A Class of Compression Systems with Model-Free Encoding

Ying-zong Huang and Gregory W. Wornell
Dept. Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: {zong,gww}@mit.edu

***Abstract*— Practical compression systems are constrained by their bit-stream standards, which define the source model together with the coding method used. We introduce a model-free coding architecture that separates the two aspects of compression and allows the design of potentially more powerful source models, as well as more flexible use of the compressed information stream. We show that this architecture is capable of producing competitive performance while supporting new use cases.**

I. Introduction

Data compression has two conceptually separate aspects: source modeling and coding. In the current widely used systems, it is the compressed bit-stream that is standardized, and therefore, both the source model and the coding strategy are implicitly defined. For example, the JPEG image compression standard implicitly defines a source model based on decorrelated values in the discrete cosine transforms (DCT) domain, and assigns Huffman coding or arithmetic coding as the methods for extracting residual entropy and representing codewords.

In general, the choices of source model and coding strategy at the time of standardization may not be optimal, and these become difficult to change later. Moreover, there is an entanglement of the source modeling and coding aspects that, so far, has led to conceptually restrictive design choices, e.g. asking for a simple entropy coder imposes that the source has a representation that is easy to code; and transmitting compressed streams under network irregularities often requires breaking the encapsulation to deal with the source model.

In this paper, we describe an architecture for compression in which the encoder only makes choices about coding, and produces a compressed bit-stream agnostic to the source model (or indeed, to any information-preserving transformation of the source). It instead leaves to the decoder to apply the relevant knowledge about the source. The compressed stream is a true information stream in that it can be re-ordered, partially lost, accumulated from multiple origins, etc. We call such compression systems as having "model-free" (MF) encoders, or as simply "model-free" coding systems.[1]

This work was supported in part by Draper Laboratory through the UR&D Program and by AFOSR under Grant No. FA9550-11-1-0183.

[1]In universal compression such as Lempel-Ziv, there is no prior knowledge of the source model, so it is learned and used within the encoder. In model-free coding, the source model is *not used* in the encoder whether available or not.

As an illustration of what is possible within this architecture, we explore constructions based on graphical models and iterative decoding, and show that model-free coding can be competitive with existing compression systems while adding new features. While we are motivated by practical concerns, we are guided by information theory, which gives meaning to source modeling as codebook construction and to coding as assigning codebook index according to an indexing scheme. This interpretation we shall defer to a discussion near the end.

A. Prior work and contribution

Relevant prior work comes from several areas.

In distributed coding, there is interest in using "data-like" side information for compression. As in [1], key frames supply side information in Wyner-Ziv video coding. As in [2], [3], the secret key is the side information in the decoding of encrypted compressed data. More broadly, works such as [4] and references therein testify to a wide range of compression problems that take advantage of "data-like" side information.

In compressed sensing, "structure-like" side information such as sparsity is used for reconstruction with fewer measurements [5]. In works such as [6], structures additional to sparsity are exploited — in this case, wavelet-domain structures for the sensing of images.

The common thread in these works is this: the encoder decimates the data to a lower rate by a relatively simple process (e.g. projection, sub-sampling); and the decoder, which possesses side information about the source, chooses the correct sample from among the numerous otherwise ambiguous possibilities. Our work builds on this suggestive idea in several ways, noting that:

1) the decimation process can be any code;
2) the side information can be any source model;
3) the choice of code can be agnostic to the source, i.e. model-free.

With this view, we propose a compression architecture based on random hashing as model-free encoding, and inference over combined source model and coding constraints as decoding. To our knowledge, such a combination has not been attempted for compressing general sources. While we believe this architectural assertion is valid for a wide variety of compression problems, this paper considers essentially the case of lossless

compression with known source model. Furthermore, we restrict attention to decoders based on graphs — graphical source models, codes on graphs, belief propagation on graphs.

As a well studied and algorithmically rich general inferential toolkit [7], [8], graphical methods have become useful to compression or signal reconstruction problems, see for instance [9], [10]. More closely relevant, graphical source models are used for direct image reconstruction given some cutset pixels reminiscent of trivial coding [11], and low-density codes on graphs show good performance for compressing structureless (i.e. memoryless) sources [12].

B. Paper organization

A basic system construction for direct compression is described in Section II. A generalization that handles serialized sources of any alphabet size is described in Section III. In Section IV, performance on some sources is reviewed. Finally, in Section V, we leverage model-free coding to compress encrypted data without the secret key to demonstrate new features this architecture enables.

II. Basic system construction

We describe a prototype of model-free coding and sketch the requisite components that make a working compression system.

A. Core components

Suppose s is an n-vector drawn from a Markov random field (MRF) $p(s)$, and H is a $k \times n$ parity-check matrix of a rate $r_{\text{code}} = k/n$ LDPC code. Both are over $\mathbf{GF}(q)$.

1) Encoding algorithm: The encoder is nearly trivial. Using H as the encoding matrix, it produces

$$x = Hs \tag{1}$$

as the compressed result. k is chosen so that $r_{\text{code}} = k/n$ is the nominal compression rate. The encoder produces some additional output described in Section II-B.

2) Decoder structure – the code subgraph: Since H enforces k hard constraints of the form $x_a = \sum_{i \in S} H_{a,i} s_i$, it can be represented by a bipartite factor graph $\mathcal{C} = (S \cup X, F)$, where there are k factor nodes $X = \{x_1, ..., x_k\}$ and n source nodes $S = \{s_1, ..., s_n\}$. There is an edge between factor node x_a and source node s_i if and only if $H_{a,i} \neq 0$.

The number of edges between factor and source nodes scales the difficulty of inference on $\mathcal{C}$. In LDPC factor graphs, the number of edges (and thus complexity) grows linearly with the source length n so it is entirely feasible.

3) Decoder structure – the source subgraph: The MRF $p(s)$ can be represented by an undirected graph $\mathcal{G} = (S = \{s_1, ..., s_n\}, E)$ over the n source nodes $S = \{s_1, ..., s_n\}$, in the sense that $p(s)$ is factored over the maximal cliques of $\mathcal{G}$:

$$p(s) = \frac{1}{Z} \prod_{C \in \mathbf{cl}(\mathcal{G})} \psi_C(s_C) \tag{2}$$

This expression can represent any source model, but the complexity of inference on $\mathcal{G}$ depends on the number of cliques and their sizes [13]. Not all compressible structures in data (in the computational sense) result in low-complexity factorization in the native domain of s, but many do. In particular, we are interested in the less general pairwise factorization:

$$p(s) = \frac{1}{Z} \prod_{i \in S} \phi_i(s_i) \prod_{(i,j) \in E} \psi_{ij}(s_i, s_j) \tag{3}$$

If indeed $p(s)$ has this factorization, then inference has complexity at most quadratic in source length n. We use this factorization in subsequent parts, but note that the essential methods apply to the general factorization (Eq. 2) as well, if the equivalent factor graph is substituted.

4) Decoding algorithm: The decoder runs (loopy) belief propagation on the combined source-code graph $\mathcal{U} = (S \cup X, E \cup F)$ where the source nodes S are shared between the source and code subgraphs (Fig. 1a). For each source node s_i, there is an edge with potential $\psi_{ij}(s_i, s_j)$ to each of its source neighbors $s_j \in \mathcal{N}_i^{\mathcal{G}}$. There is an attached node potential $\phi_i(s_i)$, sometimes interpreted as bias. There is an edge to each of its factor node neighbors (i.e. compressed symbols) $x_a \in \mathcal{N}_i^{\mathcal{C}}$. For each factor node x_a, there is an edge to each of its source node neighbors $s_k \in \mathcal{N}_a^{\mathcal{C}}$. Messages are passed along all edges. However, we do this modularly by alternating between source message passing and code message passing.

- Source message update: Let $m_{a \to i}(s_i)$ be the message (a function over the alphabet of s_i) from the code subgraph factor x_a to the source node s_i. Let $m_{k \to i}(s_i)$ be the message from the source subgraph neighbor s_k to s_i. For each $s_j \in \mathcal{N}_i^{\mathcal{G}}$, we update the message $m_{i \to j}(s_j)$ according to standard sum-product as

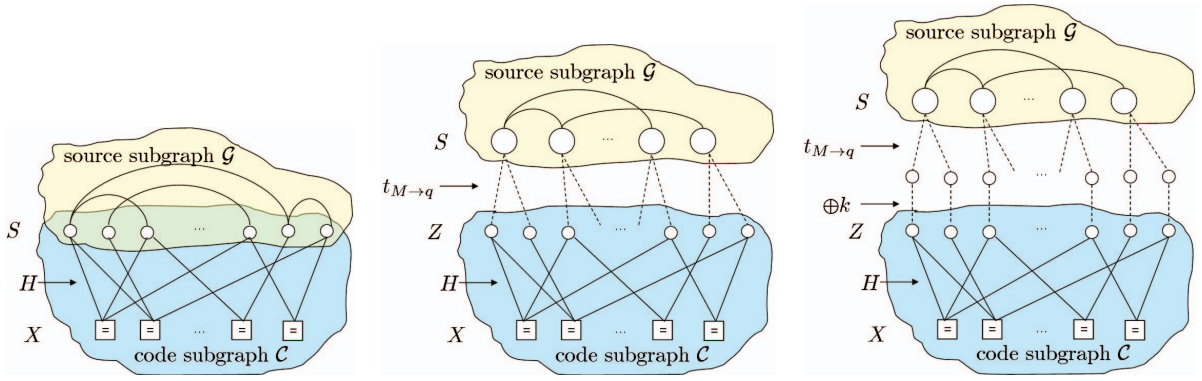
$$m_{i \to j}(s_j) \Leftarrow \sum_{s_i} \left[\prod_{x_a \in \mathcal{N}_i^{\mathcal{C}}} m_{a \to i}(s_i) \right] \phi_i(s_i) \psi_{ij}(s_i, s_j) \prod_{s_k \in \mathcal{N}_i^{\mathcal{G}} \setminus s_j} m_{k \to i}(s_i) \tag{4}$$

  where the bracketed term is the code subgraph belief at s_i encapsulated as an external message to the source subgraph.
- Code message update: Let $m_{b \to i}(s_i)$ be the message from the factor node x_b to its connected source node s_i. We update the source-to-factor message $m_{i \to a}(s_i)$ from s_i to a factor node x_a to which it is connected, according to standard sum-product as

$$m_{i \to a}(s_i) \Leftarrow \left[\prod_{s_j \in \mathcal{N}_i^{\mathcal{G}}} m_{j \to i}(s_i) \phi_i(s_i) \right] \prod_{x_b \in \mathcal{N}_i^{\mathcal{C}} \setminus x_a} m_{b \to i}(s_i) \tag{5}$$

  where the bracketed term is the source subgraph belief at s_i encapsulated as an external message to the code subgraph.

  Next, we update the factor-to-source message. Define



(a) (b) (c)

Fig. 1: The structure of the combined decoder source-code graph $\mathcal{U}$ for the system in (a) Section II; (b) Section III; and (c) Section V.

an indicator for the code subgraph contraint $x_a = \sum_{i\in S} H_{a,i}s_i$ as a function over the source nodes in $\mathcal{N}_a^{\mathcal{C}}$:

$$f_a(\mathcal{N}_a^{\mathcal{C}}) = \begin{cases} 1 & \text{if constraint satisfied} \\ 0 & \text{if not} \end{cases}$$

Let $m_{j\to a}(s_j)$ be the message from the source node s_j to the factor x_a to which it is connected. For each $s_i \in \mathcal{N}_a^{\mathcal{C}}$, we update the message $m_{a\to i}(s_i)$ from x_a to s_i according to standard sum-product as

$$m_{a\to i}(s_i) \Leftarrow \sum_{\mathcal{N}_a^{\mathcal{C}}\setminus s_i} f_a(\mathcal{N}_a^{\mathcal{C}}) \prod_{s_j\in\mathcal{N}_a^{\mathcal{C}}\setminus s_i} m_{j\to a}(s_j)$$

The algorithm is run until convergence or declaration of failure. In the first case, final decoding uses the standard belief equation, which is interpreted as combining partial beliefs from the two subgraphs.

- Belief update:

$$\hat{s}_i = \arg\max_{s_i} \left[\prod_{x_a\in\mathcal{N}_i^{\mathcal{C}}} m_{a\to i}(s_i)\right] \left[\prod_{s_j\in\mathcal{N}_i^{\mathcal{G}}} m_{j\to i}(s_i)\phi_i(s_i)\right] \tag{6}$$

B. Doping symbols

Depending on the source model, the decoding process may not begin without some initial beliefs. Therefore the encoder randomly selects a fraction r_{dope} of source nodes to describe directly to the decoder. These known "doping" symbols anchor the decoding process, and only a small amount — which can be optimized — is necessary. We can consider the doping process as augmenting H with additional unit-weight check rows to make the true encoding matrix H' with true coding rate $r = r_{\text{code}} + r_{\text{dope}}$, and encoding as $x = H's$.[2]

[2]We do not need to communicate the actual content of the matrix H or the locations of the selected doping symbols, if the encoder/decoder pair synchronize on a single random seed. This can be done out-of-band, or the random seed can be included in an initial uncoded header.

C. Parameter estimation

If the decoder does not have access to the full source model, it needs to learn the missing portion from data, generally by training on known samples, or sequentially on the history of decoded data. But it would be useful if decoding can take place with a partial model. This is the case if the source model is among a parametric family for which parameters are unknown. The following is a heuristic that jointly estimates parameters within the decoding loop itself.

Suppose s is drawn from $p(s;\theta)$ where θ denotes unknown parameters. Define the checksum-correctness objective

$$F(\hat{s}) = \frac{1}{k}\|H\hat{s} - x\| \tag{7}$$

where $\|\cdot\|$ is some appropriate norm (e.g. ℓ_0). At each iteration t of the decoding algorithm (Section II-A.4), evaluate $F(\cdot)$ on the contemporary source estimate $\hat{s}^{(t)}(\theta)$ obtained from the total belief update (Eq. 6). The value θ^* that minimizes $F(\hat{s}^{(t)}(\theta))$ among choices in a neighborhood $B_{\delta^{(t)}}(\theta^{(t-1)})$, for some diminishing sequence $\delta^{(1)} > \delta^{(2)} > \cdots > 0$ of bounded sum (e.g. $\delta^{(t)} = \delta/\alpha^t$ for $\delta > 0, \alpha > 1$), is chosen as the parameter estimate $\theta^{(t)}$ for that iteration. At the end of decoding, the sequence of estimates for θ are also converged within some neighborhood.

D. Rate selection

In model-free coding, if the encoder processing does not involve a simulation of the decoding process or upstream hints — which would be the case if the encoder has no source model or cannot access raw data — the encoder cannot anticipate the exact minimum rate to decode. In communication scenarios, some feedback from the decoder is required, such as by acknowledgement of sufficiency as x is sent letter-by-letter. In storage scenarios, this feedback can be used to truncate the compressed output after the fact.

III. Generalized construction

In Section II-A, we introduced a basic compression system where we assumed the source s and the code as represented by

H are over the same field. If coding is to be truly model-free, perhaps it is not realistic to assume the field structure or even the alphabet size of s is known. In real systems, even though s may be highly structured internally, it is often presented to the compression system after it has already been serialized into an opaque bit-stream. LDPC codes and decoding algorithms as well are well developed for $\mathbf{GF}(2)$ and we would want to keep using these no matter the alphabet size of the source. Next we describe how to compress in this setting by inserting a translation layer.

A. Representation and translation

Suppose $s = \{s_1, ..., s_n\}$ is an abstract n-symbol source serialized by symbol-level representational maps. For ease of discussion, we assume all s_i belong to the same alphabet $\mathcal{S}$ of size M, and so there is one map for all n symbols, though this need not be the case. The representation map is a bijective function $t_{M\to q} : \mathcal{S} \to \mathbf{GF}(q)^B$ where $B \geq \log_q M$. For integer symbols s_i serialized into $\mathbf{GF}(2)$, this can be as simple as their machine representations, or other binary expansions like Gray-coding. Likewise, let $t_{M\to q} : \mathcal{S}^n \to \mathbf{GF}(q)^{nB}$ operate on an n-tuple symbol-by-symbol in the obvious way.

When belief messages are passed to or from source nodes, there are related messages on their serialized representations. Define a pair of message translation functions $T_{M\to q} : (\mathcal{S} \to \mathbf{R}^+) \to (\mathbf{GF}(q) \to \mathbf{R}^+)^B$ and $T_{q\to M} : (\mathbf{GF}(q) \to \mathbf{R}^+)^B \to (\mathcal{S} \to \mathbf{R}^+)$ that convert between a message $m^{(M)}$ over $\mathcal{S}$ and a B-tuple of messages $m^{(q)} = m_1^{(q)}, ..., m_B^{(q)}$ over $\mathbf{GF}(q)$. Assuming messages are properly normalized probabilities, for $\omega \in \{1, ..., B\}$ and $\beta \in \mathbf{GF}(q)$,

$$T_{M\to q}(m^{(M)})_\omega(\beta) = \sum_{\alpha\in\mathcal{S}} m^{(M)}(\alpha)\mathbf{1}\left\{t_{M\to q}(\alpha)_\omega = \beta\right\}$$

and for $\alpha \in \mathcal{S}$,

$$T_{q\to M}(m^{(q)})(\alpha) = \prod_{\omega=1}^{B} m_\omega^{(q)}(t_{M\to q}(\alpha)_\omega)$$

These conversions just state that $m^{(M)}$ is a product of marginals $m_1^{(q)}, ..., m_B^{(q)}$, which is not true in general, so there is some potential for loss.

B. Generalized encoding

Encoding takes place in the representational domain of bit-streams, which because of this, should also be called the code domain. Given an LDPC parity-check matrix $H \in \mathbf{GF}(q)^{k\times nB}$ and rate $r_{\text{code}} = k/nB$, the compressed output is

$$x = Hz = Ht_{M\to q}(s) \tag{8}$$

C. Generalized decoding

Recall in Sections II-A.2-II-A.4, the code subgraph $\mathcal{C} = (S\cup X, F)$ and the source subgraph $\mathcal{G} = (S = \{s_1, ..., s_n\}, E)$ share the n source nodes $S = \{s_1, ..., s_n\}$ in $\mathcal{U}$. This is no longer the case here.

Instead, let $\mathcal{G} = (S = \{s_1, ..., s_n\}, E)$ and $\mathcal{C} = (Z \cup X, F)$ where $Z = t_{M\to q}(S)$. The combined source-code graph is $\mathcal{U} = (S\cup Z\cup X, E\cup F)$ (Fig. 1b). The message passing strictly within each subgraph is unchanged from Section II-A.4, but whenever messages cross alphabet/representation boundaries they are translated. Refer to the inset labeled Algorithm 1 on how this is done.

D. Doping symbols

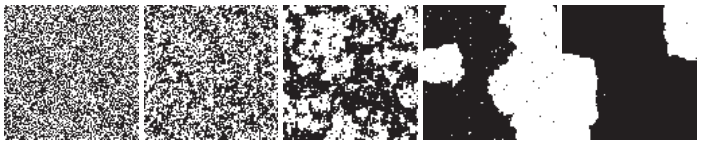
The doping process can occur in either domain, but more naturally it occurs in the code domain unless the representational map is known to the encoder, e.g. it is the encoder that applies serialization. Whichever domain this takes place in, some nodes in $\mathcal{U}$ become observed variables with definite messages, and we obtain the equivalent effect in the other domain by message translation.

IV. Performance

We characterize the compression performance of the system on two sources.

A. Synthetic bi-level images

Sample bi-level images are drawn from a stationary 2D Ising model common in computer vision research. The source model is Eq. 3 with $\phi(1) = 1-\phi(0) = p_{\text{bias}}$ and $\psi(0,0) = \psi(1,1) = 1-\psi(0,1) = 1-\psi(1,0) = p_{\text{stay}}$. The graph $\mathcal{G} = (S, E)$ for an $h \times w$ image has $n = hw$ source nodes, one for each pixel, and an edge between every neighbor pair of pixels in the four cardinal directions for a total of $2hw - (h + w)$ edges in E. Here, we focus on the case of $p_{\text{bias}} = 1/2$ and $p_{\text{stay}} \in [1/2, 1]$ (Fig. 2). For these values, the entropy rate at each p_{stay} is known [14].



(a) (b) (c) (d) (e)

Fig. 2: 100×100 Gibbs sampled images according to the model in Section IV-A. $p_{\text{bias}} = 1/2$. From left to right, $p_{\text{stay}} = 0.5, 0.6, 0.7, 0.8, 0.9$.

1) Results: We generate 100×100 Gibbs samples from this source and compare average rate performance over 20 trials at each parameter value of p_{stay}. Rate performance (output bits per input bit) of the proposed system is compared against some common compressors:

- `MF`: The system proposed in Section II ("MF" for model-free). An off-the-shelf library of regular-(6,3) binary LDPC codes is used. The rate performance denotes the minimal total rate r for which decoding converges (in this case, within 100 iterations) to the correct result.
- `JBIG2`: This state-of-the-art bi-level image compressor is based on 2D context dictionaries. We operate the encoder in lossless mode. The output length is the file size of raw-stream compression, less the compressed file size of a 1-pixel image to account for headers.

Algorithm 1 Decoding with message translation.

Let $\mathcal{N}_{i,\omega}^{\mathcal{C}}$ denote the neighbors of $z_{i,\omega} = t_{M\to q}(s_i)_\omega$ in X,

- Source message update:

$$m_{i\to j}^{(M)}(s_j) \Leftarrow \sum_{s_i} \left[m_{\mathcal{C}\to i}^{(M)}(s_i)\right] \phi_i(s_i)\psi_{ij}(s_i,s_j) \prod_{s_k\in\mathcal{N}_i^{\mathcal{G}}\setminus s_j} m_{k\to i}^{(M)}(s_i) \tag{9}$$

  where $m_{\mathcal{C}\to i}^{(M)}(s_i) = T_{q\to M}\left(\prod_{x_a\in\mathcal{N}_{i,1}^{\mathcal{C}}} m_{a\to i,1}^{(q)}(z_{i,1}), ..., \prod_{x_a\in\mathcal{N}_{i,B}^{\mathcal{C}}} m_{a\to i,B}^{(q)}(z_{i,B})\right)(s_i)$.
- Code message update (source to factor):

$$m_{i,\omega\to a}^{(q)}(z_{i,\omega}) \Leftarrow \left[m_{\mathcal{G}\to i,\omega}^{(q)}(z_{i,\omega})\right] \prod_{x_b\in\mathcal{N}_{i,\omega}^{\mathcal{C}}\setminus x_a} m_{b\to i,\omega}^{(q)}(z_{i,\omega}) \tag{10}$$

  where $m_{\mathcal{G}\to i,\omega}^{(q)}(z_{i,\omega}) = T_{M\to q}\left(\prod_{s_j\in\mathcal{N}_i^{\mathcal{G}}} m_{j\to i}^{(M)}(s_i)\phi_i(s_i)\right)_\omega(z_{i,\omega})$;
  and (factor to source):

$$m_{a\to i,\omega}^{(q)}(z_{i,\omega}) \Leftarrow \sum_{\mathcal{N}_a^{\mathcal{C}}\setminus z_{i,\omega}} f_a(\mathcal{N}_a^{\mathcal{C}}) \prod_{z_{j,\omega'}\in\mathcal{N}_a^{\mathcal{C}}\setminus z_{i,\omega}} m_{j,\omega'\to a}^{(q)}(z_{j,\omega'}) \tag{11}$$

- Belief update:

$$\hat{s}_i = \arg\max_{s_i} \left[m_{\mathcal{C}\to i}^{(M)}(s_i)\right] \left[\prod_{s_j\in\mathcal{N}_i^{\mathcal{G}}} m_{j\to i}^{(M)}(s_i)\phi_i(s_i)\right] \tag{12}$$

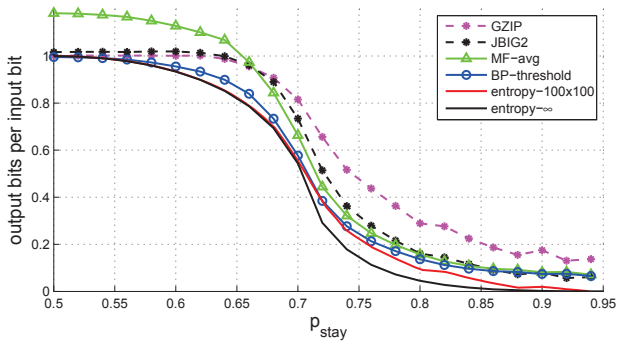


Fig. 3: Rate performance compared among three compression algorithms on 100×100 synthetic bi-level images (see Section IV-A). Entropy bounds for the infinite-extent Ising model and the 100×100 Ising model are also plotted for reference.

- `GZIP`: This Lempel-Ziv class universal compressor operates on the raw image bitmap as a 1D scanline stream. The output length is the compressed file size, less the compressed file size of a zero-length file to account for headers.

Additionally, in the case of `MF`, for each minimal rate LDPC code found, we can identify its BP decoding threshold (for the BEC), ϵ^{BP}. This threshold rate serves as a more accurate proxy for the "utilizable" rate of that code than its total rate r, in that the gap between r and ϵ^{BP} is not primarily an architectural loss, but that associated with poor code selection. Better code selection such as through degree-distribution optimization or e.g. [15] can be expected to close the gap.

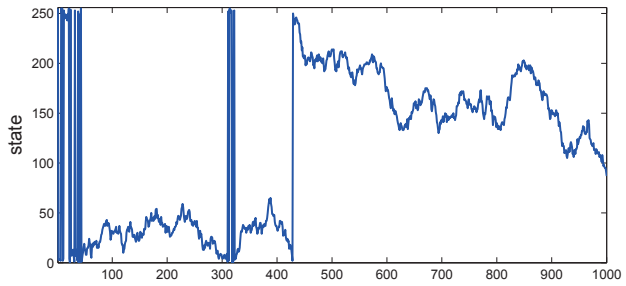


Fig. 4: A sample path of the Markov source of Section IV-B with entropy rate 0.5.

Referring to Fig. 3, in the low-rate regime (p_{stay} nearer to 1), the performance of `MF` beats `GZIP`, rivals `JBIG2`, and is very close to the best possible (entropy bound). In the high-rate regime, excess rate of `MF` is attributed to the lower BP decoding threshold of the particular chosen LDPC code. Overall it strongly suggests that with a properly chosen LDPC code (outside the scope of our work), `MF` performs well over all regimes for this source.

B. Large alphabet sources

Homogenous Markov chains over a cyclic group $\mathbf{Z}_M$ are among the simplest large-alphabet sources with memory. Here we model a family of Wiener-process-like, smoothly transitioning Markov chains over $\mathbf{Z}_M$ by defining transition probabilities as discretized Gaussian densities of certain variances centered around the current state, i.e the self transition has the

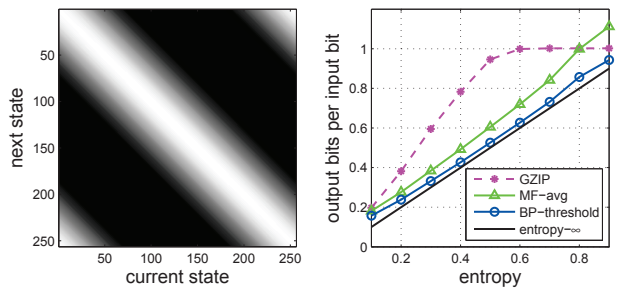


Fig. 5: On the left is depicted the transition matrix of an 8-bit Markov chain (Section IV-B) with entropy rate 0.9 (brighter value is higher probability; the bright band would be narrower at lower entropies). On the right is the rate performance comparison of compressing the Markov chains by `MF` and `GZIP`, compared to source entropy.

highest probability, followed by "nearer" neighbors:

$$\mathbb{P}\{s_{t+1} = m_{t+1} | s_t = m_t\} = \mathbb{P}\left\{|m_{t+1} - m_t| - \frac{1}{2} < Z < |m_{t+1} - m_t| + \frac{1}{2}\right\}$$

Here, $m_{t+1} - m_t$ is understood cyclically, and $Z \sim \mathcal{N}(0, \Sigma)$ for some Σ.

1) Results: We generate Markov chains of this type over $\mathbf{Z}_{256}$ with length 1000, beginning at steady state (Fig. 4). They are compressed by the system in Section III, using the Gray-coding representational map. A range of entropy rates are targeted by adjusting the variance parameter Σ for the Gaussian densities that underlie the transition probabilities. There are 20 trials for each entropy rate. We compare `MF` with `GZIP`, with the same compressor setup as in Section IV-A; here, `GZIP` compresses the $\mathbf{Z}_{256}$ symbols as a stream of bytes in their canonical representation.

We see in Fig. 5 that the `MF` performance is very close to the entropy lower bound, especially when accounting for the BP threshold rates rather than nominal rates. `GZIP` performance is poor, and does not improve substantially even when sample length is increased to 100,000.

This is a positive result in that one may expect some loss due to the non-bijective message translation procedure in `MF`, but coding for this large alphabet source is remarkably efficient. We conjecture that the symbol-level macro-structure, being preserved by the source subgraph, is repeatedly injected into the inference by the iterative nature of decoding, so that the message translation loss is not significant over multiple iterations. Although the existence of pathological sources cannot be ruled out without further investigation, it at least appears that this architecture retains the macro-structure of the source better than would independent coding of bit planes, which is another commonly used approach for large alphabet sources.

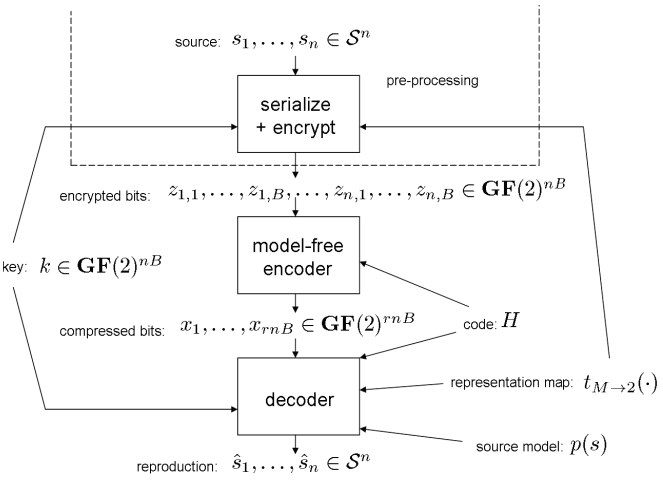


Fig. 6: Compressing an encrypted source with a model-free coding system. The already encrypted bits are presented to the encoder to compress, which without the secret key, is impossible to do with a traditional encoder.

V. An Enabled Application: Compressing an Encrypted Source

We now describe an encryption system that benefits from the model-free coding framework. Refer to Fig. 6 for the system diagram.

A. Encrypted encoding

Sometimes the owner of data wishes to retain total secrecy, and so presents to the compressor a version of the data that has already been encrypted by, e.g., adding a secret key to its bit-stream representation. In the language of Section III, the encoder receives $t_{M\to 2}(s) \oplus k$ where $k = \{k_1, ..., k_{nB}\}$ is nB bits of key from a one-time pad to go along the nB bits representing s. Normally, this stream is totally uncompressible without the encoder also having the key. However, since our encoder is model-free, we can simply view $\oplus k$ as part of the representational map and apply coding to $t_{M\to 2}(s) \oplus k \equiv t'_{M\to 2}(s)$.

B. Encrypted decoding

Note that in $\mathcal{U}$, the code subgraph is entirely encrypted since $z = t_{M\to 2}(s) \oplus k$ is encrypted and $x = Hz$ is encrypted. Note further that the source subgraph is entirely unencrypted since s, the object of decoding, is unencrypted. Therefore the encryption boundary coincides with the representation boundary (Fig. 1c). Then apply generalized decoding of Section III-C, using $t'_{M\to 2}$ and the corresponding $T'_{M\to 2}$ and $T'_{2\to M}$ pair. In practice this means during each iteration, $m^{(M)}_{\mathcal{C}\to i}(s_i)$ is obtained by message decryption followed by conventional translation $T_{2\to M}$, and $m^{(q)}_{\mathcal{G}\to i,\omega}(z_{i,\omega})$ by conventional translation $T_{M\to 2}$ followed by message encryption, with the rest of the decoding algorithm unchanged. With the key, message encryption and decryption simply amount to permuting the labels of symbols in messages.

VI. Discussion

While it seems fortuitous that combining a graphical source model and a factor graph for an LDPC code could directly result in a practical decoder for compression systems, this is in fact rooted firmly in the information theory of source coding. In that theory, the source $p(s)$ plays a role in the generation of a random codebook of typical codewords. The compressed output is solely concerned with conveying the index into this codebook, while the codebook itself is shared between encoder and decoder, and does not need to be conveyed. This shared knowledge of the codebook, which is exactly to be interpreted as the source model, is the root of all compression gain, and the basis of model-free coding. We may now interpret $p(s)$ not as the source model itself, but as the kernel from which the codebook can be independently reconstructed at the decoder.

In a model-free encoder, only the size of the codebook (hence, rate r) is needed to create an index from hashing the data itself. The actual content of the codebook is irrelevant. In the decoding process, inference over the source subgraph $\mathcal{G}$ plays the role of dynamically re-generating from $p(s)$ a portion of the codebook near the doping initialization, while inference over the code subgraph $\mathcal{C}$ plays the role of hash verification. Rather than an exponential-time enumeration of the entire codebook, the decoding process is guided by the low-complexity inversion of the hash function due to the sparsity of $\mathcal{C}$. This class of compression systems with model-free encoding continue to intrigue us because they operationalize source coding theory in this direct yet feasible way.

VII. Conclusion

We presented a model-free coding framework and several variations of system constructions that enable the practical compression of structured data, large-alphabet data, and encrypted data, with promising performance characteristics. The use of inference to support source modeling and coding/decoding in a compatible framework opens up many possibilities of innovation in source models and new avenues of effectively processing and communicating data.

References

[1] A. Aaron, S. D. Rane, E. Setton, and B. Girod, "Transform-domain Wyner-Ziv codec for video," in *Proceedings of SPIE*, vol. 5308, 2004, pp. 520–528.

[2] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.

[3] D. Schonberg, S. Draper, and K. Ramchandran, "On compression of encrypted images," in *Proceedings of the International Conference on Image Processing*, 2006, pp. 269–272.

[4] S. Jalali, S. Verdu, and T. Weissman, "A universal scheme for Wyner-Ziv coding of discrete sources," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1737–1750, Apr. 2010.

[5] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[6] L. He and L. Carin, "Exploiting structure in wavelet-based bayesian compressive sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 9, pp. 3488–3497, 2009.

[7] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations Trends in Machine Learning*, vol. 1, pp. 1–305, 2008.

[8] T. Richardson and R. Urbanke, *Modern Coding Theory*, Mar. 2008.

[9] M. Wainwright, E. Maneva, and E. Martinian, "Lossy source compression using low-density generator matrix codes: Analysis and algorithms," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 1351–1368, Mar. 2010.

[10] V. Chandar, D. Shah, and G. W. Wornell, "A simple message-passing algorithm for compressed sensing," in *2010 IEEE International Symposium on Information Theory Proceedings (ISIT)*, Jun. 2010, pp. 1968–1972.

[11] M. G. Reyes, "Cutset based processing and compression of markov random fields," Ph.D. dissertation, The University of Michigan, 2011.

[12] G. Caire, S. Shamai, and S. Verdú, "Noiseless data compression with low-density parity-check codes," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 66, pp. 263–284, 2004.

[13] B. Potetz and T. Lee, "Efficient belief propagation for higher order cliques using linear constraint nodes," *Computer Science Department*, May 2008.

[14] D. Cimasoni, "A generalized Kac-Ward formula," *J. Stat. Mech.*, 2010.

[15] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.