Positional Binding with Distributed Representations

Stephen I. Gallant, Phil Culliton
MultiModel Research
Cambridge, MA, USA
e-mail: {sgallant, pculliton}@mmres.com

Abstract—**Positional binding specifies feature positions for an image (or for text). We show how to incorporate position into a fully distributed vector formed from Vector Quantization, or add position to a vector formed from a Vector Symbolic Architecture. The method guarantees that small shifts in position result in small changes to the representation vector, and does not require an increase in vector size. The incorporation of positional binding improves performance on CIFAR-10 and on a new database of noisy abstract face images, which we hereby make public. For Deep Learning approaches, we emphasize the importance of positional binding, and this sheds light on why multiple layers and pooling are beneficial.**

Keywords-representation; distributed representation; vector symbolic archichetecture; image representation; binding

I. INTRODUCTION

Binding is the association of objects and attributes. Within images, we may want to associate a low-level feature's shape, orientation, color and motion. For text, we need to associate adjectives with the particular noun they modify, and associate the words in a subject, verb, or other clause.

Positional binding for images is the association of bound groups to positions within the image. We need to know whether a feature is in the upper-left of the image or the lower-right. Similarly, for text, we want to represent the order of paragraphs in a news story or the order of clauses in a sentence. In all cases, we want positional binding to be robust with respect to small changes in location, so that if we change feature positions by a small amount then this results in a small change to the resulting representation. This is important in helping machine learning generalize over similar data.

Here we examine the role of positional binding for two types of representations for learning.

Constructive representations start with vector representations of low-level objects, for example, image patches after vector quantization with a coding dictionary or words in sentences. We then build a representation for the text or image in a bottom-up fashion. Two examples are Coates and associates' Vector Quantization systems [1], [2], and Vector Symbolic Architectures (VSAs) [3]-[9]. For Constructive representations, positional binding must be explicitly defined and added to the representational architecture. Up to now, this has not been possible for fully distributed, robust representations, other than by concatenating vectors for different regions. Constructive representations are appealing because they result in a single fixed-length vector. Such vectors are well-suited for machine learning approaches (e.g. ridge regression, support vector machines, etc.) which are orders of magnitude faster than deep learning. Furthermore, algorithm convergence is no longer an issue.

Deep Learning architectures, by contrast, learn the vector representations for objects in a top-down fashion. Here we will argue that positional binding is an important consequence of the layer and pooling structure for these architectures.

The following section looks at Constructive representations, presenting a way to accomplish positional binding with these architectures. Section 3 examines Deep Learning architectures in light of positional binding, and suggests a simple experiment to show its importance. Section 4 describes simulations, and introduces a new public database of noisy faces and non-faces. Section 5 gives some additional previous work, and we conclude with a discussion.

II. POSITIONAL BINDING WITH CONSTRUCTIVE REPRESENTATIONS

Coates and associates [1], [2] investigated a number of Vector Quantization (VQ) methods for images that convert an image patch (small square sub-region) to a vector using a coding dictionary. To do this, we flatten the patch into a vector by concatenating rows (or columns), than pre-multiply this vector by a fixed matrix (the Coding Dictionary), and apply a non-linear transformation element-wise to produce the coded patch vector. (This is a simplified description which omits some additional helpful image processing details, such as normalizations and whitening.)

Coates et al. showed good image recognition results with several orders of magnitude less computation than Deep Learning methods that require gradient descent through a network. They also demonstrated that good results can be obtained where the coding matrix rows consist of randomly selected patches or, with better results, k-means clustering applied to patches. This confirms previous findings by Jarret et al. [10].

Each row of the coding matrix is either a k-means cluster center or a randomly chosen patch exemplar. Therefore, intuitively, we are multiplying the coding matrix and a patch to obtain a vector where each component represents the similarity of the corresponding coding matrix row with the patch. Note also that flattening a patch vector and

multiplying by a coding matrix is the same computation as applying several fixed convolution kernels, one for each row of the coding matrix.

Before applying supervised learning algorithms, Coates et al. pool the resulting sets of vectors into 4 quadrants by adding (or taking the max of) the corresponding components of vectors in each quadrant, and then concatenating the 4 resulting vectors to obtain the vector for the image. This reduces the dimension of the resulting representation from approximately the number of pixels times the vector size (number of rows in the coding matrix) to 4 times the vector size.

In addition to dimensionality reduction, this pooling by quadrants serves another purpose: positional binding. The resulting representation separates the image quadrant locations of features given by VQ yet preserves some robustness of position, because shifting any feature within its quadrant produces little or no change to the resulting vector. This simple method of positional binding proves quite effective for many applications. However, there is an issue with shifting an image part across either the horizontal or the vertical center lines, because this produces very different final vectors after the concatenation of the 4 quadrant vectors.

Another type of Constructive representation is Vector Symbolic Architectures (VSAs). Here we make reference to the MBAT system (Matrix Binding of Additive Terms) of Gallant & Okaywe [9] which primarily focuses upon 1-dimensional sentence structures. To produce a distributed vector for a sentence that includes parse information, we used a fixed set of "binding matrices" that correspond to syntactic groupings. Example matrices, with simplified notation, are $^{\text{subject-phrase}}\mathbf{M}$, $^{\text{verb-phrase}}\mathbf{M}$, $^{\text{object-phrase}}\mathbf{M}$. Individual words have their own vectors, which are generated so that similar words have similar vectors [11] [12]. Then the vector for a phrase consists of multiplying the corresponding matrix with the sum of the term vectors in the phrase. For example, to represent a subject clause "*The graceful yellow elephant*", we start with vectors corresponding to the words. By convention, we write **elephant** (in bold) to represent the *vector* for "*elephant"*. Then the vector for the clause is

<subject-clause> = $^{\text{subject-clause}}\mathbf{M}$ **(the + graceful + yellow + elephant).**

The resulting vector may recursively participate in larger structures. Thus the sentence vector might be

$^{\text{sentence}}\mathbf{M}$ **(**<subject-clause> + <verb-clause> + <object-clause>**).**

Gallant & Okaywe showed that MBAT permits learning algorithms to be sensitive to binding. For example, it is easy to learn cases where the target function depends upon a particular word appearing in the subject clause, but not in the object clause. Moreover, there is great flexibility in the choice of binding matrices, which can even be chosen as random matrices.

In general, VSA's have no positional binding; the best they can do "out-of-the-box" is to use one-dimensional sequence representations. For MBAT (similarly for other VSAs) this means multiplying the vector at each position in a sequence by a matrix raised to the corresponding power, and summing the resulting vectors:

$\sum \mathbf{M}^{(\text{position\_number})}$ <vector for this position>.

Using sequence representations to represent positions satisfies the VSA requirement of producing a fixed-length vector from sequences of different lengths. However, it has the drawback of not yielding similar vectors for consecutive positions. Note that positional binding is important even for the 1-dimensional case where we want to capture the order of text paragraphs within a (single) document vector in a robust way.

A. MOMBAT Positional Representations

We can extend MBAT to achieve positional binding for Constructive representations, including Vector Quantization representations.

To motivate our approach, note that a way to encode a (1-dimensional) sequence would be to use different matrices for each position and sum them:

$$\mathbf{M}(\mathbf{V}^1 + \mathbf{V}^2 + \mathbf{V}^3) + \mathbf{N}(\ldots) + \mathbf{O}(\ldots) + \ldots$$

For example, if we want to encode positions of paragraph vectors in a document, we can let M be the Binding Matrix for the first paragraph, N be the matrix for the second paragraph, etc. Now if nearby paragraphs have similar Binding Matrices, this will achieve robust positional encoding due to continuity and linearity with respect to the matrices in matrix multiplication.

One easy way to generate such positional binding matrices is to start with two matrices, **K** and **L**, possibly randomly generated.[1] Then for position indicator $0 \leq \alpha \leq 1$ define

$$^{\alpha}\mathbf{M} = \alpha\mathbf{K} + (1-\alpha)\,\mathbf{L}$$

Then $^{\alpha}\mathbf{M}$ (V1 + V2) and $^{\beta}\mathbf{M}$ (V1 + V2) will be more similar when position α is close to β than if they are more separated. Thus $^{\alpha}\mathbf{M}$ achieves positional binding in one dimension, in a way that is robust to small shifts in position.

We can think of $^{\alpha}\mathbf{M}$ as a matrix moving over a sequence of positions, and this motivates the acronym *MOMBAT* for *MOving Matrix Binding of Additive Terms*.

How is α set? If there are 5 items in a sequence, we can set for the 5 matrices

$$\alpha = \{0/4, 1/4, 2/4, 3/4, 4/4\}.$$

Thus neighboring positions will have similar binding matrices, but not the first and last positions.

[1] For simulations, these matrices have Gaussian random components, with mean 0 and standard deviation of 1. We tried using different random seeds to generate matrices, and this did not make a difference in performance.

Further, we can control the similarity of the sequence with a hyper-parameter S. We represent the position in a sequence of length P+1 by i, where i = 0, 1, …, P, and then we define

$$\alpha_{i,S} = (i + S) / (P + 2S)$$

As S gets larger, position matrices become more similar to each other, because $\boldsymbol{\alpha}$ converges to ½ for all positions.

B. Image Binding in Two Dimensions

For images, we can also do positional binding in two (or more) dimensions. We define two pairs of matrices, (K, L) and (Q, R), and set ${}^{\alpha}M$ as before. ${}^{\beta}N$ is similarly defined by

$${}^{\beta}\mathbf{N} = \beta\mathbf{Q} + (1-\beta)\mathbf{R}$$

Finally, we define ${}^{\boldsymbol{\alpha},\boldsymbol{\beta}}\mathbf{T}$, the two-dimensional Binding Matrix for position ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$), by the product of the horizontal and vertical positional matrices

$${}^{\alpha,\beta}\mathbf{T} = {}^{\alpha}\mathbf{M}\ {}^{\beta}\mathbf{N}$$

It is clear that the Binding Matrices for two similar positions are similar matrices. Thus we can easily bind multiple feature groups in an image by using positional Binding Matrices in order to group features at each location.

III. Positional Binding with Deep Learning

Why does Deep Learning use multiple layers, some of which are pooling layers? (*Pooling* takes a collection of vectors and replaces them with a single vector by taking the componentwise sum or max.) In addition to being able to learn more complex functions more easily, various authors give justifications that focus primarily upon dimensionality reduction, especially with respect to pooling [1], [2], [10], [13]. Boureau et al. [14] additionally cite invariance to image transformations, and better robustness to noise and clutter.

Achieving the excellent performance that Deep Learning modeling has shown requires robust positional binding. This is clear from the following "thought experiment."

Take a multi-layer convolutional neural network architecture and, for the lowest layer only, randomly permute the positions in the image for vectors before going into the first pooling layer. (Use the same permutation for all training and testing.) No simulation is needed to know that shifting positions will harm performance, because small image shifts will now drastically alter vectors going into the first pooling layer.[2]

Although many authors cite a primary reason for pooling and multiple layers to be dimensionality reduction, our thought experiment – which does not alter the network's dimensionality reduction – produces degraded results because it prevents positional binding from maintaining topological correspondence between layers.

This suggests that positional binding provides an important lens for viewing Deep Learning architectures with multiple layers, because the layers provide multiple granularities for positional binding.

This is consistent with Graham's view [15] that "Each layer of pooling is an opportunity to view the input image at a different scale. Viewing images at the 'right' scale should make it easier to recognize the tell-tale features that identify an object as belonging to a particular class." Graham also finds [16] that deeply layered networks can give improved results.

IV. Simulations

To see if positional binding techniques can improve performance with constructive image processing models, we explored positional binding with Vector Quantization architectures using two sets of data: a new database of noisy abstract faces, and CIFAR-10, a database of images with 10 classes of images [17].

A. Noisy Abstract Faces

The reason for creating the noisy abstract faces database was to have complete control over images and noise used for experiments. We hereby make publicly available the generation code for this database[3].

For the face data, each position on a 32 X 32 grid has an oval, square, line, or is empty. Each non-empty position has an orientation of horizontal, vertical, or the 2 diagonals.

Images are either a face (50%) or blank (50%). Noise-free faces have 2 eyes (horizontal ovals), a nose (vertical line), and a mouth (horizontal line) as illustrated in the upper-left box of Fig. 1. Actual positions for the features are displaced from noise-free ideal positions by a Gaussian with standard deviation being 1 position horizontally and 1 position vertically. Then every image position has noise added with probability 5%, consisting of a random shape and orientation. For face images, the noise can overwrite existing face features, thereby making 100% identification impossible. See Fig. 1 for examples. Training data consists of 80,000 examples, and test data has 10,000 examples. For all experiments, we used patch size of 3X3. For MOMBAT, we used S = 6 for horizontal and vertical directions.

Table I gives experimental results for predicting noisy faces. The Coates et al. [18] Vector Quantization baseline performance using K-Means to generate coding matrix rows and quad pooling is 88.79%. Adding a MOMBAT layer that encodes position within each quadrant gives 97.90% performance.

When we use a Coates representation with one big pooling of 12,000 dimensions, rather than concatenated quad pooling of 3,000 dimensions, performance drops significantly while keeping the same number of trainable parameters, thereby illustrating the importance of positional binding. Adding MOMBAT to the 12,000 dimensional representation improves results, but not as much as positional binding using quad pooling. Best constructive learning results were adding MOMBAT on each quadrant in quad pooling.

[2] We conducted the simulation anyway, getting an 8% reduction in accuracy using the artificial face data described below.

[3] http://MMRES.com/download/public/artificial_faces

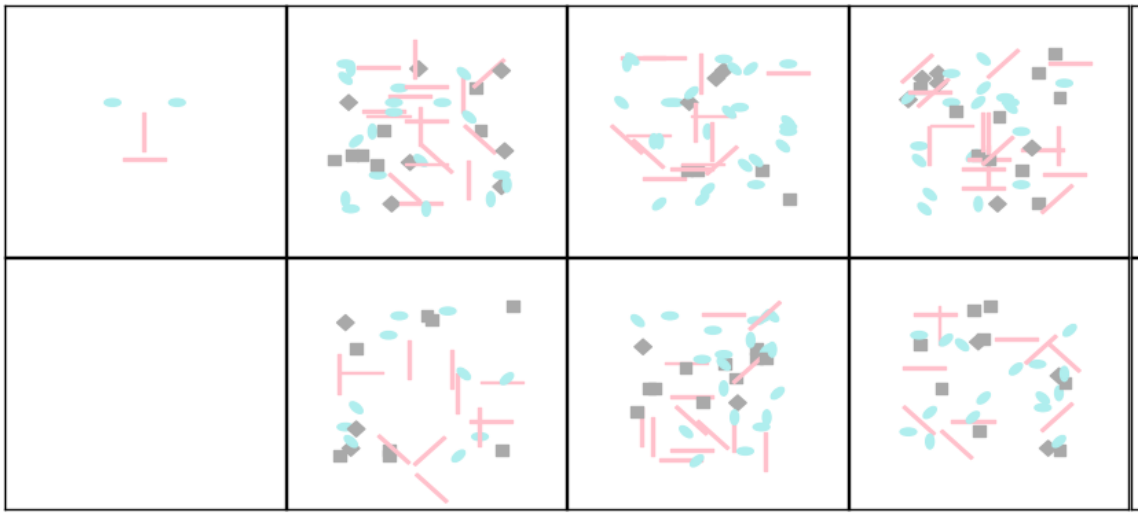

Figure 1. Noisy Abstract Faces. Templates for an ideal face and non-face are in the left column. The remainder of the top row contains noisy face examples, and the bottom row contains noisy non-face examples. This is a visualization of the actual data, which for each position consists of a 2-vector giving the feature number and orientation for each position in a 32 X 32 image.

TABLE I. SIMULATION RESULTS AND TIMES FOR NOISY ARTIFICIAL FACES AND FOR CIFAR-10

Architecture and Positional Binding	Vector Dimension for Training	Pooling	Accuracy on test set	Processing time	CPU or GPU
Artificial Face Results					
Coates [1] with quadrant pooling	4x3000	Sum within quadrants	88.79%	60 min	CPU
Coates with quadrants and MOMBAT	4x3000	Sum within quadrants	97.90%	70 min	CPU
Coates without quadrants	12000	Sum of all patches	73.10%	65 min	CPU
Coates without quadrants, but with MOMBAT	12000	Sum of all patches	76.20%	75 min	CPU
5-layer CNN	4.1 million parameters		98.30%	90 min	GPU
CIFAR-10 results					
Coates [1] with quadrant pooling	4x4000	Sum within quadrants	79.60%	15 min	CPU
Coates with quadrants and MOMBAT	4x4000	Sum within quadrants	81.00%	16 min	CPU
Coates without quadrants	16000	Sum of all patches	77.37%	11 min	CPU
Coates without quadrants, but with MOMBAT	16000	Sum of all patches	78.32%	13 min	CPU
6-layer sparse CNN w/ Dropout (Graham 2014)	29 million parameters		95.53%	70 hours	GPU

For comparison, we also implemented a 5-layer convolutional neural network. CNN performance was 98.30%, and training time was 90 minutes on a GTX 750 GPU rather than 60-75 minutes on a CPU for Constructive methods. (We estimate the GPU has 20 to 40 times the computational power of the CPU in question.)

B. CIFAR-10

For the CIFAR-10 data (Table I), using MOMBAT on top of Coates quadrant pooling again slightly increases performance to 81.00% from 79.60% baseline. Here again, quad pooling gives improved performance over single pooling using the same number of trainable parameters.

We used patch size of 6X6. For MOMBAT, we used S = 50 for horizontal and vertical directions. Each image has 3x6x6 patches extracted and flattened into a set of 729 108-dimensional vectors. Each patch was divided by its standard deviation and whitened using ZCA.

A random set of 200,000 patches was collected and clustered using k-means with k=4000, and a dictionary was created from the resulting centroids, transposed, with each row of the matrix normalized to unit length. All patches from the training and test sets were multiplied by this dictionary to produce 4000-dimensional vectors for each patch, and a soft threshold with an alpha of 0.25 was applied to the resulting vectors.

Each vector was then multiplied by a MOMBAT matrix determined by the patches' positions in the image, and the resulting vectors were pooled by quadrant and summed to create 4x4000 vectors representing the entire image. These were then concatenated to produce a final 16,000-dimensional vector for the image.

All vectors are then normalized and ridge regression is used for training and prediction. Note that we also saw good results from stacking ridge regression per-quadrant with a final logistic regression on the predictions.

For this data, Graham [19] reports significantly better performance of 95.53% using 6-layer spatially sparse CNN's with dropout. Computation with MOMBAT/VQ is about 15 minutes on a CPU, versus the CNN's time of 70 hours on a

GPU. The spatially sparse CNN also requires significant data augmentation (affine transformations, etc.)

V. Additional Prior Research

For distributed representations, Hinton [20] did seminal early work. He was also an early researcher in the area of "reduced representations" [21], which is closely tied to the notion of binding in Vector Symbolic Architectures.

VSA research dates to Smolensky [22], whose early Tensor Product system was hampered by exponential growth of representation size. Kanerva's early Sparse Distributed Memory systems [3] use hashing-like techniques for creating vectors, and Plate's Holographic reduced representations [5] [6] use a convolution operator (related to, but different from convolutions in image processing) for combining vectors without increasing size. Rachkovskij [7] explored a binary vector system with interesting part-whole properties. They also mention [23] "To get correlated X for nearby places, correlated permutations can be used for them," but this appears difficult for preserving similarity between neighboring positions, particularly with multi-dimensional position representations. Levy and Gayler [8] proposed the term "Vector Symbolic Architecture" to capture the notion of a fixed-length vector participating in symbolic computations, as well as exploring term wise vector multiplication for use in reduced representations. Gallant and Okaywe [9] gave a set of "constraints" that are necessary for representations used with machine learning, and showed that matrix multiplication can accomplish binding in a neurally plausible manner.

Boureau et al. [14] gives a good overview for pooling, which they trace back to Hubel and Wiesel [24]. For neural network research, Fukushima and Miyake [25] proposed the influential neocognitron with aspects of pooling. Average pooling is attributed to LeCun et al. [26], and max pooling was proposed by Ranzato et al. [27].

Lazebnik et al. [28] were the first to use pooling over pyramid structures, which profiled (counted) features over various rectangular divisions of an image. Later, Jia et al. [29] investigated learning pooling strategies.

Many Deep Learning techniques date to LeCun's work [26] on recognizing digits. Excitement for Deep Learning was re-kindled by several works of Socher and associates [30] [31] showing new and improved performance with sentiment analysis and vision.

VI. Discussion

We have demonstrated the first fully-distributed method for adding positional binding to Constructive representations. Moreover, for Deep Learning the notion of positional binding gives insight as to why multiple layers with pooling are helpful for modeling.

The utility of positional binding comes as no great surprise – for example, LeCun [32] refers to the need to "build robustness to small distortions" being the purpose of pooling and subsampling layers. Deep Learning researchers typically presume that positional binding will come from the underlying architecture, focusing upon making it more robust to positional shifts by techniques such as jittering image presentations. We suggest that the necessity of positional binding for various resolutions should enter into consideration more actively when choosing a network architecture.

Our simulations suggest an emerging picture that Deep Learning methods can achieve better performance, but require several orders of magnitude more computation, as well as being more difficult to train.

Current research focuses upon trying to reduce the performance difference between Vector Quantization methods + positional binding versus Deep learning. We are also exploring more neurally plausible methods for positional binding.

Acknowledgment

Research supported by the National Science Foundation.

References

[1] A. Coates and A. Y. Ng, "The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization," in *ICML 28*, 2011.

[2] A. Coates, H. Lee and A. Y. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," in *AISTATS 14*, 2011.

[3] P. Kanerva, Sparse distributed memory, Cambridge, MA: MIT Press, 1988.

[4] P. Kanerva, "Fully distributed representation," in *Proc. 1997 RealWorld Computing Symposium (Report TR-96001)*, pp. 358–365, Tsukuba-City, Japan, 1997.

[5] T. A. Plate, "Holographic recurrent networks," in *C. L. Giles, S. J. Hanson, & J. D. Cowan (Eds.), Advances in neural information processing systems*, 5, San Mateo, Morgan Kaufmann, 1992.

[6] T. A. Plate, Holographic reduced representation: Distributed representation of cognitive structure, Stanford, CA: CSLI Publications, 2003.

[7] D. A. Rachkovskij and E. M. Kussul, "Binding and normalization of binary sparse distributed representations by context-dependent thinning," *Neural Computation*, vol. 13, pp. 411–452, 2001.

[8] S. D. Levy and R. W. Gayler, "Vector symbolic architectures: A new building material for artificial general intelligence," in *Proceedings of the First Conference on Artificial General Intelligence*. IOS Press, 2008.

[9] S. I. Gallant and T. W. Okaywe, "Representing Objects, Relations, and Sequences," *Neural Computation,* vol. 25, pp. 2038–2078, 2013.

[10] K. Jarrett, K. Kavukcuoglu, M. Ranzato and Y. LeCun, "What is the Best Multi-Stage Architecture for Object Recognition?," in *Proc. International Conference on Computer Vision (ICCV'09), IEEE*, 2009.

[11] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of Workshop at ICLR*, 2013.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proceedings of NIPS*, 2013.

[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *International Conference on Learning Representations (ICLR2014), CBLS, (OpenReview)*, Arxiv:1312.6229, 2014.

[14] Y.-L. Boureau, J. Ponce and Y. LeCun, "A theoretical analysis of feature pooling in vision algorithms," in *Proc. International Conference on Machine learning (ICML'10)*, 2010.

[15] B. Graham, "Fractional Max-Pooling," arXiv preprint arXiv:1412.6071, 2014.

[16] B. Graham, "Spatially-sparse convolutional neural networks," arXiv preprint arXiv:1409.6070, 2014.

[17] A. Krizhevsky, I. Sutskever, Hinton and G. E, "Imagenet classification withdeep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc, 2012, pp. 1097–1105.

[18] A. Coates and A. Y. Ng, "Learning Feature Representations with K-means," in *G. Montavon, G. B. Orr, K.-R. Muller (Eds.), Neural Networks: Tricks of the Trade, 2nd edn*, Springer, 2012.

[19] A. Coates and A. Y. Ng, "Learning Feature Representations with K-means," in *Neural Networks: Tricks of the Trade, 2nd edn*, G. Montavon, G. B. Orr and K. Muller, Eds., Springer, 2012.

[20] G. E. Hinton, "Distributed representations," in *Parallel distributed processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, Cambridge, Ma, MIT Press, 1986.

[21] G. E. Hinton, "Mapping part-whole hierarchies into connectionist networks," *Artificial Intelligence,* vol. 46, pp. 47–75, 1990.

[22] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artificial Intelligence,* vol. 46, pp. 159–216, 1990.

[23] D. Rachkovskij, E. Kussul and T. Baidyk, "Building a world model with structure-sensitive sparse binary distributed representations," *Biologically Inspired Cognitive Architectures,* vol. V3, no. 1, pp. 64-86, 2013.

[24] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocularinteraction and functional architecture in the cat's visual cortex," *J Physiol,* vol. 160, pp. 106-154, 1962.

[25] K. Fukushima and S. Miyake, "Neocognitron: A new algorithmfor pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition,* 1982.

[26] Y. Lecun, B. Boser, J. S. Denker, D. Henderson, W. Howard and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *NIPS*, 1989.

[27] M. Ranzato, Y. Boureau and Y. LeCun, "Sparse featurelearning for deep belief networks," in *NIPS*, 2007.

[28] S. Lazebnik, C. Schmid and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition (CVPR), Vol. 2. IEEE*, 2006.

[29] Y. Jia, C. Huang and T. Darrell, "Beyond spatial pyramids: Receptive field learning for pooled image features," in *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] R. Socher, C. D. Manning and A. Y. Ng, "Learning continuous phrase representations and syntactic parsing with recursive neural networks," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2010.

[31] R. Socher, C. Lin, A. Y. Ng and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th International Conference on Machine Learning*, New York, ACM, 2011.

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