2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)
Allerton Park and Retreat Center
Monticello, IL, USA, September 24-27, 2019

Learning to Compress Using Deep AutoEncoder

Qing Li and Yang Chen

***Abstract*—A novel deep learning framework for lossy compression is proposed. The framework is based on Deep AutoEncoder (DAE) stacked of Restricted Boltzmann Machines (RBMs), which form Deep Belief Networks (DBNs). The proposed DAE compression scheme is one variant of the known fixed-distortion scheme, where the distortion is fixed and the compression rate is left to optimize. The fixed distortion is achieved by the DBN Blahut-Arimoto algorithm to approximate the N^{th}-order rate distortion approximating posterior. The trained DBNs are then unrolled to create a DAE, which produces an encoder and a reproducer. The unrolled DAE is fine-tuned with back-propagation through the whole autoencoder to minimize reconstruction errors.**

***Index Terms*—rate distortion, lossy source coding, Restricted Boltzmann Machine, Deep Auto-Encoder, Deep Belief Network, Blahut-Arimoto algorithm.**

I. Introduction

In recent years, deep learning methods have demonstrated great achievements in various areas, including machine translation and speech processing [9]. Additionally, deep learning combined with channel coding techniques can improve decoding performance (e.g., bit error rate) for error correcting codes such as high density parity check codes [23].

Lossy source coding is a technique that represents a source with fewer bits and less-than-perfect fidelity. The trade-off between rate and distortion in representing a source is termed *rate-distortion function* and was characterized by Shannon [27]. For uniformly distributed binary sources, modern coding techniques such as polar codes and spatially coupled low-density generator-matrix codes approach rate-distortion function for infinitely large code length [14], [31]. For binary sources with memory (e.g., stationary ergodic source), techniques based on machine learning are proposed to construct rate-distortion approaching lossy source codes [13], [32], [18], [19]. For binary sources neither independent and identically distributed nor stationary ergodic distributed (e.g., binarized images), relatively few codes are available [22], [28]. In this work, we focus on binary lossy source coding and demonstrate that deep autoencoders can be used to construct lossy source codes.

A. Rate Distortion and Lossy Compression

Let $\mathcal{X} = \{0, 1\}$, $N \in \mathbb{N}^+$, $\mathbf{x} = (x_0, x_1, \cdots, x_{N-1}) \in \mathcal{X}^N$ be a source codeword, and $\mathbf{y} = (y_0, y_1, \cdots, y_{N-1}) \in \mathcal{X}^N$ be a reproduced codeword. For any $(x, y) \in (\mathcal{X}, \mathcal{X})$, denote the distortion of x and y as $\varphi(x, y) \in \mathbb{R}^+$. Let $\varphi(\mathbf{x}, \mathbf{y}) = \frac{1}{N}\sum_{i=0}^{N-1} \varphi(x_i, y_i)$ be the average distortion of $\mathbf{x}$ and $\mathbf{y}$.

Qing Li is now with Western Digital Research, Milpitas, CA, 95035, USA (e-mail: Qing.Li7@wdc.com). Yang Chen is now with the Department of Statistics and Michigan Institute for Data Science, University of Michigan, Ann Arbor, MI 48109, USA (e-mail: ychenang@umich.edu).

For source and encoded codeword sizes $N, M \in \mathbb{N}^+$ and a distortion $D \in \mathbb{R}^+$, an (N, M, D)-*rate distortion code* consists of: 1) an index set $\mathcal{D} = \{0, 1, \cdots, M-1\}$ and a codebook $\mathcal{C} \subseteq \mathcal{X}^N$, 2) an encoding function $f_N : \mathcal{X}^N \to \mathcal{D}$, and 3) a reproducing function $g_N : \mathcal{D} \to \mathcal{C}$ such that $\bar{D} \stackrel{\text{def}}{=} \mathbb{E}(\varphi(\mathbf{x}, g_N(f_N(\mathbf{x})))) \le D$, and the expectation $\mathbb{E}(\cdot)$ is taken with respect to the probability distribution on $\mathbf{x} \in \mathcal{X}^N$.

The *rate distortion* function is defined as $R(D) = \lim_{N\to\infty} R_N(D)$, where $R_N(D)$ is the N^{th} order rate distortion function: $R_N(D) \stackrel{\text{def}}{=} \inf_{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \in P_N(D)} \frac{I(\mathbf{x},\mathbf{y})}{N}$, where $P_N(D) \stackrel{\text{def}}{=} \{P_{\mathbf{Y}|\mathbf{X}} : \sum_{\mathbf{x},\mathbf{y}} P_{\mathbf{X}}(\mathbf{x}) P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \varphi(\mathbf{x}, \mathbf{y}) \le D\}$, $P_{\mathbf{X}}$ is a given source probability distribution, $P_{\mathbf{Y}|\mathbf{X}}$ is the conditional probability distribution, and $I(\mathbf{x}, \mathbf{y})$ is the mutual information between $\mathbf{x}$ and $\mathbf{y}$. For a fixed N, let $\mu_{\mathbf{Y}|\mathbf{X}}$ be the conditional distribution achieving the N^{th} order rate distortion, and let $\mu_{\mathbf{Y}}$ be the resulting marginal distribution.

B. Deep Learning Models

Deep learning models such as Restricted Boltzmann Machines (RBMs [7]) and Deep Belief Networks (DBNs [10]) are universal approximators, that is, they are able to approximate any binary sequence distribution. The representation power of DBNs does not decrease by adding layers [15]. Thus RBMs and DBNs can be used to learn $\mu_{\mathbf{Y}|\mathbf{X}}$ and $\mu_{\mathbf{Y}}$.

1) Restricted Boltzmann Machines: An RBM (see Fig. 1) can be represented with parameters $(\mathbf{W}, \mathbf{b}, \mathbf{c})$: a binary stochastic unit *visible layer*, $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1}) \in \mathcal{X}^N$, is connected to a binary stochastic unit *hidden layer*, $\mathbf{h} = (h_0, h_1, \cdots, h_{K-1}) \in \mathcal{X}^K$; the weights of the edges are denoted by a matrix $\mathbf{W} = [w_{ij}]_{N\times K} \in \mathbb{R}^{N\times K}$, and $\mathbf{v}$ and $\mathbf{h}$ are also attached to bias terms $\mathbf{b} = (b_0, b_1, \cdots, b_{N-1}) \in \mathbb{R}^N$ and $\mathbf{c} = (c_0, c_1, \cdots, c_{K-1}) \in \mathbb{R}^K$, respectively.

The joint distribution of $\mathbf{v}$ and $\mathbf{h}$, $P_{\mathbf{VH}}(\mathbf{v}, \mathbf{h})$, is defined as a Boltzmann distribution, $P_{\mathbf{VH}}(\mathbf{v}, \mathbf{h}) = Z^{-1}\exp(-E(\mathbf{v}, \mathbf{h}))$, where the energy function $E(\mathbf{v}, \mathbf{h})$ is given by $E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} w_{i,j} h_j v_i - \sum_i b_i v_i - \sum_j c_j h_j$, and the partition function Z is given by $Z = \sum_{\mathbf{h},\mathbf{v}} \exp(-E(\mathbf{v}, \mathbf{h}))$. Integrating out the hidden layer $\mathbf{h}$ gives the marginal distribution of the variables $\mathbf{v}$:

$$P_{\mathbf{V}}(\mathbf{v}) = \frac{1}{Z} \prod_{i=0}^{N-1} \exp(b_i v_i) \prod_{j=0}^{K-1} (1 + \exp(c_j + \sum_{i=0}^{N-1} w_{ij} v_i)). \tag{1}$$

2) Deep Belief Network: DBNs (see Fig. 1) can be viewed as a stack of RBMs, where each RBM's hidden layer serves as the visible layer for the next. A DBN with L ($L \in \mathbb{N}^+$) layers models the joint distribution between observed vector

978-1-7281-3151-1/19/$31.00 ©2019 IEEE

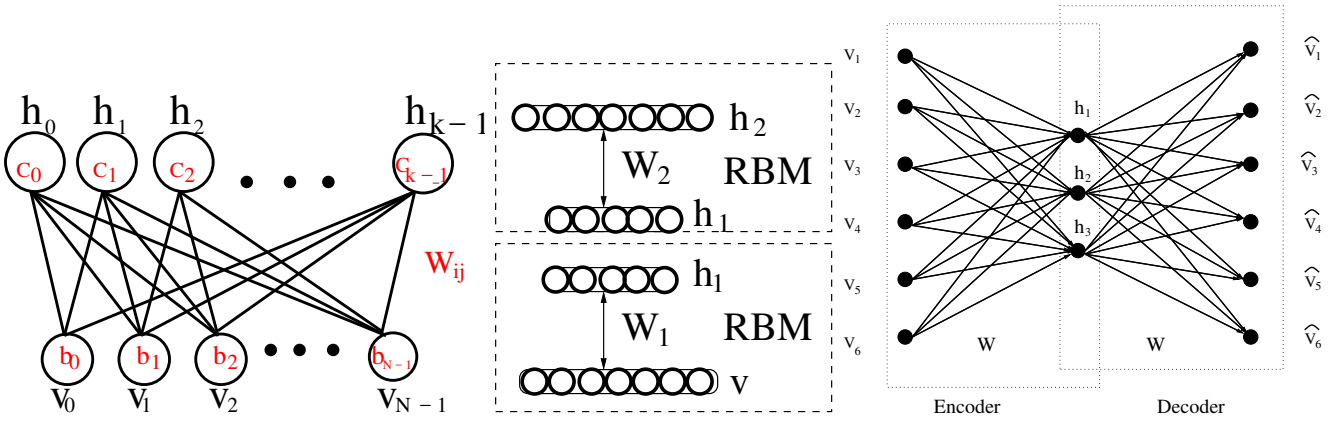

Fig. 1: Deep Learning models. **Left**: Restricted Boltzmann Machines; **Middle**: Deep Belief Networks; **Right**: AutoEncoder.

$\mathbf{v}$ and L hidden layer $\mathbf{h}^1 \in \mathcal{X}^{K_1}, \mathbf{h}^2 \in \mathcal{X}^{K_2}, \cdots, \mathbf{h}^L \in \mathcal{X}^{K_L}$ for $K_1, K_2, \cdots, K_l \in \mathbb{N}^+$ as follows:

$$P(\mathbf{v}, \mathbf{h}^1, \cdots, \mathbf{h}^L) = (\prod_{k=1}^{L-1} P(\mathbf{h}^{k+1}|\mathbf{h}^k)) P(\mathbf{h}^0, \mathbf{h}^L), \quad (2)$$

where $\mathbf{v} = \mathbf{h}^0$, $P(\mathbf{h}^k|\mathbf{h}^{k+1})$ is a visible-given-hidden conditional distribution in an RBM associated with level k of the DBN, and $P(\mathbf{h}^0, \mathbf{h}^L)$ is the joint distribution in the top-level RBM. For convenience, we represent l layers DBN by $\{(\mathbf{W}^l, \mathbf{b}^l, \mathbf{c}^l), l = 1, 2, \cdots, L\}$. DBNs are trained in a greedy layer-wise fashion [10].

C. Related work

In 2006, Hinton et al. presented a dimension reduction method in Science [11], which broke the stalemate that neural network was in low tide for a long time. As we show later, this method is essentially a lossless compression method from the coding point of view. Another machine learning based compression scheme was put forth by Jalali et al. [12], where ideas such as Markov Chain Monte Carlo (MCMC) and simulated annealing were used to study rate distortion and lossy source coding for stationary ergodic sources. Interestingly, MCMC and annealing are widely used by RBMs/DBNs to learn Boltzmann distribution, to which $\mu_{\mathbf{Y}|\mathbf{X}}$ and many other information theoretic capacity approaching probabilities belong. Motivated by [11], [12], our previous work [18], [19] revealed the connection between rate distortion and RBMs/DBNs, showed that RBMs/DBNs can be trained to learn $\mu_{\mathbf{Y}|\mathbf{X}}$ and $\mu_{\mathbf{Y}}$, approximate $R_N(D)$, and derived a lossy source coding scheme for stationary ergodic sources.

D. Contributions and Outlines

In this work, we extend Hinton's methods and our previous work to build a lossy source compression scheme based on RBM-based Deep AutoEncoder (DAE). Unlike traditional linear schemes (e.g., [14], [31]), our scheme is non-linear, suitable for any binary additive distortion measure and for any binary distributed source (though may not be optimal). Although autoencoders play an active role in deep learning based image compression (see references [3], [29]), to the best knowledge of the authors, this is the first time RBM-based DAE is studied from the coding perspective. Another way to evaluate our contribution is that we provide an alternative for dimension reduction based on lossy source coding.

The paper is organized as follows. We review our previous work, DBN Blahut-Arimoto algorithm, in Section II. A lossy compression based on RBM-DAE scheme is presented in Section III. Experimental results are presented in Section IV. Section V gives conclusions and future work.

II. DBN Blahut-Arimoto Algorithm

In this part, we briefly review the DBN Blahut-Arimoto (DBN-BA) algorithm proposed to learn $\mu_{\mathbf{Y}|\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ in [18], [19], which is the corner stone of the proposed DAE compression scheme.

A. DBN interpretation of Rate Distortion

The well-studied algorithms of DBN (e.g., sampling and learning) makes it an efficient model to learn $\mu_{\mathbf{Y}|\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ when the source is no longer independent and identically distributed. The key idea behind DBN-BA is to represent $\mu_{\mathbf{Y}}$ by one DBN and $\mu_{\mathbf{Y}|\mathbf{X}}$ by the modified DBN of $\mu_{\mathbf{Y}}$.

Lemma 1. [5, chapter 10, p.p.330] [1]

$$\mu_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z'_\beta(\mathbf{x})} \mu_{\mathbf{Y}}(\mathbf{y}) \exp(-\beta\varphi(\mathbf{y}, \mathbf{x})), \quad (3)$$

$$R_N(D) = \frac{\mathbb{E}(-\log_2 Z'_\beta(\mathbf{x}))}{N} - \frac{\beta D}{\ln 2}, \quad (4)$$

where the expectation is with respect to the probability distribution on $\mathbf{x}$,

$$Z'_\beta(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{y}} \mu_{\mathbf{Y}}(\mathbf{y}) \exp(-\beta\varphi(\mathbf{x}, \mathbf{y})),$$

and β is the Lagrange multiplier of the minimization of $I(\mathbf{x}, \mathbf{y}) + \beta\mathbb{E}(\varphi(\mathbf{x}, \mathbf{y}))$.

[1] [5, chapter 10, p.p.330] presents the single-letter characterization, and it holds for the block case by regarding $\mathbf{x}$ and $\mathbf{y}$ as elements of the alphabet $\{0, 1, \cdots, |\mathcal{X}|^N\}$.

Theorem 1. [19] For the distortion $\varphi(0,0)=\varphi(1,1)=0$, $\varphi(0,1)=a$, $\varphi(1,0)=b$, $a\geq b>0$, assume $\mu_{\mathbf{Y}}$ can be represented by one DBN of L layers, $\{(\mathbf{W}^l,\mathbf{b}^l,\mathbf{c}^l), l=1,2,\cdots,L\}$. Then, $\mu_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ can be represented by the DBN $\{(\mathbf{W}^l_{\mathbf{Y}|\mathbf{X}},\mathbf{b}^l_{\mathbf{Y}|\mathbf{X}},\mathbf{c}^l_{\mathbf{Y}|\mathbf{X}}), l=1,2,\cdots,L\}$:

$$\begin{cases} \mathbf{W}^1_{\mathbf{Y}|\mathbf{X}} &= & \mathbf{W}^1_{\mathbf{Y}}, \\ \mathbf{c}^1_{\mathbf{Y}|\mathbf{X}} &= & \mathbf{c}^1_{\mathbf{Y}}, \\ b^1_{\mathbf{Y}|\mathbf{X},i} &= & b^1_{\mathbf{Y},i}-\beta a 1_{x_i=0}+\beta b 1_{x_i=1}, \end{cases}$$

$$\begin{cases} \mathbf{W}^l_{\mathbf{Y}|\mathbf{X}} &= & \mathbf{W}^l_{\mathbf{Y}}, \\ \mathbf{c}^l_{\mathbf{Y}|\mathbf{X}} &= & \mathbf{c}^l_{\mathbf{Y}}, \qquad l\geq 2. \\ b^l_{\mathbf{Y}|\mathbf{X}} &= & b^l_{\mathbf{Y}}. \end{cases} \tag{5}$$

For the distortion $\varphi(0,0)=\varphi(1,1)=0$, $\varphi(0,1)=a$ $\varphi(1,0)=b$, $a\geq b>0$, when $a=b=1$, it is effectively the hamming distance; When $a\neq b$, it is the asymmetric distortion measure; When $a=\infty$, it is the deterministic WEM measure, which is commonly used in defect storage [24], [8], [20].

B. DBN Blahut-Arimoto Algorithm

DBN is a good candidate to learn $\mu_{\mathbf{Y}}$ when the source is binary memory case, and we present DBN Blahut-Arimoto (DBN-BA) algorithm in Algorithm 1, where t_{max} denotes the maximal number of DBN-BA iterations, n denotes the number of sample size, and $(\mathbf{W}^{l,t}_{\mathbf{Y}},\mathbf{b}^{l,t}_{\mathbf{Y}},\mathbf{c}^{l,t}_{\mathbf{Y}})$ denotes the l^{th} layer of DBN after t iterations for $l=1,2,\cdots,L$.

Algorithm 1 DBN Blahut-Arimoto

1: **procedure** DBN-BA($P_{\mathbf{X}}(\mathbf{x}),\beta,\varphi(\cdot)$)
2: initialize $\{(\mathbf{W}^{l,0}_{\mathbf{Y}},\mathbf{b}^{l,0}_{\mathbf{Y}},\mathbf{c}^{l,0}_{\mathbf{Y}})\}$ arbitrarily.
3: **for** $t=1,\ldots,t_{\max}$ **do**
4: sample $\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n$ based on $P_{\mathbf{X}}(\mathbf{x})$.
5: **for** $\mathbf{x}_i\in\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$ **do**
6: sample $\mathbf{y}_i$ from $\{(\mathbf{W}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{b}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{c}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i))\}$.
7: **end for**
8: train $\{(\mathbf{W}^{l,t}_{\mathbf{Y}},\mathbf{b}^{l,t}_{\mathbf{Y}},\mathbf{c}^{l,t}_{\mathbf{Y}})\}$ with $\{\mathbf{y}_1,\mathbf{y}_2,\cdots,\mathbf{y}_n\}$.
9: **end for**
10: **return** $\{(\mathbf{W}^{l,t_{max}}_{\mathbf{Y}},\mathbf{b}^{l,t_{max}}_{\mathbf{Y}},\mathbf{c}^{l,t_{max}}_{\mathbf{Y}})\}$.
11: **end procedure**

The above algorithm applies the well-known alternating optimizing algorithm [33, chapter 9]: First, we arbitrarily choose a $\{(\mathbf{W}^{l,0}_{\mathbf{Y}},\mathbf{b}^{l,0}_{\mathbf{Y}},\mathbf{c}^{l,0}_{\mathbf{Y}})\}$ and let it represent $\mu^0_{\mathbf{Y}}$ (i.e., line 2); Then in general $\mu^t_{\mathbf{Y}}$ is defined as $\mu^t_{\mathbf{Y}}(\mathbf{y})=\sum_{\mathbf{x}}P_{\mathbf{X}}(\mathbf{x})\mu^t_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$, and $\mu^t_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ relates with $\mu^{t-1}_{\mathbf{Y}}(\mathbf{y})$ in the form of Equation (4). Line 6 obtains $\{(\mathbf{W}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{b}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{c}^{l,t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i)\}$ representing $\mu^{t-1}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$. Line 8 obtains $\{(\mathbf{W}^{l,t}_{\mathbf{Y}},\mathbf{b}^{l,t}_{\mathbf{Y}},\mathbf{c}^{l,t}_{\mathbf{Y}}\}$ representing $\mu^t_{\mathbf{Y}}(\mathbf{y})$. The alternating optimizing guarantees the following converenge [33, chapter 9.3]: $\mu^t_{\mathbf{Y}}\to\mu_{\mathbf{Y}}$ as $t\to\infty$.

The characterization of the above DBN-BA performance was obtained in [19, Thorem 3], which claims that for a fixed t_{max} there exists a sample size number n such that Kolmogorov-Smirnov distance between the learned marginal $Q^t_{\mathbf{Y}}$ and $\mu_{\mathbf{Y}}$ is small enough.

III. Deep-AutoEncoder Lossy Source Codes

In this part, we present a compression scheme based on stacked RBMs (or DBN) Deep AutoEncoder (DAE) [11]. The proposed DAE compression scheme is one variant of the known fixed-distortion scheme, where the distortion is fixed and compression rate is left to optimize. The fixed distortion is achieved by DBN Blahut-Arimoto algorithm to approximate the N^{th}-order rate distortion approximating posterior. The trained DBNs are then unrolled to create a DAE, which produces an encoder and a reproducer. The unrolled DAE is fine-tuned with back-propagation through the whole autoencoder to minimize reconstruction errors. The two stage training is based on the well-known deep learning principle that a better training a deep network strategy is not to directly optimize the objective of interest (for our case, the target distortion) by gradient descent but to initially use a local unsupervised criterion to pre-train each layer in turns [30].

A. Deep AutoEncoder for Compression

1) Background: We brief the deep autoencoder framework. More details can be found in [2]. By a slight abuse of notations from RBM, we denote $\theta=(\mathbf{W},\mathbf{b},\mathbf{c})$ and the *sigmoid* function as $s(x)=\frac{1}{1+e^{-x}}$ for $x\in\mathbb{R}$. An autoencoder (see Fig. 1) consists of an *encoder* $f_\theta(\mathbf{v})=s(\mathbf{W}\mathbf{v}+\mathbf{b})=\mathbf{h}$, which is to transform an input vector $\mathbf{v}$ to a hidden representation $\mathbf{h}$, and a *decoder*, $g_\theta(\mathbf{h})=s(\mathbf{W}^T\mathbf{h}+\mathbf{c})=\hat{\mathbf{v}}$, which is to map $\mathbf{h}$ back to a reconstructed $\hat{\mathbf{v}}$. The autoencoder is trained to minimize the *reconstruction error*, which can be either the squared error (when $\mathbf{v}=(v_0,v_1,\cdots,v_{N-1})\in\mathbb{R}^N$, and v_i is probability of being one) or the cross-entropy (when $\mathbf{v}\in\mathcal{X}^N$). The deep autoencoder is viewed as a stack of autoencoders, and we use $\theta^L=\{(\mathbf{W}^l,\mathbf{b}^l,\mathbf{c}^l), i=1,2,\cdots,L\}$ to denote a DAE of L layers.

2) Proposed scheme: Now we present the DAE-compression algorithm (see Fig. 2), which consists of

1) Training stage, which consists of deep learning BA process and fine tuning process.
   a) Deep learning BA process. The purpose of this step is to learn $\{(\mathbf{W}^l_{\mathbf{Y}},\mathbf{b}^l_{\mathbf{Y}},\mathbf{c}^l_{\mathbf{Y}})\}$ for $\mu_{\mathbf{y}}$ based on DBN-BA (i.e., Algorithm 1).
   b) Unrolling. After pre-training a DBN, the DBN is unrolled to create a deep autoencoder. The unrolled autoencoder produces an encoder and a reproducer for lossy compression.
   c) Fine tuning process. The fine tuning stage uses back-propagation [26] through the whole autoencoder to fine-tune the optimal reconstruction error between the autoencoder input and the autoencoder output. More precisely, given the training set $\mathcal{T}=\{\mathbf{x}_0,\mathbf{x}_1,\cdots,\mathbf{x}_n\}\in\mathcal{X}^{n\times N}$ to compress, denote $\mathcal{T}_{DBN}=\{\mathbf{y}'_0,\mathbf{y}'_1,\cdots,\mathbf{y}'_n\}\in\mathcal{X}^{n\times N}$ the transposed set, where $\mathbf{y}'_i$ is sampled from $\{(\mathbf{W}^l_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{b}^l_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i),\mathbf{c}^l_{\mathbf{Y}|\mathbf{X}}(\mathbf{x}_i))\}$, and denote $\mathcal{T}_{DAE}=\{\mathbf{y}_0,\mathbf{y}_1,\cdots,\mathbf{y}_n\}\in\mathcal{X}^{n\times N}$ the reconstructed set by the DAE, and the fine-tuning process is to minimize the reconstruction error

between $\mathcal{T}_{DBN}$ and $\mathcal{T}_{DAE}$ by adjusting θ^L. Denote the fined-tuned encoder and the decoder as $\{(\mathbf{W}_{\mathbf{Y}}^{l,en}, \mathbf{b}_{\mathbf{Y}}^{l,en}, \mathbf{c}_{\mathbf{Y}}^{l,en})\}$ and $\{(\mathbf{W}_{\mathbf{Y}}^{l,re}, \mathbf{b}_{\mathbf{Y}}^{l,re}, \mathbf{c}_{\mathbf{Y}}^{l,re})\}$, respectively.

2) Compression/Reproduction stage, which consists of the compression (Algorithm 2) and the reproduction (Algorithm 3).

Algorithm 2 Deep Autoe-ncoder compression

procedure DAE-ENCODING($\mathbf{x}, \beta, \{(\mathbf{W}_{\mathbf{Y}}^{l,en}, \mathbf{b}_{\mathbf{Y}}^{l,en}, \mathbf{c}_{\mathbf{Y}}^{l,en})\}$)
  Obtain $\{(\mathbf{W}_{\mathbf{Y}|\mathbf{X}}^{l,en}(\mathbf{x}), \mathbf{b}_{\mathbf{Y}|\mathbf{X}}^{l,en}(\mathbf{x}), \mathbf{c}_{\mathbf{Y}|\mathbf{X}}^{l,en})(\mathbf{x})\}$ based on Theorem 1.
  Sample the top hidden layer variable, $\mathbf{m}$, from $\{(\mathbf{W}_{\mathbf{Y}|\mathbf{X}}^{l,en}(\mathbf{x}), \mathbf{b}_{\mathbf{Y}|\mathbf{X}}^{l,en}(\mathbf{x}), \mathbf{c}_{\mathbf{Y}|\mathbf{X}}^{l,en}(\mathbf{x}))\}$.
  return $\mathbf{m}$
end procedure

Algorithm 3 Deep autoencoder reproduction

procedure DAE-REPRODUCING($\mathbf{m}, \{(\mathbf{W}_{\mathbf{Y}}^{l,re}, \mathbf{b}_{\mathbf{Y}}^{l,re}, \mathbf{c}_{\mathbf{Y}}^{l,re})\}$)
  Sample the visible layer variable, $\mathbf{y}$, from $\{(\mathbf{W}_{\mathbf{Y}}^{l,re}, \mathbf{b}_{\mathbf{Y}}^{l,re}, \mathbf{c}_{\mathbf{Y}}^{l,re})\}$.
  return $\mathbf{y}$.
end procedure

B. Analysis and notes

1) Rate and distortion: For a fixed β and N, let $(R_{N,\beta}, D_{N,\beta})$ denote the point lying on the N^{th} rate distortion curve. The $\{(\mathbf{W}_{\mathbf{Y}}^{l}, \mathbf{b}_{\mathbf{Y}}^{l}, \mathbf{c}_{\mathbf{Y}}^{l})\}$ is trained to approximate $\mu_{\mathbf{Y}}$, and theoretical analysis in [18] shows that the Kolmogorov-Smirnov distance between $\mu_{\mathbf{Y}}$ and $P_{\mathbf{Y}}$ represented by the $\{(\mathbf{W}_{\mathbf{Y}}^{l}, \mathbf{b}_{\mathbf{Y}}^{l}, \mathbf{c}_{\mathbf{Y}}^{l})\}$ decreases with larger n and t in Algorithm 1. Thus, the average distortion with $\{(\mathbf{W}_{\mathbf{Y}}^{l}, \mathbf{b}_{\mathbf{Y}}^{l}, \mathbf{c}_{\mathbf{Y}}^{l})\}$ is approximately $D_{N,\beta}$, i.e., $\bar{D} \sim D_{N,\beta}$. The fine-tuning process (i.e., step (1.c)) causes $\{(\mathbf{W}_{\mathbf{Y}}^{l,re}, \mathbf{b}_{\mathbf{Y}}^{l,re}, \mathbf{c}_{\mathbf{Y}}^{l,re})\}$ and $\{(\mathbf{W}_{\mathbf{Y}}^{l,re}, \mathbf{b}_{\mathbf{Y}}^{l,re}, \mathbf{c}_{\mathbf{Y}}^{l,re})\}$ shift from $\{(\mathbf{W}_{\mathbf{Y}}^{l}, \mathbf{b}_{\mathbf{Y}}^{l}, \mathbf{c}_{\mathbf{Y}}^{l})\}$, which further leads to the difference between $\bar{D}$ and $D_{N,\beta}$ (see Fig. 4).

For a DAE-encoder $\{(\mathbf{W}_{\mathbf{Y}}^{l,en}, \mathbf{b}_{\mathbf{Y}}^{l,en}, \mathbf{c}_{\mathbf{Y}}^{l,en})\}$, the compressed information is represented by the top layer (i.e., the *bottleneck* layer) hidden variable, $\mathbf{h}^L$, therefore, the rate of the proposed scheme is $R = \frac{K_L}{N}$. The compressed size can be further reduced by lossless compress $\mathbf{h}^L$, however, we leave that for future work.

2) Fixed-distortion and fixed-rate schemes: Note that for a fixed β and N, the above DAE-compression first fixes the average distortion $\bar{D}$ to $D_{N,\beta}$ by iteratively approaching $\mu_{\mathbf{y}}$ (i.e, step (1.a)), and then minimizes the reconstruction error via back-propagation (i.e., step (1.c)). By fixing β and varying the bottleneck layer (i.e., K_L), the above DAE compression scheme is essentially the known *fixed-distortion* scheme. Particularly, when $\beta = 0$, the proposed DAE scheme degenerates to the one proposed by Hinton et al [11], which is a lossless compression by optimizing K_L.

Similarly by fixing K_L and varying β the above DAE compression scheme is another instance of the *fixed-rate* scheme ([32]).

IV. Experiments

In this section, we present our experimental results to validate the proposed DAE compression scheme. A simple and widely used compression method is the principal components analysis (PCA)[28], which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. Therefore, we also compare our results with those of PCA.

A. Dataset

We use the MNIST digit dataset [16], which is a standard dataset for deep learning community. The MNIST digit dataset contains 60,000 training images and 10,000 test images of ten handwritten digits (0 to 9) with 28 $\times$ 28 pixels. The data set is binarized: each pixel value was statistically set to 1 in proportion to its pixel intensity. Therefore, each image of MNIST is 784 bits. The binarized MNIST is suitable to test our DAE compression, because each digit is neither i.i.d. distributed nor stationary distributed. The distortion metric is the Hamming distance, i.e., $\varphi(0,0) = \varphi(1,1) = 0$, $\varphi(0,1) = \varphi(1,0) = 1$.

B. Training details

We built our DAE compression scheme on top of the TensorFlow framework and used an NVIDIA GeForce GTX 960 GPU for accelerated training. We implemented the fixed-distortion scheme by fixing β and varying K_L. For each fixed β, we first started with a large K_L, then gradually decreased K_L till the reconstructed digits are hard to recognize (see Fig. 3 for one example). In all our experiments, we use DAE of four layers, of which the number of first four layers neurons are fixed as 784, 1000, 900, 800, respectively, and K_L varies for different β's. The training of DBN was conducted using greedy training algorithm proposed by Hinton et al. [10], and the training of the DAE was conducted by the Adam's optimizer. The DBN-BA maximal iteration was set to 12, DBN-BA sample size was 60,000, the mini-batch size was 30, number of epochs was 50, and the learning rate was 0.01.

C. Results

The lossy compression results of MNIST images with DAE are presented in Table. I, and the rate-distortion curve is drawn in Fig. 4. For reference, the compressed size by lossless compression scheme presented in [11] is 30 bits with 784-1000-500-250-30 autoencoder. In Fig. 4, we also present the average distortion after the DBN-BA algorithm. Compared with the eventual DAE compression scheme, there is a distortion loss, and this is due to the fine-tuning process where $\{(\mathbf{W}_{\mathbf{Y}}^{l,en}, \mathbf{b}_{\mathbf{Y}}^{l,en}, \mathbf{c}_{\mathbf{Y}}^{l,en})\}$ deviates from $\{(\mathbf{W}_{\mathbf{Y}}^{l}, \mathbf{b}_{\mathbf{Y}}^{l}, \mathbf{c}_{\mathbf{Y}}^{l})\}$.

The lossy compression results of MNIST image with PCA are presented in Table. II, where the number of principle components are the same for the DAE counterpart, and the rate-distortion curve in drawn in Fig. 4. We can see that for high compress rates, DAE achieves lower distortion loss as compared to the PCA while under-performs the PCA with lower compress rates.

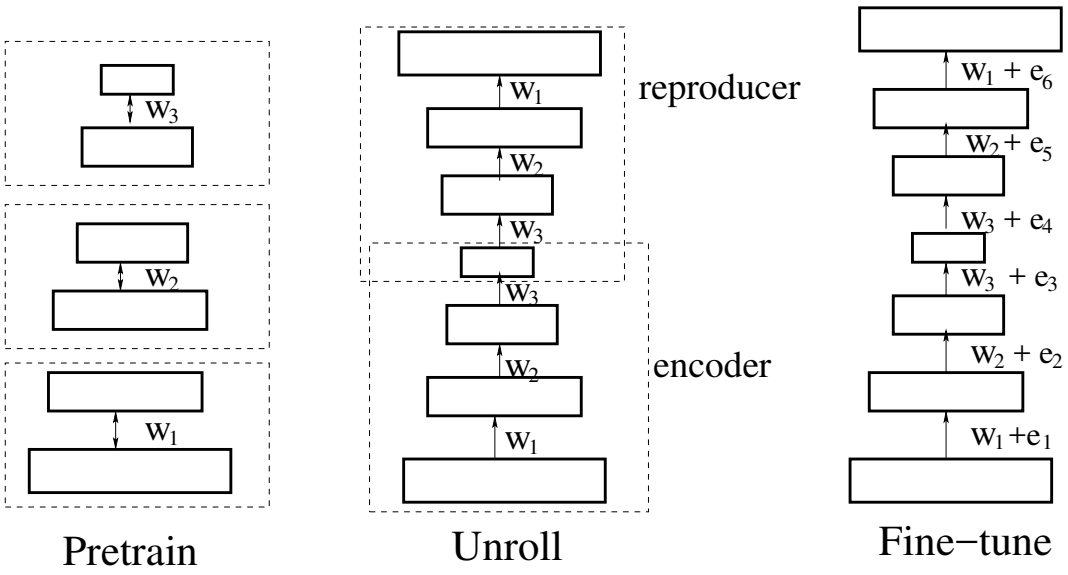

Fig. 2: Block diagram of the proposed compression training scheme. **Left**: Pre-training consists of running deep learning-BA on a stack of RBMs; **Middle**: After pre-training, the RBMs are unrolled to create a deep autoencoder, which is served as encoder and reproducer for lossy source coding; **Right**: The unrolled autoencoder is then fine-tuned using back-propagation.

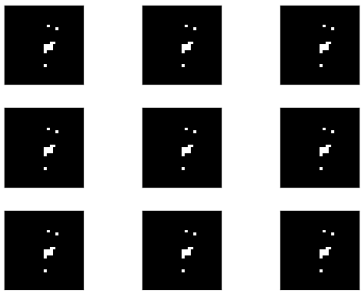

Fig. 3: An example of a unrecognizable image due to excessive compression.

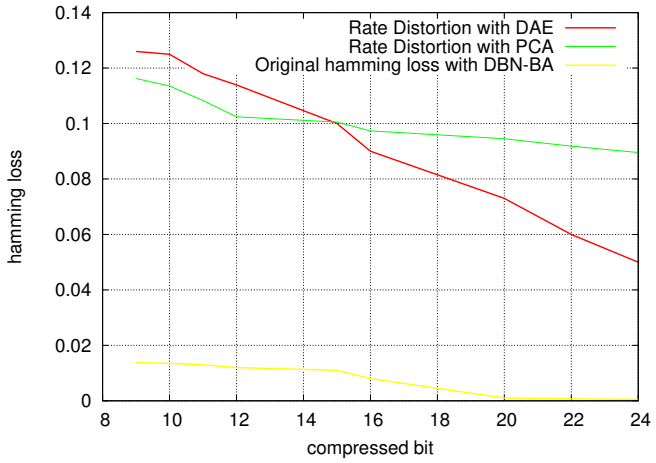

Fig. 4: Rate Distortion curves of the DAE/PCA scheme on MNIST images.

V. Conclusion and Future Work

This paper presents the applications of Restricted Boltzmann Machines (RBMs) based deep autoencoder to lossy compression of binary sources. The key steps are: 1) learn the optimal posterior through the Blahut-Arimoto algorithm with stacked RBMs (or Deep Belief Network), which is to then initialize the weight parameters for the deep auto-encoder (DAE); 2) the stacked RBMs are unrolled to create a deep auto-encoder; and 3) the DAE parameters are further fine-tuned for optimal reconstruction.

Lossy compression is the foundation of other information theory/computer science problems, such as Write-efficient Memories [1], [17], information embedding [4], [21], stenography [6], and clustering [25]. Therefore, promising directions for future work are to extend the techniques presented here to the problems listed above.

References

[1] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Information and Computation*, vol. 83, no. 1, pp. 80–97, October 1989.

[2] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[3] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[4] R. J. Barron, B. Chen, and G. W. Wornell, "The duality between information embedding and source coding with side information and some applications," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1159–1180, 2003.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[6] T. Filler and J. Fridrich, "Gibbs construction in steganography," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 705–720, 2010.

[7] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, vol. 7441, 2012.

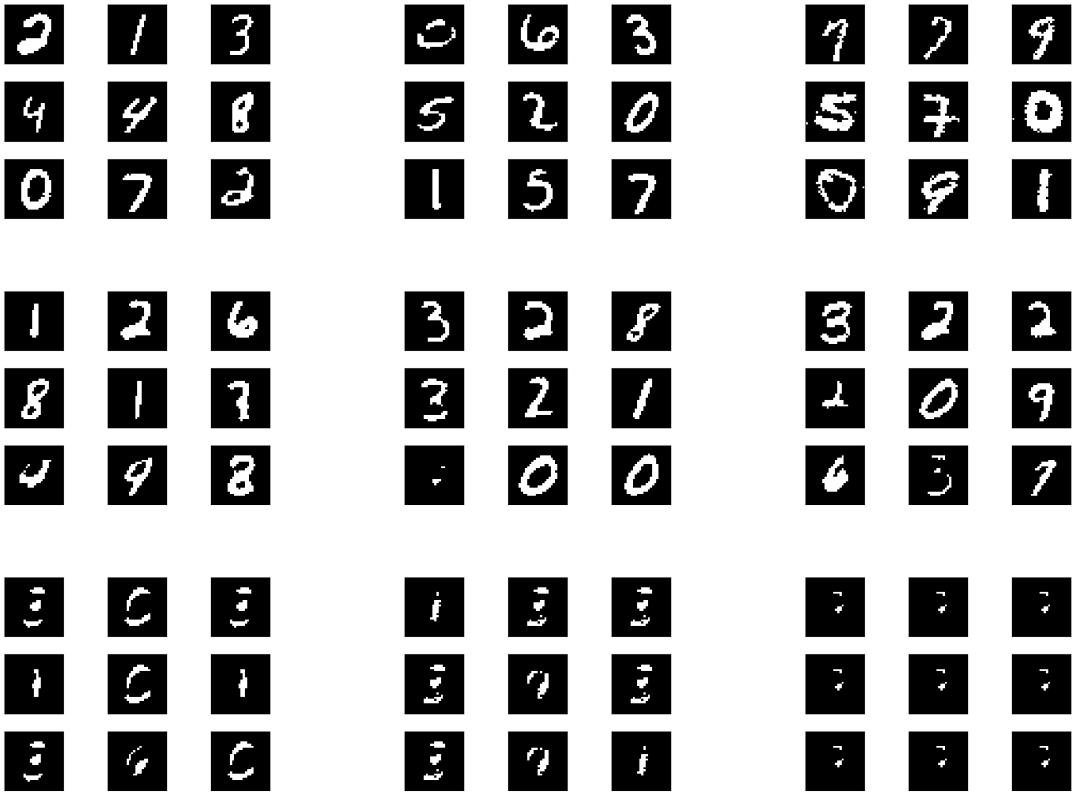

TABLE I: Lossy source compression on MNIST images with DAE. The β's from top to bottom and from left to right are 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, respectively, and the K_L's are 24, 22, 20, 18, 16, 15, 12, 10, 9, respectively.

[8] F. Fu and R. W. Yeung, "On the capacity and error-correcting codes of write-efficient memories," *IEEE Transcations on Information Theory*, vol. 46, no. 7, pp. 2299–2314, November 2000.

[9] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[12] S. Jalali, A. Montanari, and T. Weissman, "Lossy compression of discrete sources via the viterbi algorithm," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2475–2489, 2012.

[13] S. Jalali and T. Weissman, "Block and sliding-block lossy compression via mcmc," *IEEE Transactions on Communications*, vol. 60, no. 8, pp. 2187–2198, 2012.

[14] S. B. Korada and R. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1751–1768, April 2010.

[15] N. Le Roux and Y. Bengio, "Representational power of restricted boltzmann machines and deep belief networks," *Neural computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[16] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[17] Q. Li, "Linear code duality between write-efficient memories and lossy source coding," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2206–2209, 2018.

[18] Q. Li and Y. Chen, "Rate distortion via restricted boltzmann machines," in *Proc. 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2018)*, Monticello, IL, October 2018, pp. 505–512.

[19] ——, "Loss source coding via deep learning," in *Data Compression Conference (DCC), 2019*, 2019.

[20] Q. Li and A. Jiang, "Polar codes are optimal for Write-efficient Memories," in *Proc. 51st Annual Allerton Conference on Communication, Control and Computing (Allerton 2013)*, Monticello, IL, October 2013, pp. 660–667.

[21] P. Moulin and J. A. O'Sullivan, "Information-theoretic analysis of information hiding," *IEEE Transactions on information theory*, vol. 49, no. 3, pp. 563–593, 2003.

[22] T. Murayama, "Thouless-anderson-palmer approach for lossy compression," *Physical Review E*, vol. 69, no. 3, p. 035105, 2004.

[23] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Beery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[24] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Informaton and Control*, vol. 55, pp. 1–19, 1982.

[25] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[27] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec*, vol. 4, no. 142-163, p. 1, 1959.

[28] L. I. Smith, "A tutorial on principal components analysis," Tech. Rep., 2002.

[29] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[30] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[31] M. J. Wainwright, E. Maneva, and E. Martinian, "Lossy Source Com-

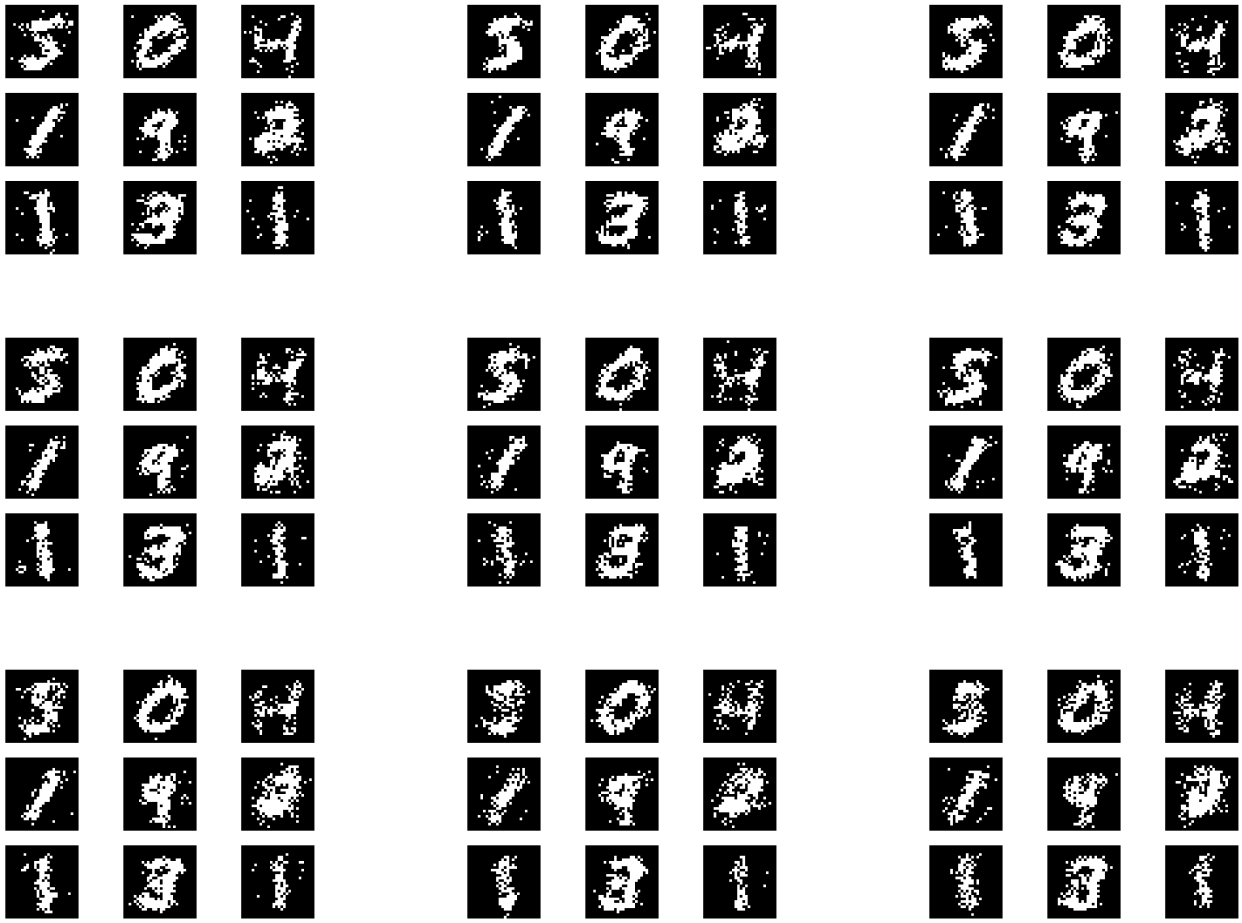

TABLE II: Lossy source compression on MNIST images with PCA. The compressed size from top to bottom and from left to right are 24, 22, 20, 18, 16, 15, 12, 10, 9, respectively.

pression Using Low-Density Generator Matrix Codes: Analysis and Algorithms," *IEEE Transacations on Information Theory*, vol. 56, no. 3, pp. 1351–1367, March 2010.

[32] E.-h. Yang, Z. Zhang, and T. Berger, "Fixed-slope universal lossy data compression," *IEEE Transactions on Information theory*, vol. 43, no. 5, pp. 1465–1476, 1997.

[33] R. W. Yeung, *Information theory and network coding*. Springer Science & Business Media, 2008.