On Effects of Compression with Hyperdimensional Computing in Distributed Randomized Neural Networks

Antonello Rosato[1], Massimo Panella[1(✉)], Evgeny Osipov[2], and Denis Kleyko[3,4]

[1] University of Rome "La Sapienza", Rome, Italy
{antonello.rosato,massimo.panella}@uniroma1.it
[2] Luleå University of Technology, Luleå, Sweden
evgeny.osipov@ltu.se
[3] Research Institutes of Sweden, Kista, Sweden
denis.kleyko@ri.se
[4] University of California, Berkeley, USA

Abstract. A change of the prevalent supervised learning techniques is foreseeable in the near future: from the complex, computational expensive algorithms to more flexible and elementary training ones. The strong revitalization of randomized algorithms can be framed in this prospect steering. We recently proposed a model for distributed classification based on randomized neural networks and hyperdimensional computing, which takes into account cost of information exchange between agents using compression. The use of compression is important as it addresses the issues related to the communication bottleneck, however, the original approach is rigid in the way the compression is used. Therefore, in this work, we propose a more flexible approach to compression and compare it to conventional compression algorithms, dimensionality reduction, and quantization techniques.

Keywords: Distributed randomized neural networks · Compression · Vector symbolic architectures · Hyperdimensional computing

1 Introduction

In this work, we are exploring the use of compression in a framework for distributed classification. The main motivation of this work is the optimization of the distributed machine learning algorithm when computational and communication costs are to be preserved as is the case in, e.g., resource-constrained scenarios (e.g., edge machine learning [43], smart sensing and privacy-preserving algorithms [8]). Broadly, the objective for compression can be formulated as

finding the best trade-off between classification performance and the communication overhead, which is crucial in the distributed scenario, instead of adopting direct methods for data compression on specific frameworks, as typical for multimedia applications [34]. In this scenario, and generally in distributed solutions, the communication bottleneck is often overlooked, since the tendency is to focus on efficient algorithmic choices and implementations improving classification performance. We, however, aim at developing a flexible solution, which allows controlling the trade-off between classification performance and reduction of the communication overheads. In [36] we presented a novel concept for distributed learning based on a class of randomized neural networks known as Random Vector Functional Link (RVFL) networks [14,16,37] using framework of hyperdimensional computing also known as vector symbolic architectures (HDC/VSA) [10,11,20,32,33].

The main contribution of this article is the in-depth empirical evaluation of the effect of compressing information shared between agents participating in the distributed learning. We present the trade-offs between the accuracy of the neural model and various compression approaches as means of reducing the risk for the communication bottleneck. These results further extend the functionality of our distributed RVFL learning solution with a generalized approach for controlling the compression ratio under different communication bottleneck conditions.

The experiments reported in this paper were done with two approaches to forming network's classifier: regularized least squares (RLS) and centroids. Though, other alternatives are available [1,5,22,38]. The activations of the hidden layer of the network were formed according to a recently proposed version of RVFL network [25], which simplifies the conventional architecture [16] using some of the HDC/VSA principles [24,27].

When studying the effect of compressing the classifier, we focused on analysing our inherently lossy compression approach in a range of compression ratios, along with the conventional, entropic lossless compression. It was also important to contrast the proposed approach with some other conventional lossy approaches. However, we have not identified such an approach, which would compress generic numeric matrices (i.e., classifiers) without assuming a particular data modality (e.g., images) since for such data lossy compression is well performed based on perceptual (e.g., visual) models. However, a lossy compression of the matrix containing the classifier can be done with the conventional methods for dimensionality reduction. In particular, we implemented a lossy approach, which relies on the eigenvectors and eigenvalues obtained from the Singular Value Decomposition (SVD) of the classifier. The approach uses the fact that some of the eigenvalues with the smallest values and the corresponding eigenvectors can be ignored when reconstructing the classifier.

2   Compression Algorithms

As already stated, the main purpose of the compression in the considered distributed scenario is in reducing the communication overhead, thus, improving

the feasibility and applicability of the distributed classification. We are particularly interested in the lossy compression, which might results in decreased classification performance compared to the uncompressed solution, since it should allow controlling the trade-off between the compression ratio and classification performance losses.

To better understand the experiments reported hereafter, in which we assess and analyse the performance of the proposed approach, it is important to clarify the parts of the model involved in the compression. The network consists of connected agents (we assume full connectivity in our experiments); each agent has its own training dataset. We form agents' datasets by equally splitting samples of the whole dataset between agents without replacement. So each agent has the access to a subset of samples of the whole dataset with all the features. Each agent trains its own local model with its portion of the dataset and shares information about its own classifier (denoted as $\mathbf{W}^{\text{out}}$) with all other agents (due to the full connectivity), thus, the training data are not actually shared but kept private. Additionally, in this work, before sharing values of its local classifier with others, each agent compresses it. At the receiving end, each agent decompresses the data it received and incorporates them in the aggregated model to be able to create a more powerful classifier.

Below, we describe three compression approaches used herein highlighting their differences. Classification performance obtained with these approaches is reported and discussed in Sect. 3.2.

2.1   Compression Based on Conventional Algorithms

We describe hereafter the two compression approaches used to compare with the proposed approach. As already stated, they are general and, although, implemented specifically for the purpose of the presented distributed classification, they only make use of well-known conventional algorithms.

Lossless Compression. The most common and effective way to perform compression on a matrix is using one of the many conventional lossless compression algorithms readily available and widely assessed. We opted for the ones in the `zlib` software library for data compression, which use the public domain ZLIB Deflator algorithm. The algorithm uses the Deflate file format, employing a combination of LZSS and Huffman coding (see [4] for more details). In this work, the lossless compression is given as a baseline of how strong is the gain of the proposed approach in terms of information compression. Since it is impractical from the application point of view, given that the operations needed to perform it at the local nodes are computationally expensive with respect to the actual classification model. Moreover, when using the lossless compression, the classification performance of the model is the same as in the uncompressed case but the compression ratio is fixed to the one obtained by the compression algorithm.

Lossy Compression. Lossy quantization of a data matrix is well performed in the case of audio and video applications, using specific techniques. Such techniques are not of interest herein since they are based on auditory and visual perceptual models that are not applicable in our scenario. Nevertheless, one way to decrease the size of the classifier is by quantizing matrix values to some fixed number of quantization levels. We will discuss this approach in Sect. 4.3.

In the absence of quantization, dimensionality reduction methods can serve for the purpose of compression. In particular, we use the eigenvectors and eigenvalues obtained from the SVD and the compression is based on the fact that eigenvalues with smaller, negligible norm can be discarded.

In practice, we carry out the SVD-based compression by first transforming the classifier $\mathbf{W}^{\text{out}}$ with the SVD. Given that the values of $\mathbf{W}^{\text{out}}$ are real, this results in the well-known factorization: $\mathbf{U\Sigma V}$. Given the properties of the SVD, if the eigenvalues and both the left and right eigenvectors are transmitted as is, another agent is able to losslessly reconstruct the original matrix. Of course, this is completely impractical for compression and sharing, since the dimension of the transmitted information in the form of the three matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ is much larger than the original classifier $\mathbf{W}^{\text{out}}$.

For the compression purpose, to ensure the reduction of the size of the matrices to be transmitted, each agent selects and transmits only a portion of the SVD matrices, corresponding to a number t of the largest eigenvalues, resulting in what is often called a "truncated SVD" ($\mathbf{U}_t$, $\mathbf{\Sigma}_t$ and $\mathbf{V}_t$), as it is used for proper dimensionality reduction [41]. This way, the receiving agent is able to reconstruct a version of the original classifier: $\hat{\mathbf{W}}^{\text{out}} = \mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t$, which is not an exact replica, since some information is lost due to the truncation. To be able to effectively control the lossy compression ratio of the classifier, we reshape $\mathbf{W}^{\text{out}}$ into a square matrix (with zero padding if necessary) and then select t eigenvalues based on the desired compression ratio.

2.2   Compression Based on HDC/VSA Principles

In our previous work [36], we proposed how to use the principles of HDC/VSA to compress the classifier $\mathbf{W}^{\text{out}}$. Here we recapitulate it. The key idea is that before being shared with the other agents a locally computed version of $\mathbf{W}^{\text{out}(p)}$ can be compressed into a single D-dimensional vector (hypervector, denoted as $\mathbf{w}$) For the proposed approach we use Holographic Reduced Representations model [32].

The compression procedure uses the idea of key-value pair representations in HDC/VSA[1]. In the compression context, a key hypervector is generated randomly where a value hypervector contains some of the values from $\mathbf{W}^{\text{out}(p)}$. The hypervector of a key-value pair is formed via the binding operation. In the Holographic Reduced Representations model, the binding operation is realized via

---

[1] This is possible since HDC/VSA provide primitives for representing structured data in hypervectors such as sequences [9,12,20,40], sets [20], [29], state automata [44], [31], hierarchies, or predicate relations [32], [33]. Please consult [23] for a general overview.

the circular convolution (denoted as ⊛), which is formulated through the outer product of the hypervectors being bound. The value of the jth component of $\mathbf{z}$ is calculated as:

$$\mathbf{z}_j = \sum_{k=0}^{D-1} \mathbf{y}_k \mathbf{x}_{j-k \mod D}$$

In [36], it was proposed to form L key-value pairs where L denotes the number of classes in the task. The disadvantage of this proposal is that the compression ratio is fixed to L to so the procedure is inflexible. Here we show that the procedure can be simply modified so that the compression ratio can take any positive integer value. To do so, for the chosen compression ratio (denoted as R), we first calculate the number of dimensions D in $\mathbf{w}$ as: $D = \lceil HL/R \rceil$, where H denotes the size of the hidden layer. Then $\mathbf{W}^{\text{out}}$ should be reshaped (denoted as $\mathbf{S}$) such that is has D rows and R columns. Note that some zero padding might be necessary when HL is not a multiple of R. Next, R random hypervectors (denoted as $\mathbf{K}_i$) are generated. They act as keys. Each column in $\mathbf{S}$ is then bound with the corresponding key: $\mathbf{K}_i \circledast \mathbf{S}_i$. Finally, these key-value hypervectors are used to form the compressed version of the classifier:

$$\mathbf{w} = \sum_{i=1}^{R} \mathbf{K}_i \circledast \mathbf{S}_i;$$

$\mathbf{w}$ can be shared with other agents in an attempt to improve the classification.

When the agent p receives $\mathbf{w}$ from its neighbour q it needs to decompress $\mathbf{W}^{\text{out}(q)}$ from $\mathbf{w}$. The decompression is done for each $\mathbf{S}_i^{(q)}$ using the inverses of the corresponding key hypervectors of q (see [32] for the details) as:

$$\hat{\mathbf{S}}_i^{(q)} \approx \mathbf{w} \circledast \mathbf{K}_i^{-1}.$$

Finally, $\hat{\mathbf{S}}^{(q)}$ has to be reshaped to get $\hat{\mathbf{W}}^{\text{out}(q)}$. Note that the reconstructed classifier $\hat{\mathbf{W}}^{\text{out}(q)}$ will not be the exact replica of the original one. This is explained by the fact that the superposition operation is lossy in a sense that hypervectors of other key-value pairs add their crosstalk noise to the resultant hypervector [9,30]. It was, however, shown in [36] that when many reconstructed classifiers are combined, the noise will average out without affecting much the classification performance. At the same time, it is expected that the compression ratio R will regulate how lossy is the compression and, therefore, it will affect the classification performance. The effect of R on the classification performance is shown experimentally in Sect. 3.2.

3   Experiments

3.1   Data and Setup

The experiments reported in this paper were performed on a collection of 18 real-world classifications datasets from the UCI Machine Learning Repository [6]. The
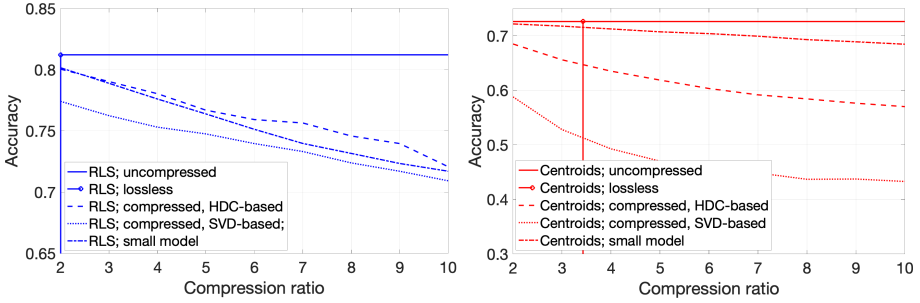
Fig. 1. The average accuracy on the datasets against different compression ratios in the proposed procedure. The number of agents was set to 10.

datasets are a subset of a larger collection used in the seminal work [7]. The only addition, which was introduced to the preprocessing, was the normalization of input features to $[0, 1]$ range as this is necessary to form the thermometer codes used in the model. In the distributed scenario, it is important that each agent gets some training samples. Since we limited ourselves to splitting training data between agents without the replacement, only datasets with more than $1,000$ samples in the training part were considered. Another requirement was the size of the hidden layer, H. In order to be able to make a comparison between compressed classifiers and smaller models of the corresponding size, only the datasets where the optimal H was more than $1,300$ were picked. Only 18 datasets in the collection satisfied both requirements[2].

The optimal hyperparameters of the models (H, λ, and κ) for each dataset were obtained with the grid search using the same steps as in [7] using the centralized scenario with the RLS classifier. H varied in the range $[50, 1500]$ with step 50; λ varied in the range $2^{[-10,5]}$ with step 1; and κ varied between $\{1, 3, 7, 15\}$. The chosen values were used for both classifiers and in all the experiments reported below. To avoid the influence of a particular random initialization on the performance, all results reported below were averaged for 10 random initializations of $\mathbf{W}^{in}$, which is the input projection matrix.

3.2   Results

For the lossless compression, the results were analysed only in terms of a single compression ratio. This is straightforward because, being lossless, no information is missing at the decompression receiving end of the network, and each agent can retrieve the exact copy of the classifier sent by its neighbours. Thus, in Fig. 1 for

---

[2] The names of the datasets were: Abalone, Cardiotocography (10 classes) Chess (King-Rook vs. King-Pawn), Letter, Oocytes merluccius nucleus 4D, Oocytes merluccius states 2F, Pendigits, Plant margin, Plant shape, Plant texture, Ringnorm, Semeion, Spambase, Statlog landsat, Steel plates, Waveform, Waveform noise, Yeast.
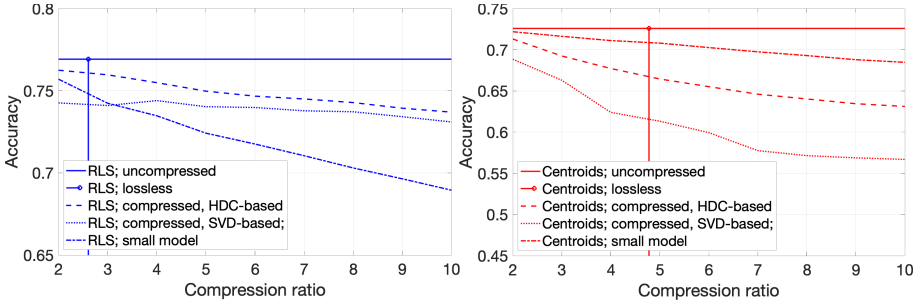
Fig. 2. The average accuracy on the datasets against different compression ratios in the proposed procedure. The number of agents was set to 100.

$N = 10$ agents and Fig. 2 for $N = 100$ agents, the achieved compression ratio for the lossless algorithm is depicted by bars. Naturally, the accuracies were equal to the corresponding uncompressed versions of the classifiers. We stress the fact that this case has only the effect of making the lossy results be examined in terms of compression ratio, giving us the comparison with a compression procedure optimized for much larger models and agents with superior computational power.

The advantage of the lossless compression is that it preserves all of the accuracy but there is no way to control its compression ratio. The proposed compression procedure (Sect. 2.2) and the SVD-based one (Sect. 2.1) are lossy but they allow varying the compression ratio.

In order to investigate the effect of the compression ratio on the accuracy, we performed the experiments in the distributed scenario with both procedures for both classifiers. Figure 1 presents the results of the experiments for the case when the number of agents was set $N = 10$ while Fig. 2 reports the results for $N = 100$. As the baseline, the figures depict the accuracies of the distributed scenario when no compression was evolved (solid lines). For both classifiers, this baseline acted as an upper bound as losses introduced to the classifiers during the decompression incurred decreased classification performance. Obviously, the accuracy of both classifiers was getting worse with the increased compression ratio but given the same ratio, the RLS classifier always outperformed the centroids one.

Note that an approach alternative to the compression would be to train a smaller model and share it with other agents without involving any compression. Therefore, for each compression ratio we trained a model with smaller hidden layer (denoted as "small model" in the figure) such that the size of smaller model's classifier would equal to the compressed classifier of the full-size model. In the case of the centroids classifier for both N, small models (dash-dotted line) performed much closer to the baseline than the compressed centroids. The explanation is likely rooted in the fact that to have a fair comparison in the experiments we used the size of the hidden layer optimized for the RLS classifier. Since the centroids classifier is much simpler it is likely that it would need smaller
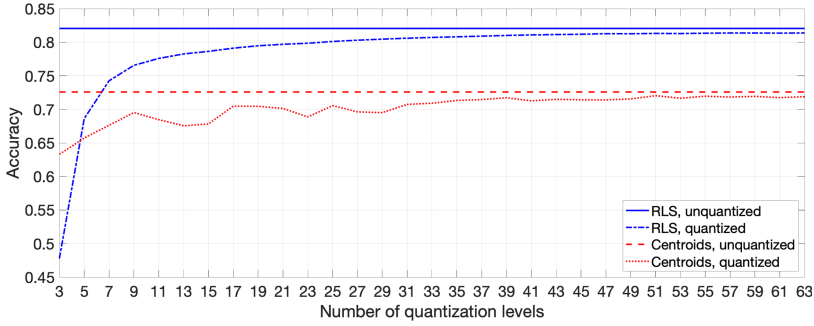
Fig. 3. The average accuracy against different number of quantization levels for $\mathbf{W}^{\mathrm{out}}$.

size of the hidden layer to get most of its classification performance and so its small models were pretty close to the baseline. In contrast, for the RLS classifier and the proposed approach when $N = 10$, for small compression ratios (2 or 3) the compressed classifier and small model performed on a par. However, when the compression ratio was increasing, the compressed classifier was performing better than the corresponding small model. In the case of $N = 100$, the proposed approach was noticeably better than the small models. That is expected because in the proposed approach noise introduced to the decompressed classifiers gets averaged out and the more classifiers get aggregated the better it is for mitigating the noise.

With respect to the SVD-based compression, for both values of N, the trend of results in terms of accuracy/compression ratio trade-off was as expected. In particular, the mean values follow the same trend as for the proposed approach, with the difference that the achieved accuracy was always lower. This can be likely caused by the fact that even least significant eigenvectors bear important information about the classifier so when truncated, the decompressed classifier misses important information. Results indicated that SVD-based compression, while remaining a solid dimensionality reduction method, was inferior in terms of accuracy/compression ratio trade-off compared to the proposed approach.

4   Discussion

4.1   Results

The main point is that, when choosing the compression approach, it is important to consider the burden of the cost of performing the compression procedure for each local agent, given that the studied solution should be deployed on resource-constrained devices. To this end, we note that the proposed compression approach based on HDC/VSA is the most efficient one, being tailored for such devices, while, to a greater extent, the SVD and the lossless ones can be considered as having a lower efficiency but the exact comparison in terms of computational costs is outside the scope of this paper.

4.2 Relation to Other Areas

The proposed approach hinges on principles and premises, which are studied in the federated learning framework [19,42]. Namely, the availability of raw data only at local agents (i.e., "siloed data") and the impossibility of sharing such raw data are a standard premise in the federated learning. In fact, depending on the definition used, there are certain aspects which might differ; in particular, in our work, there is an absence of a "master" agent orchestrating the training, which is often present in the federated learning. This fundamental distinction makes our work challenging in several aspects (computation, combination) that are specific to the fully distributed scenarios. Thus, while this work could be considered as a particular case of the federated learning, our work has a significance and purpose by itself and should be studied as a distinct field, given also the different practical application. Similarly, there are previous studies in HDC/VSA domain [17,28], which investigated the distributed scenario but all of them assumed some elements of centralization.

It also worth noting that while the experiments reported here were done with shallow randomized neural networks, the proposed approach is applicable to deep randomized neural networks [2], which recently gained quite a lot of attraction. This is so since the same principle can be used – only information about the trainable part of the model (i.e., classifier) should be exchanged by the agents, while the randomly chosen part of the network can be the same for every agent. In principle, the proposed compression approach could be used even for fully trained neural networks; there is, however, a risk that decompression losses at earlier layers might results in additional errors at the later layers. Nevertheless, it is worth clarifying this in the experiments.

The proposed compression approach is conceptually related to the recent idea of using the binding and superposition operations of HDC/VSA to represent parameters of many deep neural networks in a single hypervector [3,13,45]. The difference with the proposed approach is that the classifier is decompressed back to its original shape, which was not the case in [3]. The attempts to apply HDC/VSA in the communication domain [18,21,26,39] are also of relevance but there the goal would usually be to extract the data back from the hypervector without any losses, which is not the case in this work.

4.3 Quantization of the Classifier

In the previous experiments, the weights in $\mathbf{W}^{\text{out}}$ being compressed were real-valued since the proposed compression approach is real-valued as well. It is, however, known that neural networks can perform well even when the weights are limited to a few quantization levels [15,35]. Therefore, we did an experiment to demonstrate what to potentially expect for the quantization of $\mathbf{W}^{\text{out}}$. For the sake of simplicity, the experiment was done using the centralized scenario so $N = 1$. The goal of this experiment was to explore whether it is worth considering quantizing the classifier before the compression.

Figure 3 presents the average accuracy of the RLS (dash-dotted line) and centroids (dotted line) classifiers (uncompressed) on all 18 datasets against the number quantization levels being used. The corresponding accuracies (solid and dashed lines) from the unquantized real-valued $\mathbf{W}^{out}$ were used as the baselines. When $\mathbf{W}^{out}$ of the RLS classifier was quantized to very few levels (3 or 5) the accuracy was affected significantly. It is, however, clear that for the increased number of levels the accuracy of both classifier started to reach their unquantized baselines. The results, thus, suggest that allocating one byte per weight should preserve most of the classification performance. Note that in the proposed approach, $\mathbf{w}$ with the compressed version of $\mathbf{W}^{out}$ can also be represented with fewer than 32 bits per weight. It is expected that decreasing the precision of $\mathbf{w}$ will just add some additional noise during the reconstruction. Obtaining the quantitative results to characterize the effect of this noise will be a part of the future extension of this work.

5   Conclusions

We presented a study exploring the effect of compression on classification performance in a distributed classification scenario. The main goal was to explore the proposed compression approach, which is based on HDC/VSA principles. By examining the numerical results obtained on a collection of 18 datasets for three compression approaches, it is concluded that the proposed approach has a favourable trade-off in terms of accuracy and communication costs. Also, the finding that small models perform well or on a par with respect to other schemes is worth exploring, and can be considered as enriching the presently relevant discussion regarding the convenience of training large, deep models. Further, future studies should investigate the sensitivity of the overall performance of the proposed approach with respect to residual coding schemes of matrix coefficients, similarly to what is performed in linear predictive coding and adaptive coding of spectral coefficients, as well as considering some parsimonious model representation strategies as, for instance, Minimum Description Length (MDL) and Bayesian Information Criterion (BIC).

References

1. Alonso, P., Shridhar, K., Kleyko, D., Osipov, E., Liwicki, M.: HyperEmbed: Hyper-Embed: Trade-offs Between Resources and Performance in NLP Tasks with Hyperdimensional Computing enabled Embedding of n-gram Statisticss. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–9 (2021)
2. Ben-Nun, T., Hoefler, T.: Demystifying parallel and distributed deep learning: an in-depth concurrency analysis. ACM Comput. Surv. **52**(4), 1–43 (2019)
3. Cheung, B., Terekhov, A., Chen, Y., Agrawal, P., Olshausen, B.A.: Superposition of Many Models into One. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 10868–10877 (2019). https://papers.nips.cc/paper/2019/file/4c7a167bb329bd92580a99ce422d6fa6-Paper.pdf

4. Deutsch, P., Gailly, J.L.: RFC1950: ZLIB Compressed Data Format Specification Version, vol. 3, p. 3 (1996)
5. Diao, C., Kleyko, D., Rabaey, J.M., Olshausen, B.A.: Generalized Learning Vector Quantization for Classification in Randomized Neural Networks and Hyperdimensional Computing. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–9 (2021)
6. Dua, D., Graff, C.: UCI Machine Learning Repository (2017). http://archive.ics.uci.edu/ml
7. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(90), 3133–3181 (2014)
8. Fierimonte, R., Scardapane, S., Uncini, A., Panella, M.: Fully decentralized semisupervised learning via privacy-preserving matrix completion. IEEE Trans. Neural Netw. Learn. Syst. **28**(11), 2699–2711 (2017)
9. Frady, E.P., Kleyko, D., Sommer, F.T.: A theory of sequence indexing and working memory in recurrent neural networks. Neural Comput. **30**, 1449–1513 (2018)
10. Frady, E.P., Kleyko, D., Sommer, F.T.: Variable binding for sparse distributed representations: theory and applications. arXiv:2009.06734, pp. 1–16 (2020)
11. Gayler, R.W.: Multiplicative binding, representation operators and analogy. In: Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences, pp. 1–4 (1998)
12. Hannagan, T., Dupoux, E., Christophe, A.: Holographic string encoding. Cogn. Sci. **35**(1), 79–118 (2011)
13. Hersche, M., Rupp, P., Benini, L., Rahimi, A.: Compressing subject-specific braincomputer interface models into one model by superposition in hyperdimensional space. In: 2020 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 246–251. IEEE (2020)
14. Huang, G., Zhu, Q., Siew, C.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
15. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: training neural networks with low precision weights and activations. J. Mach. Learn. Res. **18**, 1–30 (2018)
16. Igelnik, B., Pao, Y.: Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans. Neural Netw. **6**, 1320–1329 (1995)
17. Imani, M., et al.: A framework for collaborative learning in secure highdimensional space. In: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), pp. 435-446. IEEE (2019)
18. Jakimovski, P., Schmidtke, H.R., Sigg, S., Chaves, L.W.F., Beigl, M.: Collective communication for dense sensing environments. J. Ambient Intell. Smart Environ. **4**(2), 123–134 (2012)
19. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., et al.: Advances and Open Problems in Federated Learning. arXiv:1912.04977, pp. 1–121 (2019)
20. Kanerva, P.: Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. Cogn. Comput. **1**(2), 139–159 (2009)
21. Kim, H.: HDM: Hyper-dimensional modulation for robust low-power communications. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2018)

22. Kim, Y., Imani, M., Rosing, T.S.: Efficient human activity recognition using hyper-dimensional computing. In: Proceedings of the 8th International Conference on the Internet of Things, pp. 1–6 (2018)
23. Kleyko, D., et al.: Vector Symbolic Architectures as a Computing Framework for Nanoscale Hardware. arXiv:2106.05268, pp. 1–28 (2021)
24. Kleyko, D., Frady, E.P., Kheffache, M., Osipov, E.: Integer echo state networks: efficient reservoir computing for digital hardware. IEEE Trans. Neural Netw. Learn. Syst. **PP(99)**, 1–14 (2020)
25. Kleyko, D., Kheffache, M., Frady, E.P., Wiklund, U., Osipov, E.: Density encoding enables resource-efficient randomly connected neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3777–3783 (2021)
26. Kleyko, D., Lyamin, N., Osipov, E., Riliskis, L.: Dependable MAC layer architecture based on holographic data representation using hyper-dimensional binary spatter codes. In: Multiple Access Communications (MACOM). Lecture Notes in Computer Science, vol. 7642, pp. 134–145 (2012)
27. Kleyko, D., Osipov, E., De Silva, D., Wiklund, U., Alahakoon, D.: Integer self-organizing maps for digital hardware. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
28. Kleyko, D., Osipov, E., Papakonstantinou, N., Vyatkin, V.: Hyperdimensional computing in industrial systems: the use-case of distributed fault isolation in a power plant. IEEE Access **6**, 30766–30777 (2018)
29. Kleyko, D., Rahimi, A., Gayler, R.W., Osipov, E.: Autoscaling bloom filter: controlling trade-off between true and false positives. Neural Comput. Appl. **32**, 3675–3684 (2020)
30. Kleyko, D., Rosato, A., Frady, E.P., Panella, M., Sommer, F.T.: Perceptron Theory for Predicting the Accuracy of Neural Networks. arXiv:2012.07881, pp. 1–12 (2020)
31. Osipov, E., Kleyko, D., Legalov, A.: Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing. In: IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 3276-3281. IEEE (2017)
32. Plate, T.A.: Holographic Reduced Representations: Distributed Representation for Cognitive Structures. Center for the Study of Language and Information (CSLI), Stanford (2003)
33. Rachkovskij, D.A.: Representation and processing of structures with binary sparse distributed codes. IEEE Trans. Knowl. Data Eng. **3**(2), 261–276 (2001)
34. Rizzi, A., Buccino, N.M., Panella, M., Uncini, A.: Genre classification of compressed audio data. In: 2008 IEEE 10th Workshop on Multimedia Signal Processing, pp. 654–659. IEEE (2008)
35. Rosato, A., Altilio, R., Panella, M.: Finite precision implementation of random vector functional-link networks. In: 2017 22nd International Conference on Digital Signal Processing (DSP), pp. 1–5. IEEE (2017)
36. Rosato, A., Panella, M., Kleyko, D.: Hyperdimensional Computing for Efficient Distributed Classification with Randomized Neural Networks. In: International Joint Conference on Neural Networks(IJCNN), pp. 1–10 (2021)
37. Scardapane, S., Wang, D.: Randomness in neural networks: an overview. Data Min. Knowl. Disc. **7**(2), 1–18 (2017)
38. Shridhar, K., Jain, H., Agarwal, A., Kleyko, D.: End to end binarized neural networks for text classification. In: Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pp. 29–34 (2020)

39. Simpkin, C., Taylor, I., Bent, G.A., de Mel, G., Rallapalli, S., Ma, L., Srivatsa, M.: Constructing distributed time-critical applications using cognitive enabled services. Futur. Gener. Comput. Syst. **100**, 70–85 (2019)
40. Thomas, A., Dasgupta, S., Rosing, T.: Theoretical Foundations of Hyperdimensional Computing. arXiv:2010.07426, pp. 1–32 (2020)
41. Xu, P.: Truncated SVD methods for discrete linear Ill-posed problems. Geophys. J. Int. **135**(2), 505–514 (1998)
42. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)
43. Yazici, M.T., Basurra, S., Gaber, M.M.: Edge machine learning: enabling smart internet of things applications. Big Data Cogn. Comput. **2**(3), 1–17 (2018)
44. Yerxa, T., Anderson, A., Weiss, E.: The hyperdimensional stack machine. In: Cognitive Computing, pp. 1–2 (2018)
45. Zeman, M., Osipov, E., Bosnic, Z.: Compressed Superposition of Neural Networks for Deep Learning in Edge Computing. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2021)