieved item $\mathbf{x}_2$, we look up its true factor values, (f_1^S, f_2^S, f^2), and plot each against z_j. The results are shown in Fig. 1. Notably for our RIVAE, each latent variable corresponds to only one true shared factor exclusively, implying that the disentangled factors are accurately identified. RBi-VAE partially identifies f_1^S in z_1, but both shared factors are entangled in z_2. We also run the latent traversal with the DCCA, which is done by converting the learned CCA model to the dual-view latent variable linear Gaussian model following [2]. However, as this maximum-likelihood estimated model only fits well to the data, we see that each of the learned latent variables retain both factors entangled in it. Finally, the quantitative D/C/I metrics in Table 2 show that our RIVAE yields significantly better latent representations than competing models.

3.4 Sprites

Using the benchmark dSprites dataset [25], we devise an experimental setup for the bi-modal retrieval task. First, we assume that the shape of sprites induces the modalities, specifically, $\mathbf{x}_1$ is *square*, and $\mathbf{x}_2$ *oval*. We then consider only the X, Y positions and the scale of the sprite as the underlying shared factors, with the other factors being fixed. There are 32 variations in each of the X, Y positions and 6 variations in scale, which are independent from one another, resulting in 6144 samples. Image size is (64×64) pixels. The dimension of the embedded space $\mathcal{V}$ is set to 10, and the latent space $\dim(\mathbf{z}) = 3$, which matches ground-truth.

The retrieval results on 1000 randomly selected search set are summarized in Table 3. Most approaches yield near perfect performance except for the decoder training models (Bi-VAE and Bi-VAE-on-$\mathcal{V}$), while our RIVAE performs marginally the best. The plots of the ground-truth factors of the retrieved images due to the latent space traversal, similar to Fig. 1, can be found in the Supplement, where our RIVAE shows near one-to-one correspondence between the true and learned factors, while other models exhibit considerable entanglement. We also visualize the retrieved images obtained by latent traversal in Fig. 2. The result again verifies the high quality of disentanglement in the learned factors, very close to the true latent factors.



(a) RIVAE (b) RBi-VAE (c) DCCA

Figure 2: (Sprites) Retrieved images from latent traversal. The top left image is the query image that determines the reference point $\mathbf{z}^{\text{Ref}}$ (the same for all models). For each model, each of the three columns depicts the retrieved images due to z_1, z_2, and z_3 changes (progresses vertically). Visually, it is clear that with RIVAE, varying z_1 alone results in change in the Y-pos with the scale and X-pos intact, z_2 exclusively affects the X-pos, and z_3 only affects the scale. Such interpretation is not clear for other models.

Table 3: (Sprites) Retrieval performance among a 1000 randomly selected search set (averaged over 10 random runs).

Method	R@1 ↑	R@5 ↑	R@10 ↑	Med-R ↓
Cos-Sim-LVM	0.13	0.46	0.67	6.00
Bi-VAE [36]	0.02	0.07	0.10	127.75
Bi-VAE on $\mathcal{V}$	0.69	0.97	0.99	1.00
DCCA [1]	0.96	1.00	1.00	1.00
RBi-VAE	0.99	1.00	1.00	1.00
RIVAE	1.00	1.00	1.00	1.00

Table 4: (Sprites) Goodness of the learned latents.

	Disent. ↑	Comple. ↑	Inform. ↑
Cos-Sim-LVM	0.6664	0.5634	0.8417
Bi-VAE	0.6821	0.0779	0.1259
Bi-VAE on $\mathcal{V}$	0.1367	0.6812	0.3119
DCCA	0.4163	0.1178	0.1432
RBi-VAE	0.5814	0.4863	0.7234
RIVAE	**0.8280**	**0.9133**	**0.8476**

The D/C/I scores in Table 4 also support this claim quantitatively.

3.5 Split-MNIST

Following the setup in [1], we form a retrieval setup from the MNIST dataset [20] by taking the left half of each image as modality-1 and the right half as modality-2. So, each view contains images of size ($H = 28 \times W = 14$) pixels. Note that the shared factors for both modalities would be the digit class and the writing style, which are deemed independent (disentangled) from each other. We followed the standard data splits, where 2000 images are randomly sampled from the test set to serve as the retrieval search set.

We set $\dim(\mathcal{V}) = 50$ and $\dim(\mathbf{z}) = 10$. The retrieval scores are reported in Table. 5. Consistent with previous experiments, our RIVAE attains the best scores. To see how the two underlying factors, writing style and digit class, are disentangled in the learned latent variables for our RIVAE, we show the latent traversal results visually in Fig. 3. Refer to the caption of the figure for details.

3.5.1 Quantitative Analysis for Split-MNIST

Although we know that the shared factors between the left and right halves deem to be partitioned to those related to digit class and those non-digit related (e.g., writing style), there are only digit labels ($0 \sim 9$) available, and it is not even clear how the writing style can be formally described or specified. Hence, we devise some reasonable quantitative measures that can reflect how the underlying variability in digit transitions is captured in the model's latent variables.

Table 5: (Split-MNIST) Retrieval performance with 2000 randomly selected test images as the search set.

Method	$R@1 \uparrow$	$R@5 \uparrow$	$R@10 \uparrow$	Med-R $\downarrow$
Cos-Sim-LVM	0.19	0.46	0.61	6.35
Bi-VAE [36]	0.01	0.03	0.05	524.80
Bi-VAE on $\mathcal{V}$ [36]	0.21	0.51	0.66	5.30
DCCA [1]	0.47	0.79	0.87	2.00
RBi-VAE	0.43	0.81	0.91	2.00
RIVAE	0.52	0.89	0.96	1.00

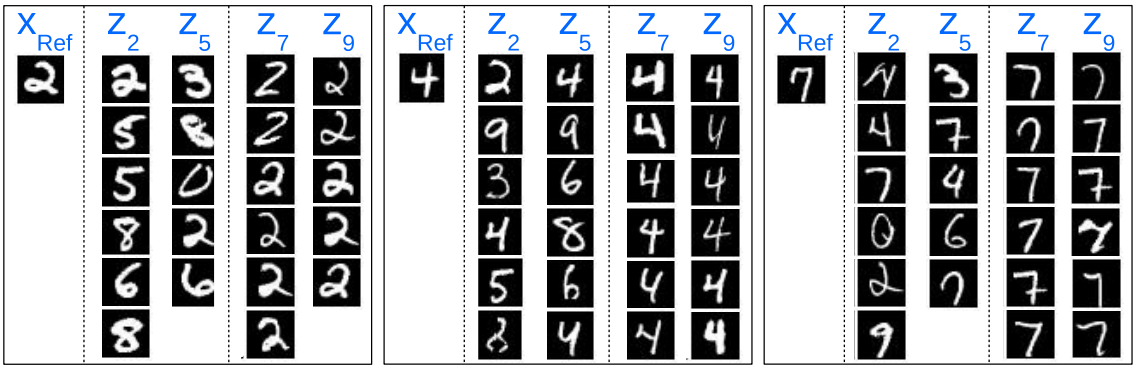


Figure 3: (Split-MNIST) For three query references, latent traversal along four latent variables (z_2, z_4, z_7, z_9) in RIVAE. Visually, z_2, z_5 = digit class, z_7, z_9 = writing style. See Supplement for other query references.

Let's say that we traverse along the axis z_j in the latent space, while fixing the rest dimensions of $\mathbf{z}$. After retrieving the data items, we record the number of unique digit transitions (e.g., $0 \rightarrow 2$ or $3 \rightarrow 8$) in the retrieved images in the traversal. The results, over all dimensions, can be summarized into a $(d \times 45)$ table, where $d = \dim(\mathbf{z})$ and 45 $(= 10C2)$ is the number of direction-free transitions (e.g., $0 \rightarrow 2$ and $2 \rightarrow 0$ are regarded identical). Note that discarding the directions makes sense considering traversal in the reverse direction. More concretely, if the digit classes of the retrieved items for the traversal z_5 are: $2 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 8 \rightarrow 9 \rightarrow 3 \rightarrow 2 \rightarrow 2 \rightarrow 2$, then the fifth row of the table has value 1 at four columns, $2 \rightarrow 3$, $3 \rightarrow 8$, $8 \rightarrow 9$, $3 \rightarrow 9$, with all the other entries 0. There will be one such table for each reference query ($\mathbf{x}_1^{\text{Ref}}$). And we will collect many (e.g., 1000) such tables/queries, and take the average to get the global statistics.

The key idea is that this (averaged) digit transition table can tell us possible digit transition types specific to each latent dimension. For instance, z_0 has large entries in the table for the transitions among digits $(1, 4, 7)$, while z_1 covers the clique $(2, 3, 5, 8)$, and so on. And we expect that if latent factors are well trained, the transition cliques for different dimensions tend to be less overlapped, and overall the union of the cliques has good coverage of all 45 possible transitions. The former is related to *disentanglement* of the latent variables, and the latter being *thoroughness* or *coverage*, how many different transitions the learned latent variables can capture.

To formalize, let's say $D = (d \times 45)$ be the table of averaged digit transitions. We normalize each row of D as a probability distribution. Then we measure the overlap between two rows i and j, e.g., $overlap(i, j) = \frac{1}{45} \sum_{k=1}^{45} \min(D[i, k], D[j, k])$, similarly as the histogram intersection. Then we measure the average overlap over all $i \neq j$. The smaller the overlap is, the better. And, for the *coverage*, we take the union of the rows, e.g., simply the average of the rows in D. Then we compute the entropy of the union. The larger the entropy is, the union covers more digit transitions. The results are summarized in Table 6, and we see that the learned latent variables in our RIVAE exhibit low overlap and the highest coverage among others.

Table 6: (Split-MNIST) Goodness of the learned latents. Two quantitative measures regarding the variability in digit transitions. Interpretation: *overlap* is related to disentanglement (the lower, the better), and *coverage* to thoroughness (the higher, the better). The differences in the last column.

	Overlap $\downarrow$	Coverage $\uparrow$	$C - O \uparrow$
Cos-Sim-LVM	0.0168	0.8881	0.8713
Bi-VAE	**0.0091**	0.4863	0.4772
Bi-VAE on $\mathcal{V}$	0.0181	0.6418	0.6237
DCCA	0.0191	0.6377	0.6186
RBi-VAE	0.0189	0.8932	0.8743
RIVAE	0.0121	**0.9675**	**0.9554**

Table 7: (Recipe1M) Retrieval performance with size 1000 random search set (averaged over 10 random runs).

Method	$R@1 \uparrow$	$R@5 \uparrow$	$R@10 \uparrow$	Med-R $\downarrow$
Cos-Sim-LVM	0.45	0.74	0.82	2.00
Bi-VAE [36]	Failed	Failed	Failed	Failed
Bi-VAE on $\mathcal{V}$ [36]	0.22	0.45	0.55	7.70
DCCA [1]	Failed	Failed	Failed	Failed
RBi-VAE	0.29	0.56	0.66	4.00
RIVAE	0.39	0.70	0.79	2.00

Table 8: (R1M) Goodness of the learned latents.

	Disent. $\uparrow$	Comple. $\uparrow$	Inform. $\uparrow$
Cos-Sim-LVM	0.6203	0.6027	0.5983
Bi-VAE	Failed	Failed	Failed
Bi-VAE on $\mathcal{V}$	0.5128	0.5423	0.5670
DCCA	Failed	Failed	Failed
RBi-VAE	0.4954	0.6079	0.5664
RIVAE	**0.8569**	**0.8500**	**0.8615**

3.6 Food Image to Recipe Retrieval (Recipe1M)

Recipe1M [28] is the dataset comprised of about 1M cooking recipes (titles, instructions, ingredients) and images. In this work, a subset of about 0.4M recipes containing at least one image, no more than 20 ingredients or instructions, and at least one ingredient and instruction was used. Data is split into $70\%/15\%/15\%$ train/validation/test sets. The underlying embedding networks used in this experiments combines architectural and training strategies from [28, 34, 7], and the related Cos-Sim model performs comparable to the state-of-the-arts. The embedded space has $\dim(\mathcal{V}) = 1024$, and the latent variables $\dim(\mathbf{z}) = 30$. Table 7 shows retrieval performance. The Cos-Sim-LVM attains the best retrieval performance even with the introduced bottleneck layers, where our RIVAE performs nearly comparably to it. Note that both DCCA [1] and Bi-VAE [36] completely failed to converge.

3.6.1 Quantitative Analysis for Recipe1M

In the Recipe1M dataset, there are no labels available for the ground-truth factors, those that commonly govern the variability of recipes and food images, deemed to be *food factors*. To measure the goodness of the learned latent variables of the competing methods, we aim to select a small subset of food factors that look the most pronounced and capturing the shared variability in recipes and images. To this end, we re-crape all the recipes in the Recipe1M that are associated with the Internet domain `food.com`, and parse their keywords and categories. Then we form the recipe *tags* as the unique terms union between keywords and categories. This subset represents about 50% of the whole dataset.

Then we manually group the tags that are related to one another, and among the tag groups, we choose 8 factors the most dependent on the latent variables by visually inspecting the latent traversal results for the competing methods. They are: 1) wateriness, 2) greenness, 3) stickiness, 4) oven-baked-or-not, 5) food container longishness (e.g., bottle/cup or plate), 6) grains, 7) savory-or-dessert, and 8) fruit-or-no-fruit. They are intuitively very appealing. For the association between the tags and these 8 factors, refer to the Supplement. All these factors that consider in this analysis are ordinal, and we consider 5 scales/levels for each factor (e.g., wateriness$= 1$ means very dry food, while wateriness$= 5$ implies containing lots of water).

We sample about 20 random reference/query images ($\mathbf{x}_1^{\text{Ref}}$) in the traversal/retrieval. For the 100 traversal points along each latent dimension, we collect retrieved items ($\mathbf{x}_2$), and manually label the values of the 8 food factors. Then we select 10 latent dimensions that have the highest correlations with the 8 factors, and form a (10×8) correlation table. The D/C/I measures are summarized in Table 8. As shown, our RIVAE attains the highest scores among the competing models by large margin.

3.6.2 Qualitative (Visual) Analysis for Recipe1M

Next we qualitatively assess the discovered hidden factors through visual inspection of the retrieved items from latent traversal. We also generate the word cloud images using the ingredients in the retrieved recipes. For the four latent variables that have the highest correlation with the factors: wateriness, greenness, savoriness, and fruit-or-no-fruit, Fig. 4 shows traversal results for top-3 retrieved items, which are visually very coherent to the corresponding true factors.

In Fig. 5, we show the ingredient word clouds generated from the top-10 retrieved items from 20 query data points for the latent variable corresponding to the *savoriness*. We see that the word cloud on the left end contains ingredients

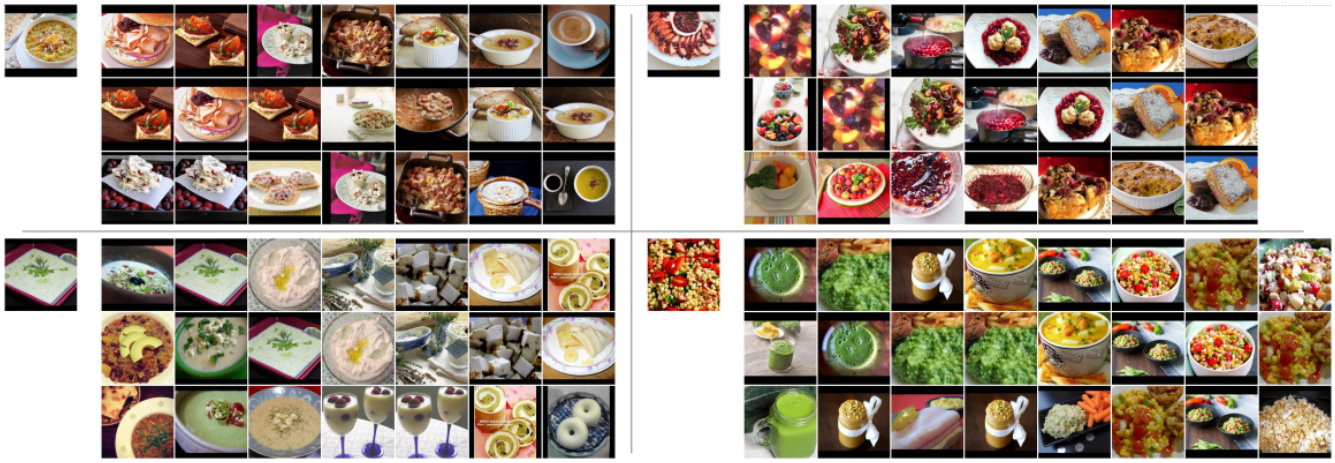


Figure 4: Retrieved images from latent traversal. **Top left** latent variable for *wateriness*, **top right** *fruit-no-fruit*, **bottom left** *savoriness*, and **bottom right** *greenness*. For each panel, the query image is shown on the leftmost, and each column has top-3 retrieved items at each traversal point. Larger images, more examples, and other discovered factors can be found in Supplement.

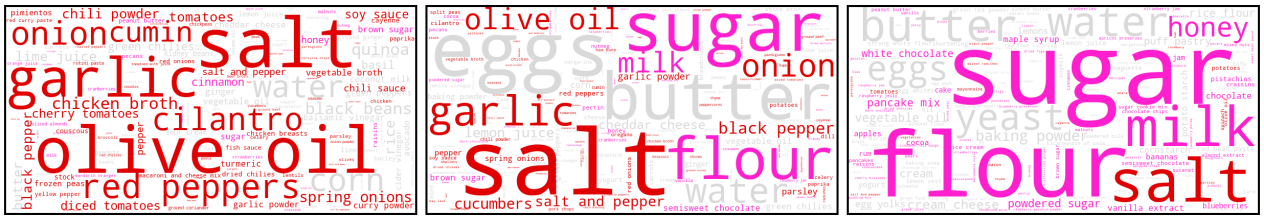


Figure 5: Ingredient word clouds for the latent variable *savoriness*, obtained from top-10 retrieved items over 20 queries. Ingredient color indicates typical use; red = savory and pink = non-savory (sweet).

mostly associated with savory dishes, the middle contains both savory and sweet ingredients, while the right end has typical dessert ingredients. This highlights the main advantage of our model where it accurately identifies true factors, and allows us to directly control the latent variables to generate (retrieve) desired data items. This feature can also be easily extended to *manipulation of multiple factors*. Fig. 6 shows top-5 retrieved examples when we control two or three latent variables corresponding to *savoriness*, *wateriness*, and *greenness*, simultaneously.

3.7 Ablation Study: Embedder Regularization

In our Retrieval-IVAE model, we employed the embedder regularization loss, specifically (11). To verify the impact of this regularization term, we run our RIVAE model without this loss term, and the retrieval results are summarized in Table 9, in conjunction with the results with the embedder regularization. The result demonstrates that without embedding network regularization, either the performance becomes considerably degraded or simply fails.

Table 9: RIVAE with/without embedder regularization.

Dataset	Reg.	$R@1 \uparrow$	$R@5 \uparrow$	$R@10 \uparrow$	Med-R $\downarrow$
Synth	Yes	0.35	0.80	0.92	2.00
	No	0.29	0.73	0.90	3.00
Sprites	Yes	1.00	1.00	1.00	1.00
	No	0.01	0.03	0.06	144.40
Split-MNIST	Yes	0.52	0.89	0.96	1.00
	No	0.35	0.75	0.88	2.00

Figure 6: **Left**: Retrieved images from activation of multiple latent variables. **Middle**: indicates which combinations of latent variables are activated. **Right**: Ingredient word clouds of the retrieved recipes.

4 Conclusion

We have proposed a novel Retrieval-IVAE model, extension of the identifiable VAE for cross-modal retrieval, where it completely removes the ambient data decoding module via implicit encoder inversion that is achieved by Jacobian regularization of the low-dimensional embedding function. The model is shown to have the capability of learning disentangled and complete latent representations that can explain the variability of the bi-modal data, on both controlled and the large-scale Recipe1M datasets. As our future work, we plan to apply our approaches to retrieval tasks with more complex structured data in multiple modalities greater than two, including video, text, and audio data.

References

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis, 2013. International Conference on Machine Learning. 1, 4, 5, 7, 8, 9

[2] F. R. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis, 2005. Technical Report 688, Department of Statistics, University of California, Berkeley. 6

[3] Philemon Brakel and Yoshua Bengio. Learning independent features with adversarial nets for non-linear ICA. In *arXiv preprint*, 2017. 1

[4] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu. Deep visual-semantic hashing for cross-modal retrieval, 2016. In Proceedings of the 22nd ACM SIGKDD. 2

[5] J. Chen, C. W. Ngo, and T. S. Chua. Cross-modal recipe retrieval with rich food attributes, 2017. Proceedings of the 25th ACM international conference on Multimedia. 2

[6] J. Chen, L. Pang, and C. W. Ngo. Cross-modal recipe retrieval: How to cook this dish?, 2017. International Conference on Multimedia Modeling. 2

[7] Jing-Jing Chen, Chong-Wah Ngo, Fu-Li Feng, and Tat-Seng Chua. Deep understanding of cooking procedure for cross-modal recipe retrieval. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 1020–1028. ACM, 2018. 9, 13

[8] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018. 1, 18

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing Generative Adversarial Nets, 2016. In Advances in Neural Information Processing Systems. 1

[10] Cian Eastwood and Christopher K. I. Williams. A framework for the quantitative evaluation of disentangled representations, 2018. In Proceedings of the Second International Conference on Learning Representations, ICLR. 5

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. 1

[12] Geoffrey E. Hinton. Products of experts, 1999. Proceedings of the Ninth International Conference on Artificial Neural Networks. 18

[13] Q. Jiang and W. Li. Deep cross-modal hashing, 2017. IEEE Conference on Computer Vision and Pattern Recognition. 2

[14] Ilyes Khemakhem, Diederik P. Kingma, Ricardo Pio Monti, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework, 2020. Artificial Intelligence and Statistics. 2, 3

[15] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. International Conference on Machine Learning, 2018. 1, 18

[16] Minyoung Kim, Yuting Wang, Pritish Sahu, and Vladimir Pavlovic. Bayes-factor-VAE: Hierarchical Bayesian deep auto-encoder models for factor disentanglement. In *IEEE International Conference on Computer Vision, ICCV*, 2019. 1

[17] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2015. In Proceedings of the Second International Conference on Learning Representations, ICLR. 14

[18] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes, 2014. In Proceedings of the Second International Conference on Learning Representations, ICLR. 1

[19] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variation inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018. 1

[20] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 7

[21] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations, 2019. International Conference on Machine Learning. 1

[22] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016. 1

[23] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1M+: A Dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 1, 2, 4, 5

[24] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling disentanglement in variational autoencoders, 2019. International Conference on Machine Learning. 1

[25] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dSprites: Disentanglement testing Sprites dataset, 2017. 6

[26] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010. 18

[27] Y. Peng, J. Qi, X. Huang, and Y. Yuan. Ccl: Cross-modal correlation learning with multigrained fusion by hierarchical network. *IEEE Transactions on Multimedia*, 20(2):405–420, 2018. 2

[28] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 4, 5, 9, 13

[29] Yuming Shen, Li Zhang, and Ling Shao. Semi-supervised vision-language mapping via variational learning, 2017. International Conference on Robotics and Automation. 4

[30] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models, 2015. In Advances in Neural Information Processing Systems. 4

[31] Shupeng Su, Zhisheng Zhong, and Chao Zhang. Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[32] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the Bregman divergence: A unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012. 18

[33] B. Wang, Y. Yang, X. Xu, A. Hanjalic, and H. T. Shen. Adversarial cross-modal retrieval, 2017. In Proceedings of the ACM on Multimedia Conference. 2

[34] Hao Wang, Doyen Sahoo, Chenghao Liu, Ee-peng Lim, and Steven CH Hoi. Learning cross-modal embeddings with adversarial networks for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11572–11581, 2019. 9, 13

[35] Weiran Wang, Xinchen Yan, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis, 2016. arXiv:1610.03454. 1, 4

[36] Mike Wu and Noah Goodman. Multimodal generative models for scalable weakly-supervised learning, 2018. In Advances in Neural Information Processing Systems. 1, 4, 5, 7, 8, 9

[37] F. Zheng, Y. Tang, and L. Shao. Hetero-manifold regularisation for cross-modal hashing. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1059–1071, 2018. 2

[38] B. Zhu, C. W. Ngo, J. Chen, and Y. Hao. R2gan: Cross-modal recipe retrieval with generative adversarial network, 2019. IEEE Conference on Computer Vision and Pattern Recognition. 2

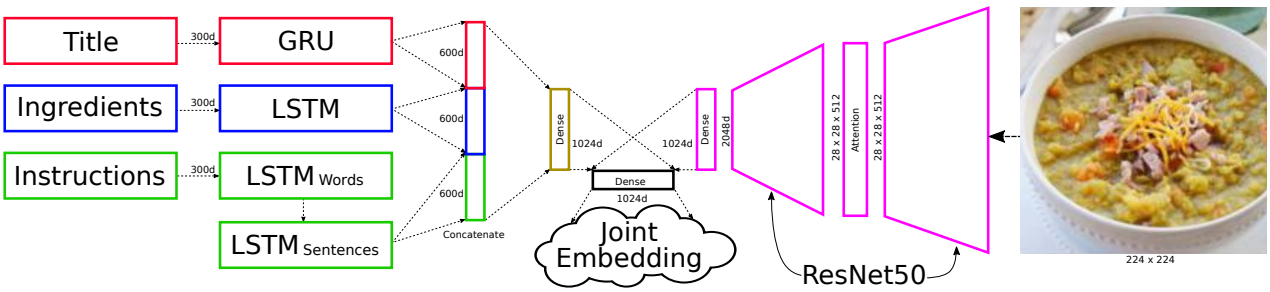


Figure 7: (Recipe1M) Diagram of the embedding networks.

Supplementary Material

This supplement consists of the following materials:

- Experimental details (Sec. A).
- Additional experimental results (Sec. B).
  - Sprites: Plots of true factors vs. latent variables (Sec. B.1)
  - Split-MNIST: Visual results of latent traversal (Sec. B.2)
  - Receipe1M: Visual results of latent traversal (Sec. B.3)
- Recipe1M: Association between food factors and tags from `food.com` (Sec. C).
- Details of the joint model, Retrieval-Bi-VAE (RBi-VAE) (Sec. D).

A Experimental Details

A.1 Network Architectures

A.1.1 Embedding networks

For all competing models that employ the embedding networks $\mathbf{e}_1(\cdot)$ and $\mathbf{e}_2(\cdot)$, we use the same network architectures. They are Cos-Sim-LVM, Bi-VAE-on-$\mathcal{V}$, DCCA, RBi-VAE, and our RIVAE. The learned embedding network parameters of the Cos-Sim-LVM are used as initial parameters for the rest models. Detailed embedding network architectures for different benchmark datasets are defined as follows:

- **Synth**: A fully connected network with one hidden layer of 25 hidden units. The $tanh()$ nonlinear link is used.
- **Sprites**: Four 2D convolutional layers (filter size (4×4) pixels) with the number of channels 32, 32, 64, 64, followed by two fully connected layers with output dimensions 128 and $\dim(\mathcal{V}) = 10$. The rectified linear unit (ReLU) is used as nonlinear components.
- **Split-MNIST**: Four fully connected layers of hidden dimension 1024 are applied with the ReLU nonlinearity.
- **Recipe1M**: Our model combines architectural and training strategies from [28, 34, 7], mainly, hard-example mining, a modality-shared fully connected layer, and full RNN text encoder. Word embeddings used by our model are initialized from a recipe1M pre-trained word2vec 300 dimensional model, fine-tuned during training. RNN layers are all bi-directional and have a 300-dimensional hidden state, with forward and backward states concatenated into a single 600-dimensional vector. Additionally, similar to other works, our image embedding module is based on ResNet50 pre-trained on ImageNet, however, we have introduced an attention module at the 28x28 level. This attention module helps the image encoder focus on the parts of the image that are related to the dish of food as described by the text recipe. Fig. 7 gives an schematic of the general architecture of the embedding modules.

As usual practice, we also apply the L_2 normalization to the final outputs of the embedding networks to make the embedded vectors $\mathbf{v}$ unit norm.

A.1.2 Model-Specific Network Architectures

For each competing model, we employ the following network architectures (and strategies):

- **RIVAE**: For the two conditional models $P(\mathbf{z}|\mathbf{v}_1)$ and $P(\mathbf{v}_2|\mathbf{z})$, we use fully connected networks with two hidden layers of 10 hidden units, and the leaky-ReLU with slope 0.2 is used for nonlinearity. For the Recipe1M, we used a single hidden layer with 100 hidden units, and the same leaky-ReLU nonlinearity as before.
- **RBi-VAE**: (See Sec. D for the details.) For the bi-modal VAE model on the embedded spaces $\mathcal{V}$, we run several different model architectures, either linear or nonlinear, ranging from simpler to more complex architectures for the latter, and we report the ones with the highest performance on the validation sets. For the nonlinear networks, in both encoder and decoder we use one or three fully connected hidden layers. Note that when we adopt a linear (Gaussian) model in the place of the VAE, all inferences can be done analytically without introducing variational densities. For all datasets, the linear Gaussian models perform better than the nonlinear VAE models. For the discriminator networks, used for estimating the total correlation loss via the density ratio trick, we adopted 6 fully connected layers with hidden dimension 300. For the Recipe1M, we also used 6 fully connected layers, however, with only 100 hidden dimensions.
- **Bi-VAE**: We need to model the ambient encoder $P(\mathbf{z}|\mathbf{x}_i)$ and decoder networks $P(\mathbf{x}_i|\mathbf{z})$ for $i = 1, 2$. Similarly as RBi-VAE, we run different (linear/nonlinear and simple/complex) network architectures, and select the best performing one. The nonlinear network architectures include: the convolutional and transposed-convolutional networks of similar structures as the embedding networks above, and the fully connected networks with one or three hidden layers.
- **Bi-VAE-on-$\mathcal{V}$**: This model fixes the embedding networks and learns only the bi-modal VAE model built on the embedded spaces. Hence, the overall model architectures are identical to the RBi-VAE.
- **DCCA**: Instead of building the nonlinear feature extraction networks $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$ from the scratch, we compose the embedding networks $\mathbf{v} = \mathbf{e}(\mathbf{x})$ with the additional nonlinear mappings $\mathbf{z} = \mathbf{g}(\mathbf{v})$ to form $\boldsymbol{\phi}$ (i.e., $\boldsymbol{\phi} = \mathbf{g} \circ \mathbf{e}$). The network $\mathbf{g}$ is defined as fully connected networks with three hidden layers with hidden dimension 10, hence fairly comparable to RBi-VAE. For the Recipe1M, similarly to our RIVAE, we used a single hidden layer with 100 hidden units as the network $\mathbf{g}$.

A.2 Optimization Strategies

We use the Adam optimizer [17] for training of all models. The batch size is 64 and the maximum number of epochs is 2000. For our RIVAE, at the first stage we train the IVAE module alone with the embedding networks fixed, then at the second stage we jointly train the entire networks. The joint training starts at epoch 100 except for the Split-MNIST dataset (epoch 5). The learning rates are: 0.005 (Synth), 0.001 (Sprites), and 0.0001 (Split-MNIST). We also decay the learning rates by roughly half at epoch 200 and 1000 (at epoch 20 and 50 for the Split-MNIST). The trade-off parameter λ for the embedding regularization $\text{Reg}(\mathbf{e}_2)$ is fixed as 0.1 throughout the experiments. For the Recipe1M the first stage (IVAE only) was trained for 20 epochs using a learning rate of 0.001. The second stage (joint embedding and IVAE) observes annealing of the learning by a factor of 0.1 at epochs 30 and 60.

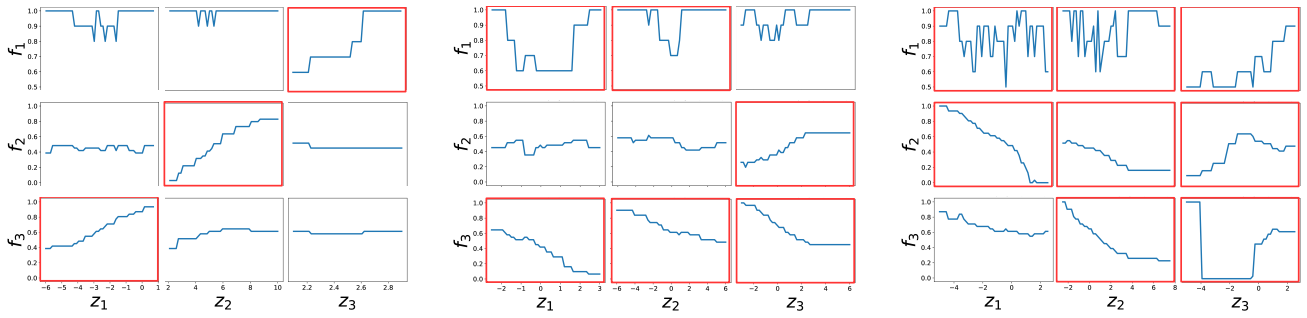
B Additional Experimental Results

B.1 Sprites: Plots of true factors vs. latent variables

Recall that we assumed that the shape from sprites induces the modalities, specifically, $\mathbf{x}_1$ is a *square* image, and $\mathbf{x}_2$ an *oval*, and considered only the X, Y positions and the scale of the sprite as the underlying shared factors, with the other varying source *rotation* being fixed. The plots of the ground-truth factors of the retrieved images due to the latent space traversal are shown in Fig. 8. Our RIVAE shows near one-to-one correspondence between the true and learned factors, while other models exhibit considerable entanglement. More specifically, for RIVAE, we see that z_1 *exclusively* corresponds to f_3 (highlighted by red box), $z_2 = f_2$ and $z_3 = f_1$, signifying that our model learns the disentangled latent representation very accurately. On the other hand, for the DCCA, there is no exclusive correspondence, but significant entanglement. E.g., change in z_1 results in considerable changes in both f_1 and f_2, indicating that the scale and X-pos factors are entangled in the latent variable z_1.

B.2 Split-MNIST: Visual results of latent traversal

For this dataset, we formed a bi-modal setup by taking the left half of each image as modality-1 and the right half as modality-2. The shared factors for both modalities would be the digit class and the writing style, which are independent (disentangled)



(a) RIVAE (b) RBi-VAE (c) DCCA

Figure 8: (Sprites) True factors (f_1 = scale, f_2 = X-pos, and f_3 = Y-pos) of the retrieved images vs. latent traversed points. Each column shows traversal of one latent z_j with the other fixed. The Y axes are true factors obtained from retrieved items $\mathbf{x}_2$. Red boxes indicate significant changes in the true factors, due to latent traversal, i.e., high correlation.

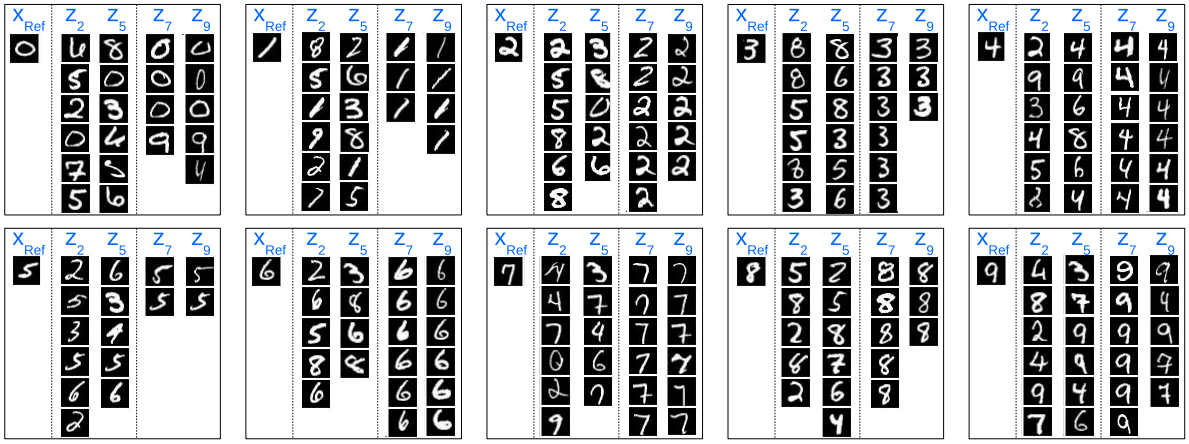


Figure 9: (Split-MNIST) Latent traversal in our RIVAE model. Among $\dim(\mathbf{z}) = 10$ latent variables, we highlight four (z_2, z_5, z_7, z_9); for each we perform traversal by varying the latent values while fixing the rest variables. We depict 10 cases (panels), where each panel consists of: i) $\mathbf{x}_{\text{Ref}}$ = the anchor/reference input whose latent vector $\mathbf{z}$ is inferred using its left half $\mathbf{x}_1$, ii) each column (z_j) contains the retrieved images (the completed images are shown by merging the retrieved right half images $\mathbf{x}_2$ with their corresponding ground-truth left halves) obtained by varying z_j while the rest variables being fixed. Visually, it is clear varying z_2/z_5 leads to dominant changes in digit class with writing style roughly remaining intact. Traversal along z_7/z_9 produces large variation in style with digit classes remaining mostly unchanged. This result shows that our RIVAE accurately disentangles the two main factors, writing style and digit class, in the learned latent representation.

from each other. To see how the two underlying factors, writing style and digit class, are disentangled in the learned latent variables for our RIVAE, we show the latent traversal results in Fig. 9. For the four highlight latent variables (z_2, z_5, z_7, z_9), we can see that the first two explain the digit class variation, while the other two correspond to the writing style. Refer to the caption of the figure for details.

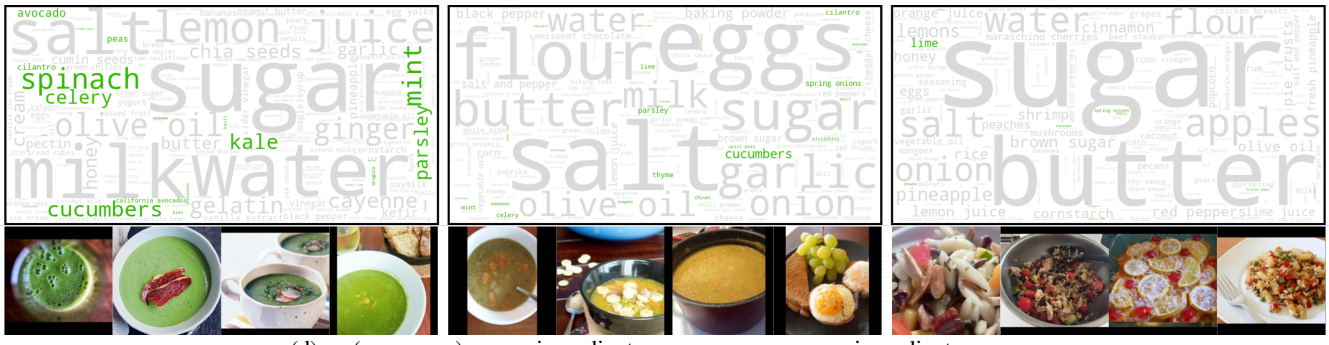
B.3 Receipe1M: Visual results of latent traversal

The qualitative latent traversal results for our RIVAE are depicted in Figs. 12, 13, 14, and 15, each of which shows food images of the retrieved recipes in the latent traversal along the latent variables corresponding to: *wateriness*, *savoriness*, *greenness*, and *fruit-no-fruit*, respectively. Fig. 10 also displays the word clouds of the food ingredients, which are generated from the retrieved top 10 nearest neighbors (NN) from 20 query data points, where each latent variable takes three different values (two extrema and the mean, namely $\mu_i - 10\sigma_i$, μ_i, and $\mu_i + 10\sigma_i$). Hence, they corresponding to *weak*, *mild*, and *strong* impact of each factor. The size of each ingredient word in the word clouds corresponds to its relative frequency in the top 10 NN (from each of the 20 queries) set. Interpretation: For instance, in Fig. 10a, traversal along z_{23} (*fruit-no-fruit*), we see that the ingredients are mostly associated with fruits on the left panel, while on the right panel no fruit related ingredients are found. Also, in Fig. 10b, traversal along z_2 (*savoriness*), the ingredients are mostly associated with savory dishes on the left panel, while typical dessert ingredients on the right, and a mixture of these on the middle panel. In Fig. 10c, traversal along z_0 (*wateriness*, there is an inclination of more liquid ingredients on the right side. In particular, it is noticeable that the word

(a) z_{23} (*fruit-no-fruit*): fruit ingredients as purple – non-fruit ingredients as orange.

(b) z_2 (*savoriness*): savory ingredients as red – non-savory as pink.

(c) z_0 (*wateriness*): watery ingredients as blue – non-watery ingredients as gray.



(d) z_3 (*greenness*): green ingredients as green – non-green ingredients as gray.

Figure 10: (Recipe1M) Ingredient word clouds obtained from latent traversal (at three values, low/medium/high, in left/middle/right columns) along each of four latent variables. For each row, ingredients are colored differently according to the relevance/irrelevance to the governing factor (best seen in color). Below each word cloud image, we also place four retrieved food images corresponding to it. For instance, in (b), the factor of *savoriness*, we manually color the savory ingredients as red, while typical dessert (non-savory) ingredients are shown as pink. The light grey color is reserved for ingredients with no particular association to the factor in question.

water increases in frequency (word size) from left to right. Similarly, in Fig. 10d, z_3 (*greenness*), the left side contains more green ingredients than the right.

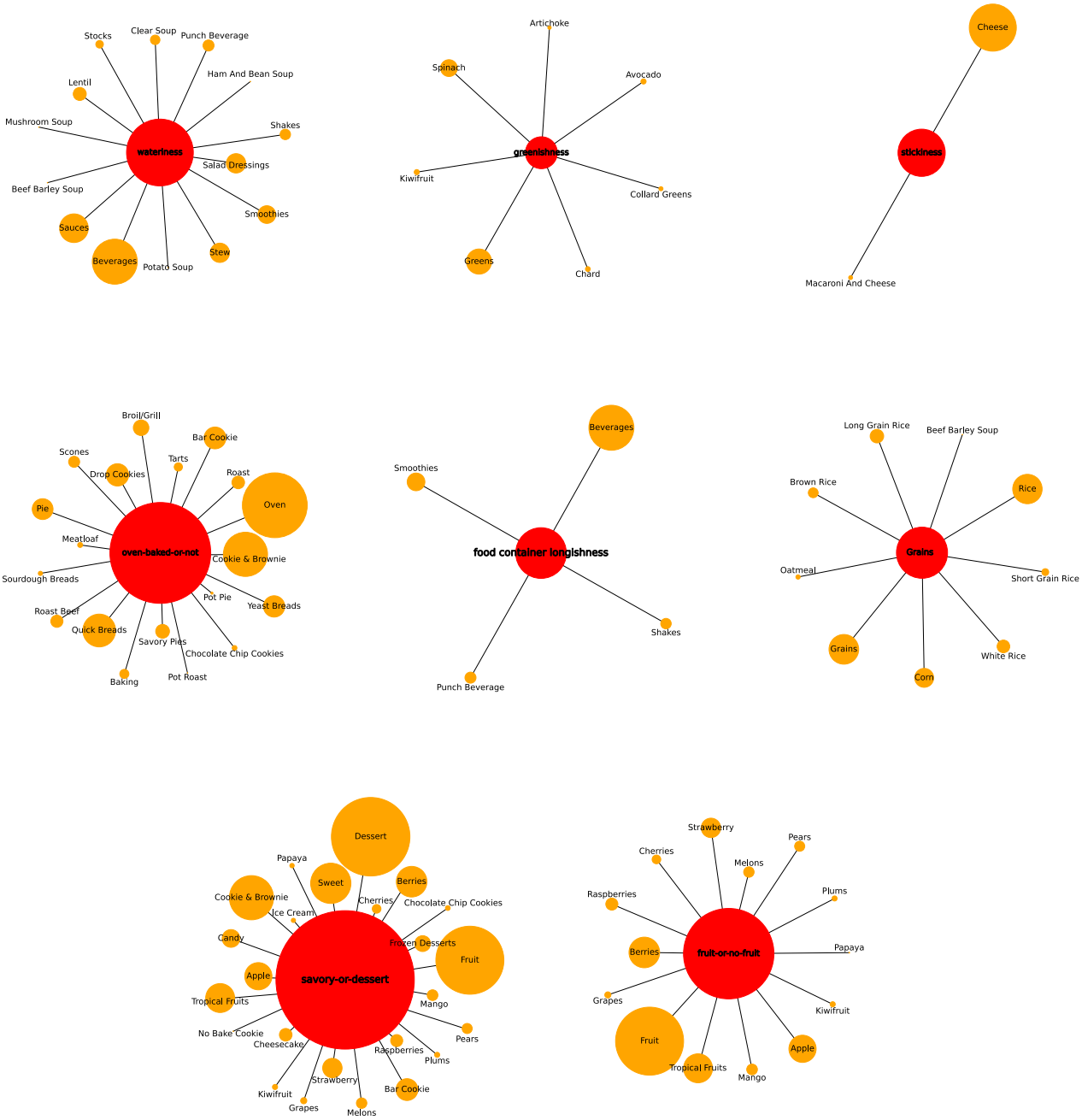


Figure 11: (Recipe1M) Association between tags (shown as orange) extracted from food.com recipes (a subset of Recipe1M) and our 8 food factors (red). The sizes of the nodes are proportional to their frequencies.

C Recipe1M: Association between food factors and food.com tags

Recall that we selected a small subset of food factors by re-craping all the recipes in the Recipe1M that are associated with the Internet domain food.com, and parsing their keywords and categories. We then formed the recipe *tags* as the unique terms union between keywords and categories, and manually grouped the tags that are related to one another. Among the tag groups, we chose 8 factors the most dependent on the latent variables by visually inspecting the latent traversal results for the competing methods. They are: 1) wateriness, 2) greenness, 3) stickiness, 4) oven-baked-or-not, 5) food container longishness (e.g., bottle/cup or plate), 6) grains, 7) savory-or-dessert, and 8) fruit-or-no-fruit. Fig. 11 shows the association between the tags and these 8 factors.

D Details of the joint model, Retrieval-Bi-VAE (RBi-VAE)

In this section we describe the details of the joint model RBi-VAE that served as one of the competing approaches in the empirical study. Note that unlike our proposed RIVAE, the model has no *model identifiability* property, which leads to inferior performance to our RIVAE in the experimental results.

This model builds a latent variable generative model (e.g., VAE) on top of the shared embedding space $\mathcal{V}$. To this end, the embedded vectors $\mathbf{v}$'s are regarded as the observed data for the VAE. That is, the RBi-VAE model is defined as:

$$P(\mathbf{z}) = \mathcal{N}(0, \mathbf{I}),\ P(\mathbf{v}_i|\mathbf{z}) = \mathcal{N}(\mathbf{v}_i; \boldsymbol{\mu}_i(\mathbf{z}), \text{Diag}(\boldsymbol{\sigma}_i(\mathbf{z}))) \text{ for } i = 1, 2, \tag{14}$$

$$\mathbf{v}_1 = \mathbf{e}_1(\mathbf{x}_1),\ \mathbf{v}_2 = \mathbf{e}_2(\mathbf{x}_2). \tag{15}$$

The latent variables $\mathbf{z} \in \mathbb{R}^d$ represent the underlying factors, and $\boldsymbol{\mu}_{1/2}(\cdot)$ and $\boldsymbol{\sigma}_{1/2}(\cdot)$ are the deep networks whose outputs determine means and variances of $P(\mathbf{v}_{1/2}|\mathbf{z})$.

Similarly as RIVAE, we train RBi-VAE with two goals: maximizing the cross-modal prediction performance and maximizing the (embedded) data likelihood. Given the embedding networks, with $\mathbf{v}_1$ and $\mathbf{v}_2$ fixed, learning the Bi-VAE model can be done by maximizing the variational evidence lower bound (ELBO),

$$\text{ELBO} := \mathbb{E}_{Q(\mathbf{z}|\mathbf{v}_1,\mathbf{v}_2)}[\log P(\mathbf{v}_1, \mathbf{v}_2|\mathbf{z})] - \text{KL}(Q(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2)||P(\mathbf{z})) \tag{16}$$

where the variational density $Q(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2)$ is defined as a product of the single-modal inference networks [12], $Q(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2) \propto Q(\mathbf{z}|\mathbf{v}_1) \cdot Q(\mathbf{z}|\mathbf{v}_2)$, where

$$Q(\mathbf{z}|\mathbf{v}_i) := \mathcal{N}(\mathbf{z}; \mathbf{m}_i(\mathbf{v}_i), \text{Diag}(\mathbf{s}_i(\mathbf{v}_i))) \text{ for } i = 1, 2. \tag{17}$$

Furthermore, to encourage the model to learn disentangled latent variables, we adopt the total correlation (TC) loss [15, 8], $\text{TC} = \text{KL}\big(Q(\mathbf{z})\|\prod_j Q(z_j)\big)\big)$ where $Q(\mathbf{z}) = \mathbb{E}_{(\mathbf{x}_1,\mathbf{x}_2)\sim\text{Data}}[Q(\mathbf{z}|\mathbf{v}_1, \mathbf{v}_2)]$. Estimating the TC loss can be done by the density ratio estimation proxy [26, 32] followed by adversarial training [15], or the weighted sampling strategy [8]. Combining the two leads to the following objective function for the Bi-VAE model:

$$\mathcal{L}_{\text{Bi-VAE}}(\Theta_{\text{Bi-VAE}}) = -\text{ELBO} + \gamma \cdot \text{TC}, \tag{18}$$

where $\Theta_{\text{Bi-VAE}}$ in (18) refers to all parameters of the Bi-VAE model (i.e., parameters of $\boldsymbol{\mu}_{1/2}(\cdot)$, $\boldsymbol{\sigma}_{1/2}(\cdot)$, $\mathbf{m}_{1/2}(\cdot)$, $\mathbf{s}_{1/2}(\cdot)$), and γ trades off the TC loss against the ELBO. We use $\gamma = 10.0$ for all datasets except $\gamma = 100.0$ for Recipe1M.

The cross-modal retrieval loss, similar to our RIVAE, is also adopted to train both embedding networks and the Bi-VAE model. The related loss is:

$$\mathcal{L}_{\text{Retrieval}}(\Theta_{\text{Bi-VAE}}, \Theta_{\text{Emb}}) = \Big(1 + \mathbb{E}_{Q(\mathbf{z}|\mathbf{v}_1)}\big[\log P(\mathbf{v}_2'|\mathbf{z}) - \log P(\mathbf{v}_2|\mathbf{z})\big]\Big)_+ \tag{19}$$

where $\mathbf{v}_2' = \mathbf{e}_2(\mathbf{x}_2')$ is the embedding of the mismatch sample $\mathbf{x}_2'$, Θ_{Emb} indicates the parameters of the embedding networks $\mathbf{e}_1(\cdot)$ and $\mathbf{e}_2(\cdot)$, and $(a)_+ = \max(0, a)$. Finally, we impose the embedding regularization loss for $\mathbf{e}_2(\cdot)$, similar to RIVAE:

$$\mathcal{L}_{\text{Reg}}(\Theta_{\text{Emb}}) = \mathbb{E}_{\mathbf{x}_2,\boldsymbol{\epsilon}}\Big[\big(||\mathbf{e}_2(\mathbf{x}_2) - \mathbf{e}_2(\mathbf{x}_2 + \boldsymbol{\epsilon})|| - c\big)^2\Big] \tag{20}$$

where $\boldsymbol{\epsilon} \sim P(\boldsymbol{\epsilon})$ is a random sample from a noise distribution $P(\boldsymbol{\epsilon})$ with small magnitude (e.g., $||\boldsymbol{\epsilon}|| = 0.001$). Then the training of the RBi-VAE can be written as the following optimization:

$$\min\ \mathcal{L}_{\text{Bi-VAE}}(\Theta_{\text{Bi-VAE}}) + \lambda_{\text{Retrieval}}\mathcal{L}_{\text{Retrieval}}(\Theta_{\text{Bi-VAE}}, \Theta_{\text{Emb}}) + \lambda_{\text{Reg}}\mathcal{L}_{\text{Reg}}(\Theta_{\text{Emb}}). \tag{21}$$

Note that the arguments in each loss $\mathcal{L}(\cdot)$ indicates which parameters should be updated regarding the loss.

Figure 12: Images of the retrieved recipes from latent traversal along the latent variable z_0 (*wateriness*). **Image grids**: the leftmost are the query images, and each column contains top-3 retrieved items at a point in traversal. Among the traversed points, we only show those that have change in the top-1 retrieved recipes from the previous traversed/retrieved ones. Note that the images on the left end of the traversal are less watery, while those on the right are more watery.

Figure 13: Images of the retrieved recipes from latent traversal along z_2 (*savoriness*). **Image grids**: the leftmost are the query images, and each column contains top-3 retrieved items at a point in traversal. Among the traversed points, we only show those that have change in the top-1 retrieved recipes from the previous traversed/retrieved ones. Note that the images on the left end of the traversal are more savory, while those on the right are less savory or sweet.

Figure 14: Images of the retrieved recipes from latent traversal along z_3 (*greenness*). **Image grids**: the leftmost are the query images, and each column contains top-3 retrieved items at a point in traversal. Among the traversed points, we only show those that have change in the top-1 retrieved recipes from the previous traversed/retrieved ones. Note that the images on the left end of the traversal are more greenish, while those on the right are less greenish.

Figure 15: Images of the retrieved recipes from latent traversal along z_{23} (*fruit-no-fruit*). **Image grids**: the leftmost are the query images, and each column contains top-3 retrieved items at a point in traversal. Among the traversed points, we only show those that have change in the top-1 retrieved recipes from the previous traversed/retrieved ones. Note that the images on the left end of the traversal contain fruits, while those on the right have less or no fruits.