ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413447

# PREDICTIVE CODING FOR LOSSLESS DATASET COMPRESSION

*Madeleine Barowsky* *Alexander Mariona* *Flavio P. Calmon*

Harvard University, mbarowsky@g.harvard.edu, agmariona@college.harvard.edu, flavio@seas.harvard.edu

## ABSTRACT

Lossless compression of datasets is a problem of significant theoretical and practical interest. It appears naturally in the task of storing, sending, or archiving large collections of information for scientific research. We can greatly improve encoding bitrate if we allow the compression of the original dataset to decompress to a permutation of the data. We prove the equivalence of dataset compression to compressing a permutation-invariant structure of the data and implement such a scheme via predictive coding. We benchmark our compression procedure against state-of-the-art compression utilities on the popular machine-learning datasets MNIST and CIFAR-10 and outperform for multiple parameter sets.

***Index Terms*—** Compression, source coding, dataset, predictive coding

## 1. INTRODUCTION

Source coding is a central task in signal processing and information theory whose goal is to create a compressed binary representation of a sequence of symbols [1]. Whether the compression is lossy or lossless, the original sequence's *order* is always reconstructed exactly. Common lossless compression schemes like LZ78 [2] or `bz2` [3] operate on data streams and are thus order-preserving.

Existing source coding algorithms are suboptimal for the problem of losslessly compressing large datasets where each sample's relative placement doesn't matter. This use case arises in many applications: large datasets used for fitting machine learning models, a photo collection, research archive, file directory, or tabular statistical databases. In these cases, the position of each item in the dataset is irrelevant. Consequently, the order-preservation constraint on the compression algorithm can be removed—potentially dramatically improving the achievable compression rate.

We define *dataset compression* as the problem of creating source codes that allow datasets to be reconstructed up to a permutation of their entries. The challenge of dataset compression is timely given the ever-increasing data requirements of complex machine learning models; communicating and storing massive training and benchmark data is bandwidth- and memory-intensive. In this paper, we introduce a theoretical framework and practical algorithm for *lossless dataset compression*. We make four primary contributions:

1. We present a new information-theoretic formulation for the dataset compression problem as an extension of the standard lossless source coding framework.
2. We formulate and prove a theorem which reduces the task of compressing a dataset to the task of compressing a (deterministic) structure on the data which is invariant to ordering. This points to a general strategy of finding computationally and theoretically tractable structures that fit the dataset well and then compressing such structures.
3. We propose a new predictive coding procedure for dataset compression. This procedure reorders the dataset to minimize differences between elements and fits a predictive model on features from adjacent elements.
4. We build a robust software package implementing predictive coding for dataset compression. Our code offers multiple parameters for encoding and will be made available upon publication. We apply the scheme to two well-known ML datasets, MNIST and CIFAR-10, and outperform competitive compressors such as `xz` and JPEG-LS.

We summarize next related literature for mathematical descriptions of dataset compression, predictive coding, and compression of permutation-invariant structures on data.

**Dataset compression.** Dataset compression can be cast in terms of communicating multisets. In 1986 Lempel showed that a multiset code contains no more codewords than a sequence code [4]. He further conjectured that all multiset codes satisfy Kraft's Inequality, i.e. they have prefix-free equivalents. Later work by Varshney and Goyal [5–7] lower bounded the entropy of a multiset by decomposing it into the entropy of a deterministic ordering of the elements and the entropy of a uniform distribution over all possible permutations [5, Eq. (2)]. They noted that a multiset drawn from a finite alphabet can be fully described by its type, the empirical distribution of elements. Hence a multiset can be communicated with sub-linear rate by enumerating all types of the given size. However, this does not consider the implementation cost of enumerating and encoding types (which is non-negligible in many applications such as images). In contrast, we demonstrate precisely how to operationalize dataset compression. Our approach is both information-theoretically sound and amenable towards algorithmic implementation.

There is an extensive literature on universal source cod-

M. Barowsky was supported by a GFSD Fellowship. A. Mariona was supported by a PRISE Fellowship. This material is based upon work supported by the National Science Foundation under Grant Nos. 1845852 and 1900750.

ing in information theory. Universal source coding of i.i.d. sources can be achieved by encoding parameters of the estimated source distribution (e.g. a type) and then selecting a code for that distribution [8]. In [9–11] the authors introduce bounds related compressing the *profile* of a sequence (the prevalence of multiplicities of symbols)—a quantity that can also be interpreted as the histogram of a dataset. As before, their work has a theoretical focus, requiring each symbol in the sequence to be described in full.

In a similar vein, [12, 13] describe compression schemes for when the alphabet size is much larger than the sequence length. Although compressing sequences, they separately encode the values and ordering of samples in the sequence. Note that coding the values of a sequence is equivalent to coding a dataset. These works also consider sequences where data is generated conditionally from other samples in the sequence, similar to our predictive coding procedure (see Sec. 3).

**Predictive coding.** Predictive coding is a compression technique which factors out repetition and similarity inherent in data by means of a predictive model. If the model fits the data well, its prediction is often correct. Even when the model is wrong, it should be close to the correct value. Thus, the sequence of errors between predicted and true values can be approximated as an i.i.d. stream concentrated around zero. This stream can then be encoded using techniques for i.i.d. sources [14]. Compressing only the errors requires fewer bits than compressing the original data. Predictive coding is a consequence of the Minimum Description Length principle [15].

For images, there is often a strong relation between nearby pixels. This relation is leveraged in both image coding (e.g., PNG [16] and WebP [17]) for intra-frame prediction, and modern video codecs (e.g., H.264/H.265 [18, 19], VVC [20], VP9 [21], AV1 [22]) for inter-frame prediction. The image codec JPEG-LS uses predictive coding for lossless image compression [23] and is a direct inspiration for our approach. More recently, JPEG XL also supports this mode [24].

A novel predictive coding method for single high-resolution images was introduced in [25]. To our knowledge, [26] is the only existing use of predictive models for dataset compression where predictors consider features from similar elements. However, their goal is to encode a queryable database and must preserve information in the code used for indexing.

**Compressing data structures.** We formalize the motivation for compressing a permutation-invariant structure on a dataset in Section 2. Here we briefly review literature in data compression that also builds structures based on element similarity prior to compressing. In [26–28], the authors build a tree from the data and traverse it to obtain a reordering that leads to efficient lossless compression. However, [27,28] do not use predictive coding to leverage similarity between samples in a dataset. Prior work on tabular data compression has showed that carefully reordering columns before coding improves up to 20% over using initial order [29]. [30] approaches the problem of graph compressibility and presents two codes which perform well on particular Web graphs. They also find that breadth-first search (BFS)/depth-first search (DFS) orders do better than random. We replicate this in Section 4.

## 2. PROBLEM FORMULATION

We next formalize the dataset compression setting studied in this paper. This formulation is a direct extension of the usual information-theoretic lossless source coding framework (see [31]). We denote by $\{0,1\}^*$ the set of all binary strings, $[n] \triangleq \{1, 2, \ldots, n\}$, and define $\mathcal{S}_n$ as the set of permutation of $[n]$. The type of a sequence $X^n$ describes the relative proportion of each symbol in the sequence and induces a probability distribution. We notate type as $T_{X^n} : \mathcal{X} \to \Delta^n$ where $\Delta^n$ is the $n-$dimensional probability simplex. The length of a string (in bits) is given by the function $l : \{0,1\}^* \to \mathbb{N}$.

A *lossless source code* (l.s.c.) for a random variable $X$ with distribution $P_X$ over alphabet $\mathcal{X}$ is given by functions $(f, g)$ where $f : \mathcal{X} \to \{0,1\}^*$, $g : \{0,1\}^* \to \mathcal{X}$ such that $g(f(x)) = x$ for all $x \in \mathcal{X}$. The shortest average length of a prefix-free [31, Chap. 5] source code for $X$ is denoted by

$$M^*(P_X) \triangleq \min\left\{\mathbb{E}\left[l(f(X))\right] \mid (f, g) \text{ is a prefix-free l.s.c.}\right\}. \tag{1}$$

Now consider a *dataset* $X^n = (X_1, \ldots, X_n)$ drawn i.i.d. from a distribution $P_X$ over a finite alphabet $\mathcal{X}$. For example, $X^n$ may be a sequence of $n$ greyscale images or rows in a tabluar dataset. A pair of functions $f : \mathcal{X}^n \to \{0,1\}^*$ and $g : \{0,1\}^* \to \mathcal{X}^n$ is called a *lossless dataset source code* (l.d.s.c.) if for each $x^n \in \mathcal{X}^n$ there is $\pi \in \mathcal{S}_n$ (dependent on $x^n$) such that $g(f(x^n)) = \pi \circ x^n$. A lossless dataset source code reconstructs a dataset $X^n$ up to a permutation of its samples. In contrast, a lossless source code would require exact reconstruction of the original sequence.

Analogously to (1), we denote

$$M^*_n(P_X) \triangleq \min\left\{\mathbb{E}\left[l(f(X^n))\right] \mid (f, g) \text{ is a prefix-free l.d.s.c.}\right\}. \tag{2}$$

Clearly, $M^*_n(P_X) \leq M^*(P_{X^n})$. The following theorem (proved in the extended version) gives a characterization of minimum average length of a prefix-free lossless dataset compression code and has an important practical implication: compressing a dataset is equivalent to compressing a permutation-invariant structure $S$ on that dataset (e.g. type or neighbor graph).

**Theorem 1.** *Let $\widetilde{X}^n = \pi \circ X^n$, where $\pi$ is drawn uniformly and at random from $\mathcal{S}_n$. Moreover, let $S$ be such that*

*(i) $S \to X^n \to \widetilde{X}^n$ and $X^n \to S \to \widetilde{X}^n$;*

*(ii) $H(S|X^n) = H(S|\widetilde{X}^n) = 0$.*

*Then $M^*(P_S) = M^*_n(P_X)$,*

$$H(S) \leq M^*_n(P_X) < H(S) + 1, \tag{3}$$

*and*

$$H(S) = I(X^n; \widetilde{X}^n) \leq \min\{|\mathcal{X}| \log_2(n+1), n \log_2 |\mathcal{X}|\} \tag{4}$$

*Proof.* Available in the extended version of our paper. □

This theorem's operational consequence of finding a "good" structure $S$ for a dataset leads directly into discussion of our predictive dataset compression pipeline. As mentioned in [5, 7] and evidenced in (4), dataset compression has a growth rate that is sublinear in sequence length.

## 3. A METHOD FOR DATASET COMPRESSION

Here we describe a methodology for lossless dataset compression. The procedure extends predictive coding techniques found in the literature [14, 32] and used, for example, in JPEG-LS [23]. Our approach has three core steps: (i) reordering dataset elements, (ii) predictive coding, (iii) entropy coding. We are motivated by Theorem 1 to find a computationally tractable structure $S$ based on reordering the dataset.

Unlike JPEG-LS, which primarily uses fixed coefficients for predictive coding, we train a predictive model over the entire data set. Our model includes input features not only from within a sample, but also from related samples. The approach outlined below applies broadly to either tabular or image datasets. For tabular data, reordering by similarity can also use Euclidean distance, context strings to train the model may be constructed in the same way, and predictive model(s) may be used to estimate the features in a row. It will be clear when we describe image-only implementation details.

**Dataset reordering.** We wish to reorder the dataset entries (i.e. produce $S$ in Theorem 1) to maximize the accuracy of predictive coding. Predictive coding is most effective when the predictor error string is concentrated around zero, that is, when the model performs well. Our experiments confirm that ordering to reduce the Euclidean distance between successive dataset entries results in better compression than random ordering. Towards this goal, we develop a method named *kNN-MST ordering*, described below.

First, note that reordering dataset elements to maximize similarity between them is equivalent to finding a minimum Hamiltonian path [33] on the complete nearest-neighbors graph. Computing a globally optimal traversal of a neighbor graph is the Metric Traveling Salesman Problem. Although MetricTSP is NP-hard [34], there exist good approximations. Simply traversing a minimum spanning tree (MST) of the complete neighbor graph gives a solution which is within twice the optimal cost [34, Thm 2].

Of course, computing the MST on the complete neighbor graph of a large dataset may be computationally infeasible. We make the following elementary graph theory observation:

**Lemma 1.** *Let $G = (V, E)$. For all $1 \leq k \leq |V|$, if kNN$(G)$ is connected, then a MST of kNN$(G)$ is a MST of $G$.*

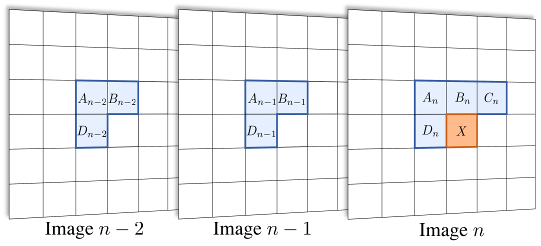

**Fig. 1**. $X$ is the current pixel under prediction.

This lemma allows us to find $k$ such that the kNN graph of $G$ is fully connected and then compute a reordering of the dataset by traversing the MST. We note the result of [35], which found the minimum $k$ needed for kNN connectivity in statistical simulations over i.i.d. data was $O(\log n)$. In practice we set our initial $k = \ln n$ and increase only if the resulting kNN graph is disconnected.

**Context strings.** Our core idea in predictive coding for dataset compression is to use features from neighboring samples and within a sample produced by the kNN-MST ordering of the dataset. In contrast, JPEG-LS operates on single images and only predicts on features within an image. To train the predictor, we extract one *context string* per pixel from each image in the dataset. A context string is an array of features related to the feature under prediction, e.g. a pixel in an image or cell in a table. Certain pixels within the current image can only be decoded after others have been decoded—any pixels to the right of a pixel cannot be included in its context string. However, we can include the corresponding pixel in previous images in the ordered dataset, which improves prediction when there are multiple similar images. We refer to the pattern of relative pixels used in prediction as a *context strategy*. We allow separate context strategies for previous and current images. Each context string includes a fixed number of previous images. We show an example with strategy DAB/DABC and two previous images in Figure 1.

**Color images.** Our compression procedure has two modes for handling RGB data: *single mode* and *triple mode*. The latter trains one predictor for each color channel. In the former, there is a single predictor which outputs a triple of channel values. Experimentally, we find triple mode performs somewhat worse than single mode. This may be because there is mutual information between color channels, such as brightness, which is captured by the single mode predictor.

**Predictor model classes.** The accuracy of the predictor function is crucial to minimizing entropy in the error string and improving the compression rate. We use a two-pass approach for prediction. First, we generate the context string for each pixel in the dataset. We then use this information to train the predictive model over all elements. We train until it overfits because we do not need to consider how accuracy generalizes to samples beyond the dataset. After the model has been fit, we apply it to each pixel and record the error or 0 if the predicted value equals the true value.

**Algorithm 1** Predictive Dataset Compression

```
Input: a dataset $\mathcal{D} = \{x_i\}_{i=1}^n$
$(x_{i_j})_{j=1}^n = \text{kNN-MST-REORDER}(x^n)$
$\{\{\text{context}_a, a\}_{a \in x_t}\}_{t=1}^n = \text{EXTRACTCONTEXT}(\mathcal{D})$
$h = \text{TRAINPREDICTIVEMODEL}(\{\{\text{context}_a, a\}_{a \in x_t}\}_{t=1}^n)$
error_string = [ ]
for $x_t$ in $\mathcal{D}$ do
    for $a$ in $x_t$ do
        error_string += $h(\text{context}_a) - a$
    end for
end for
HUFFMANENCODE(metadata, f.params, error_string)
```

## 4. EXPERIMENTS

This section describes the implementation and application of the lossless dataset compression procedure from Section 3 to two popular image datasets. This procedure is benchmarked against standard lossless compression schemes. We focus on final compressed byte size with a motivating use case in which the party holding uncompressed data will expend computational power to reduce file size for storage or communication.

We evaluate our pipeline on the MNIST [36] and CIFAR-10 [37] datasets, which are 60000- and 50000-sample, well-structured image datasets heavily used in signal processing and machine learning research. MNIST consists of centered $28 \times 28$ grayscale images of handwritten digits from 0 to 9, while CIFAR-10 is composed of $32 \times 32$ pictures which partition into ten subsets of common objects.

We vary predictor class, previous context strategy, and number of previous images. We use current context strategy DAB. For CIFAR-10, we test both single and triple mode for predicting RGB pixel values (though triple mode is not available for logistic regression). In all cases, we used a traditional Huffman encoder to compress the error string. Our predictive coding compression algorithm is presented in Algorithm 1.

The final metadata and bytestream consist of model parameters, sufficient starting pixels to bootstrap context strings, and the error string. We pass this through Huffman coding, a common step in image codecs [16, 17]. Then we apply `xz` [38], a lossless compression utility we benchmark against, in order to pare down additional redundancy. Even without this final step, our approach beats many baselines.

**Results.** We considered a number of existing compression schemes for comparison and ran each on MNIST and CIFAR-10. To encode JPEG-LS images, we modified the Python package CharPyLS [39] to handle RGB data. Encoding images in PNG, a lossless codec, significantly increased dataset size; therefore results are not reported.

Because `xz` performed best across datasets, we use it as the final step in our compression pipeline. The **post-XZ (MB)** column in Tables 2–3 indicates size after applying `xz`.

**Analysis.** The most analogous comparison for our dataset predictive coding scheme is to JPEG-LS because it also uses predictive coding but operates per-image. Our best performing dataset compression of MNIST is $18\%$ smaller than a JPEG-LS encoding of each image and $20\%$ smaller once passing both through `xz`. For CIFAR-10, our strongest compression is $5\%$ smaller than JPEG-LS and $4.5\%$ after compressing both with `xz`. Thus, our improvements can be attributed to the dataset-specific approach. All our results shown for CIFAR-10 beat all compression benchmarks. The best parameters improve $10\%$ on `xz`, $26\%$ on `gz`, and $15\%$ for `bz2`.

The compression utilities we benchmark against have been tightly optimized at the bit-level over years of iteration in the open source community. We do not over-engineer our code to directly compete with their engineering, but rather, put forth a new high-level compression methodology, note its successes despite straightforward bitstream formatting, and campaign for further improvements.

| Compressor | MNIST Size (MB) | CIFAR-10 Size (MB) |
|---|---|---|
| Uncompressed | 47.04 | 155.18 |
| gz | 9.66 | 141.7 |
| bz2 | 8.40 | 122.6 |
| xz | 8.11 | 116.4 |
| JPEG-LS | 16.58 | 118.66 |
| JPEG-LS + xz | 13.41 | 109.39 |

**Table 1**. Compression Benchmarks

| Predictor | Ordering | Context strategy | Final size (MB) | post-XZ (MB) |
|---|---|---|---|---|
| Linear | Random | DAB | 21.43 | 18.66 |
| Linear | 11-nn MST | DABX | 18.13 | 15.80 |
| Logistic | 11-nn MST | DABC | 13.56 | 10.72 |

**Table 2**. Compression on MNIST (2 context images).

| Predictor | Ordering | Context strategy | Final size (MB) | post-XZ (MB) |
|---|---|---|---|---|
| Linear (single) | Random | DAB | 112.41 | 104.56 |
| | 10-nn MST | DABX | 112.47 | 104.39 |
| Linear (triple) | 10-nn MST | DAB | 112.44 | 110.55 |

**Table 3**. Compression on CIFAR-10 (2 context images).

## 5. CONCLUSION

Efficient lossless dataset compression algorithms can significantly reduce storage and transmission bandwidth requirements in several applications of practical interest, including everything from disk backups and scientific results, to folders with vacation photos and cat memes. We have introduced a theoretical formulation of the problem based on permutation-invariant structures, which indicates new avenues for code design. Our work also implements a predictive coding scheme for image dataset compression, benchmarks it on MNIST and CIFAR-10 for a variety of parameters, and outperforms existing lossless stream coding baselines such as `xz` and JPEG-LS.

## 6. REFERENCES

[1] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] Jacob Ziv and Abraham Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. on Info. Theory*, vol. 24, pp. 530 – 536, 10 1978.

[3] Julian Seward, "bzip2," `https://sourceware.org/bzip2/`, Accessed: 2020-10-21.

[4] A. Lempel, "On multiset decipherable codes (corresp.)," *IEEE Trans on Info. Theory*, vol. 32, no. 5, pp. 714–716, 1986.

[5] L. R. Varshney and V. K. Goyal, "Toward a source coding theory for sets," in *DCC Proceedings*, 2006, pp. 13–22.

[6] L. R. Varshney and V. K. Goyal, "On universal coding of unordered data," in *2007 Information Theory and Applications Workshop*, 2007, pp. 183–187.

[7] L. R Varshney and V. K. Goyal, "Ordered and Disordered Source Coding," in *Proc. UCSD Workshop Inform. Theory Its Applications*, La Jolla, CA, 02 2006.

[8] O. Kosut and L. Sankar, "Asymptotics and non-asymptotics for universal fixed-to-variable source coding," *IEEE Trans. on Info. Theory*, vol. 63, no. 6, pp. 3757–3772, 2017.

[9] Alon Orlitsky, Narayana P. Santhanam, Krishnamurthy Viswanathan, and Junan Zhang, "On modeling profiles instead of values," in *UAI '04*, Arlington, Virginia, USA, 2004, UAI '04, p. 426–435, AUAI Press.

[10] Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang, "Universal compression of memoryless sources over unknown alphabets," *IEEE Trans. on Info. Theory*, vol. 50, no. 7, pp. 1469–1481, jul 2004.

[11] A. Orlitsky and N. P. Santhanam, "Speaking of infinity [i.i.d. strings]," *IEEE Trans. on Info. Theory*, vol. 50, no. 10, pp. 2215–2230, 2004.

[12] Xiao Yang, *Compression and Predictive Distributions for Large Alphabets*, Ph.D. thesis, Yale University, New Haven, May 2015.

[13] X. Yang and A. R. Barron, "Compression and Predictive Distributions for Large Alphabet i.i.d and Markov Models," in *ISIT '14*, 2014, pp. 2504–2508.

[14] Hisashi Kobayashi and Lalit R. Bahl, "Image data compression by predictive coding i: Prediction algorithms," *IBM J. Res. Dev.*, vol. 18, no. 2, pp. 164–171, 1974.

[15] Jorma Rissanen, *Minimum Description Length Principle*, pp. 666–668, Springer US, Boston, MA, 2010.

[16] "Compression and Filtering (PNG: The Definitive Guide)," Accessed: 2020-10-13.

[17] "WebP Compression Techniques," `developers.google.com/speed/webp/docs/compression`.

[18] "Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264, ISO/IEC 14496-10 AVC)," JVT of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.

[19] Gary J. Sullivan, Jens Rainer Ohm, Woo Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE TCSVT*, vol. 22, 2012.

[20] S. De-Luxán-Hernández, V. George, J. Ma, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, "An Intra Subpartition Coding Mode for VVC," in *ICIP '19*, 2019, pp. 1203–1207.

[21] "VP9 Video Codec Summary," `webmproject.org/vp9/`, 04 2017, Accessed: 2020-10-13.

[22] "AV1 Specification Syntax Structures," `aomedia.org/av1/specification/syntax/`, 01 2019, Accessed: 2020-10-13.

[23] Marcelo J Weinberger, Gadiel Seroussi, and Guillermo Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE Trans. on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[24] A. Rhatushnyak, J. Wassenberg, J. Sneyers, J. Alakuijala, L. Vandevenne, L. Versari, R. Obryk, Z. Szabadka, E. Kliuchnikov, I.-M. Comsa, K. Potempa, M. Bruse, M. Firsching, R. Khasanova, R. van Asseldonk, S. Boukortt, S. Gomez, and T. Fischbacher, "Committee Draft of JPEG XL Image Coding System," 2019.

[25] Sheng Cao, Chao-Yuan Wu, and Philipp Krähenbühl, "Lossless Image Compression through Super-Resolution," Tech. Rep.

[26] Lionel Gueguen and Mihai Datcu, "A similarity metric for retrieval of compressed objects: Application for mining satellite image time series," *TKDE '08*, vol. 20, no. 4, pp. 562–574, apr 2008.

[27] Christian Steinruecken, "Compressing Sets and Multisets of Sequences," *IEEE Trans. on Info. Theory*, vol. 61, no. 3, pp. 1485–1490, mar 2015.

[28] Yuriy A. Reznik, "Coding of sets of words," in *Data Compression Conference Proceedings*, 2011, pp. 43–52.

[29] Adam L. Buchsbaum, Glenn S. Fowler, and Raffaele Giancarlo, "Improving table compression with combinatorial optimization," *J. ACM*, vol. 50, no. 6, pp. 825–851, 11 2003.

[30] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan, *On Compressing Social Networks*, 2009.

[31] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, Wiley, 2 edition, 07 2006.

[32] LR Bahl and H Kobayashi, "Image data compression by predictive coding ii: Encoding algorithms," *IBM Journal of Research and Development*, vol. 18, no. 2, pp. 172–179, 1974.

[33] Eric W. Weisstein, "Hamiltonian path," `mathworld.wolfram.com/HamiltonianPath.html`, Accessed: 2020-10-21.

[34] Markus Bläser, *Metric TSP*, pp. 517–519, Springer US, Boston, MA, 2008.

[35] José María González-Barrios and Adolfo J. Quiroz, "A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree," *Stat. Probab. Lett.*, vol. 62, no. 1, pp. 23–34, March 2003.

[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[37] Alex Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[38] "XZ Utils," `tukaani.org/xz/`, Accessed: 2020-09-26.

[39] Pierre V. Villeneuve, "CharPyLS," `github.com/Who8MyLunch/CharPyLS`, 2018, Commit: 9a91590.