Article

Symmetric Encoding-decoding Framework without Backpropagation

Pengyuan Zhai [1]

[1] Harvard University; pzhai@g.harvard.edu

Abstract: We propose a forward-only multi-layer encoding-decoding framework based on the principle of Maximal Coding Rate Reduction (MCR^2), an information-theoretic metric that describes a statistical distance between two sets of feature vectors up to the second moment. The encoder directly transforms data vectors themselves via gradient ascent to maximize the MCR^2 distance between different classes in the feature space, resulting in class-wise mutually orthogonal subspace representations. The decoder follows a process symmetric to the encoder, and transforms the subspace feature vectors via gradient descent to minimize the MCR^2 distance between the reconstructed data and the original data. We show that the encoder transforms data to linear discriminative representations without breaking the higher-order manifolds, preserving higher-moment correlation/information. It is exactly due to this information-preserving property that the decoder is able to reconstruct the data with high fidelity.

Keywords: Backpropagation-free Network, Information Theory, Maximal Coding Rate Reduction

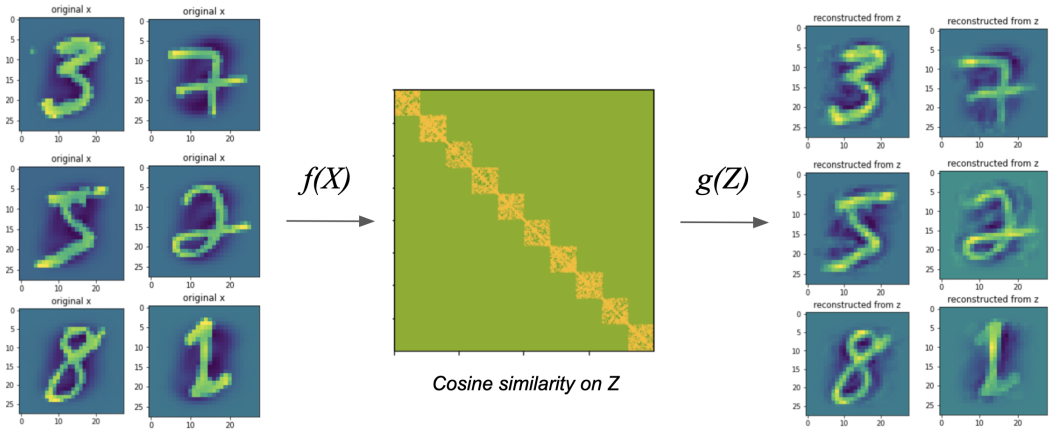

Figure 1. We propose a non-backpropagation encoding-decoding pipeline based on the Maximal Coding Rate Reduction (MCR^2) principle. Given data vectors $\{\boldsymbol{X}^{(c)}\}_C$ from C classes, the forward transformation corresponds to a gradient-based transformations on data vectors $\{\boldsymbol{X}^{(c)}\}_C$ themselves, which simultaneously maps multi-class data vectors into linear subspaces $\{\boldsymbol{Z}^{(c)}\}_C$ respective to each class. The decoding process follows a multi-layer pipeline that mirrors the structure of the encoding process, reconstructing $\{\boldsymbol{Z}^{(c)}\}_C$ into $\{\tilde{\boldsymbol{X}}^{(c)}\}_C$ with high visual fidelity with the original $\{\boldsymbol{X}^{(c)}\}_C$.

0. Introduction and Related Work

One important goal of modern machine learning is to learn and model complex distributions of real-world data. A typical learning paradigm is to establish an (often parameterized) mapping between the distribution of the real data, ie., $\boldsymbol{x} \in \mathbb{R}^D$, and a more compact representation, i.e., $\boldsymbol{z} \in \mathbb{R}^d$:

$$\text{encode: } f(\cdot, \theta_f) : \boldsymbol{x} \in \mathbb{R}^D \mapsto \boldsymbol{z} \in \mathbb{R}^d \quad \text{and/or decode:} \quad g(\cdot, \theta_g) : \boldsymbol{z} \in \mathbb{R}^d \mapsto \boldsymbol{x} \in \mathbb{R}^D, \quad (1)$$

where $\boldsymbol{z}$ has simpler or more discriminative structures. The learnt feature $\boldsymbol{z}$ is thus easier to manipulate for both generative or discriminative purposes.

Citation: Zhai, P. Symmetric Encoding-decoding Framework without Backpropagation. *Preprints* **2022**, *1*, 0. https://doi.org/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

0.1. Auto-Encoding

One typical approach to encoding-decoding is the "Auto Encoder" (AE) [1,2], which learns an encoding mapping f from $\boldsymbol{x}$ to $\boldsymbol{z}$ and a decoding mapping g from $\boldsymbol{z}$ back to $\boldsymbol{x}$:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta_f)} \boldsymbol{Z} \xrightarrow{g(\boldsymbol{z},\theta_g)} \hat{\boldsymbol{X}}. \tag{2}$$

To make the learned distribution $\hat{p}(\boldsymbol{z})$ close to a target $p(\boldsymbol{z})$, we minimize some distance metric, e.g., the KL-divergence: $\min \mathcal{D}_{KL}(\hat{p}, p)$. Between two arbitrary degenerate distributions in high-dimensional spaces, however, KL-divergence is often intractable. In practice, one instead minimizes instead certain bounds of the distance, e.g., the variational bound in *variational auto-encoding* (VAE) [3,4]. The exact distribution/structure of $\hat{p}(\boldsymbol{z}|\boldsymbol{X},\theta)$ is thus unclear.

0.2. Coding Rate Reduction

Recently, [5] [6] proposed a novel objective for learning a linear discriminative representation (LDR) for multi-class data, called "coding rate reduction". Unlike conventional discriminative methods only trying to predict class labels, the LDR learns the structure of the data, and is more suitable for both discriminative and generative purposes. [6] maps distributions of the input data on *multiple* nonlinear submanifolds to multiple distinctive linear subspaces. Intuitively, this approach generalizes nonlinear PCA [1] to settings where one simultaneously applies *multiple nonlinear PCAs* to data on multiple nonlinear submanifolds.

In this work, we extend the work of [6] and show that the *coding rate reduction* principle is also suitable for encoding-decoding purposes, and that one can build a pipeline that learns LDRs in the feature space and also reconstructs high-fidelity data in a forward-only fashion without back-propagation. The manifold structures can be learned directly from data itself, free of explicit parameterization.

1. Problem Formulation

In this section, we first explain the objective formulations for the forward (encoding) mapping and the backward (decoding) mapping, and then cover the details of the network structure. The main measurement of "closeness" between data distributions is the coding rate reduction metric [5]. Our goal is to come up with a encoding-decoding scheme:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x})} \boldsymbol{Z} \xrightarrow{g(\boldsymbol{z})} \hat{\boldsymbol{X}}, \tag{3}$$

such that the learnt features $\boldsymbol{Z}$ of different classes are maximally discriminative, and the reconstructed data $\hat{\boldsymbol{X}}$ is the "closest" to the real data $\boldsymbol{X}$ by the coding rate reduction metric.

1.1. Encoding mapping via maximal coding rate reduction

The (labeled) training data $\boldsymbol{X}$ is the concatenation of data of all k classes: $\boldsymbol{X} \doteq \left[\boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^k\right] \in \mathbb{R}^{D\times m}$, where $\boldsymbol{X}^j \in \mathbb{R}^{D\times m_j}$ denotes the j-th class's data matrix, D is the dimension of each data sample, and m_j is the number of samples of class j. Following the notation in [6], we use the matrix $\boldsymbol{\Pi}^j(i,i) = 1$ to denote the membership of sample i belonging to class j (and $\boldsymbol{\Pi}^j = 0$ otherwise). For any general encoding mapping $f(\cdot,\theta_f) : \boldsymbol{x} \mapsto \boldsymbol{z}$ from $\boldsymbol{X}$ to an optimal representation $\boldsymbol{Z} \in \mathbb{R}^{d\times m}$:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta_f)} \boldsymbol{Z}, \tag{4}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

we wish to make each class's features more discrimative of others by maximize the coding rate reduction objective [5]:

$$\max_{\boldsymbol{Z}} \Delta R(\boldsymbol{Z} \,|\, \boldsymbol{\Pi}, \epsilon) \doteq \underbrace{\frac{1}{2} \log\det\Big(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^*\Big)}_{R(\boldsymbol{Z}\,|\epsilon)} - \underbrace{\sum_{j=1}^{k} \frac{\gamma_j}{2} \log\det\Big(\boldsymbol{I} + \alpha_j \boldsymbol{Z}\boldsymbol{\Pi}^j\boldsymbol{Z}^*\Big)}_{R_c(\boldsymbol{Z}\,|\boldsymbol{\Pi},\epsilon)}, \tag{5}$$

where $\alpha = \frac{d}{m\epsilon^2}$, $\alpha_j = \frac{d}{\mathrm{tr}(\boldsymbol{\Pi}^j)\epsilon^2}$, $\gamma_j = \frac{\mathrm{tr}(\boldsymbol{\Pi}^j)}{m}$ for $j = 1, \ldots, k$, and $\boldsymbol{Z}^*$ denotes the transpose of $\boldsymbol{Z}$. For simplicity we denote $\Delta R(\boldsymbol{Z}\,|\boldsymbol{\Pi}, \epsilon)$ as $\Delta R(\boldsymbol{Z})$, assuming $\boldsymbol{\Pi}, \epsilon$ are known and fixed. The first term $R(\boldsymbol{Z}\,|\epsilon)$ is the coding rate of the whole set of features $\boldsymbol{Z}$ (coded as a Gaussian source) with precision ϵ; the second term $R_c(\boldsymbol{Z}\,|\boldsymbol{\Pi}, \epsilon)$ is the weighted average coding rate of the k subsets of feature, each coded as a Gaussian.

For more readability, Eq. 5 is equivalently:

$$\max_{\boldsymbol{Z}} \Delta R(\boldsymbol{Z} \,|\, \boldsymbol{\Pi}, \epsilon) \doteq \underbrace{\frac{1}{2} \log\det\Big(\boldsymbol{I} + \alpha \boldsymbol{Z}\boldsymbol{Z}^*\Big)}_{R(\boldsymbol{Z}\,|\epsilon)} - \underbrace{\sum_{j=1}^{k} \frac{\gamma_j}{2} \log\det\Big(\boldsymbol{I} + \alpha_j \boldsymbol{Z}^j\boldsymbol{Z}^{j*}\Big)}_{R_c(\boldsymbol{Z}\,|\boldsymbol{\Pi},\epsilon)}, \tag{6}$$

given known class partitions of the feature set: $\boldsymbol{Z} = \Big[\boldsymbol{Z}^1, \boldsymbol{Z}^2, \ldots, \boldsymbol{Z}^k\Big] \in \mathbb{R}^{d\times m}$.

1.2. Decoding via coding rate matching.

The pair-wise coding rate reduction metric for two identical sets of data samples is equal to the minimal value of zero. To match the distribution of the reconstructed data to that of the real data, for any general decoding mapping $g(\cdot, \theta_g) : \boldsymbol{z} \mapsto \hat{\boldsymbol{x}}$ from the encoded $\boldsymbol{Z}$ features to an optimal reconstructed $\hat{\boldsymbol{X}}$:

$$\boldsymbol{Z} \xrightarrow{\;g(\boldsymbol{z},\theta_g)\;} \hat{\boldsymbol{X}}, \tag{7}$$

we propose the following objective:

$$\begin{aligned}\min_{\hat{\boldsymbol{X}}} \sum_{j=1}^{k} \underbrace{\Delta R\Big(\Big[\boldsymbol{X}^j, \hat{\boldsymbol{X}}^j\Big]\Big)}_{\text{coding rate matching}} = \sum_{j=1}^{k} \Bigg\{ &\underbrace{\frac{1}{2} \log\det\Big(\boldsymbol{I} + \alpha(\boldsymbol{X}^j\boldsymbol{X}^{j*} + \hat{\boldsymbol{X}}^j\hat{\boldsymbol{X}}^{j*})\Big)}_{R(\boldsymbol{Z}\,|\epsilon)} \\ &- \underbrace{\frac{1}{4} \log\det\Big(\boldsymbol{I} + 2\alpha\boldsymbol{X}^j\boldsymbol{X}^{j*}\Big) - \frac{1}{4} \log\det\Big(\boldsymbol{I} + 2\alpha\hat{\boldsymbol{X}}^j\hat{\boldsymbol{X}}^{j*}\Big)}_{R_c(\boldsymbol{Z}\,|\boldsymbol{\Pi},\epsilon)} \Bigg\},\end{aligned} \tag{8}$$

where $\alpha = \frac{d}{2m_j\epsilon^2}$, if we assume that the number of data samples in $\boldsymbol{X}^j$ and $\hat{\boldsymbol{X}}^j$ are equal.

Eq. 8 minimizes the pairwise coding rate reduction of the real data $\boldsymbol{X}^j$ and the reconstructed data $\hat{\boldsymbol{X}}^j$ of class j. If the reconstructed data is exactly identical to the real data for all classes, the objective evaluates to zero.

2. Forward-only Encoding-decoding Pipeline

The encoding and decoding objectives (Eq. 6 and Eq. 8) are suitable for any general parameterized functions, which can be readily modeled by deep neural networks. In this work, we propose a non-parameterized, forward-only pipeline that does not require training through back-propagation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

There are two key factors that make such pipeline viable (details will be discussed in later sections):

1. **Differentiable gradients.** Eq. 6 and Eq. 8 are differentiable with $\boldsymbol{Z}$ and $\hat{\boldsymbol{X}}$ respectively. Given any initial configuration of $\boldsymbol{Z}$ or $\hat{\boldsymbol{X}}$, one can directly apply gradient ascent/descent steps to transform the data features to increase/decrease the objective function, following a normalization step after each update.
2. **Structure-preserving transformation.** According to our numerical simulations, both the encoding and decoding mappings (through gradient ascent/descent) preserve the local topological relationships between data points. The encoding and decoding transformations resembles a Ricci flow, which preserves local structures of the manifold [7]. Additionally, due to the normalization (by Frobenius norm) after each gradient update, the Riemannian metric of the manifold is constant, providing a fair comparison of coding rate reduction metrics at different layers.

2.0.1. Encoding via Gradient-based Transformation

Consider the encoding objective in Eq.6, if we directly optimize the objective $\Delta R(\boldsymbol{Z})$ as a function of the (normalized) data feature $\boldsymbol{Z}$. The gradient ascent step is:

$$\boldsymbol{Z}_{\ell+1} \propto \boldsymbol{Z}_\ell + \eta \cdot \frac{\partial \Delta R}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} \text{ subject to } ||\boldsymbol{Z}_{\ell+1}||_F = 1. \tag{9}$$

This means that at each layer, we apply a gradient ascent transformation directly on data features themselves to incrementally improve the coding rate reduction metric ΔR.

The each part of the gradient $\frac{\partial \Delta R}{\partial \boldsymbol{Z}}$ is computed as (refer to [8]):

$$\begin{aligned}
\frac{1}{2}\frac{\partial \log\det(\boldsymbol{I}+\alpha \boldsymbol{Z} Z^*)}{\partial \boldsymbol{Z}}\bigg|_{\boldsymbol{Z}_\ell} &= \underbrace{\alpha(\boldsymbol{I}+\alpha \boldsymbol{Z}_\ell \boldsymbol{Z}_\ell^*)^{-1}}_{\boldsymbol{E}_\ell \in \mathbb{R}^{d\times d}} \boldsymbol{Z}_\ell \in \mathbb{R}^{d\times m}, \\
\frac{1}{2}\frac{\partial \left(\gamma_j \log\det\left(\boldsymbol{I}+\alpha_j \boldsymbol{Z}^j \boldsymbol{Z}^{j*}\right)\right)}{\partial \boldsymbol{Z}^j}\bigg|_{\boldsymbol{Z}_\ell^j} &= \gamma_j \underbrace{\alpha_j\left(\boldsymbol{I}+\alpha_j \boldsymbol{Z}_\ell^j \boldsymbol{Z}_\ell^{j*}\right)^{-1}}_{\boldsymbol{C}_\ell^j \in \mathbb{R}^{d\times d}} \boldsymbol{Z}_\ell^j \in \mathbb{R}^{d\times m_j},
\end{aligned} \tag{10}$$

where $\alpha = \frac{d}{m\epsilon^2}$. Thus the total gradient of the encoding objective with respect to $\boldsymbol{Z}_\ell^j$ of class j is:

$$\frac{\partial \Delta R}{\partial \boldsymbol{Z}^j}\bigg|_{\boldsymbol{Z}_\ell^j} = \underbrace{\boldsymbol{E}_\ell}_{\text{Expansion}} \boldsymbol{Z}_\ell^j - \sum_{j=1}^{k} \gamma_j \underbrace{\boldsymbol{C}_\ell^j}_{\text{Compression}} \boldsymbol{Z}_\ell^j \quad \in \mathbb{R}^{n\times m}. \tag{11}$$

2.0.2. Symmetric Decoding

Each of the k terms in the summation of Eq. 8 has the following gradient with respect to the reconstructed data $\hat{\boldsymbol{X}}^j$ of class j (with $\alpha = \frac{d}{2m_j\epsilon^2}$):

$$\frac{\partial \Delta R\left(\left[\boldsymbol{X}^j, \hat{\boldsymbol{X}}^j\right]\right)}{\partial \hat{\boldsymbol{X}}^j}\bigg|_{\hat{\boldsymbol{X}}_\ell^j} = \underbrace{\alpha\left(\boldsymbol{I}+\alpha(\boldsymbol{X}_\ell^j \boldsymbol{X}_\ell^{j*} + \hat{\boldsymbol{X}}_\ell^j \hat{\boldsymbol{X}}_\ell^{j*})\right)^{-1}}_{\boldsymbol{E}_\ell^j} \hat{\boldsymbol{X}}_\ell^j - \frac{1}{2}\underbrace{2\alpha\left(\boldsymbol{I}+2\alpha \hat{\boldsymbol{X}}_\ell^j \hat{\boldsymbol{X}}_\ell^{j*}\right)^{-1}}_{\boldsymbol{C}_\ell^j} \hat{\boldsymbol{X}}_\ell^j. \tag{12}$$

Thus, the total gradient of the decoding objective with respect to $\hat{\boldsymbol{X}}^j$ is

$$\frac{\partial \Delta R\left(\left[\boldsymbol{X}^j, \hat{\boldsymbol{X}}^j\right]\right)}{\partial \hat{\boldsymbol{X}}^j}\bigg|_{\hat{\boldsymbol{X}}_\ell^j} = \left(\boldsymbol{E}_\ell^j - \frac{1}{2}\boldsymbol{C}_\ell^j\right)\hat{\boldsymbol{X}}_\ell^j. \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

3. Pipeline Architecture

With the knowledge of the gradients, we now build the layer-by-layer pipeline for forward-only encoder f and forward-only decoder g:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x})} \boldsymbol{Z} \xrightarrow{g(\boldsymbol{z})} \hat{\boldsymbol{X}}. \tag{14}$$

3.1. Encoding Pipeline

Assuming a fixed number of L_f layers, define the encoding mapping function as $f(\boldsymbol{X}) \doteq f_{L_f} \circ f_{L_f-1} \circ \cdots \circ f_2 \circ f_1(\boldsymbol{X})$, where $\boldsymbol{Z}_\ell = f_\ell(\boldsymbol{Z}_{\ell-1})$ is the transformation function acting on the input $(\boldsymbol{Z}_{\ell-1})$ to the ℓ-th layer:

$$\begin{aligned} \boldsymbol{Z}_\ell^j = f_\ell(\boldsymbol{Z}_{\ell-1}^j) &\doteq FN\left(\boldsymbol{Z}_\ell^j + \eta \left.\frac{\partial \Delta R}{\partial \boldsymbol{Z}^j}\right|_{\boldsymbol{Z}_\ell^j}\right) = FN\left(\boldsymbol{Z}_\ell^j + \eta\left(\boldsymbol{E}_\ell \boldsymbol{Z}_\ell^j - \sum_{j=1}^k \gamma_j \boldsymbol{C}_\ell^j \boldsymbol{Z}_\ell^j\right)\right) \\ &= FN\left(\boldsymbol{Z}_\ell^j + \eta\left(\boldsymbol{E}_\ell - \sum_{j=1}^k \gamma_j \boldsymbol{C}_\ell^j\right)\boldsymbol{Z}_\ell^j\right) = FN\left(\left[\boldsymbol{I} + \eta\left(\boldsymbol{E}_\ell - \sum_{j=1}^k \gamma_j \boldsymbol{C}_\ell^j\right)\right]\boldsymbol{Z}_\ell^j\right), \end{aligned} \tag{15}$$

where $FN(\cdot)$ is the Frobenius normalization function:

$$FN(\boldsymbol{Z}) = \frac{\boldsymbol{Z}}{||\boldsymbol{Z}||_F} \tag{16}$$

3.2. Decoding pipeline.

Assuming a fixed number of L_f layers, define the decoding mapping function as $g(\boldsymbol{Z}_L) \doteq g_{L_g} \circ g_{L_g-1} \circ \cdots \circ g_2 \circ g_1(\boldsymbol{Z}_L) = \hat{\boldsymbol{X}}$, where $\hat{\boldsymbol{X}}$ denotes the decoded features in the data space (referred to as the "reconstructed data"). We use a similar procedure:

$$\begin{aligned} \hat{\boldsymbol{X}}_\ell^j = f_\ell(\hat{\boldsymbol{X}}_{\ell-1}^j) &\doteq FN\left(\hat{\boldsymbol{X}}_\ell^j + \eta \left.\frac{\partial \Delta R_g}{\partial \hat{\boldsymbol{X}}^j}\right|_{\hat{\boldsymbol{X}}_\ell^j}\right) = FN\left(\hat{\boldsymbol{X}}_\ell^j + \eta\left(\boldsymbol{E}_\ell^j - \frac{1}{2}\boldsymbol{C}_\ell^j\right)\hat{\boldsymbol{X}}_\ell^j\right) \\ &= FN\left(\left[\boldsymbol{I} + \eta\left(\boldsymbol{E}_\ell^j - \frac{1}{2}\boldsymbol{C}_\ell^j\right)\right]\hat{\boldsymbol{X}}_\ell^j\right). \end{aligned} \tag{17}$$

We make the following remarks:

1. For encoding (f), we have that $\boldsymbol{Z}_0 = \boldsymbol{X}$, and the last layer outputs $\boldsymbol{Z}_L$. The deepest-layer features $\boldsymbol{Z}_L$ is then fed into the decoder (g) to produce $\hat{\boldsymbol{X}}$.
2. The structure of the pipeline bears resemblance to that of a neural network with residual connections (by the appearance of the identity matrix $\boldsymbol{I}$). The normalization steps can be seen as a non-linear function applied on the piece-wise linearly transformed features (piece-wise linear by each class).

4. Numerical Simulations

In this section, we present results from the experiments to demonstrate:

1. How the encoder transforms the data $\boldsymbol{X}$ to a linear discriminative representation (LDR) $\boldsymbol{Z}$, while preserving the local manifold structure;
2. How the decoder transforms/unfolds $\boldsymbol{Z}$ to $\hat{\boldsymbol{X}}$ such that the principal components of $\hat{\boldsymbol{X}}^j$ align well with the those of $\boldsymbol{X}^j$ at the optimal configuration;
3. The importance of preserving the local manifold structures during encoding in order to assure successful decoding;
4. The performance of our pipeline on the MNIST dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

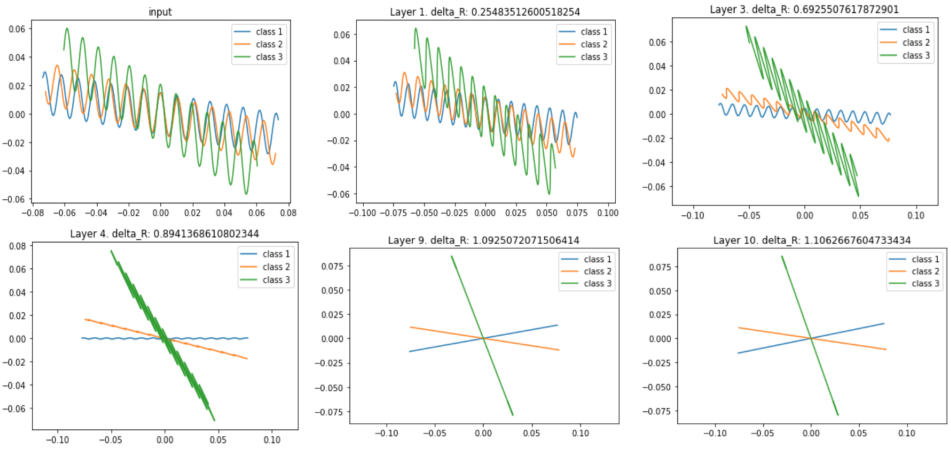

Figure 2. Visualization of the encoding process on 2-dimensional data vectors from 3 classes. At deeper layers, data vectors from each class are compressed to a linear subspace. Later we show that the higher-order "micro" manifolds are preserved in this process, allowing reconstruction/decoding with high-fidelity.

4.1. Experiment 1: Encoding while preserving manifold structures

For each of the tow classes, we generate 500 2-dimensional (x, y) pairs as the training data with the relationship:

$$y = \frac{(2 \cdot \cos(x \cos a - x \sin a) - x \sin a)}{0.3 \cos a}, \tag{18}$$

where a controls the rotation around the origin.

We use a 10-layer encoder with $\epsilon = 0.01$ and the learning rate $\eta_f = 0.1$ and plot the features output by each layer. Two key observations are: i. the dominant principal components of each class becomes farther apart in deeper layers, in the last layer, the manifolds are compressed to LDRs; ii. the local structure of the manifolds are preserved (notice the zig-zag shapes of the manifolds). Although the manifolds are hard to visualize in deeper layers, we demonstrate in the next experiment that the manifolds structures are indeed preserved (Fig. 2).

4.2. Experiment 2: Decoding by manifold unfolding and component alignment

We first use a 30-layer encoder with $\epsilon = 0.01$ and the learning rate $\eta_f = 0.01$ and plot the features output by the last layer (Fig. 3), which will be the starting $\boldsymbol{Z}$ which we will decode from. We transform this last layer $\boldsymbol{Z}$ using the decoding pipeline (Section 3) and reconstruct $\tilde{\boldsymbol{X}}$ to match the distribution of $\boldsymbol{X}$. The decoder has 10 layers with the learning rate $\eta_g = 0.1$ (Fig. 4)

Notice that the decoder does two things: i. it "unfolds" the manifolds from a compressive state to a more expanded state; ii. it aligns the principal components of $\hat{\boldsymbol{X}}$ to those of $\boldsymbol{X}$ by gradually rotating the manifolds.

4.3. Experiment 3: Matching two different manifolds

In this experiment we transform a $\boldsymbol{Z}$ to match $\boldsymbol{X}$ with a different manifold structure. Namely we feed into the decoder a Gaussian-like $\boldsymbol{Z}$ to match $\boldsymbol{X}$, which has the sinusoidal manifolds. We use a 20-layer decoder with learning rate $\eta_g = 0.1$:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

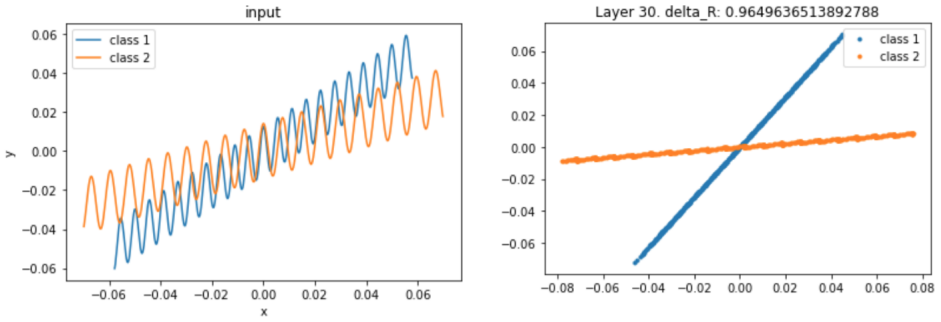

Figure 3. To prepare for the decoding, we first prepare the encoding process with two classes. We then transform the compressed data vectors (right-hand side) into the original manifold as shown below (Fig. 4).

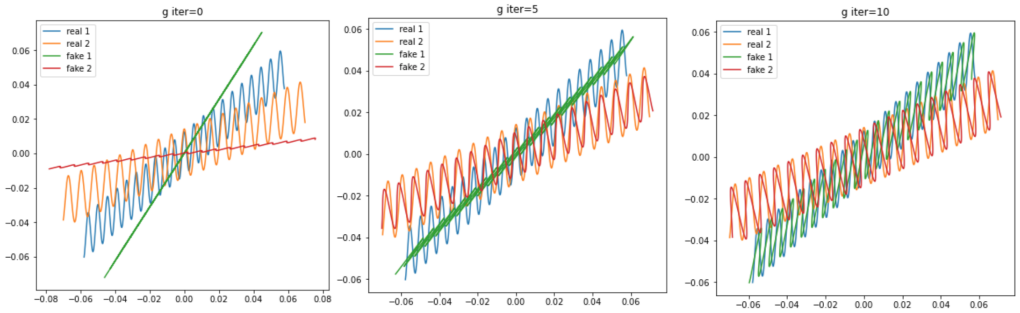

Figure 4. Visualization of the decoding process. The green (compressed class 1) and red (compressed class 2) linear subspaces are the compressed data vectors from Fig. 3, juxtaposed on the blue (original class 1) and orange (original data 2) data points from the true distribution. We transform the compressed data vectors based on the decoding pipeline described in Section 3 until convergence. The reconstructed data vectors (red and green) match the true data distribution with high fidelity, as the "micro-manifolds" in the compressed state are able to expand back to the original shape.

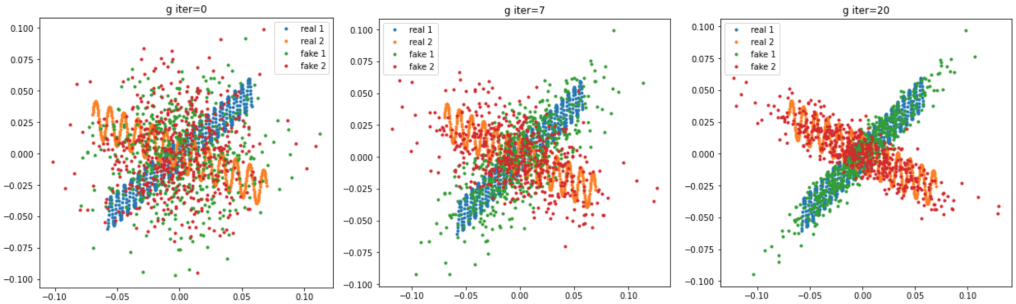

Figure 5. Visualization of a decoding process, where the target $\boldsymbol{X}$ are the blue and orange "spiral" distributions, and the input $\boldsymbol{Z}$ are two random Gaussian distributions (green and red). The decoding progress matches the second moment of the reconstructed $\tilde{\boldsymbol{X}}$ and the real $\boldsymbol{X}$ in the data space.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

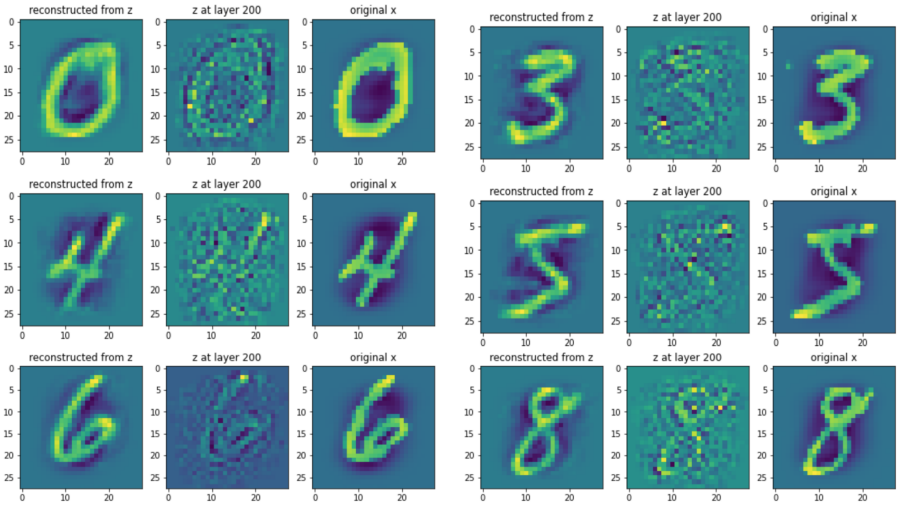

Figure 6. The original data $\boldsymbol{X}$ (right column), the compressed data in respective linear subspaces $\boldsymbol{Z}$ (middle column) and the reconstructed data $\tilde{\boldsymbol{X}}$ (left column) for a selection of MNIST digits.

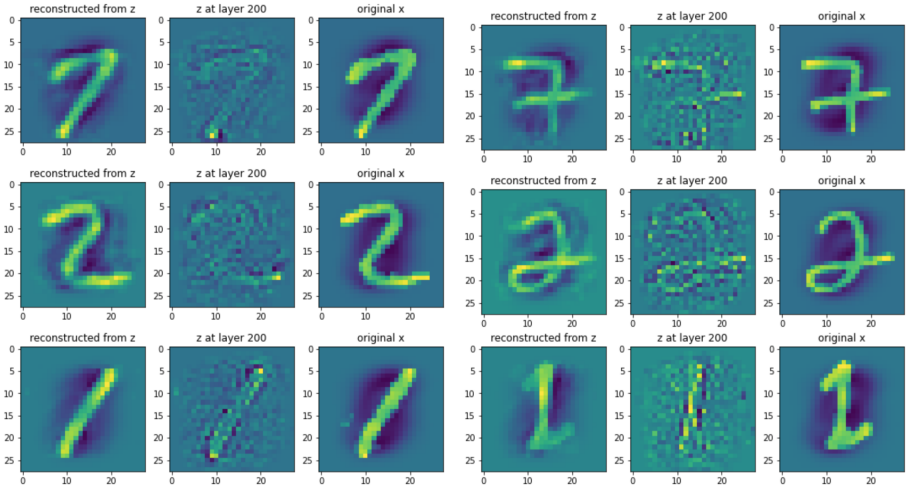

Figure 7. For different styles of digits, the decoder is able to reconstruct with high visual fidelity. For both sub-panels, right columns are the real input $\boldsymbol{X}$, middle columns are the transformed images $\boldsymbol{Z}$ that lie on respective linear subspaces, and the left columns are the reconstructed images $\tilde{\boldsymbol{X}}$ from $\boldsymbol{Z}$.

Notice that the principal components of $\hat{\boldsymbol{X}}$ gradually align with those of $\boldsymbol{X}$, but due to the intrinsic differences of the manifolds (Gaussian vs sinusoidal), the reconstructed $\hat{\boldsymbol{X}}$ does not learn the local manifold structures if only using the coding rate reduction objective.

4.4. Experiment 4: MNIST

We perform encoding-decoding on randomly selected MNIST digits [9]. For each of the 10 classes, we randomly select 200 images, resize each $28p \times 28p$ image into a 784-dimensional vector, and then center the data. With $\epsilon = 0.1$ we use a 200-layer encoder with learning rate $\eta_f = 0.003$ to arrive at the deepest $\boldsymbol{Z}_L \in R^{784\times 2000}$ (we keep the number of features to 784). From $\boldsymbol{Z}_L$, we reconstruct $\hat{\boldsymbol{X}}$ using a 200-layer decoder with learning rate $\eta_g = 0.001$. We present some selected samples to showcase the qualitative decoding performance (Fig 6, 7).

For each digit, the left column is the reconstructed image, the middle is the deepest features $\boldsymbol{Z}_L$ plotted as an image, and the right is the original input. The reconstructed $\hat{\boldsymbol{X}}$ almost exactly replicate the original input, given a seemingly low-quality $\boldsymbol{Z}_L$. We also notice that different modes of the same digits are also captured by the decoder (Fig. 7).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2022 doi:10.20944/preprints202211.0537.v2

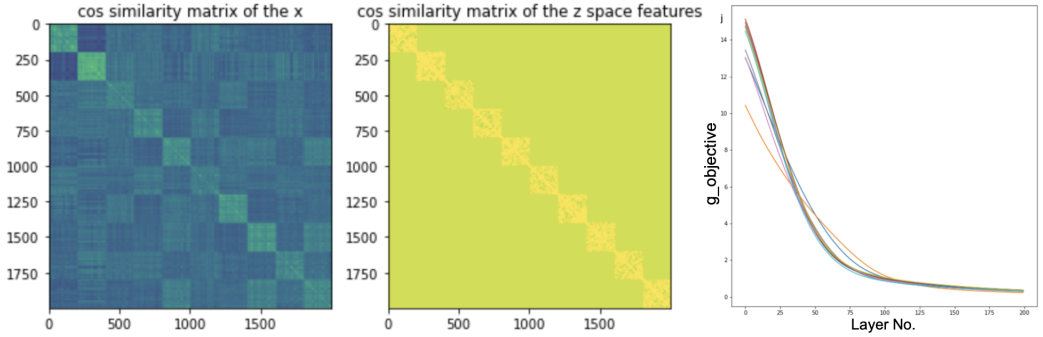

Figure 8. Left: the cosine similarity bewteen any two pair of MNIST images (grouped by the 10 classes). In the original data space, there is no distinct similarity pattern. However, when we transform the original image vectors $\boldsymbol{X}$ into $\boldsymbol{Z}$, and visualize the cosine similarty of vectors from $\boldsymbol{Z}$ (middle), we see high intra-class similarities as showcased by the block-diagonal pattern.

To investigate whether $\boldsymbol{Z}^j$ for class j is truly a linear discriminative representation, we plot the cosine similarity matrix of the the deepest learnt features Z_L and the cosine similarity matrix of the original data $\boldsymbol{X}$, where the data points are grouped by classes (Fig. 8). It is clear that in the Z space, the learnt data features $\boldsymbol{Z}^j$ for class j has high cosine similarity with other data points in the same class, while far away from other classes (shown by the 10 bright diagonal blocks on the left plot), meanwhile the unencoded $\boldsymbol{X}$ is not a good linear discriminative representation (right plot).

Additionally, we show the stability of the decoder but plotting the decoder objective value (Eq. 8) at different layer of g, which shows very nice convergent properties for all classes (represented by each curve, Fig. 8 right).

5. Conclusions and Future Work

We proposed a non-parametric, forward-only, multi-layer encoding-decoding framework based on the principle of maximal coding rate reduction [5], an information-theoretic metric that measures the distance between two sets of feature vectors which are often degenerate with high dimensions. By designing the encoding and decoding objectives based on the *coding rate reduction* principle, we are able to derive gradients for both the encoding and decoding mappings with respect to data/features themselves. The encoder directly applies gradient ascent on data itself to maximize the distances between different classes, while the decoder transforms the learnt features by directly applying gradient descent to minimize the distance between the reconstructed and the original data. Through numerical simulations, we show that the encoder transforms data to linear discriminative representations, and the decoder reconstructs the data with high fidelity. Future work includes investigating the exact connections to the Ricci Flow, and under what conditions this pipeline will converge/diverge. More experiments shall also be done to investigate the performance on larger, more complex datasets such as the Cifar [10] datasets and the ImageNet[11] dataset.

References

1. Kramer, M.A. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal* **1991**, *37*, 233–243.
2. Hinton, G.E.; Zemel, R.S. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In Proceedings of the Proceedings of the 6th International Conference on Neural Information Processing Systems; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993; NIPS'93, p. 3–10.
3. Kingma, D.P.; Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* **2013**.
4. Zhao, S.; Song, J.; Ermon, S. InfoVAE: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262* **2017**.
5. Yu, Y.; Chan, K.H.R.; You, C.; Song, C.; Ma, Y. Learning Diverse and Discriminative Representations via the Principle of Maximal Coding Rate Reduction. In Proceedings of the Advances in neural information processing systems, 2020.
6. Chan, K.H.R.; Yu, Y.; You, C.; Qi, H.; Wright, J.; Ma, Y. ReduNet: A White-box Deep Network from the Principle of Maximizing Rate Reduction. *CoRR* **2021**, *abs/2105.10446*, [2105.10446].

 doi:10.20944/preprints202211.0537.v2

7. Li, Y.; Lu, R. Applying Ricci Flow to High Dimensional Manifold Learning, 2017, [arXiv:cs.LG/1703.10675].
8. Chan, K.H.R.; Yu, Y.; You, C.; Qi, H.; Wright, J.; Ma, Y. Deep Networks from the Principle of Rate Reduction, 2020, [arXiv:cs.LG/2010.14765].
9. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* **2012**, *29*, 141–142.
10. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009.
11. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.