ARTICLE Communicated by Michael Hersche

Efficient Decoding of Compositional Structure in Holistic Representations

Denis Kleyko
denis.kleyko@ri.se
Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA 94720, U.S.A., and Intelligent Systems Laboratory, Research Institutes of Sweden, 16440 Kista, Sweden

Connor Bybee
bybee@berkeley.edu
Ping-Chen Huang
pingchen.huang@berkeley.edu
Christopher J. Kymn
cjkymn@berkeley.edu
Bruno A. Olshausen
baolshausen@berkeley.edu
Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA 94720, U.S.A.

E. Paxon Frady
e.paxon.frady@intel.com
Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA 95054, U.S.A.

Friedrich T. Sommer
fsommer@berkeley.edu
Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA 94720, U.S.A., and Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA 95054, U.S.A.

We investigate the task of retrieving information from compositional distributed representations formed by hyperdimensional computing/vector symbolic architectures and present novel techniques that achieve new information rate bounds. First, we provide an overview of the decoding techniques that can be used to approach the retrieval task. The techniques are categorized into four groups. We then evaluate the considered techniques in several settings that involve, for example, inclusion of external noise and storage elements with reduced precision. In particular, we find that the decoding techniques from the sparse coding and compressed sensing literature (rarely used for hyperdimensional computing/vector symbolic architectures) are also well suited for decoding information

Neural Computation **35, 1–28** (2023)
https://doi.org/10.1162/neco_a_01590

from the compositional distributed representations. Combining these decoding techniques with interference cancellation ideas from communications improves previously reported bounds (Hersche et al., 2021) of the information rate of the distributed representations from 1.20 to 1.40 bits per dimension for smaller codebooks and from 0.60 to 1.26 bits per dimension for larger codebooks.

1 Introduction

Hyperdimensional computing (Kanerva, 2009) also known as vector symbolic architectures (HD/VSA; Gayler, 2003) allows the formation of rich, compositional, distributed representations that can construct a plethora of data structures (Demidovskij, 2021; Kleyko, Davies et al., 2022). Although each individual field of a data structure is encoded in a fully distributed manner, it can be decoded (and manipulated) individually. This decoding property provides the remarkable transparency of HD/VSA, in stark contrast to the opacity of traditional neural networks (Shwartz-Ziv & Tishby, 2017). For example, decoding of distributed representations enables the tracing (or explanation) of individual results. It even led to the proposal of HD/VSA as a programming framework for distributed computing hardware (Kleyko, Davies et al., 2022). However, there are capacity limits on the size of data structures that can be decoded from fixed-sized distributed representations, and these limits depend on the decoding techniques used. Here, we characterize different techniques for decoding information from distributed representations formed by HD/VSA and provide empirical results on the information rate, including results for novel decoding techniques. The reported results are interesting from a theoretical perspective, as they exceed the capacity limits previously thought to hold for distributed representations (Frady et al., 2018; Hersche et al., 2021). From a practical perspective, many applications of HD/VSA hinge on efficiently decoding information stored in distributed representations, including, communications (Guirado et al., 2022; Hsu & Kim, 2020; Jakimovski et al., 2012; Kim, 2018; Kleyko et al., 2012) and distributed orchestration (Simpkin et al., 2019).

The problem of decoding information from distributed representations has similarities to information retrieval problems in other areas, such as in communications, reservoir computing, sparse coding, and compressed sensing. Here, we describe how techniques developed in these areas can be applied to HD/VSA. This study makes the following major contributions:

- A taxonomy of decoding techniques suitable for retrieval from representations formed by HD/VSA
- A comparison of 10 decoding techniques on a retrieval task
- A qualitative description of the trade-off between the information capacity of distributed representations and the amount of computation the decoding requires

- Improvements on the known bounds on information capacity for distributed representations of data structures (in bits per dimension) (Frady et al., 2018; Hersche et al., 2021).

The article is structured as follows. Section 2 introduces the approaches suitable for decoding from distributed representations. The empirical evaluation of the introduced decoding techniques is reported in section 3. The findings are discussed in section 4.

Readers interested in further background for HD/VSA are encouraged to read appendix A.1. To evaluate the considered decoding techniques, we consider the case of an n-dimensional vector, $\mathbf{y}$, that represents a sequence of symbols $\mathbf{s}$ of length v. The symbols are drawn randomly from an alphabet of size D. Symbols are represented by n-dimensional random bipolar vectors that are stored in the codebook $\mathbf{\Phi}$. The permutation and superposition operations are used to form $\mathbf{y}$ from representations of sequence symbols $\mathbf{\Phi}_{\mathbf{s}_i}$. We then use the decoding techniques to construct $\hat{\mathbf{s}}$, a reconstruction of $\mathbf{s}$ using $\mathbf{y}$ and the codebook $\mathbf{\Phi}$. Further details on the encoding scheme are in appendix A.2. We evaluate the quality of $\hat{\mathbf{s}}$ based on accuracy and information rate; further details are provided in appendix A.3.

2 Decoding Techniques

In this section, we survey decoding techniques for retrieving sequence symbols (denoted as $\hat{\mathbf{s}}$) from their compositional distributed representation $\mathbf{y}$ (see appendix A.2 for a detailed problem formulation). The decoding techniques can be taxonomized into two types, selective (section 2.1) and complete (section 2.2). In selective decoding, a query input selects a particular field in the data structure, which is then decoded individually. Conversely, in complete decoding, all fields of the data structure are decoded simultaneously. In the following sections, we introduce concrete decoding techniques pertaining to the two different types. The taxonomy of these decoding techniques and their relationships are summarized in Figure 1.

2.1 Techniques for Selective Decoding. In techniques for selective decoding, a query input selects a particular field of the data structure represented by a distributed representation (vector $\mathbf{y}$). The content of the selected data field $\hat{\mathbf{s}}_i$ is then decoded. In techniques for selective decoding, information about the field i of the query in the data structure is translated into a readout matrix (denoted as $\mathbf{W}^{\text{out}}(i) \in [D \times n]$), which is then used for decoding the data field. We adopt here the term *readout matrix* from the reservoir computing literature (Lukosevicius & Jaeger, 2009). In reservoir computing, many readout matrices can be specified for a single distributed representation depending on the task. In the framework of HD/VSA, the different readout matrices correspond to queries of different

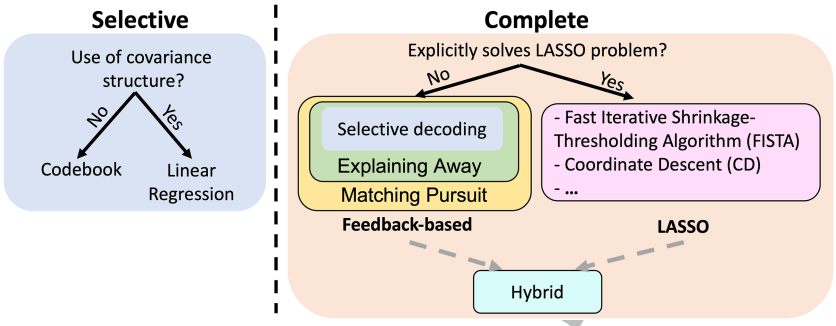

Figure 1: The taxonomy of decoding techniques surveyed and investigated. Techniques for selective decoding require two steps: (1) the transformation of the query vector to select a particular field and (2) a simple feedforward matrix-vector multiplication, followed by arg max (see section 2.1). In contrast, techniques for complete decoding (see section 2.2) do not retrieve a single field but rely on iterative procedures that are computationally more intensive. Ellipses indicate that there are additional techniques within a group that are not considered here.

fields in the data structure represented by the compositional distributed representation $\mathbf{y}$.

Once the readout matrices are known, a prediction for symbol in position i is computed as

$$\hat{\mathbf{s}}_i = \arg\max(\mathbf{W}^{\text{out}}(i)\mathbf{y}),$$

where $\arg\max(\cdot)$ returns the symbol with the highest dot product. Thus, selective techniques make their predictions by choosing a symbol corresponding to a row in $\mathbf{W}^{\text{out}}(i)$ that has the highest dot product with $\mathbf{y}$. Below, we consider two ways of forming $\mathbf{W}^{\text{out}}(i)$.

2.1.1 Codebook Decoding. Codebook decoding is the simplest technique for selectively decoding $\hat{\mathbf{s}}$ from $\mathbf{y}$. It corresponds to a matrix-vector multiplication between the query high-dimensional vector (also known as a hypervector) such as $\mathbf{y}$ and the algorithm's codebook, a matrix containing all hypervectors representing symbols. Here, the codebook $\mathbf{\Phi} \in \{-1, 1\}^{n \times D}$ stores atomic n-dimensional random independent and identically distributed (i.i.d.) bipolar hypervectors assigned to D unique atomic symbols of the alphabet. Because this technique of decoding is omnipresent in HD/VSA, we call it "Codebook decoding." Further, we use a permutation with a long cycle length, denoted as $\rho^i(\cdot)$ to bind a symbol with its position in the sequence (see details in appendix A.2). For example, to represent the ith position, the corresponding entry of the codebook $\mathbf{\Phi}_{\mathbf{s}_i}$ can be permuted

$v - i$ times. Then the readout matrix for the symbol at position i in the sequence is constructed as

$$\mathbf{W}^{\text{out}}(i) = \rho^{v-i}(\mathbf{\Phi}^\top). \tag{2.1}$$

Note that an equivalent way to decode is by applying an inverse permutation operation (denoted as ρ^{-1}) $v - i$ times to $\mathbf{y}$ without permuting $\mathbf{\Phi}$:

$$\hat{\mathbf{s}}_i = \arg\max\left(\rho^{v-i}(\mathbf{\Phi}^\top)\mathbf{y}\right) = \arg\max\left(\mathbf{\Phi}^\top \rho^{-(v-i)}(\mathbf{y})\right). \tag{2.2}$$

This formulation is more practical, as only a single vector (rather than a matrix) is being permuted. Finally, it is worth noting that the performance of Codebook decoding could be predicted analytically using the methodologies from Frady et al. (2018). This point was confirmed in our experiments that follow (see Figures 2 to 5), where the gray dashed lines depicting the analytical predictions closely match our experiments.

2.1.2 Linear Regression Decoding. Note that the hypervectors in $\mathbf{\Phi}$ are not perfectly orthogonal to each other. Therefore, Codebook decoding benefits from adjustments based on the expected covariance matrix of hypervectors interacting in $\mathbf{y}$ (denoted as $\tilde{\mathbf{C}}$):

$$\mathbf{W}^{\text{out}}(i) = \left(\tilde{\mathbf{C}}^{-1}\rho^{v-i}(\mathbf{\Phi})\right)^\top. \tag{2.3}$$

The readout matrix in equation 2.3 does not differ much from the one in equation 2.1 used for Codebook decoding, yet it substantially improves the information rate for small D (section 3.1). Nevertheless, this technique does not seem to be widely known within the HD/VSA literature; hence, this study aims at placing it on the map for those in the area. We will abbreviate this technique as "LR decoding."

For the considered transformation (see equation A.3 in section A.2), $\tilde{\mathbf{C}}$ can be calculated analytically (Frady et al., 2018) as

$$\begin{aligned}\tilde{\mathbf{C}} = &\sum_{i=1}^{v} \rho^i(\mathbf{\Phi})\left(\frac{1}{D}\mathbf{I}\right)\rho^{-i}(\mathbf{\Phi})^\top + \\ &+ \sum_{i=1}^{v}\sum_{j=1;\, j\neq i}^{v} \rho^i(\mathbf{\Phi})\left(\frac{1}{D^2}\mathbf{J}\right)\rho^{-j}(\mathbf{\Phi})^\top,\end{aligned} \tag{2.4}$$

where $\mathbf{I} \in [D \times D]$ is the identity matrix while $\mathbf{J} \in [D \times D]$ is the unit matrix. Note that compared to Codebook decoding, LR decoding incurs additional computational costs for obtaining $\tilde{\mathbf{C}}^{-1}$ and computing $\mathbf{W}^{\text{out}}(i)$ that scale with n, D, and, v. In general, however, the covariance-based readout

matrix cannot be computed analytically. Therefore, the standard practice in the randomized neural networks literature (Kleyko et al., 2021; Lukosevicius & Jaeger, 2009; Scardapane & Wang, 2017) is to collect some training data and use minimum mean squared error to estimate the optimal values of the readout matrix by solving a linear regression (LR) problem.

2.2 Techniques for Complete Decoding. Since hypervectors in $\mathbf{\Phi}$ and their permuted versions are not completely orthogonal, summing them in $\mathbf{y}$ produces cross-talk noise that degrades the result of the selective decoding. There are techniques for attempting the complete decoding of the data structure, for example, by first selectively decoding all fields of the data structure and then using the decoding results to remove cross-talk noise introduced by other fields and repeating the selective decoding. We overview three kinds of techniques: feedback based, least absolute shrinkage and selection operator (LASSO), and hybrid (combining elements of both).

Prior to introducing the techniques for complete decoding, we can make an interesting observation by reconsidering the transformation used to form a sequence's compositional hypervector $\mathbf{y}$ (see equation A.3). Similar to readout matrices for Codebook decoding in equation 2.1, we can make v permuted versions of the codebook $\mathbf{\Phi}$ and concatenate them horizontally into one large codebook (denoted as $\mathbf{A} \in [n \times vD]$) as

$$\mathbf{A} = [\rho^{v-1}(\mathbf{\Phi}), \rho^{v-2}(\mathbf{\Phi}), \ldots, \rho^{1}(\mathbf{\Phi}), \rho^{0}(\mathbf{\Phi})]. \tag{2.5}$$

Next, we can form a vector $\mathbf{x} \in [vD \times 1]$ that will contain the concatenation of D-dimensional one-hot encodings of symbols in the original sequence $\mathbf{s}$ (denoted as $\mathbf{o}_{\mathbf{s}_i}$):

$$\mathbf{x} = [\mathbf{o}_{\mathbf{s}_1}^\top, \mathbf{o}_{\mathbf{s}_2}^\top, \ldots, \mathbf{o}_{\mathbf{s}_v}^\top]^\top. \tag{2.6}$$

Note that multiplication of $\mathbf{A}$ by $\mathbf{x}$ is now equivalent to equation A.3:

$$\mathbf{A}\mathbf{x} = \mathbf{y} = \sum_{i=1}^{v} \rho^{v-i}(\mathbf{\Phi}_{\mathbf{s}_i}). \tag{2.7}$$

In this formulation, the complete decoding can be seen as an optimization problem of finding $\hat{\mathbf{x}}$ for given $\mathbf{A}$ and $\mathbf{y}$ such that

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2. \tag{2.8}$$

Below, we consider several approaches to solving equation 2.8.

2.2.1 Feedback-Based Techniques. The key idea of feedback-based techniques is to leverage initial predictions $\hat{\mathbf{s}}$ (obtained, e.g., with one of the selective techniques from section 2.1) to remove cross-talk noise in the decoding of one field by subtracting the hypervectors for all (or some) other fields in $\mathbf{y}$. Similar ideas for such a feedback mechanism have been developed in other fields of research and referred to as "explaining away," "interference cancellation," or "peeling decoding." We consider two feedback-based decoding techniques: explaining-away feedback (EA) and matching pursuit with explaining away.

Explaining away. To reduce the cross-talk noise in decoding one field i of the data structure, this technique constructs the corresponding hypervector from the decoding predictions of all other data fields and subtracts it from $\mathbf{y}$. Under the assumption that most of the decoding predictions $\hat{\mathbf{s}}$ are correct, this subtraction significantly reduces cross-talk for decoding the data field i. Formally, this can be written as

$$\hat{\mathbf{y}}(i) = \mathbf{y} - \sum_{j=1;\, j\neq i}^{v} \rho^{v-j}(\mathbf{\Phi}_{\hat{\mathbf{s}}_j}). \tag{2.9}$$

The hypervector $\hat{\mathbf{y}}(i)$ with reduced cross-talk noise is then used with the corresponding readout matrix $\mathbf{W}^{\text{out}}(i)$ to revise the prediction for the ith position of $\hat{\mathbf{s}}$:

$$\hat{\mathbf{s}}_i = \arg\max(\mathbf{W}^{\text{out}}(i)\hat{\mathbf{y}}(i)). \tag{2.10}$$

This process is repeated iteratively until either the predictions in $\hat{\mathbf{s}}$ stop changing or the maximum number of iterations (denoted as r) is reached. We consider two variants of EA using the two variants of selective decoding described above: Codebook EA and LR EA.

Matching pursuit with explaining away. One issue with EA is that when many of the decoding predictions in $\hat{\mathbf{s}}$ are wrong, the subtraction adds rather than removes noise. One possibility to counteract this problem is by successively subtracting individual decoded fields in the hypervector, starting with the ones for which the confidence of the correct decoding is highest. As a confidence measure for selective decoding, we choose the cosine similarity between the (residual) hypervector and the best, appropriately permuted, matching codebook entry. A confidence score is calculated as the difference between the highest and the second-highest cosine similarities. Intuitively, we expect that a high confidence score should correlate with the decoding result being correct.

Using a technique for selective decoding for all positions, we choose the decoding result in position c with the highest confidence score and remove this prediction $\hat{\mathbf{s}}_c$ from the compositional hypervector $\mathbf{y}$:

$$\tilde{\mathbf{y}} = \mathbf{y} - \rho^{v-c}(\mathbf{\Phi}_{\hat{\mathbf{s}}_c}). \tag{2.11}$$

From now on, the prediction for position c is fixed, and we assume that $\tilde{\mathbf{y}}$ stores only $v-1$ symbols.[1] The new hypervector $\tilde{\mathbf{y}}$ can be used with EA to make new predictions for the remaining $v-1$ symbols. Then we choose the most confident prediction among these $v-1$ symbols, fix the prediction, remove it from $\tilde{\mathbf{y}}$, and repeat the EA decoding for the remaining $v-2$ symbols. In such a manner, the decoding proceeds successively until complete. This type of confidence-based EA is similar to matching pursuit (MP), a well-known greedy technique for sparse signal approximation (Mallat & Zhang, 1993). In a step of MP, the best matching codebook element is weighted with the dot product between signal and codebook element to explain as much as possible of the signal. The next MP step continues on the residual. Essentially, equation 2.11 is also the residual of an MP approximation. However, the goal here is to explain away an element of the hypervector representing one field of the data structure. As the encoding procedure weights all used codebook elements with a value of one, the weight chosen in the residual is also one. Similar to the case of EA, we investigate MP decoding with the two variants of selective decoding: Codebook MP and LR MP.

2.2.2 LASSO Techniques. The formulation in equation 2.8 can be conceptualized as trying to infer a solution simultaneously (i.e., trying to decode the whole data structure at once). Note that this problem formulation is a relaxed version of the original task, as it does not take into account the constraint that there is only one nonzero component within each D-dimensional segment of $\hat{\mathbf{x}}$. We can simply impose this constraint and form $\hat{\mathbf{s}}$ from $\hat{\mathbf{x}}$ by assigning $\hat{s}_i$ to the position of the highest component of the ith D-dimensional segment of $\hat{\mathbf{x}}$.

Another way to think about the constraint above is that it is as if we have prior knowledge that $\hat{\mathbf{x}}$ has only v nonzero components. This means that the density of $\hat{\mathbf{x}}$ should be $1/D$ so the expected solution becomes quite sparse even for moderately large values of D. Therefore, the problem in equation 2.8 can be treated as the sparse inference procedure used in the areas of sparse coding (Olshausen & Field, 1996) and compressed sensing (Donoho, 2006). Thus, the natural choice for techniques to solve equation 2.8 should come from an arsenal of methods developed within the sparse coding/compressed sensing literature. The most common approach to do so is via a well-known lasso regression (Tibshirani, 1996) that adds L1 norm regularization to equation 2.8,

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 + \lambda\|\mathbf{x}\|_1, \tag{2.12}$$

[1] Note that it is still important to keep track of the original positions in the sequence due to the use of position-dependent permutations.

where λ is a standard hyperparameter denoting the importance of the L1 regularization term. Coordinate descent (CD) and gradient descent algorithms are investigated for solving the lasso. CD is implemented by Pedregosa et al. (2011; CD decoding) and the fast iterative shrinkage-thresholding algorithm (FISTA, FISTA decoding; Beck & Teboulle, 2009) is used for gradient descent.

2.2.3 Hybrid Techniques. Hybrid techniques combine primitives from the previous techniques as indicated by the dashed arrows in Figure 1. Although there is no fixed recipe for combining techniques, we show that one particularly powerful technique is in combining CD or FISTA decoding and LR decoding with MP (CD/LR MP and FISTA/LR MP). In these techniques, either CD or FISTA decoding is used every time when the current most confident prediction is explained away from $\mathbf{y}$ according to equation 2.11[2] while LR EA decoding is used to improve CD's or FISTA's predictions for the symbols that are not yet fixed.

3 Empirical Evaluation

In the experiments, we focus on three settings for the decoding:

- Decoding in the absence of external noise (section 3.1)
- Decoding in the presence of external noise (section 3.2)
- Decoding from storage elements with limited precision (section 3.3).

Before going into the results of evaluation, we briefly repeat the notations introduced so far because we will be using them intensively below. $\mathbf{y}$ is an n-dimensional vector that represents a sequence $\mathbf{s}$ of v symbols where the symbols are chosen from an alphabet of size D whose representations are stored in the codebook $\mathbf{\Phi}$. A hypervector of an ith symbol in $\mathbf{s}$ is denoted by $\mathbf{\Phi}_{\mathbf{s}_i}$ while the reconstructed sequence is denoted as $\hat{\mathbf{s}}$.

3.1 Noiseless Decoding. We begin by comparing the surveyed decoding techniques in section 2 in a scenario when no external noise is added to $\mathbf{y}$. This follows the setup of Hersche et al. (2021), which previously reported the highest information rate of HD/VSA in bits per dimension that could be achieved in practice. In the experiments in Hersche et al. (2021), n was set to 500 (see appendix B, which reports the effect of n); D was chosen from $\{5, 15, 100\}$, and v varied between 0 and 300 (we used 400 for $D = 5$). The results of the experiments for the techniques from section 2 are presented in Figure 2. In Hersche et al. (2021), only the first 4 out of 10 techniques (see the legend in Figure 2) compared here[3] were considered. The best

[2] CD or FISTA decoding is also used to make the initial predictions from the original $\mathbf{y}$.

[3] It also reported a "soft-feedback" technique that we do not report here due to its high similarity to EA.

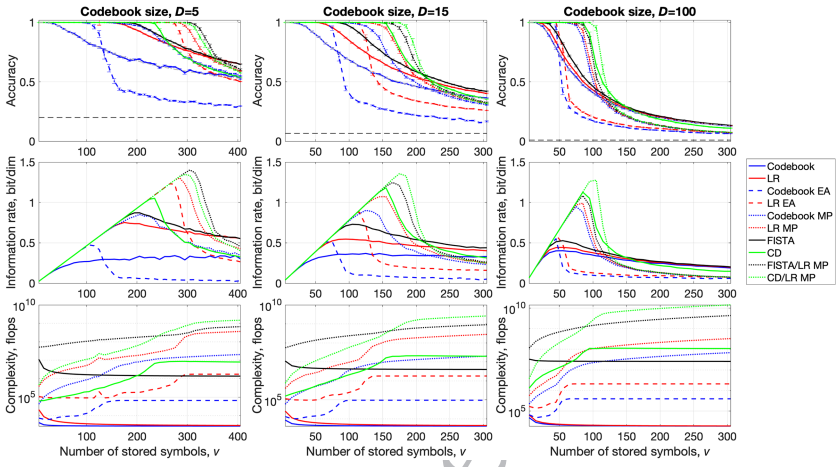

Figure 2: The decoding accuracy, information rate, and computational complexity against sequence length v for three different codebook sizes D. No external noise was added to $\mathbf{y}$. The upper panels depict accuracy, the middle panels correspond to information rate, and the lower panels show complexity in terms of floating point operations. The reported results are averages obtained from 20 randomly initialized codebooks. Ten random sequences were simulated for each codebook per each v. In the upper panels, bars depict 95% confidence intervals, the thin dashed black lines indicate the corresponding random guess at $1/D$, and gray dashed lines correspond to analytical predictions for Codebook decoding.

information rate (see the definition in equation A.7 in section A.3.2) achieved in Hersche et al. (2021), was approximately 1.20, 0.85, and 0.60 bits per dimension for D equal to 5, 15, and 100, respectively. The key takeaway from Figure 2 is the improvement over previously achieved information rates. The new highest results for information rate are 1.40 (17% improvement), 1.34 (58% improvement), and 1.26 (110% improvement) bits per dimension for D equal to 5, 15, and 100, respectively.

Note that for all values of D, the highest information rate was obtained with the hybrid techniques. This matches the fact that the lasso techniques alone demonstrated high-fidelity (i.e., close to perfect) regimes of decoding accuracy (see the definition in equation A.4 in section A.3.1), longer than the ones obtained with the selective techniques.

For selective techniques, the observations are consistent with previous reports (Frady et al., 2018; Hersche et al., 2021) where LR decoding (red solid lines) improves over Codebook decoding (blue solid lines) for small values of D (e.g., $D = 5$), but the improvement diminishes as D increases (see the right-most panels in Figure 2).

As for feedback-based techniques, it is clear that EA (dashed lines) extended the high-fidelity regime for the corresponding selective techniques. At the same time, there was a critical value of the accuracy of a selective technique, after which the accuracy of EA reduced drastically (since incorrect predictions added noise rather than removing it). The use of MP (dotted lines) partially alleviated this issue, as the cross-talk noise was removed symbol after symbol. This resulted in even longer high-fidelity regimes and a more gradual transition from the high-fidelity to the low-fidelity regimes (where performance was near chance).

In order to compare the computational complexity of different techniques, we measured the average number of floating point operations (flops) per decoding of a symbol using the PAPI (Performance Application Programming Interface) library (Terpstra et al., 2010) (see Figure 2, lower panels). Not surprisingly, selective techniques (especially Codebook decoding) were the cheapest to compute.[4] The key observation to make, however, is that the techniques that provided the highest information rate (e.g., the hybrid techniques) also required the largest number of computations.[5] This observation suggests that there is a trade-off between the computational complexity of a decoding technique and the amount of information it can decode from a distributed representation. As an example, we can consider techniques using EA (dashed lines) and MP (dotted lines). We already noted that the use of MP noticeably improves the high-fidelity regime. However, there is a computational price to be paid for the improvement since EA involves up to υr repetitions (grows linearly with υ) of some selective decoding while MP using EA as a part of its algorithm requires up to $\upsilon(\upsilon+1)r/2$ repetitions (grows quadratically with υ), which contributes substantially to the computational complexity (see the corresponding curves in the lower panels in Figure 2).

3.2 Noisy Decoding. In the previous experiment, no external noise was added to $\mathbf{y}$. However, in many scenarios, distributed representations are exposed to noisy environments—for example, during data transmission. Therefore, it is worth investigating the behavior of the considered decoding techniques in the presence of noise. We performed the experiments with the same setup as in the previous experiment using additive white gaussian

[4] Out of full fairness, we should note that we have not included the one-time cost of computing $\tilde{\mathbf{C}}$ for techniques that used LR decoding, which would surely add to the already reported costs.

[5] The values reported in Figure 2 assume that the whole sequence needs to be decoded. In the case when only a single symbol of the sequence should be decoded, the gap between selective and complete decoding techniques will be even larger since the techniques for complete decoding would need to decode the whole sequence anyway, while the techniques for selective decoding would be able to decode an individual field without accessing the other ones.

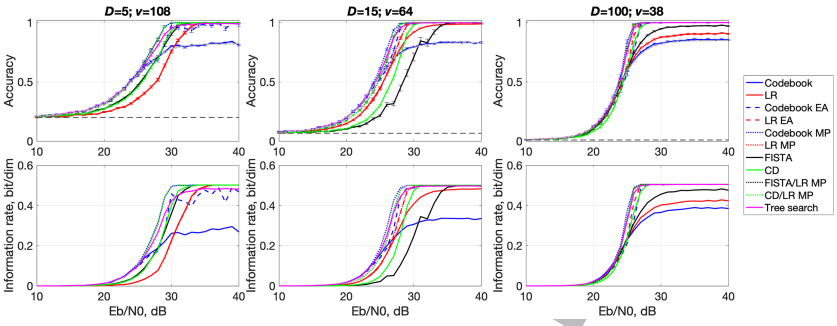

Figure 3: The decoding accuracy and information rate against the amount of external noise added to $\mathbf{y}$. The upper panels depict accuracy, while the lower panels correspond to information rate. For each value of D, the value of v was chosen such that the coding rate $v \log_2(D)/n$ would be approximately 0.5. The reported results are averages obtained from 20 randomly initialized codebooks. Ten random sequences were simulated for each codebook per each value of Eb/N0. In the upper panels, bars depict 95% confidence intervals, thin dashed black lines indicate the corresponding random guess at $1/D$, and gray dashed lines correspond to analytical predictions for Codebook decoding.

noise (AWGN). In order to account for the varying magnitude of $\mathbf{y}$, the normalized signal-to-noise ratio (known as Eb/N0) was used to regulate the amount of noise being added to $\mathbf{y}$ as

$$\hat{\mathbf{y}} = \mathbf{y} + \varepsilon, \tag{3.1}$$

where ε is an n-dimensional vector with AWGN that is randomly sampled for the given signal-to-noise ratio. The signal-to-noise ratio for a chosen Eb/N0 was computed as $\text{Eb/N0} + 10 \log_{10}(v \log_2(D)/n)$, and the power of the signal was $\langle \mathbf{y}^2 \rangle$. The noisy version of $\mathbf{y}$, $\hat{\mathbf{y}}$, was an input for all the decoding techniques. Figure 3 reports the results. Note that for this experiment, we have considered an additional decoding technique, labeled "Tree search" (with magenta solid lines). It was introduced in Hsu and Kim (2020), where the problem of decoding individual symbols from the superposition hypervector was formulated as an optimization problem (similar to the lasso techniques) but instead using a tree-based search to iteratively decode each symbol. Instead of concatenating one-hot encodings of symbols into a single vector to do the joint optimization according to equation 2.12, the technique in Hsu and Kim (2020) searches individual symbols one by one and keeps track of K best candidates for each symbol. We implemented the tree-based search with the universal sorting, keeping $K = 2$ best candidates at each step.

In Figure 3, for each value of D, the value of v was chosen to match the information rate of 0.5 bits per dimension. Clearly, if the amount of added noise was too high, the accuracies were down to random guess values ($1/D$) so no information was retrieved; hence, the information rate was zero. Once signal-to-noise improved, each decoding technique reached its highest accuracy matching the corresponding noiseless value (see Figure 2). The tree-based search included in the comparison, as expected, performed better than the EA-based techniques but slightly worse than the hybrid techniques. Also, with the increased value of D (see the right-most panels), the difference in the performance of different techniques during the transition from the low-fidelity regime to the high-fidelity regime was not significant. This was, however, not the case for lower values of D, where the first thing to notice was that Codebook decoding was the first technique to demonstrate accuracies that were higher than the random guesses $1/D$. As a consequence, feedback-based techniques using Codebook decoding also performed well; for example, the Codebook MP decoding was either on a par with or better than the CD/LR MP decoding. Thus, it is useful to keep in mind that in some scenarios, rather simple decoding techniques might be still worthwhile in terms of both performance and computational cost.

3.3 Decoding with Limited Precision. Before, we assumed that the superposition operation used when forming $\mathbf{y}$ was linear. A practical disadvantage of this assumption is that for large values of v, a possible range of values of $\mathbf{y}$ is also large, making it expensive to store $\mathbf{y}$. Therefore, it is practical to consider limiting the precision of components of $\mathbf{y}$. Since $\mathbf{y}$ consists of only integer values, a clipping function that is commonly used in neural networks (Frady et al., 2018; Kleyko et al., 2019, 2021) is a simple choice to keep the values of components of $\mathbf{y}$ in a limited range that is regulated by a threshold value (denoted as κ):

$$f_\kappa(y) = \begin{cases} -\kappa & y \leq -\kappa \\ y & -\kappa > y > \kappa \\ \kappa & y \geq \kappa \end{cases}. \tag{3.2}$$

Thus, when using the clipping threshold κ, the values of $f_\kappa(\mathbf{y})$ will be integers in the range between $-\kappa$ and κ. In this case, each neuron can be represented using only $\log_2(2\kappa + 1)$ bits of memory. For example, when $\kappa = 3$, there are seven unique values that can be stored with just three bits.

In order to investigate the effect of the limited precision, we used the clipping function with the following values of $\kappa \in \{1, 3, 7, 15, 31, 63, 127, 255, 511\}$ that approximately required $[2 : 10]$ bits of storage per dimension, respectively. The results are reported in Figure 4, where the upper panels show the accuracy, and the middle and the lower panels depict the information rate in bits per dimension and bits per storage bits, respectively. The

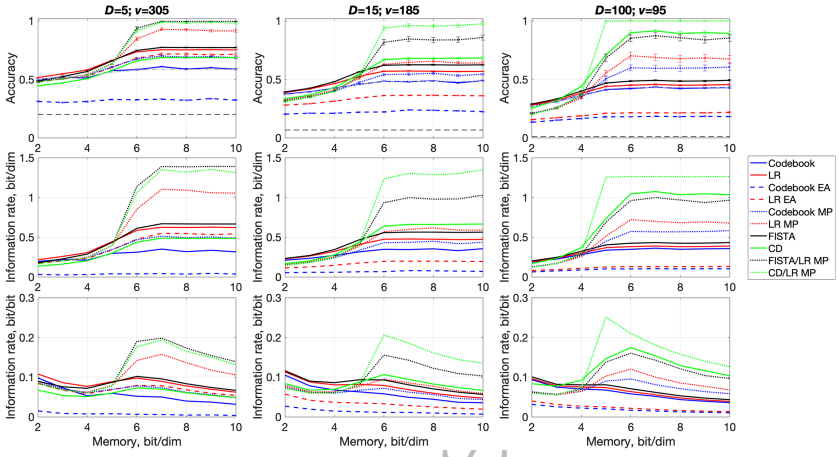

Figure 4: The decoding accuracy and information rate against the limited precision used to store $\mathbf{y}$. The top panels depict accuracy, the middle panels correspond to the information rate in bits per dimension, and the bottom panels show the information rate in information bits per storage bits. The reported results are averages obtained from 20 randomly initialized codebooks. Ten random sequences were simulated for each codebook per each memory size. In the upper panels, bars depict 95% confidence intervals, thin dashed black lines indicate the corresponding random guess at $1/D$, and gray dashed lines correspond to analytical predictions for Codebook decoding.

values of υ for each D were chosen to match their peaks of the information rate as observed in Figure 2. In order to account for the effect of the reduced scaling due to the use of the clipping function, the clipped representations were rescaled by a constant factor, which was estimated analytically based on υ and κ, to match the power of the original representations.

First, it is clear that making the superposition operation to be nonlinear was detrimental for some of the decoding techniques. This is particularly true for EA techniques that performed worse than their corresponding selective techniques. EA was so sensitive to the use of the clipping function as it is based on the assumption that the superposition operation to form $\mathbf{y}$ is linear. The other techniques, however, managed to get close to their information rate in bits per dimension (middle panels in Figure 4; see the middle panels in Figure 2) once κ was sufficiently large. We could also see that the best information rate in information bits per storage bits (lower panels) was achieved for the smallest value of κ that allowed reaching the maximum of the information rate in bits per dimension. Note also that for larger values of D, the information rate in information bits per storage bits was higher.

This is the case as for lower values of D the peak in the information rate was observed for higher values of υ that in turn means a large range of values of $\mathbf{y}$ and, hence, larger values of κ to preserve most of the range. Finally, it is worth noting that the second smaller peak in the information rate in information bits per storage bits was observed for the smallest value of $\kappa = 1$, where the selective techniques were the best option. This is expected since the range is so limited that neither feedback-based nor lasso techniques can benefit from it.

4 Discussion

4.1 Summary of the Study. In this article, we have focused on the problem of retrieving information from compositional distributed representations obtained using the principles of HD/VSA. To the best of our knowledge, this is the first attempt to survey, categorize, and quantitatively compare decoding techniques for this problem. Our taxonomy reveals that decoding techniques from other research areas can be utilized, such as reservoir computing, sparse signal representation, and communications. In fact, some of the investigated techniques were not used previously in HD/VSA but improved the information rate bounds beyond the state-of-the-art. We also introduced a novel decoding technique: matching pursuit with explaining away (see section 2.2.1). It should be noted that the experiments in this study used the multiply-add-permute model. While we showed before (Frady et al., 2018; Schlegel et al., 2022) that for Codebook decoding some HD/VSA models are more accurate (e.g., Fourier holographic reduced representations model; Plate, 1995a), this observation should not affect the consistency of relative standing of the considered decoding techniques when evaluated on models other than multiply-add-permute. Our decoding experiments explored three different encoding scenarios: the hypervector formed by plain linear superposition, linear superposition with external noise, and lossy compression of linear superposition using component-wise clipping. The standard decoding technique in HD/VSA, Codebook decoding, was in all scenarios outperformed by other techniques. Nevertheless, it combines decent decoding performance with other advantages: absence of free parameters that require tuning and the lowest computational complexity. In the first scenario of linear superposition with no external noise, lasso techniques performed exceptionally well. In other scenarios, high noise, or compression with strong nonlinearity (see Figure 4), the assumptions in the optimization approach (see equation 2.12) are violated and, accordingly, the performance is worse than with simpler techniques. Notably, in our experiments, the hybrid decoding techniques combining lasso techniques with matching pursuit with explaining away advanced the theory of HD/VSA by improving the information rate bounds of the distributed representations reported before in Frady et al. (2018) and Hersche et al. (2021)

by at least 17%. However, this improvement comes at the price of performing several orders of magnitude more operations compared to the simplest selective techniques (see the lower panels in Figure 2, which highlight the trade-off between the computational complexity and information rate).

4.2 Related Work.

4.2.1 Randomized Neural Networks and Reservoir Computing. Decoding from distributed representations can be seen as a special case of function approximation, which connects it to randomized neural networks and reservoir computing (Scardapane & Wang, 2017). As we highlighted in section 2.1.2, this interpretation allows learning a readout matrix for each position in a sequence from training data. This technique was introduced to HD/VSA in Frady et al. (2018), which has also shown that when distributed representations are formed according to (equation A.3), the readout matrices do not have to be trained; they can be computed using the covariance matrix from (equation 2.4).

4.2.2 Sparse Coding (SC) and Compressed Sensing (CS). As indicated in section 2.2.2, the task of retrieving from equation A.3 can be framed as sparse inference procedure used within SC (Olshausen & Field, 1996) and CS (Donoho, 2006). Within the HD/VSA literature, this connection was first made in Summers-Stay et al. (2018) for decoding from sets and in Frady, Kleyko, and Somme (2021) for sets and sequences. Similar to SC and CS, L0 sparsity is more desirable than L1 sparsity since the sparse vector, $\mathbf{x} \in \{0, 1\}^{vD}$, is composed of variables that are exactly zero and one. In general, optimization of the L0 penalty is a hard problem. Optimization with the L1 penalty thresholds small values, leading to a sparse vector with many variables being zero. Efficient algorithms exist for optimization of the L1 penalty, which provides a practical technique for performing sparse inference.

4.2.3 Communications. The problem of decoding individual messages from their superposition as in equation A.3 is the classic multiple access channel (MAC) problem in communications. The capacity region, which specifies the achievable rates for all users given their signal-to-noise ratios, has been fully characterized (Cover, 1999). It is known that the capacity region of an MAC can be achieved by code-division-multiple access (CDMA), where separate codes are used by different senders and the receiver decodes them one by one. This so-called *successive interference cancellation* (onion peeling) is the key idea used for Codebook decoding with EA. Understanding how close the performance of the decoder is to the capacity will provide us insights for improving decoder design in the future.

Within HD/VSA, the decoding with inference cancellation (EA) was introduced in Kim (2018) that proposed to combine forward error correction

and modulation using the Fourier holographic reduced representations model (Plate, 1995a). Similar to the results reported here, the main motivation for using inference cancellation was that it significantly improved the quality of decoding compared to Codebook decoding. Later Hersche et al. (2021) introduced the so-called soft-feedback technique, similar to MP; it makes use of the prediction's confidence. Another improvement on top of EA was the tree-based search (Hsu & Kim, 2020). It outperformed EA-based techniques with the caveat that the complexity of the tree-based search grows exponentially with the number of branches, so only the K best candidates for each symbol were retained (Hsu & Kim, 2020). This imposes a trade-off between decoding accuracy and computation/time complexity. It is also prone to errors when several candidates share the same score.

Another development within communications that is very similar to the retrieval task considered here is sparse superposition codes (SSC; Barron & Joseph, 2010). SSCs are capacity-achieving codes with a sparse block structure that are closely related to SC (Olshausen & Field, 1996) and CS (Donoho, 2006). SSC decoding algorithms are like those studied in this work, such as L1 minimization, successive interference cancellation, and approximate belief propagation techniques. Future work should investigate SSC constructed from the encoding and decoding strategies from this work.

4.2.4 Related Work within HD/VSA Literature. Besides the work mentioned above that has been connecting the task of retrieving from distributed representations formed by HD/VSA to tasks within other areas, work has also studied the Codebook decoding technique. Early analytical results on the performance of Codebook decoding with real-valued hypervectors were given in Plate (2003). For the case of dense binary hypervectors, an important step for obtaining the analytical results is in estimating an expected Hamming distance between the compositional hypervector and a symbol's hypervector (see, e.g., expressions in Kanerva, 1997; Kleyko, Gayler et al., 2020; Mitrokhin et al., 2019). Further steps for the dense binary/bipolar hypervectors were presented in Gallant and Okaywe (2013), Kleyko et al. (2017), and Rahimi et al. (2017). The performance in the case of sparse binary hypervectors (Rachkovskij, 2001) was analyzed in Kleyko et al. (2018). The most general and comprehensive analytical studies of the performance of Codebook decoding for different HD/VSA models were recently presented in Frady et al. (2018) and Kleyko, Rosato et al. (2023) while other recent studies (Clarkson et al., 2023; Thomas et al., 2021) have provided theoretical bounds of several HD/VSA models in other scenarios. Some recent empirical studies of the capacity of HD/VSA can be also found in Mirus et al. (2020) and Schlegel et al. (2022).

Finally, it is worth noting that the problem formulation considered here is very similar to the trajectory association task that was proposed in Plate (1992) and can be used to study the memory capacity of recurrent neural networks (see, Danihelka et al., 2016; Frady et al., 2018).

4.3 Future Work. In this study, we have surveyed the key ideas and techniques to solve the retrieval task. There are, however, more specific techniques to try, as well as other angles for looking at the problem. A possibility for future work is to compare the computation complexity and decoding accuracy of different lasso techniques. Some other techniques that we have not simulated but are worth exploring include MP with several iterations, genetic algorithms for refining the best current solution, and lasso solving for a range of values of λ (see Summers-Stay et al., 2018, for some experiments within HD/VSA).

While in this study, we have fixed the formation of distributed representations (see equation A.3), it is expected that the choice of the transformation of input data can affect the performance of the decoding techniques. For instance, as we saw in Figure 2, working with smaller codebooks leads to increased information rates. Another notable example of importance of input transformation are fountain codes (MacKay, 2005), where the packet distribution can be optimized to minimize the probability of error. Therefore, in future work, we also plan to consider other transformations to distributed representations that we did not consider here.

Appendix A: Background

A.1 Vector Symbolic Architectures. In this section, we provide a summary from Kleyko, Davies et al. (2022) to briefly introduce HD/VSA (Gayler, 2003; Kanerva, 2009)[6] using the multiply-add-permute (MAP) model (Gayler, 2003) to showcase a particular HD/VSA realization. It is important to keep in mind that HD/VSA can be formulated with different types of vectors, namely, those containing real (Gallant & Okaywe, 2013; Plate, 1995a), complex (Plate, 1995a), or binary entries (Frady, Kleyko, & Sommer, 2021; Kanerva, 1997; Laiho et al., 2015; Rachkovskij, 2001). The key HD/VSA model has these key components:

- High-dimensional space (e.g., integer; n denotes the dimensionality)
- Pseudo-orthogonality (between two random vectors in this high-dimensional space)
- Similarity measure (e.g., dot (inner) product or cosine similarity)
- Atomic representations (e.g., random i.i.d. high-dimensional vectors, also known as hypervectors)
- Item memory storing atomic hypervectors and performing autoassociative search
- Operations on hypervectors

In MAP (Gayler, 2003), the atomic hypervectors are bipolar random vectors, where each vector component is selected randomly and independently

[6] See Kleyko, Rachkovskij et al. (2022, 2023) for a comprehensive survey of HD/VSA.

from $\{-1, +1\}$. These random i.i.d. atomic hypervectors can serve to represent "symbols" in HD/VSA (i.e., categorical objects), since such vectors are pseudo-orthogonal to each other (due to the concentration of measure phenomenon) and thus are treated as dissimilar.

Additionally, each HD/VSA model defines three key operations used to manipulate atomic hypervectors; we specify their implementations in MAP:

- Superposition, also known as bundling (denoted as $+$; implemented as component-wise addition possibly followed by some normalization function)
- Permutation (denoted as ρ; implemented as a rotation of components)
- Binding (denoted as $\odot$; implemented as component-wise multiplication, also known as Hadamard product; not used in this article)

HD/VSA models also need to define a similarity measure between two vector representations. For this purpose, we will use dot product and cosine similarity that are computed for two hypervectors $\mathbf{a}$ and $\mathbf{b}$ as

$$\mathbf{a}^\top \mathbf{b} \tag{A.1}$$

and

$$\frac{\mathbf{a}^\top \mathbf{b}}{||\mathbf{a}||_2 ||\mathbf{b}||_2}, \tag{A.2}$$

respectively; $||\cdot||_2$ denotes L2 norm of a hypervector.

A.2 Problem Formulation. In this section, we present a transformation (i.e., an encoding scheme) that is used to form distributed representations for this study. It is worth noting that the use of HD/VSA operations allows forming compositional distributed representations for a plethora of data structures such as sets (Kanerva, 2009; Kleyko, Rahimi et al., 2020), sequences (Hannagan et al., 2011; Kanerva, 2009; Thomas et al., 2021), state machines (Osipov et al., 2017; Yerxa et al., 2018), hierarchies, predicate relations (Gallant, 2022; Plate, 2003; Rachkovskij, 2001), and so on. (Consult Kleyko, Davies et al., 2022, for a detailed tutorial on representations of these data structures.)

To focus on decoding techniques, we use only one simple but common transformation for representing a symbolic sequence of length v. A sequence (denoted as $\mathbf{s}$—e.g., $\mathbf{s} = (a, b, c, d, e)$) is assumed to be generated randomly. Symbols constituting the sequence are drawn from an alphabet of finite size D, and the presence of each symbol in any position of the sequence is equiprobable.

In order to form a distributed representation of a sequence, first we need to create an item memory, $\mathbf{\Phi}$ (we call it the *codebook*) that stores atomic n-dimensional random i.i.d. bipolar dense hypervectors corresponding to symbols of the alphabet,[7] thus, $\mathbf{\Phi} \in \{-1, 1\}^{n \times D}$. The hypervector of the ith symbol in $\mathbf{s}$ will be denoted as $\mathbf{\Phi}_{\mathbf{s}_i}$. It should be noted that in HD/VSA, it is a convention to draw hypervectors' components randomly unless there are good reasons to make them correlated (see Frady, Kleyko, Kymn et al., 2021; Frady et al., 2022). The presence of correlation will, however, reduce the performance of the decoding techniques because their signal-to-noise ratio will be lower and, thus, the decoding becomes harder.

For sequence transformations, there is a need to associate a symbol's hypervector with a symbol's position in the sequence. There are several approaches to do so; we use one that relies on the permutation operation (Frady et al., 2018; Kanerva, 2009, 2019; Kleyko et al., 2016; Plate, 1995b; Sahlgren et al., 2008). The idea is that before combining the hypervectors of sequence symbols, the position i of each symbol is associated by applying some fixed permutation $v - i$ times to its hypervector[8] (e.g., $\rho^2(\mathbf{\Phi}_c)$ for the sequence above).

The last step is to combine the sequence symbols into a compositional hypervector (denoted as $\mathbf{y}$) representing the whole sequence. We do it using the superposition operation. For the sequence above, the compositional hypervector is

$$\mathbf{y} = \rho^4(\mathbf{\Phi}_a) + \rho^3(\mathbf{\Phi}_b) + \rho^2(\mathbf{\Phi}_c) + \rho^1(\mathbf{\Phi}_d) + \rho^0(\mathbf{\Phi}_e).$$

In general, a given sequence $\mathbf{s}$ of length v is represented as

$$\mathbf{y} = \phi(\mathbf{s}) = \sum_{i=1}^{v} \rho^{v-i}(\mathbf{\Phi}_{\mathbf{s}_i}). \tag{A.3}$$

Note that a transformation of the sequence similar to the one in equation A.3 can be obtained by assigning v random bipolar hypervectors (one for each unique position) and using the binding operation on the corresponding hypervectors to represent the association between a symbol and its position. However, we do not report the experiments on such a representation since in terms of decoding performance, it is equivalent to that in equation A.3. In this study, we also assume that the superposition operation is linear unless noted otherwise (see section 3.3), that is, that no normalization operation is applied to the result of equation A.3.

[7] For simplicity, we assume that the symbols are integers between 0 and $v - 1$. This notation makes it more convenient to introduce the decoding techniques in section 2.

[8] It is worth pointing out that the reverse order of applying successive powers of a permutation can be used as well.

The problem of decoding from compositional distributed representation $\mathbf{y}$ is formulated as follows: for given v, $\mathbf{\Phi}$, and $\mathbf{y}$, the task is to provide a reconstructed sequence (denoted as $\hat{\mathbf{s}}$) such that $\hat{\mathbf{s}}$ is as close as possible to the original sequence $\mathbf{s}$.

A.3 Performance Metrics. In this section, we introduce two performance evaluation metrics used in this study to assess the quality of the reconstructed sequence $\hat{\mathbf{s}}$.

A.3.1 Accuracy of the Decoding. For the decoding of a sequence's symbols from its compositional distributed representation $\mathbf{y}$, the accuracy of the correct decoding (denoted as a) is a natural metric to characterize the performance of a decoding technique. Since the superposition operation used in this study is linear, the accuracy of decoding symbols in different positions of the sequence is the same, and so there is no need to compute separate accuracies for different positions. The accuracy is computed empirically over g randomly generated sequences, using the predictions in the reconstructed sequence $\hat{\mathbf{s}}$ (obtained by some decoding technique) and the original sequence $\mathbf{s}$: as an average ratio of symbols, which were decoded correctly,

$$a = \frac{1}{gv} \sum_{n=1}^{g} \sum_{i=1}^{v} [\mathbf{s}_i^{(n)} = \hat{\mathbf{s}}_i^{(n)}], \tag{A.4}$$

where $[\cdot]$ is the indicator function set to one when the symbols match and to zero otherwise. Note that since symbols in sequence $\mathbf{s}$ are equiprobable, the accuracy of a random guess is $1/D$.

A.3.2 Information Decoded from Distributed Representation. The accuracy is an intuitive performance metric for the task of sequence decoding and allows characterizing performance of various decoding techniques. Its disadvantage, however, is that the accuracy also depends on the dimensionality of representation, n, size of the codebook, D, and number of symbols in the sequence, v. In order to, for example, study the effect of these parameters on distributed representation, it is convenient to use a single scalar metric characterizing the quality of decoding that would take into account particular values of n, D, and v. For this purpose, we use the total amount of information decoded per a dimension of representation. The amount of information decoded for a single symbol (denoted as I_{symb}) is calculated using the corresponding accuracy and size of the codebook as[9]

[9] We do not go into the detailed derivation of this equation here. Refer to section 2.2.3 in Frady et al. (2018) for details.

$$I_{\text{symb}}(a, D) = a \log_2(Da) + (1 - a) \log_2 \left(\frac{D}{D-1}(1 - a) \right). \tag{A.5}$$

Note that when the accuracy equals that of a random guess ($1/D$), the amount of decoded information would be zero. The total amount of information decoded from the distributed representation is calculated as a sum of information extracted for all symbols:

$$I_{\text{tot}} = \sum_{i=1}^{\upsilon} I_{\text{symb}}(a, D) = \upsilon I_{\text{symb}}(a, D). \tag{A.6}$$

Finally, since I_{tot} does not account for the dimensionality of representation, it makes sense to consider the amount of decoded information per a single dimension:

$$I_{\text{dim}} = \frac{I_{\text{tot}}}{n}. \tag{A.7}$$

Thus, I_{dim} corresponds to an information rate that is a relative measure in bits per dimension that accounts for all three parameters n, D, and υ and can be used for a fair comparison of various transformations using different choices of these parameters.

Appendix B: Additional Experiments against the Dimensionality of Representations

In the main text, the dimensionality of representations was fixed to $n = 500$. While it is well known (Frady et al., 2018) that with the increased dimensionality the decoding accuracy will also increase, it is still worth demonstrating this effect empirically within the experimental protocol on this study. To do so, we hand-picked four decoding techniques from the considered groups: Codebook, LR MP, CD, and CD/LR MP. The techniques were evaluated using the values of n from $\{128, 256, 512, 1024, 2048\}$. For each codebook size, the choice of υ was made qualitatively based on Figure 2 choosing various relative performance gaps (at $n = 500$) between the techniques. The results are reported in Figure 5, where the upper panels depict the accuracy, and the lower panels show the information rate. There are no surprising observations about Figure 5. Eventually, given a sufficient dimensionality, each technique entered the high-fidelity regime. The relative ordering of the techniques persisted with increased n (unless all were perfectly accurate), while for lower n, cross-talk noise is too high, effectively similar to adding a lot of noise as in Figure 3, where all techniques performed equally poorly.

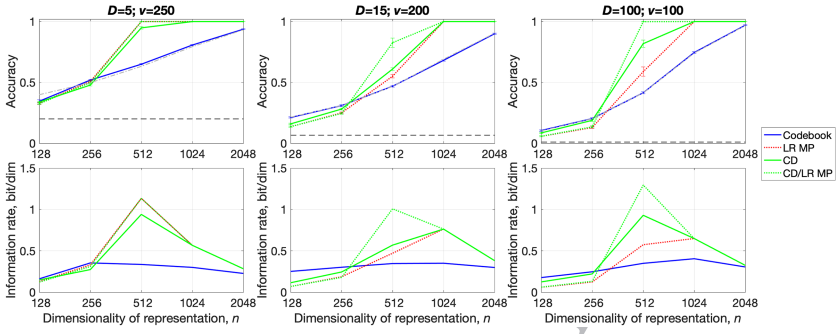

Figure 5: The decoding accuracy and information rate against dimensionality of representations n for three different codebook sizes D using a subset of the considered decoding techniques. The upper panels depict accuracy, while the lower panels correspond to information rate. The reported results are averages obtained from 10 randomly initialized codebooks. Ten random sequences were simulated for each codebook per each v. In the upper panels, bars depict 95% confidence intervals, thin dashed black lines indicate the corresponding random guess at $1/D$, and gray dashed lines correspond to analytical predictions for Codebook decoding.

Appendix C: Pseudocode for Hybrid Techniques

Section 2.2.3 stated that it is worth combining primitives from several various techniques, which results in hybrid techniques. In particular, the results in section 3 featured two such techniques: the combination of CD decoding with LR decoding with MP (CD/LR MP) as well as the combination of FISTA decoding with LR decoding with MP (FISTA/LR MP). In order to provide more details on the techniques, algorithm 1 lists the corresponding pseudocode for CD/LR MP. The algorithm for FISTA/LR MP is not shown explicitly as it is obtained simply by replacing every use of CD decoding in algorithm 1 by FISTA decoding. The notations used in the algorithm correspond to the ones introduced above. There are, however, several novel notations such as $\text{sim}_{\cos}(\cdot, \cdot,)$ to denote cosine similarity (see section A.1), **confidence** for a v-dimensional vector storing a confidence score for each position in the sequence (see section 2.2.1), **fixed** for a set storing positions with the fixed prediction (see section 2.2.1), as well as `CD()` and `LRMP()` denoting routines for performing CD and LR MP decoding, respectively. Note that these routines can take **fixed** as input, which means that the predictions in the corresponding positions will remain unchanged and values of confidence scores in these positions will be set to -1 to avoid choosing them again during the arg max() step.

Algorithm 1: A High-Level Pseudocode for CD/LR MP Decoding.

Require: $\mathbf{y}$; $\mathbf{\Phi}$; v;
  $\hat{\mathbf{s}}^{\mathrm{CD}}, \mathbf{confidence}^{\mathrm{CD}} \leftarrow \mathtt{CD}(\mathbf{y}, \mathbf{\Phi}, v, \emptyset, \emptyset)$
  $\hat{\mathbf{s}}^{\mathrm{LRMP}}, \mathbf{confidence}^{\mathrm{LRMP}} \leftarrow \mathtt{LRMP}(\mathbf{y}, \mathbf{\Phi}, v, \hat{\mathbf{s}}^{\mathrm{CD}}, \emptyset)$
  if $\mathrm{sim}_{\cos}(\mathbf{y}, \phi(\hat{\mathbf{s}}^{\mathrm{CD}})) \geq \mathrm{sim}_{\cos}(\mathbf{y}, \phi(\hat{\mathbf{s}}^{\mathrm{LRMP}}))$ **then**
    $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}}^{\mathrm{CD}}$
    $\mathbf{confidence} \leftarrow \mathbf{confidence}^{\mathrm{CD}}$
  else
    $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}}^{\mathrm{LRMP}}$
    $\mathbf{confidence} \leftarrow \mathbf{confidence}^{\mathrm{LRMP}}$
  end if
  $\tilde{\mathbf{y}} \leftarrow \mathbf{y}$
  $\mathbf{fixed} \leftarrow \emptyset$
  for $i = 1$ to $v - 1$ **do**
    $c \leftarrow \arg\max(\mathbf{confidence})$
    append c to **fixed** ▷ Confidence of positions already included in **fixed** is assumed to be equal -1
    $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \rho^{v-c}(\mathbf{\Phi}_{\hat{\mathbf{s}}_c})$
    $\hat{\mathbf{s}}^{\mathrm{CD}}, \mathbf{confidence}^{\mathrm{CD}} \leftarrow \mathtt{CD}(\tilde{\mathbf{y}}, \mathbf{\Phi}, v - i, \hat{\mathbf{s}}, \mathbf{fixed})$
    $\hat{\mathbf{s}}^{\mathrm{LRMP}}, \mathbf{confidence}^{\mathrm{LRMP}} \leftarrow \mathtt{LRMP}(\tilde{\mathbf{y}}, \mathbf{\Phi}, v - i, \hat{\mathbf{s}}^{\mathrm{CD}}, \mathbf{fixed})$
    if $\mathrm{sim}_{\cos}(\mathbf{y}, \phi(\hat{\mathbf{s}}^{\mathrm{CD}})) \geq \mathrm{sim}_{\cos}(\mathbf{y}, \phi(\hat{\mathbf{s}}^{\mathrm{LRMP}}))$ **then**
      $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}}^{\mathrm{CD}}$
      $\mathbf{confidence} \leftarrow \mathbf{confidence}^{\mathrm{CD}}$
    else
      $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}}^{\mathrm{LRMP}}$
      $\mathbf{confidence} \leftarrow \mathbf{confidence}^{\mathrm{LRMP}}$
    end if
  end for
  return $\hat{\mathbf{s}}$

Acknowledgments

We thank Spencer Kent for sharing with us the implementation of the FISTA algorithm. The work of F.T.S., B.A.O., C.B., and D.K. was supported in part by Intel's THWAI program. The work of B.A.O. and D.K. was also supported in part by AFOSR FA9550-19-1-0241. D.K. has received funding from the European Union's Horizon 2020 research and innovation program. under Marie Sklodowska-Curie grant agreement 839179. The work of C.J.K. was supported by the U.S. Department of Defense through the National Defense Science and Engineering Graduate Fellowship Program. F.T.S. was supported by Intel and NIH R01-EB026955.

References

Barron, A. R., & Joseph, A. (2010). Toward fast reliable communication at rates near capacity with gaussian noise. In *Proceedings of the IEEE International Symposium on Information Theory* (pp. 315–319). IEEE.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202. 10.1137/080716542

Clarkson, K. L., Ubaru, S., & Yang, E. (2023). *Capacity analysis of vector symbolic architectures*. arXiv:2301.10352.

Cover, T. M. (1999). *Elements of information theory*. Wiley.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., & Graves, A. (2016). Associative long short-term memory. In *Proceedings of the International Conference on Machine Learning* (pp. 1986–1994). ACM.

Demidovskij, A. (2021). Encoding and decoding of recursive structures in neural-symbolic systems. *Optical Memory and Neural Networks*, *30*(1), 37–50. 10.3103/S1060992X21010033

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306. 10.1109/TIT.2006.871582

Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A., & Sommer, F. T. (2021). *Computing on functions using randomized vector representations*. arXiv:2109.03429.

Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A., & Sommer, F. T. (2022). Computing on functions using randomized vector representations (in brief). In *Proceedings of the Neuro-Inspired Computational Elements Conference* (pp. 115–122). ACM.

Frady, E. P., Kleyko, D., & Sommer, F. T. (2018). A theory of sequence indexing and working memory in recurrent neural networks. *Neural Computation*, *30*, 1449–1513. 10.1162/neco_a_01084

Frady, E. P., Kleyko, D., & Sommer, F. T. (2021). Variable binding for sparse distributed representations: Theory and applications. *IEEE Transactions on Neural Networks and Learning Systems*, *99*(PP), 1–14.

Gallant, S. I. (2022). *Orthogonal matrices for MBAT vector symbolic archi-tectures, and a "soft" VSA representation for JSON*. arXiv:2202.04771.

Gallant, S. I., & Okaywe, T. W. (2013). Representing objects, relations, and sequences. *Neural Computation*, *25*(8), 2038–2078. 10.1162/NECO_a_00467

Gayler, R. W. (2003). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In *Proceedings of the Joint International Conference on Cognitive Science* (pp. 133–138). Erlbaum.

Guirado, R., Rahimi, A., Karunaratne, G., Alarcón, E., Sebastian, A., & Abadal, S. (2022). Wireless on-chip communications for scalable in-memory hyperdimensional computing. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1–8). IEEE.

Hannagan, T., Dupoux, E., & Christophe, A. (2011). Holographic string encoding. *Cognitive Science*, *35*(1), 79–118. 10.1111/j.1551-6709.2010.01149.x

Hersche, M., Lippuner, S., Korb, M., Benini, L., & Rahimi, A. (2021). Near-channel classifier: Symbiotic communication and classification in high-dimensional space. *Brain Informatics*, *8*, 1–15. 10.1186/s40708-021-00138-0

Hsu, C.-W., & Kim, H.-S. (2020). Non-orthogonal modulation for short packets in massive machine type communications. In *Proceedings of the IEEE Global Communications Conference* (pp. 1–6). IEEE.

Jakimovski, P., Schmidtke, H. R., Sigg, S., Chaves, L. W. F., & Beigl, M. (2012). Collective communication for dense sensing environments. *Journal of Ambient Intelligence and Smart Environments*, *4*(2), 123–134. 10.3233/AIS-2012-0139

Kanerva, P. (1997). Fully distributed representation. In *Proceedings of the Real World Computing Symposium* (pp. 358–365).

Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, *1*(2), 139–159. 10.1007/s12559-009-9009-8

Kanerva, P. (2019). Computing with high-dimensional vectors. *IEEE Design and Test*, *36*(3), 7–14. 10.1109/MDAT.2018.2890221

Kim, H.-S. (2018). HDM: Hyper-dimensional modulation for robust low-power communications. In *Proceedings of the IEEE International Conference on Communications* (pp. 1–6). IEEE.

Kleyko, D., Davies, M., Frady, E. P., Kanerva, P., Kent, S. J., Olshausen, B. A., . . . Sommer, F. T. (2022). Vector symbolic architectures as a computing framework for emerging hardware. *Proceedings of the IEEE*, *110*(10), 1538–1571. 10.1109/JPROC.2022.3209104

Kleyko, D., Gayler, R. W., & Osipov, E. (2020). *Commentaries on "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception"* [*Science Robotics* (2019), 4(30) 1–10]. arXiv:2003.11458.

Kleyko, D., Kheffache, M., Frady, E. P., Wiklund, U., & Osipov, E. (2021). Density encoding enables resource-efficient randomly connected neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(8), 3777–3783. 10.1109/TNNLS.2020.3015971

Kleyko, D., Lyamin, N., Osipov, E., & Riliskis, L. (2012). Dependable MAC layer architecture based on holographic data representation using hypedimensional binary spatter codes. In *Multiple access communications* (pp. 134–145). Springer.

Kleyko, D., Osipov, E., & Gayler, R. W. (2016). Recognizing permuted words with vector symbolic architectures: A Cambridge test for machines. *Procedia Computer Science*, *88*, 169–175. 10.1016/j.procs.2016.07.421

Kleyko, D., Osipov, E., Senior, A., Khan, A. I., & Sekercioglu, Y. A. (2017). Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(6), 1250–1262. 10.1109/TNNLS.2016.2535338

Kleyko, D., Osipov, E., Silva, D. D., Wiklund, U., & Alahakoon, D. (2019). Integer self-organizing maps for digital hardware. In *Proceedings of theInternational Joint Conference on Neural Networks* (pp. 1–8). IEEE.

Kleyko, D., Rachkovskij, D. A., Osipov, E., & Rahimi, A. (2022). A survey on hyperdimensional computing aka vector symbolic architectures, Part I: Models and data transformations. *ACM Computing Surveys*, *55*(6), 1–40.

Kleyko, D., Rachkovskij, D. A., Osipov, E., & Rahimi, A. (2023). A survey on hyperdimensional computing aka vector symbolic architectures, Part II: Applications, cognitive models, and challenges. *ACM Computing Surveys*, *55*(9), 1–52.

Kleyko, D., Rahimi, A., Gayler, R. W., & Osipov, E. (2020). AutoscalingBloom filter: Controlling trade-off between true and false positives. *Neural Computing and Applications*, *32*, 3675–3684. 10.1007/s00521-019-04397-1

Kleyko, D., Rahimi, A., Rachkovskij, D. A., Osipov, E., & Rabaey, J. M. (2018). Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristic. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(12), 5880–5898. 10.1109/TNNLS.2018.2814400

Kleyko, D., Rosato, A., Frady, E. P., Panella, M., & Sommer, F. T. (2023). Perceptron theory can predict the accuracy of neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *99*(PP), 1–15. 10.1109/TNNLS.2023.3237381

Laiho, M., Poikonen, J. H., Kanerva, P., & Lehtonen, E. (2015). High-dimensional computing with sparse vectors. In *Proceedings of theIEEE Biomedical Circuits and Systems Conference* (pp. 1–4). IEEE.

Lukosevicius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149. 10.1016/j.cosrev.2009.03.005

MacKay, D. J. C. (2005). Fountain codes. *IEE Proceedings-Communications*, *152*(6), 1062–1068. 10.1049/ip-com:20050237

Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, *41*(12), 3397–3415. 10.1109/78.258082

Mirus, F., Stewart, T. C., & Conradt, J. (2020). Analyzing the capacity distributed vector representations to encode spatial information. In *Proceedings of theInternational Joint Conference on Neural Networks* (pp. 1–7). IEEE.

Mitrokhin, A., Sutor, P., Fermuller, C., & Aloimonos, Y. (2019). Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, *4*(30), 1–10. 10.1126/scirobotics.aaw6736

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609. 10.1038/381607a0

Osipov, E., Kleyko, D., & Legalov, A. (2017). Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing. In *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society* (pp. 3276–3281). IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2825–2830.

Plate, T. A. (1992). Holographic recurrent networks. In S. Hanson, J. Cowan, & C. Giles (Eds.), *Advances in neural information processing systems*, *5* (pp. 34–41). MIT Press.

Plate, T. A. (1995a). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. 10.1109/72.377968

Plate, T. A. (1995b). *Networks which learn to store variable-length sequences in a fixed set of unit activations*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.3452&rep=rep1&type=pdf

Plate, T. A. (2003). *Holographic reduced representations: Distributed representation for cognitive structures*. Stanford: Center for the Study of Language and Information.

Rachkovskij, D. A. (2001). Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering*, *2*, 261–276. 10.1109/69.917565

Rahimi, A., Datta, S., Kleyko, D., Frady, E. P., Olshausen, B., Kanerva, P., & Rabaey, J. M. (2017). High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *64*(9), 2508–2521. 10.1109/TCSI.2017.2705051

Sahlgren, M., Holst, A., & Kanerva, P. (2008). Permutations as a means to encode order in word space. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (pp. 1300–1305). Curran.

Scardapane, S., & Wang, D. (2017). Randomness in neural networks: An overview. *Data Mining and Knowledge Discovery*, *7*, 1–18.

Schlegel, K., Neubert, P., & Protzel, P. (2022). A comparison of vector symbolic architectures. *Artificial Intelligence Review*, *55*, 4523–4555. 10.1007/s10462-021-10110-3

Shwartz-Ziv, R., & Tishby, N. (2017). *Opening the black box of deep neural networks via information*. arXiv:1703.00810.

Simpkin, C., Taylor, I., Bent, G. A., de Mel, G., Rallapalli, S., Ma, L., & Srivatsa, M. (2019). Constructing distributed time-critical applications using cognitive enabled services. *Future Generation Computer Systems*, *100*, 70–85. 10.1016/j.future.2019.04.010

Summers-Stay, D., Sutor, P., & Li, D. (2018). Representing sets as summed semantic vectors. *Biologically Inspired Cognitive Architectures*, *25*, 113–118. 10.1016/j.bica.2018.07.002

Terpstra, D., Jagode, H., You, H., & Dongarra, J. (2010). Collecting performance data with PAPI-C. In M. Müller, M. Resch, A. Schulz, & W. Nagel (Eds.), *Tools for high performance computing* (pp. 157–173). Springer.

Thomas, A., Dasgupta, S., & Rosing, T. (2021). A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research*, *72*, 215–249. 10.1613/jair.1.12664

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. 10.1111/j.2517-6161.1996.tb02080.x

Yerxa, T., Anderson, A. G., & Weiss, E. (2018). The hyperdimensional stack machine. In *Proceedings of Cognitive Computing* (pp. 1–2). IEEE.

Received October 13, 2022; accepted February 27, 2023.