Perceptron Theory Can Predict the Accuracy of Neural Networks

Denis Kleyko, *Member, IEEE*, Antonello Rosato, *Member, IEEE*, Edward Paxon Frady, Massimo Panella, *Senior Member, IEEE*, and Friedrich T. Sommer

***Abstract*— Multilayer neural networks set the current state of the art for many technical classification problems. But, these networks are still, essentially, black boxes in terms of analyzing them and predicting their performance. Here, we develop a statistical theory for the one-layer perceptron and show that it can predict performances of a surprisingly large variety of neural networks with different architectures. A general theory of classification with perceptrons is developed by generalizing an existing theory for analyzing reservoir computing models and connectionist models for symbolic reasoning known as vector symbolic architectures. Our statistical theory offers three formulas leveraging the signal statistics with increasing detail. The formulas are analytically intractable, but can be evaluated numerically. The description level that captures maximum details requires stochastic sampling methods. Depending on the network model, the simpler formulas already yield high prediction accuracy. The quality of the theory predictions is assessed in three experimental settings, a memorization task for echo state networks (ESNs) from reservoir computing literature, a collection of classification datasets for shallow randomly connected networks, and the ImageNet dataset for deep convolutional neural networks. We find that the second description level of the perceptron theory can predict the performance of types of ESNs, which could not be described previously. Furthermore, the theory can predict deep multilayer neural networks by being applied to their output layer. While other methods for prediction of neural networks performance commonly require to train an estimator model, the proposed theory requires only the first two moments of the distribution of the postsynaptic sums in the output neurons. Moreover, the perceptron theory compares favorably to other methods that do not rely on training an estimator model.**

***Index Terms*— Accuracy prediction, deep neural networks, hyperdimensional computing, perceptron theory, reservoir computing, vector symbolic architectures.**

I. Introduction

DUE to its ability to provide data-driven solutions to many previously unsolved problems, machine learning is rapidly changing many areas of our life. At the same time, there is growing demand from society [1], [2] to provide transparency and interpretability of machine learning solutions. For artificial neural networks (ANNs), this requires a deeper understanding of their underlying principles. There are many different approaches to this problem, for example, discovering and characterizing structure and dynamics emerging during the training phase, such as geometrical structures of the classifier [3], a classification behavior called shortcut learning [4], or information content about input against training class [5]. Another important avenue for understanding ANNs is through studying large ANNs in transfer learning tasks [6]. This can reveal to what extent input–output mappings are just memorized wholesale, in contrast to the input being dissected for specific parts that should elicit a particular output. To provide the transparency of ANN computations, it is crucial to analyze the decisions in trained ANNs and to predict and explain the quality of decisions. One avenue to assess the decision quality is to build models that can predict ANN's performance [7], [8], [9].[1] The most recent works [8], [9] train another estimator ANN to perform the prediction.

Here, we propose an alternative approach for predicting the expected accuracy on classification problems for different trained networks, including deep networks, as well as echo state networks (ESNs) [10]. Our approach does not require training another ANN; rather it is based on the theory of the simple one-layer perceptron, which can describe the last layer of the ANN. The perceptron theory generalizes an earlier theory [11], proposed for hyperdimensional computing or, synonymously, vector symbolic architectures (HD/VSA) and ESNs with unitary recurrent connections.

The perceptron theory presented here includes formulas for three different levels of prediction accuracy. Building on the original formulation [11], we propose two novel formulas that, under certain conditions, yield predictions with higher accuracy than the original theory. All formulas leverage the first two moments of the statistics of the postsynaptic sums at the output neurons, but differ in how coarsely the multivariate distribution is approximated. The new formula with the highest prediction accuracy captures correlations among dendritic sums of neurons, which makes it hard to evaluate numerically. The formulas for lower prediction accuracy neglect such

[1] In the scope of this study, by performance, we refer to ANN's empirical accuracy measured on the test data for the corresponding classification problem. Interchangeably, the term "actual accuracy" is also used when appropriate.

Manuscript received 17 September 2021; revised 1 September 2022 and 10 November 2022; accepted 8 January 2023. The work of Friedrich T. Sommer was supported by the National Institutes of Health (NIH) under Grant R01-EB026955. The work of Denis Kleyko was supported in part by the European Union's Horizon 2020 Research and Innovation Program within the Marie Skłodowska–Curie under Grant 839179, in part by the Defense Advanced Research Projects Agency's (DARPA) VIP (Super-HD Project) and AIE (HyDDENN Project) Programs, and in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-19-1-0241. The work of Friedrich T. Sommer and Denis Kleyko was supported in part by the Intel's THWAI Program. *(Corresponding author: Denis Kleyko.)*

Denis Kleyko is with the Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA 94720 USA, and also with the Intelligent Systems Laboratory, Research Institutes of Sweden, 164 40 Kista, Sweden (e-mail: denis.kleyko@ri.se).

Antonello Rosato and Massimo Panella are with the Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", 00184 Rome, Italy.

Edward Paxon Frady is with the Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA 95054 USA.

Friedrich T. Sommer is with the Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA 94720 USA, and also with the Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA 95054 USA.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2023.3237381.

Digital Object Identifier 10.1109/TNNLS.2023.3237381

correlations and can be easily evaluated numerically. The neglecting of correlations introduces a bias to the predictions, which, however, can be removed empirically.

To assess the quality of the predicted accuracies and to demonstrate their effectiveness in interpreting ANNs, we applied the theory was in three experimental settings. First, in Section IV-A, we applied it to a type of ESN that was not described by the earlier theory [11]. Second, in Section IV-B, we used a collection of classification datasets with shallow randomly connected ANNs exploring two strategies of training the perceptron. Third, in Section IV-C, we evaluated the accuracies of 15 deep convolutional ANNs (CNNs) trained on the ImageNet dataset. The results show high correlation between the actual accuracies and the predictions. Thus, the proposed theory identifies critical features for predicting and comparing performances of networks with different architectures. This not only helps to gain a deeper understanding of the principles at work, but also has practical implications. For example, for a particular task, the theory can be used to select the best suited network from a set of pretrained networks.

II. Related Work in Predicting ANN Performance

Investigations on how to analyze and predict the classification performance of ANNs go back to several decades [7], [12], [13]. Recent interest in this topic has been in the context of network architecture search (NAS) [14], [15], [16], i.e., studying the design space of a network and the search strategies in that space [16]. Another domain where prediction performance is crucial is knowledge distillation (KD) [17], and the related studies on how to enhance the performance of simpler models by leveraging more complex ones [18], [19]. The topic is also relevant for explaining and interpreting the overall learning mechanism in multilayer networks [20], including efforts on visualizations deciphering deep networks [21].

Here, we particularly build on earlier work on how to predict accuracy, either based on the weights in the ANN [8], [22] or based on the activations of the different layers [9]. Unterthiner et al. [8] trained estimators, which were able to reasonably rank the classification performance of a plethora of CNNs. DeChant et al. [9] trained a "meta network" to predict the correctness of an ANN on an individual input data sample, by using the activations in the hidden layers. The results of these studies, as well as [23], support our approach to bisect ANNs into a transformation and a readout perceptron stages. In [8], it was observed that parameters of the last fully connected weight layer (i.e., readout perceptron) were among the most informative and frequently used ones. DeChant et al. [9] reported that for predicting network accuracy, the activations of the last hidden and output layer were the most useful.

While most of the previous work on performance prediction relies on training an estimator model (e.g., an elementary regressor or another ANN), in the context of NAS, there is a demand for maximally reducing computational expenses associated with the search [24] that, in turn, stimulates explorations of simple metrics that do not require training an estimator model. A recent study [25] has empirically investigated several such metrics. Curiously, while some of the investigated metrics have been proposed in the NAS context [26], most of the metrics have appeared first in the context of ANNs' pruning and compression [27], [28], [29], [30]. It is worth noting that the principal motivation for developing these metrics is in their practical usage for the abovementioned applications. In contrast, here, we develop a statistical theory that relies on only a few assumptions with the aim of deepening our understanding of the principles at work in ANNs and without employing any estimator model, which would add yet another black box. In contrast, here, we develop a statistical theory that relies on only a few assumptions, but does not employ an estimator model, which yet adds another black box. Thus, potentially, our statistical approach can provide novel insights for explaining the quality of ANNs' decisions, such as the possibility to compare networks with different architectures using only the weights of their last layers (see Section IV-C6 for details).

Another interesting approach was introduced recently in [22]. It does not rely on training an estimator model. Instead, statistical metrics are computed solely from the weights of the ANN and, thus, provide a single score to characterize the trained ANN. In contrast, the complete formulation of the perceptron theory needs access to both the ANN's weights and the activation patterns in the last hidden layer to estimate two moments of the statistics of the postsynaptic sums (see Section IV-C6). We compared the methods in Section IV-C7 on predicting networks with different architectures, showing that the perceptron theory provides higher quality than the methods from [22] as well as the most performing metric from [25].

III. Perceptron Theory

The perceptron network is the ancestor of all modern ANNs. A perceptron is a simple artificial neuron introduced by Rosenblatt [31], in which inputs are weighted by synapses, added linearly, and converted to a binary output by a threshold transfer function. Classification problems can be naturally mapped to a network of parallel perceptrons (often referred to as one-layer perceptron) in which each perceptron neuron is responsible for detecting a match between the input and one of the classes. The best match of a given input can be determined by adaptively changing the threshold value in all perceptrons [32]. Alternatively, one can replace the original perceptron transfer function by a graded transfer function, such as sigmoid or rectified linear unit, so that the output vector of the network still contains graded match information for the different classes. One can also replace the individual neural transfer functions by a global mechanism for maximum detection among all neurons [33], for example, using winner-take-all competition [34], [35], [36]. Note that maximum detection was not discussed in the original works of Rosenblatt. However, for solving classification problems with multiple classes, such a mechanism is required and was introduced early on; see, e.g., [33] for a natural generalization of the original perceptron with the winner-take-all mechanism for the case of multiple classes. Conversely, modern ANNs use graded neural transfer functions, which enables maximum detection on the output vector of the network. Thus, the analysis of classification with a perceptron-like neural network layer has to take maximum detection into account, and it should be based on signal detection theory [37].

A. Perceptron Theory in the Literature

Curiously, but understandably given the lack of universality of the one-layer perceptron [38], one cannot find the theory of perceptron classification in textbooks. However, this theory has been developed piece by piece in the context of understanding more complicated networks, such as the formation of separable sensory representations [39], symbolic reasoning

in HD/VSA[2] [51], and some types of ESNs[3] [11]. These studies demonstrated that complicated neural computations, involving recurrent circuitry and nonlinear stages, can be successfully dissected into two stages: a transformation of input data samples to a new N-dimensional space and a one-layer perceptron that reads out the transformations of input data to classes or network outputs.

In the one-layer perceptron, the synaptic weights are a linear transformation of the inputs to the postsynaptic sums of the neurons. The classification is correct if the postsynaptic sum is the largest in the neuron that corresponds to the actual class of the input data sample. To date, the most complete version of a Gaussian theory of the perceptron was presented in [11]. By generalizing signal detection theory [37] and building on earlier work on how Gaussian distributions can be transformed [52], the work in [11] shows that if the input data sample belongs to one of D classes, then the predicted accuracy (denoted by a), i.e., the probability that the perceptron output ($\text{class}_{\text{out}}$) is the correct class ($\text{class}_{\text{inp}}$), is given by

$$a := p(\text{class}_{\text{out}} = \text{class}_{\text{inp}}) = \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \left[\Phi\left(\frac{\sigma_r}{\sigma_h} x + \frac{\mu_h - \mu_r}{\sigma_h} \right) \right]^{D-1}. \quad (1)$$

Here, $\Phi(x)$ is the cumulative Gaussian; μ_h and σ_h denote the mean and standard deviation of the postsynaptic sum of the output neuron that corresponds to the correct class; μ_r and σ_r denote the mean and standard deviation of the postsynaptic sum for all other neurons. The formula describes the performance of all flavors of HD/VSA models [40], [51], [53], [54], [55], [56], [57] and also describes some variants of ESNs [58], [59].

B. Novel Contribution to Perceptron Theory

Intuitively, (1) computes the probability that for any given value x of the postsynaptic sum in the output neuron that represents the correct class, the postsynaptic sums in all other neurons are smaller than x. Note that (1) makes three strong assumptions about the distributions of the postsynaptic sums.

1) The distributions are normal.
2) The distributions for "distractor" classes are all the same.
3) The distributions for different classes are independent.

For analyzing HD/VSA models, these assumptions are justified, because information is represented by normalized pseudorandom vectors, and the symbolic operations in HD/VSA conserve pseudorandomness and norm. Furthermore, these models do not have weights that are learned from data. When predicting the accuracy with weights learned from input distributions, for example, in ESNs or ANNs, some of these assumptions will be violated.

We generalize (1) in two steps and demonstrate that the generalization can predict a broader variety of ANN architectures. Our first step of generalization of (1) is to drop assumption 2) and compute individual predicted accuracies for each class $i \in \{1, 2, \ldots, D\}$

$$\mathbf{a}_i := p(\text{class}_{\text{out}} = i | \text{class}_{\text{inp}} = i) = \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}\boldsymbol{\sigma}_i} e^{-\frac{(x-\boldsymbol{\mu}_i)^2}{2\boldsymbol{\sigma}_i^2}} \prod_{j=1, j\neq i}^{D-1} \Phi(x, \boldsymbol{\mu}_j, \boldsymbol{\sigma}_j). \quad (2)$$

Here, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote vectors representing the first two moments (mean and standard deviation) of the statistics of the postsynaptic sums for all classes. The assumptions for (2) to hold are reduced to 1) and 3).

Our second generalization is to drop assumption 3); that is, we allow the postsynaptic sums for different neurons to be correlated according to a multivariate normal distribution[4]

$$\mathbf{a}_i := p(\text{class}_{\text{out}} = i | \text{class}_{\text{inp}} = i) = \int_{-\infty}^{\infty} d\mathbf{x}_1 \int_{-\infty}^{\mathbf{x}_1} d\mathbf{x}_2 \cdots \int_{-\infty}^{\mathbf{x}_1} d\mathbf{x}_D \, \mathcal{N}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (3)$$

where $\mathcal{N}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate normal distribution with the full covariance matrix $\boldsymbol{\Sigma}$ replacing the variance vector $\boldsymbol{\sigma}$ for independent Gaussians in (2); see Appendix A for connection between (2) and (3). The assumptions for (3) to hold are reduced to 1).

To aggregate the individual class accuracies in $\mathbf{a}$ into the overall prediction, we form the expectation over all D classes

$$\sum_{i=1}^{D} \mathbf{f}_i \mathbf{a}_i \quad (4)$$

where $\mathbf{f}_i$ is the prior probability of the ith class in the data. Recall that vector $\mathbf{a}$ stores predicted accuracies for all D individual classes where the predictions are obtained according to one of the above formulations of the perceptron theory [see (1)–(3)]. This single prediction score characterizes the accuracy of the network for the whole classification task. The prior probability of each class $\mathbf{f}_i$ is estimated using the frequency of the ith class labels in the empirical samples of the data, such as the training or test ones. The predicted accuracy can be compared with the actual accuracy of the network.

IV. Experiments

A. Predicting the Performance of ESNs

First, we test our generalized perceptron theory on predicting an ESN performance in the trajectory association task [41]. In Fig. 1, we compare two ESNs with different readout perceptrons.[5] The synapses of one perceptron (blue line) are set to the centroids of inputs of the different classes (codebook-based). The synapses in the other perceptron (red line) have been optimized with linear regression (regression-based). The average predicted accuracy for the both versions is plotted in Fig. 1 as a function of the delay of the input occurred in the sequence of input vectors.

The solid lines depict empirical accuracies. The dotted lines are the predictions by theory in (1)–(3), using the statistics reported in Fig. 13 of Appendix A. The original formulation in (1) correctly describes the experimental performance of

[2]For readers interested but not familiar with HD/VSA, we would like to recommend the tutorial-like article [40], which is probably one of the best starting points facilitating the entrance to the area. There is also a detailed treatment of the basics of the area that can be found in [41] and, more recently, in a two-part comprehensive survey [42], [43]. When it comes to specific aspects of the area, [44] can be consulted for representation of data structures, while for similarity-preserving representations and applications to classification problems, [45], [46], [47] and [48], [49], [50] can be looked into, respectively. For specific aspects, such as representation of data structures, similarity-preserving representations, and applications to classification problems, we recommend consulting [44]; [45], [46], and [47]; and [48] and [49], respectively.

[3]Please refer to Appendix A, which introduces a minimalistic variant of ESN as well as provides additional references.

[4]We assume that the correct class is always in the first position of $\mathbf{x}$ (i.e., $\mathbf{x}_1$). This is done just for convenience to make the writing of the next integrals in (3) more straightforward.

[5]Please see Appendix A for the detailed description of the experiment and the types of ESNs being used.

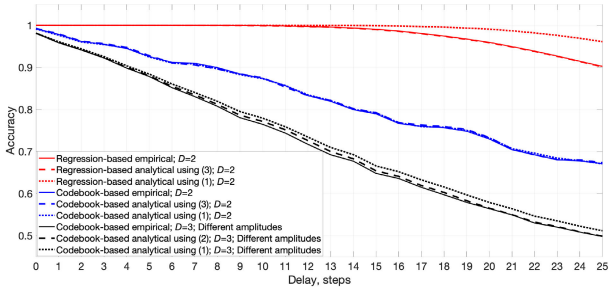


Fig. 1. Accuracy of the ESN against delay for the case of codebook-based and regression-based readout perceptrons. The following values for the ESN parameters were used: $N = 100$, $D = 2$ (for red and blues lines) or $D = 3$ (for black lines), $\kappa = 4$. The length of test sequences was 10 000. All reported values were averaged over 50 simulations with pseudorandom codebooks.

the perceptron with centroid filters but overestimates the performance of the perceptron with regression filters (i.e., it has some bias). The bias occurs, because (1) neglects the correlation between the linear filters of the regression-based readout, visible in Fig. 13 (right) of Appendix A. Fig. 1 (dashed lines) depicts the predictions by (3), and the accuracies of both networks are predicted correctly.

The predictions obtained using (2) are omitted, because they did not differ from the predictions obtained using (1) when $D = 2$. To see differences between these two formulations, we run another experiment with $D = 3$ and a perceptron with centroid filters where the inputs had different amplitudes (black solid line). For this experiment, (1) (black dotted line) overestimated the accuracy, while (2) (black dashed line) provided accurate predictions.[6]

B. Predicting the Performance of Shallow Networks

We also applied the theory in (2) to shallow feedforward randomly connected ANNs [60], which are similar to ESNs without the recurrent connections. The evaluation was done on a collection of the classification tasks where the data are provided in the form of the extracted features. In particular, the reported results are based on 121 real-world classification datasets obtained from the UCI Machine Learning Repository [61]. The considered collection of datasets has been initially analyzed in a large-scale comparison study of different classifiers, and the interested readers are kindly referred to the original work [62] for more details. The only preprocessing step was to normalize features in the range [0, 1].

For the sake of brevity, here, we do not go into the details of the transformation stage. The interested readers are kindly referred to [63]. In essence, the transformation resembles Random Vector Functional Link (RVFL) networks [60], which is a particular variation of shallow feedforward randomly connected ANNs. Moreover, it is very similar to the known approaches of using HD/VSA for classification [48], [64], [65], [66], [67], [68], [69], [70].

The search of the hyperparameters for each dataset has been done according to [62] using the grid search over λ (regularization parameter), N, and κ (activation function parameter). N varied in the range $[50, 1500]$ with step 50; λ varied in the range $2^{[-10,5]}$ with step 1, and κ varied in the set of $\{1, 3, 5, 7\}$. The obtained optimal hyperparameters were used to estimate the cross-validation accuracy on all datasets.

[6]We also studied the perceptron theory predictions for a synthetic binary classification problem in the case of nonindependent distributions of postsynaptic sums in Appendix B.

Similar to Section IV-A, we considered two ways of forming the perceptron. The first way is common in HD/VSA. It forms a linear filter as a centroid of a class by simply superimposing all transformations of input data for this class. The perceptron then is simply the collection of all the centroids. Formally, let us assume that the transformations for all training data are stored in the matrix $\mathbf{X}$, and the corresponding class labels are stored in the ground-truth vector $\mathbf{y}$; then, the centroid for the ith class (denoted as $\mathbf{W}_i$) is obtained as the sum of the corresponding transformations in $\mathbf{X}$

$$\mathbf{W}_i = \sum_{j, \mathbf{y}_j = i} \mathbf{X}_j. \tag{5}$$

In addition, it is common to normalize each centroid to unit norm

$$\mathbf{W}_i = \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|_2}. \tag{6}$$

The second way is common in shallow feedforward randomly connected ANNs. The perceptron is the result of the ridge regression, which uses the transformations of training data and the ground-truth class labels to obtain the optimal values of the perceptron $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{Y}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \tag{7}$$

where $\mathbf{I}$ denotes the identity matrix, λ is a regularization parameter used in the ridge regression, and $\mathbf{Y}$ is the one-hot representation of the ground truth from $\mathbf{y}$. Thus, the main difference between the considered approaches in forming the perceptrons is that the regression filters are obtained using the optimization procedure, while the centroid filters are nonparametric.

Across the datasets, the average cross-validation accuracy for the perceptron with centroid filters was 0.70, while that for the perceptron with regression filters was 0.80. The Pearson correlation coefficient between the obtained results was 0.80. It is clear that the perceptrons obtained from the ridge regression usually outperformed the ones with the centroids.

Fig. 2 (top) presents the cross-validation accuracies against their corresponding predicted accuracies for the perceptron with centroid filters. The top-left panel in the figure corresponds to the case when the statistics for (2) was obtained from the training data. The top-right panel in the figure corresponds to the case when the statistics for (2) was obtained from the test data. The Pearson correlation coefficient between the accuracy and the predicted accuracy calculated from the statistics for the training data was 0.79, while that of the test data was 0.90. The usage of the statistics for the test data improved the quality of the predictions. The reduced correlation for the case when the statistics for the training data was used is expected and happens due to the "leakage" of the data. Since the same data samples are used to both form the perceptron and to estimate the statistics of the postsynaptic sums, the obtained estimates might differ significantly from the ones observed for previously unseen data samples (i.e., test data).

Fig. 2 (bottom) presents the cross-validation accuracies against their corresponding predicted accuracies for the perceptron with regression filters. The Pearson correlation coefficient between the actual accuracy and the predicted accuracy for the statistics from the training data was 0.47, while that for the test data was 0.97. Similar to the results for the perceptron with centroid filters, the usage of the statistics for the test data improved the quality of the predictions. The difference, however, was that the predicted accuracies

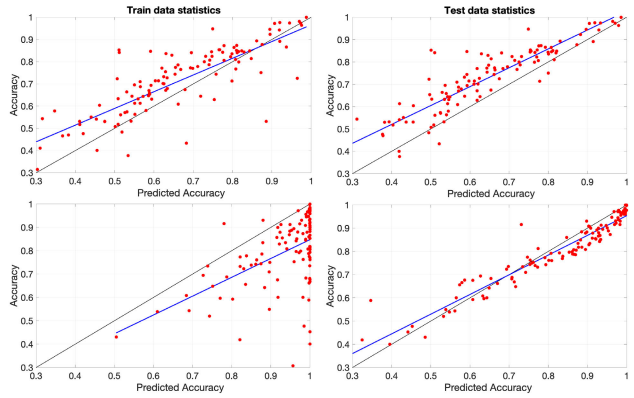


Fig. 2. Cross-validation accuracy against the predicted accuracy where the predicted accuracy for individual classes was calculated according to (2). Each point corresponds to a dataset. Top: perceptron with centroid filters. Bottom: perceptron with regression filters.

from the training data statistics did not correlate well with the accuracies. The most likely explanation is that in the case of the ridge regression, the perceptron is calculated to maximize the accuracy on the training data. As a side effect, the statistics of the postsynaptic sums is too promising, and so many networks are predicted to achieve high accuracy on the test data. Other notable differences were that compared to the perceptron with centroid filters, the errors and bias are smaller for the ridge regression with test data statistics. For instance, after removing the bias, the mean value of the absolute error was 0.03 for the perceptron with regression filters, while for the perceptron with centroid filters, it was 0.12.

Thus, the main finding here is the sequent: for both perceptrons with centroid and regression filters, the theory in (2) introduced some bias, but the Pearson correlation coefficients between the predictions and the actual accuracies were high: 0.90 and 0.97, respectively. These differences should be attributed to the effect of the assumptions used in (2). Section IV-C3 will elaborate more on this topic.

C. Predicting the Performance of Deep CNNs

1) Deep ANNs in Terms of Transformation and Perceptron Stages: In the remainder, we will apply the perceptron theory described in Section III on deep ANNs. To apply the theory, we dissect the holistic functionality of a deep network into two parts: a multilayer transformation stage and a single-layer classification by a perceptron as depicted in Fig. 3. It is not very common to look at ANNs this way, but, for example, a recent work [3] has used the same dissection to study the terminal phase of the training. Note that this dissection is conceptual, and it affects neither training nor inference processes. This bipartite view suggests that most of the network is doing a transformation, i.e., produces a useful representation of an input data sample. As we will see below, a similar conceptual dissection can be applied if the transformation stage includes convolutional layers (i.e., CNNs) or recurrent connections (i.e., RNNs), as long as the last hidden layer and the output later are densely connected. Finally, note that using the final outcomes of the multilayer transformation stage (i.e., activations of the last hidden layer) does not mean that any information about the network is lost, because obtaining these activations requires propagating the activity through all the layers that precede the last hidden one [see Fig. 3 (dashed rectangle)].

2) Setup: Here, we describe the results of experiments with a set of well-known deep CNNs, which were pretrained on the

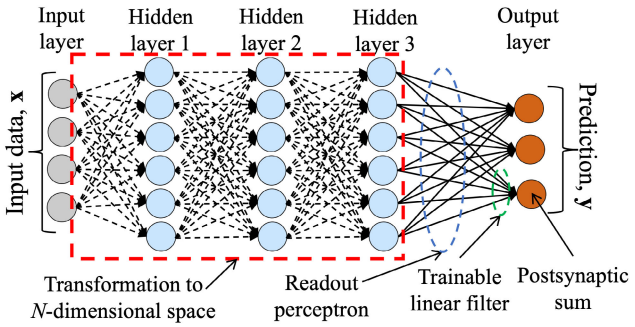


Fig. 3. Dissecting the functionality of an ANN into multilayer transformation and perceptron stages. The figure indicates components of the network by the terms used in this article.

ImageNet dataset [71]. The ImageNet dataset arose from ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Currently, it is one of the well-established benchmarks in the object category classification and detection. ImageNet includes $D = 1000$ classes corresponding to different object categories, which makes it hard to preform well on this task. The training data of ImageNet include over 14 million images. The validation set from ILSVRC 2012 challenge has 50 000 images in total; exactly 50 images per each class.

In the experiments below, 15 pretrained deep CNNs were used: AlexNet [72], GoogLeNet [73], ResNet-18, ResNet-50, ResNet-101 [74], VGG-16, VGG-19 [75], ShuffleNet [76], MobileNet-v2 [77], DenseNet-201 [78], Inception-v3 [79], Inception-ResNet-v2 [80], Xception [81], NASNet-Mobile, and NASNet-Large [82]. The actual accuracy of the deep CNNs was assessed using the validation set from ILSVRC 2012 challenge. Blue solid lines in the figures below are the lines fit to the results.

For each network, we first saved the weight matrix corresponding to the connections between the last hidden layer and the output layer. Recall that in our interpretation, this weight matrix (i.e., readout perceptron) is treated as a set of D linear filters, where, for each class, the linear filter is the corresponding N-dimensional vector. Second, for all networks, we have also obtained the activations of the last hidden layer for each image in the ILSVRC 2012. Recall that these activations are seen as the results of the multilayer transformation stage of the network. All linear filters and the last hidden layer activations of a particular class were used to calculate the postsynaptic sums. Note that in cases when it is easier to obtain the statistics in the form of activations of ANN's last layer, they can be used to directly compute the required statistics. The resultant matrix was used as a data sample to estimate moments of probability density functions, which are involved in calculating the predicted accuracy $\mathbf{a}_i$ for that class.

3) Predictions According to (2): We compared the actual accuracy of each network against its predicted accuracy, where the predicted accuracies for individual classes were calculated according to (2) and then aggregated into a single prediction [Fig. 4 (left)].[7] The results were mixed. On the one hand, the Pearson correlation coefficient between the actual accuracies and the predicted accuracies was 0.93, indicating a high similarity between the actual and predicted accuracies. Also, the Kendall's τ correlation coefficient measuring the quality of the ranking of the networks performances was 0.86, which is rather high. For instance, it is worth noting here that if

[7]In practice, it was averaging as in the ILSVRC 2012 each class has the same number of images, so $\mathbf{f}_i = 1/D = 10^{-3}$.

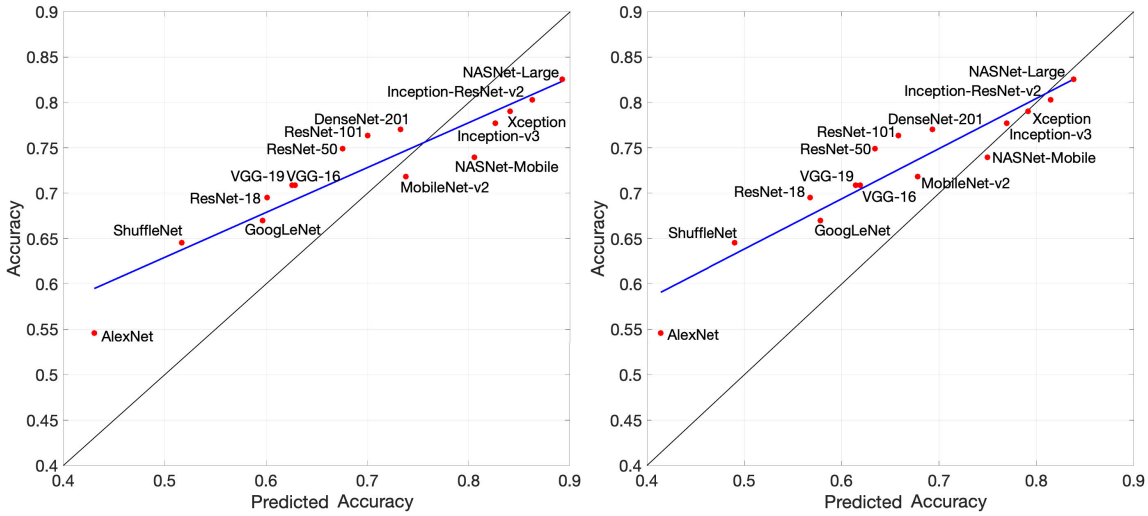


Fig. 4. Accuracy of 15 deep CNNs on the ILSVRC 2012 validation dataset against the predicted accuracy. Left: individual predicted accuracies were calculated according to (2). Right: individual predicted accuracies were calculated similar to (2) but using the kernel distributions (Pearson correlation coefficient was 0.94).

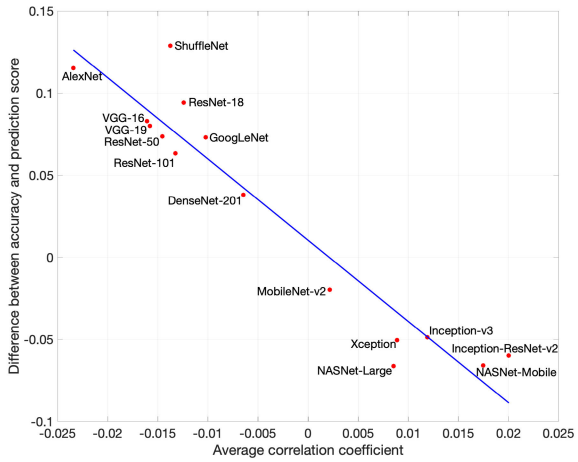


Fig. 5. Difference between the accuracy and the predicted accuracy for 15 deep CNNs against the average Pearson correlation coefficient in the covariance matrices. The predicted accuracies were calculated according to (2).

one would just use the number of flops for each network (which is a meaningful information for deep CNNs) as the performance prediction, these correlation coefficients are only 0.50 and 0.49, respectively. On the other hand, however, there are obvious issues with the predicted accuracies. First, there is a clear bias in the predicted accuracies. Second, single points deviate noticeably from the fit linear trends. For example, even after compensating for bias, the largest error was observed for AlexNet, where the predicted accuracy overestimated the accuracy by 0.05; the mean value of the absolute error was 0.02.

One possible explanation of these prediction errors is that assumption 1) in (2) is violated, the assumption that the postsynaptic sums are distributed normally. We quantified the effect of this assumption by using nonparametric kernel density estimation for the probability density functions and then calculated the predicted accuracy in the same way as in (2). The accuracies predicted with the kernel estimation [Fig. 4 (right)], indeed, differ somewhat from the ones depicted in the left panel. However, dropping the assumption of Gaussianity did not significantly reduce the error in predicted accuracies. The strong bias was still present, and the deviations of the points from the fit line were still nonnegligible. The mean value of the absolute error after compensating the bias was at 0.02. In Section IV-C7.a, we also provide the predictions produced by other prediction methods.

The next natural step was to investigate the effect of assumption 3) that the distributions of individual postsynaptic sums are all independent.

4) On Removing Bias Introduced by (2): Since (2) does not take into account the covariance matrix, it is worth checking whether some information contained in the covariance matrix itself is able to explain the bias introduced by the predicted accuracies. Fig. 5 depicts the difference between the observed accuracy and the predicted accuracy against the average value of Pearson correlation coefficients in the covariance matrices of each network. We see that the difference closely follows the average values of Pearson correlation coefficients for different networks. In particular, the Pearson correlation coefficient between the differences and the average Pearson correlation coefficients was -0.95. Thus, at least for the ImageNet dataset, we can conclude that it is possible to use the average Pearson correlation coefficient to predict whether the predicted accuracy is too optimistic or too pessimistic about the actual accuracy.

Morevoer, since it is hard to conclude much from only 15 datapoints, we used ImageNet to create subproblems with smaller numbers of classes. In our experiments, the subproblem of a given size was generated by randomly choosing (without replacement) the classes to be included in the subproblem. The following sizes of subproblems were used $\{2, 4, 8, 16, 32, 64, 128, 256, 512, 768\}$. For each network and for each subproblem size, 40 different subproblems were evaluated.

Each panel in Fig. 6 (left) corresponds to a deep CNN. Each point corresponds to a subproblem with the size of the subproblem represented by color. First of all, we see that each network develops its own bias, which is in line with the bias present in Fig. 4 (left). This suggests that assuming independence between distributions of postsynaptic sums of output neurons caused a noticeable bias effect on the predicted accuracies. Moreover, the error between the actual and predicted accuracies caused by the bias was increasing with the size of the subproblem. However, what is astonishing is that the Pearson correlation coefficients between the actual and predicted accuracies were almost exactly one for individual networks (the lowest one was 0.99 for

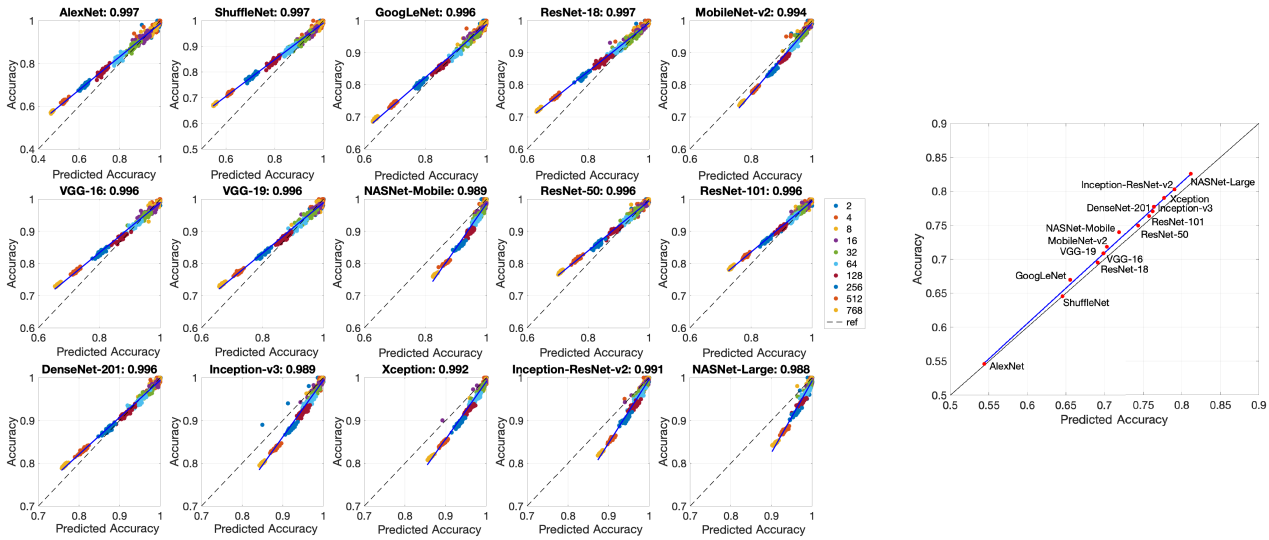


Fig. 6. Accuracy of 15 deep CNNs on the ILSVRC 2012 validation dataset against the predicted accuracy. Left: subproblems of different sizes randomly sampled from ImageNet. The predicted accuracies for individual classes were calculated according to (2). Color of point indicates the size of the subproblems. Title for each panel states network's name and the Pearson correlation coefficient. Right: predicted accuracies of networks were calculated according to (2) and then compensated using the lines from the left panel.

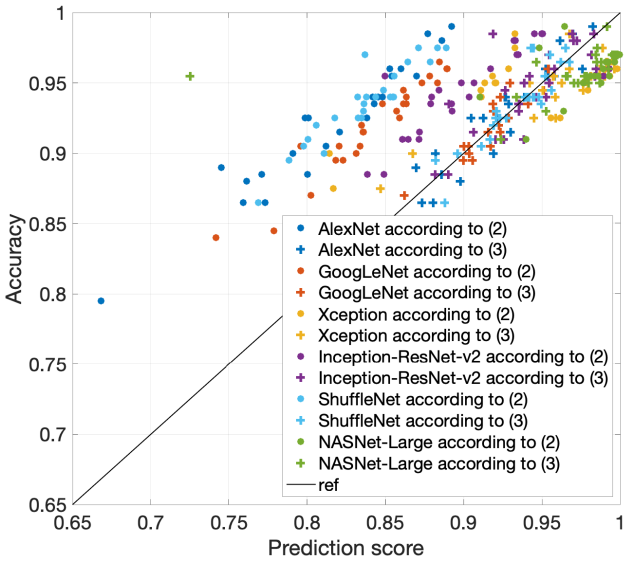


Fig. 7. Accuracy of six deep CNNs against the predicted accuracy for the subproblems of size 4. The predicted accuracies for individual classes were calculated either according to (2) or (3).

NASNet-Large). This is another implicit indication that the assumption of normal distribution is not critical for predicting accuracy. Moreover, the lines fit for each network can now be used to compensate the corresponding predicted accuracies in Fig. 4 (left). For the compensated predicted accuracies against the actual accuracies, the Pearson correlation coefficient was 0.998 [Fig. 6 (right)]. We also noticed that the compensations based on the subproblems on individual networks have almost removed bias and unsystematic deviations between the accuracies. The compensated predicted accuracies overestimate the actual accuracies to a lesser degree than that depicted in Fig. 4; e.g., the largest error (overestimated) was 0.02 for NASNet-Mobile.

5) Predictions According to (3): The results in Fig. 6 (right) are encouraging from the point that the compensated predicted accuracies nearly perfectly corresponded to the accuracies. Obviously, the Kendall's τ correlation coefficient was 1.00. On the other hand, in order to make the compensation, it is necessary to observe the accuracies of smaller subproblems, which is highly undesirable from the practical applicability point of view. The solution to this problem is to get rid of the independence assumption by calculating the predicted accuracies according to (3). This approach, however, has its own complications, as the numerical integration of (3) is challenging, even for the moderate number of classes. Thus, to demonstrate that (3) addresses the bias issue, we have performed another experiment. We selected the subproblems from ImageNet where the size of the subproblem was fixed to four classes. Different from the previous experiment, we first randomly chose 10 000 subproblems and calculated both the actual and predicted accuracies using (2). Then, we hand-picked 25 most challenging subproblems choosing the ones whose predicted accuracies calculated using (2) resulted in large errors. For these subproblems, we used both formulas, (2) and (3), to predict the accuracies (see Fig. 7). For the considered networks, there was no bias in the predictions calculated according to (3), which is an empirical demonstration that (3) can predict a wider range of network architectures.

As Fig. 7 suggests, calculating the predicted accuracies with (3) should address the problem of bias introduced by (2). However, the numerical integration for problems with many classes, such as ImageNet, is not tractable. We partially circumvented this problem by using a Monte Carlo approach for sampling from the estimated distributions. The results obtained with the Monte Carlo approach (see Fig. 8) were quite obviously better than the ones obtained with (2). The Pearson correlation coefficient was 0.98, and the Kendall's τ correlation coefficient was 0.92. Another nice outcome was that the points were arranged on a line (dashed line) parallel to the line of perfect correspondence (solid line); i.e., all the predictions slightly overestimated the actual accuracies by a constant offset. This offset just comes from the fact that one should use the normalization constant in our estimations, which Monte Carlo sampling does not provide.

Fig. 9 presents the results of applying the same sampling approach to different sizes of subproblems randomly formed from ImageNet. Overall, we still see the same trend as in Fig. 8, i.e., that the predicted accuracies were approximately offset from the line of perfect correspondence by a constant

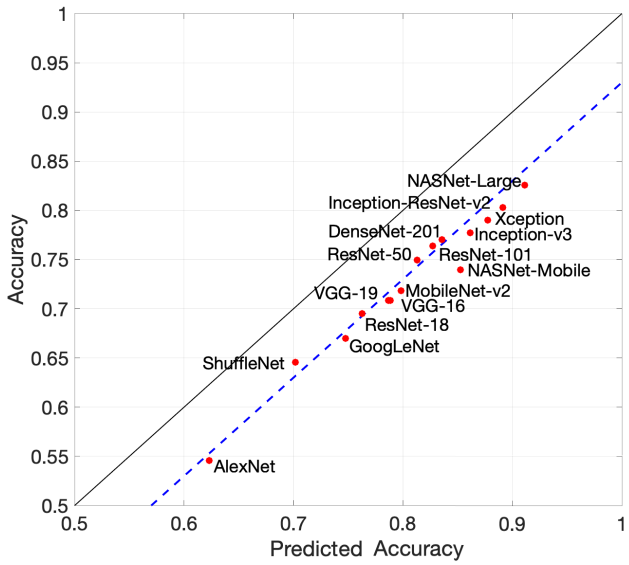


Fig. 8. Accuracy of 15 deep CNNs against the predicted accuracy. The predicted accuracies were calculated according to (3) using Monte Carlo sampling.

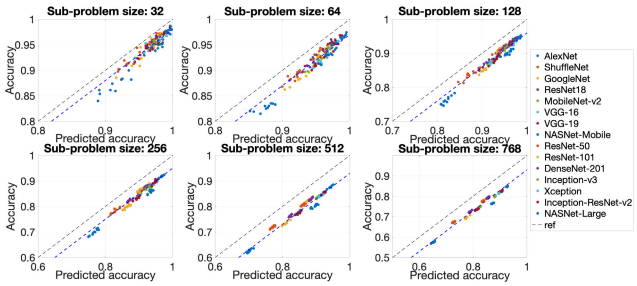


Fig. 9. Accuracy of 15 deep CNNs against the predicted accuracy. Each panel corresponds to a fixed size of a subproblem. The predicted accuracies were calculated according to (3) using Monte Carlo sampling.

(dashed lines). The difference is more noticeable with the decreased size of a subproblem, which indicates the increased importance of normalization constants.

6) Predicted Accuracy From the Readout Perceptron Only: One interesting question to the proposed theory is: how well could it work if given only the readout perceptron but no activations of the last hidden layer? Quite surprisingly, as we will see below, it was possible to achieve a τ correlation coefficient of 0.71 by computing the predicted accuracy only from the readout perceptrons, without any access to the transformations of input data samples.

Intuitively, a possible way to calculate the required statistics (i.e., $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) would be to use linear filters themselves in place of the hidden layer activations. The issue with this approach, however, is that the postsynaptic sums to linear filters themselves are much higher than that of the hidden layer activations from real data, since in reality, the activations of the hidden layer will never match perfectly their corresponding linear filters, which means that realistic postsynaptic sums are lower. Thus, this situation presents the case when we obtain the maximally possible postsynaptic sum, which is not expected in reality where hidden layer activations will not be the exact copies of the corresponding linear filter. Therefore, the predicted accuracies calculated in this way are nearly 1. A possible way to mitigate this issue is to disturb the linear filters by adding some white noise to them. The noisy versions of a particular linear filter are used in place of activations of the last hidden layer by the

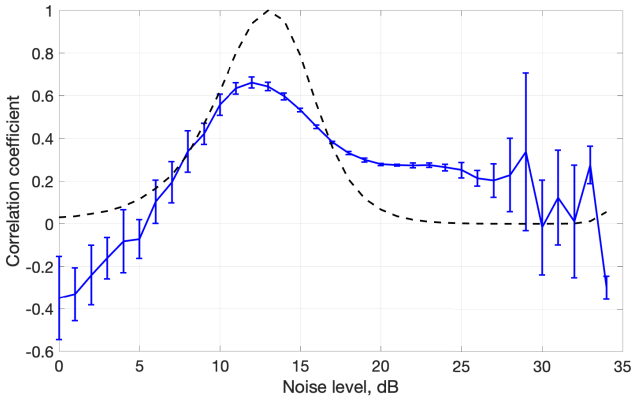


Fig. 10. Solid line depicts the mean Pearson correlation coefficients for 15 deep CNNs between their accuracy and the predicted accuracies calculated using only the perceptron when disturbing the linear filters with white noise for different level of noise. The bars depict standard deviations. The dashed line depicts standard deviations of the predicted accuracies normalized by the largest value.

samples of the corresponding class. The postsynaptic sums are calculated using the original linear filters and their noisy versions. The obtained postsynaptic sums are used to estimate the required statistics for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Thus, the white noise added to the linear filters acts as a way to obtain surrogates of the actual statistical distribution of the data.

We have made the experiments for different levels of noise. In a single experiment, each linear filter was disturbed 50 times. The disturbed versions were used as the hidden layer activations for linear filter's class. Fig. 10 (solid line) reports the mean Pearson correlation coefficients between the actual accuracy and the predicted accuracies obtained from surrogate statistics from ten experiments for each level on noise. The results suggest that either high or low noise levels did not result in high correlation, but there was a window between 8 and 17 dB where the correlation peaked getting as high as 0.66.

A natural question to ask would be: how to know which level of noise to use for obtaining the highest correlation? We think that it could be identified in a straightforward manner; we use two observations to do so. First, we expect that different networks have somewhat different accuracy. Second, we know that the best value of noise is somewhere in between the extremes. That is because when the noise is too high the predicted accuracies of all the networks would vanish to a random guess while when the noise is too low the predicted accuracies would saturate at one, as explained above. Using these observations, we expect that the best level of noise should result in the largest dispersion of the predicted accuracies, which can be simply measured by the standard deviation of the predicted accuracies. Fig. 10 (dashed line) depicts mean (across ten experiments) and standard deviations for each level of noise. As we can see from comparing this curve with the curve for the Pearson correlation coefficients, both curves peaked approximately in the same window between 8 and 17 dB. Thus, when choosing the level of noise corresponding to the largest standard deviation, we expect to get close to the peak in the Pearson correlation coefficient. The absence of a clear peak might be interpreted as the indication that the accuracy of all the networks is so close to each other that we cannot discriminate between them using only information about the perceptron.

Finally, we can use the obtained predicted accuracies in order to rank networks using the Kendall's τ correlation coefficient to measure the quality of the ranking. The largest

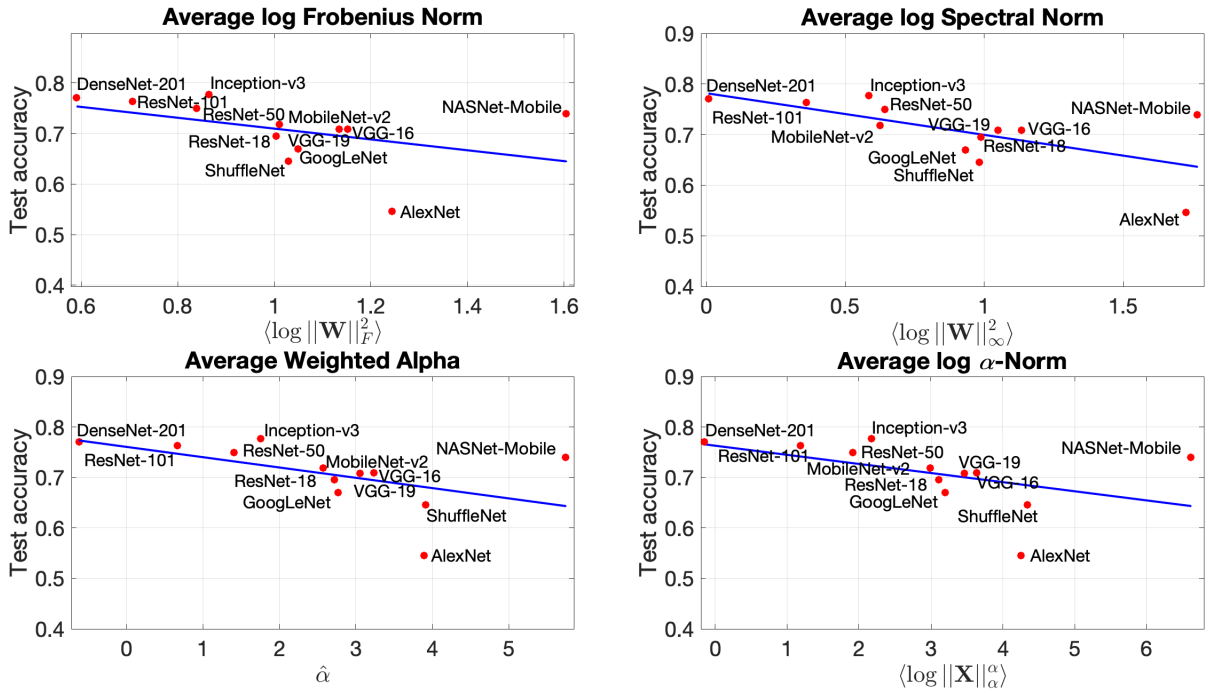


Fig. 11. Accuracy of 12 deep CNNs on the ILSVRC 2012 validation dataset against the prediction metrics from [22]. The prediction metrics were obtained using full sets of networks' weights.

mean τ was 0.71, which is not a perfect ranking ($\tau = 1$), but it is far better than random ranking ($\tau = 0$). Perhaps not surprisingly, these results are not as good as the results obtained for real hidden layer activations, but they still suggest that the perceptron itself contains rich information about possible performance of a network, which is in line with the observations in [8]. For predictions obtained from the readout perceptron by other estimation methods, please refer to Section IV-C7.b.

7) Existing Prediction Methods for ANNs: This section provides the sense on how the predictions produced by the perceptron theory compare to predictions by other existing methods. As indicated in Section II, most of the related methods rely on training estimator models. While these methods are useful in applications, such as NAS, they hardly provide novel insights into the principles underlying ANNs. Notably, however, there are recent studies [22], [25] that do not require any training of the estimator model, similar to our perceptron theory case. Therefore, we compared our results to several metrics proposed in [22] and to the best performing metric in [25]. All the metrics proposed in [22] are based on the weights of the trained ANN. While the investigation in [25] was concerned with searching for simple "zero-cost" metrics that would release the computational burden of performing NAS where the metrics were inspired from the literature related to ANNs' weights pruning [27], [29], [30]. For the sake of brevity, we omit the details of calculating the considered metrics, but refer to the corresponding equations in [22] and [25]:

1) average log Frobenius norm (denoted as $\langle \log \|\mathbf{W}\|_F^2 \rangle$; see (3) in [22]);
2) average log spectral norm (denoted as $\langle \log \|\mathbf{W}\|_\infty^2 \rangle$; see (4) in [22]);
3) average weighted alpha (denoted as $\hat{\alpha}$; see (10) in [22]);
4) average log α-norm (denoted as $\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$; see (11) in [22]);
5) `synflow`; see (1) in [25].

Among the metrics considered in [25], we use `synflow`, since it was empirically demonstrated to perform most consistently across all the alternatives. A small peculiarity was that we found that using log(`synflow`) significantly improved Pearson's r, so it was used to report the results. Since not all of the networks considered in this study were available in the software distribution for reproducing [22], in the experiments below, we limited the ANNs to the subset of 12 deep CNNs (Xception, NASNet-Mobile, and NASNet-Large were excluded).

a) Predictions obtained from complete networks: First, to provide the comparison to the perceptron theory predictions reported in Figs. 4 and 8, we obtained the metrics using full sets of weights for each ANN. Fig. 11 presents the results where each panel corresponds to a metric. Table I[8] summarizes the quality of predictions in terms of two correlation coefficients used above that were calculated with the actual accuracies: Pearson's r and Kendall's τ. Besides the statistical metrics from [22] and `synflow`, the table also reports the coefficients for the corresponding predictions (i.e., excluding three CNNs mentioned above) obtained for the perceptron theory according to (2) (see Fig. 4) and (3) (see Fig. 8).

Among the statistical metrics from [22], the average log spectral norm had the best Pearson's r, which is consistent with the original study. It is also worth pointing out that `synflow` outperformed all the metrics from [22], which might not be that surprising given that it was empirically shown to be performing well in the scenario that requires ranking of ANNs. Nevertheless, when this result is compared with the perceptron theory, we observe decent improvement in the quality of predictions in favor of the perceptron theory. It should be also noted that the above experiment used networks with different architectures; e.g., Martin et al. [22]

[8]Strictly speaking, the Pearson correlation coefficients for all metrics from [22] are negative. However, for the sake of presentation clarity, their absolute values are reported. The same note applies to the results reported in Table II.

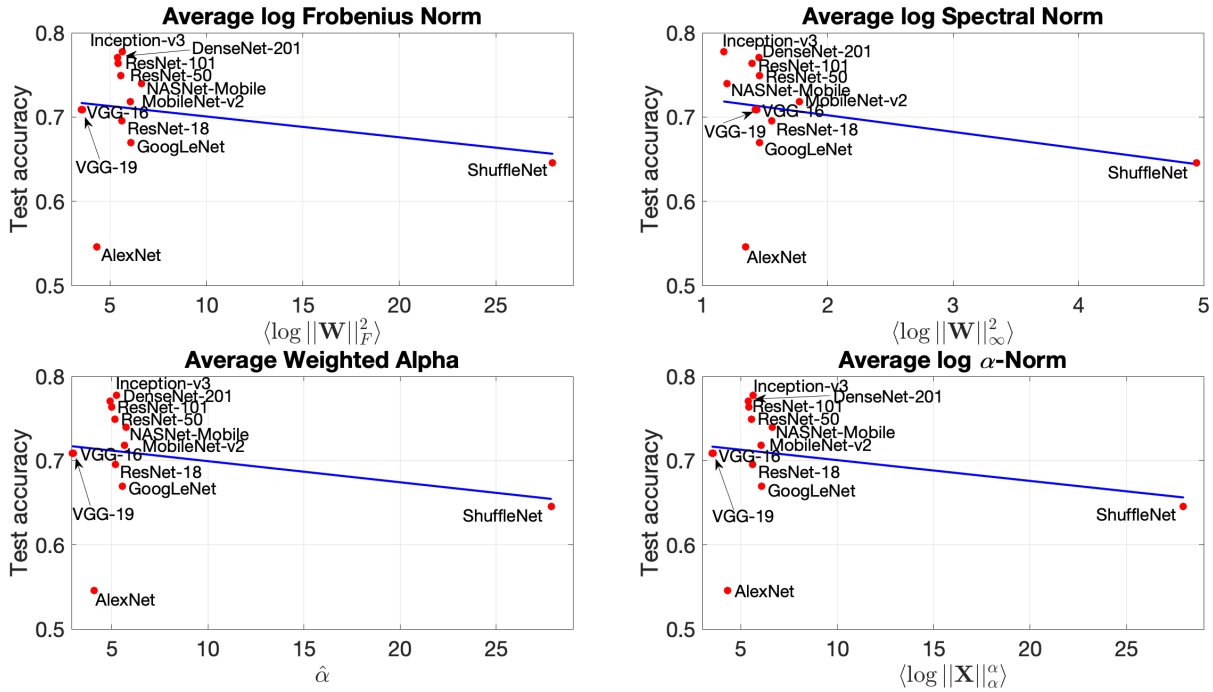


Fig. 12. Accuracy of 12 deep CNNs on the ILSVRC 2012 validation dataset against the prediction metrics from [22]. The prediction metrics were obtained using only the weights of the readout perceptrons.

TABLE I

QUALITY OF PREDICTIONS OBTAINED FROM THE COMPLETE NETWORKS

Method	Source	Pearson's r	Kendall's τ
Average log Frobenius norm	[22]	0.43	0.42
Average log spectral norm	[22]	0.65	0.42
Average weighted alpha	[22]	0.52	0.52
Average log α-norm	[22]	0.48	0.52
log `synflow`	[25]	0.66	0.71
Perceptron theory, Eq. (2)	our	0.90	0.79
Perceptron theory, Eq. (3)	our	**0.98**	**0.88**

TABLE II

QUALITY OF PREDICTIONS OBTAINED FROM THE READOUT PERCEPTRONS

Method	Source	Pearson's r	Kendall's τ
Average log Frobenius norm	[22]	0.25	0.09;
Average log spectral norm	[22]	0.31	0.30
Average weighted alpha	[22]	0.26	0.06
Average log α-norm	[22]	0.25	0.09
Perceptron theory, Eq. (2)	our	**0.66**	**0.61**

indicated that their metrics should be used with caution when comparing networks with different architectures. To us, the results in Table I are an indicator that to be able to achieve high quality of predictions among different architectures, and the knowledge of statistics of the postsynaptic sums might be an essential information.

b) Predictions obtained from readout perceptrons: To be consistent with the setup of Section IV-C6, we obtained the statistical metrics when using only the corresponding readout perceptrons (`synflow` was not considered here). These results are presented in Fig. 12; each panel corresponds to a metric. Similar to Table I, Table II reports the overall quality of predictions where the results for the perceptron theory were obtained according to the methodology described in Section IV-C6. As it could be seen from the table, similar to the previous experiment, the perceptron theory provided much higher quality of predictions. Notably, the results of the perceptron theory in Table II are comparable to those of the statistical methods from the previous experiment (see Table I). Finally, it is also worth emphasizing that in this experiment's setup, all the methods (including the perceptron theory) formed the predictions without the access to the activations of the last hidden layer (i.e., only a subset of network's weights was used), which suggests that in the case of different architectures, the perceptron theory is competitive with the state-of-the-art methods.

V. DISCUSSION

A. Summary of Results

We present a theory for classification with one-layer perceptrons, which is general in the sense that it applies to networks formed by any learning rule or optimization procedure. The curious fact that the theory for perceptrons, the earliest ANN models [31], cannot be found in textbooks might be due to the lack of universality of one-layer perceptrons as a general function approximator [38]. Here, we trace the late and gradual development of a general perceptron theory, which occurred in the context of neuroscience [39] and more complex neural networks [11], [39]. The presented perceptron theory generalizes the earlier versions and, thus, is applicable to a wider variety of neural networks that contain a perceptron-like output layer.

We first verified that our formulation of the perceptron theory (3) can accurately predict the performance of ESNs, even with the readout perceptron optimized by linear regression, the case that could not be treated with the previous formulation of the theory [11]. Next, we investigated the application of the perceptron theory to shallow and deep networks. The empirical evaluation of the predicted accuracy was performed on numerous classification datasets with shallow networks and on ImageNet with more than a dozen deep neural network architectures. In both cases, we observed high correlation between predicted and actual accuracies even when assumptions in the theory definition (Section III) were violated, for example, independence of postsynaptic sum

distributions for different classes or Gaussianity. However, neglecting dependencies introduced a bias to the predicted accuracies (Section IV-C3), which can be mitigated empirically (Section IV-C4). Alternatively, we also offer a general formulation in (3) that takes the dependencies into account. One might argue that a theory that predicts accuracies based on the activations in the last hidden layer is not useful, because the accuracies could be computed directly by going through the last layer. Note, however, that the theory does not require the activations explicitly, and it only requires some low-order statistical moments of the postsynaptic sums in the output neurons, which can be cheaply collected during the execution time. In addition, the theory also identifies the low-order statistical moments of the postsynaptic sums as essential for determining the accuracy of the network.

B. Applications and Future Directions

We foresee the following applications of this study.

1) The proposed perceptron theory is applicable to any network architecture with perceptron-like output layer. Thus, the potential application range goes far beyond the network architectures we have investigated in this article.
2) The proposed theory enables ranking of networks with different architectures for a particular task by accessing only low-order statistical moments of the postsynaptic sums in the output neurons, which is important as it avoids a common issue of data privacy.
3) A more advanced application of our theory is for comparing networks without having any access to the data or the transformation stage of the network. Our experiments showed that the results are less accurate, but maybe still sufficient as the first filtering step in a multistage evaluation process for granting access to the data only to the most promising networks.
4) Finally, our approach can not only predict classification accuracies but also be extended to estimate probabilities of observing a certain vector of postsynaptic sums in the output layer. This estimation can potentially be used to detect adversarial examples and other outliers.

The presented results suggest several novel directions for future investigation. One interesting question is whether the perceptron theory could be used to design more efficient or faster training algorithms, perhaps by using the theoretical predictions for identifying classes that need more training. Another obvious research direction is efficient numerical evaluation of the theory formulas. It would be interesting to compare existing methods for numerical multidimensional integration in their ability to evaluate the more precise formulation in (3) for larger number of classes.

Appendix A
Analyzing Short-Term Memory in ESNs

The theory in [11] was developed for studying the short-term memory of distributed representations in such models as HD/VSA[9] and recurrent randomly connected networks within reservoir computing (e.g., ESNs [58]).[10] In particular, here, we consider the trajectory association task [41], [91], [92] (see below) as one of the ways of studying the short-term memory of a simplified version of the ESN [83]. The ability to form and use the short-term memory is a key enabler for many HD/VSA use cases, such as representation of data structures [93], [94], [95], [96], [97], processing of strings [98], [99], [100], [101], and communications [102], [103], [104]. Below, we will introduce the trajectory association task together with the simplified version of the ESN. We kindly refer readers to [59] for a step-by-step tutorial on applying ESNs; to [105] for a broader overview of the area; and to [106] for the aspects of physical realization.

The trajectory association task has two stages: memorization and recall. At the memorization stage, at every time step m, the ESN stores a symbol $\mathbf{s}(m)$ from the sequence of symbols $\mathbf{s}$ to be memorized. The number of unique symbols (i.e., alphabet size) is denoted as D. The symbols are represented using N-dimensional random bipolar dense vectors stored in the codebook $\mathbf{\Phi} \in \{-1, 1\}^{N\times D}$. Thus, at every time step m, the ESN is presented with the corresponding N-dimensional vector $\mathbf{\Phi}_{\mathbf{s}(m)}$, which is added to the hidden layer of the ESN ($\mathbf{x} \in \mathbb{Z}^{N\times 1}$). The state of the hidden layer at time step m [denoted as $\mathbf{x}(m)$] is updated as follows:

$$\mathbf{x}(m) = f_\kappa(\rho(\mathbf{x}(m-1)) + \mathbf{\Phi}_{\mathbf{s}(m)})$$

where $\mathbf{x}(m-1)$ is the previous state of the hidden layer at time step $m-1$; ρ denotes the permutation operation (e.g., circular shift to the right), which acts as a simple variant of a recurrent connectivity matrix; $f_\kappa(x)$ is a clipping function—nonlinear activation function, which keeps the values of the hidden layer in the limited range using a threshold value κ as follows:

$$f_\kappa(x) = \begin{cases} -\kappa, & x \le -\kappa \\ x, & -\kappa < x < \kappa \\ \kappa, & x \ge \kappa. \end{cases}$$

In practice, the value of κ regulates the recency effect of the ESN.

At the recall stage, the ESN uses the content of its hidden layer $\mathbf{x}(m)$ as the query vector to retrieve the symbol stored d steps ago, where d denotes the delay. In the experiments below, the range of the delays varied between 0 and 25. The recall is done by using the readout perceptron for particular d, which contains one N-dimensional vector (linear filter) per each symbol. The readout perceptron is denoted as $\mathbf{W}^d \in \mathbb{R}^{D\times N}$, and the recall is done as follows:

$$\hat{\mathbf{s}}(m-d) = \arg\max(\mathbf{W}^d\mathbf{x}(m))$$

where $\arg\max(\cdot)$ returns the symbol with the highest postsynaptic sum among the output neurons for the chosen delay $\mathbf{W}^d$ value and the given hidden layer state $\mathbf{x}(m)$.

Let us consider two approaches of forming the readout perceptron. First, it can be constructed as done usually in HD/VSA from the codebook $\mathbf{\Phi}$ and the reverse permutation by d

$$\mathbf{W}^d = \rho^{-d}(\mathbf{\Phi}^\top). \tag{8}$$

The advantage of this approach is that no training is required to obtain the readout perceptron.

An alternative approach, which is more native for ESNs, is to obtain $\mathbf{W}^d$ via solving a linear regression on a given training sequence and the corresponding states of the hidden layer. The advantage of this approach is that, as we have seen

[9] A comprehensive two-part survey of HD/VSA is available in [42] and [43].

[10] It is worth noting here that HD/VSA and reservoir computing have multiple points of connection. There is both a direct connection that has been drawn in [11] and [83] as well as an implicit connection via cellular automata computations [84], [85], [86] that have also been found useful within HD/VSA [87], [88], [89], [90].

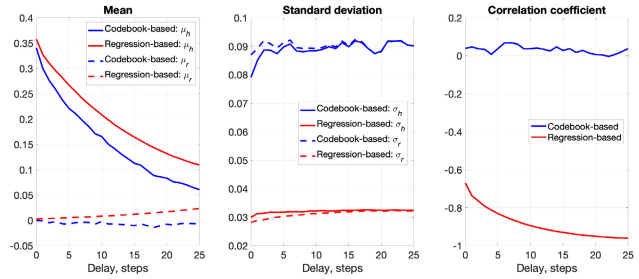


Fig. 13. Statistics extracted for the case of codebook-based and regression-based perceptrons. The following values for the ESN parameters were used: $N = 100$, $D = 2$, and $\kappa = 4$. The length of test sequences was 10 000. All reported valued were averaged over 50 simulations with random codebooks.

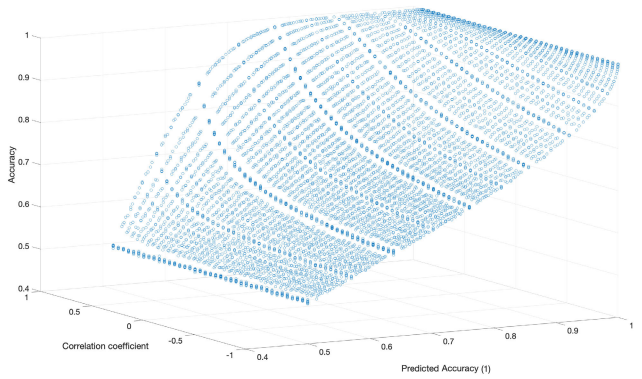


Fig. 14. Actual accuracy of a synthetic datasets for different Pearson correlation coefficients and predicted accuracy calculated assuming that the correlation between classes is zero, i.e., according to (1)/(2)/(9), which are all the same for $D = 2$.

in Fig. 1 in the main text, it has higher accuracy than the codebook-based approach.

While it is straightforward to simulate the presented network and obtain the accuracies for different values of d empirically, predicting the accuracy analytically given only the parameters (N, D, d, and κ) is challenging. Nevertheless, [11] has proposed a solution to this problem when the perceptron is based on the codebook as in (8). The solution includes two components. The first one is the equation for calculating the predicted accuracies a (i.e., the expected accuracy)

$$\begin{aligned} a &:= p(\mathbf{s}(m-d) = \hat{\mathbf{s}}(m-d)) \\ &= \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}\sigma_h} e^{-\frac{(x-\mu_h)^2}{2\sigma_h^2}} (\Phi(x, \mu_r, \sigma_r))^{D-1} \end{aligned} \tag{9}$$

where μ_h and σ_h denote the mean and standard deviation of the postsynaptic sum (i.e., dot product) between $\mathbf{x}(m)$ and the row of $\mathbf{W}^d$ (i.e., linear filter) corresponding to the correct symbol $\mathbf{s}(m-d)$, while μ_r and σ_r denote the mean and standard deviation of the postsynaptic sum for all other symbols in the codebook. Note that (9) is equivalent to (1) in the main text. Moreover, (9) is a special case of (2) where for all symbols in the codebook other than the correct symbol the same μ_r and σ_r are assumed, that is, because the codebook-based perceptron uses $\mathbf{\Phi}$, where representations for different symbols are random, and therefore, for large N, they are quasi-orthogonal to each other. Due to this fact, we know that the expected value of μ_r is 0 and that of σ_r is $(N\kappa(\kappa+1)/3)^{1/2}$. However, the values of μ_h and σ_h depend on the given values of d and κ; therefore, the second component of the solution determines them (please refer to [11, Eqs. (2.44)–(2.47)]).

It is also worth mentioning that (3) is the generalization of (2), because once we assume that $\mathbf{\Sigma}$ is diagonal (i.e., variables in $\mathbf{x}$ are independent), (3) can be simplified to (2) as follows:

$$\begin{aligned} \mathbf{a}_i &= \int_{-\infty}^{\infty} \int_{-\infty}^{\mathbf{x}_1} \cdots \int_{-\infty}^{\mathbf{x}_1} p(\mathbf{x}, \boldsymbol{\mu}, \mathbf{\Sigma}) d\mathbf{x}_L \cdots d\mathbf{x}_1 \\ &= \int_{-\infty}^{\infty} p(\mathbf{x}_1) d\mathbf{x}_1 \int_{-\infty}^{\mathbf{x}_1} p(\mathbf{x}_2) d\mathbf{x}_2 \cdots \int_{-\infty}^{\mathbf{x}_1} p(\mathbf{x}_L) d\mathbf{x}_L \\ &= \int_{-\infty}^{\infty} p(\mathbf{x}_1, \boldsymbol{\mu}_1, \boldsymbol{\sigma}_1) d\mathbf{x}_1 \Phi(\mathbf{x}_1, \boldsymbol{\mu}_2, \boldsymbol{\sigma}_2) \cdots \Phi(\mathbf{x}_1, \boldsymbol{\mu}_L, \boldsymbol{\sigma}_L) \\ &= \int_{-\infty}^{\infty} p(\mathbf{x}_1, \boldsymbol{\mu}_1, \boldsymbol{\sigma}_1) d\mathbf{x}_1 \prod_{j=2}^{L} \Phi(\mathbf{x}_1, \boldsymbol{\mu}_j, \boldsymbol{\sigma}_j). \end{aligned}$$

Note that in the case of the regression-based perceptron, one cannot assume the independence of linear filters in the perceptron. Therefore, currently, there is no way of analytically estimating μ_h, μ_r, σ_h, and σ_r for the regression-based perceptron. Nevertheless, these values could be estimated empirically using the simulations. Fig. 13 presents μ_h, μ_r, σ_h, and σ_r for different values of d for both regression-based (red color) and codebook-based (blue color) perceptrons.[11] Notice, that there are several important differences between the statistics observed for different perceptrons. First, μ_h for the regression-based perceptron is higher than that of the codebook-based perceptron; however, for both perceptrons, μ_h is decreasing with the increased delay, which is expected. Second, both σ_h and σ_r are much lower in the case of the regression-based perceptron. Both facts should positively affect the accuracy. Third, there is a strong negative correlation between the linear filters in the regression-based perceptron. As it was shown in Fig. 1 in the main text, the presence of correlation hindered the applicability of (9), but the use of (3) allowed getting the correct accuracy.

Finally, let us explain how the original problem: analysis of trajectory association with ESN relates to predicting the accuracy of a neural network. First, in the case of classification, we do not assume delay, so we can safely say that $d = 0$. Second, the activation of the last hidden layer of a neural network corresponds to the hidden layer activity in the considered ESN. Note, however, that there is no guarantee that these activations are noisy versions of the entries of some fixed random matrix, as the activations come from (usually continuous) real data. Finally, the weights of the output layer of a neural network are more similar to the regression-based perceptron than to the codebook-based perceptron, because they are obtained through an optimization procedure (e.g., error backpropagation). This implicitly explains the bias observed in Fig. 4 when using (2) to calculate the predicted accuracy. Recall, that in Fig. 4, we have observed both underestimation and overestimation of the empirical accuracy, while in Fig. 1, we have observed only the overestimation. Therefore, in Section B, we will study the effect of the correlation on the results produced by (2) or (1)/(9).

Appendix B
Effect of Correlations

In Fig. 1 in the main text, we have seen that even in the case of two symbols (or two classes), the presence of

[11] Since the regression-based and codebook-based perceptrons might have different norms, Fig. 13 was obtained using cosine similarities instead of postsynaptic sums. Obviously, the usage of the cosine similarity does not affect neither the accuracy nor the applicability of (9) to the estimated statistics.

correlation might result in inaccuracies between the predicted accuracy and the actual accuracy. Nevertheless, using (1)/(9) or (2) to calculate the predicted accuracy is tempting, because, numerically, it is a simple task compared with (3). Therefore, one interesting question is whether it is possible to compensate bias introduced by (1)/(9) or (2) without the need of, e.g., calculating the accuracies on subproblems as in Fig. 6.

In order to study this, we created a synthetic binary classification problem ($D = 2$), assuming that the mean value of postsynaptic sums of the incorrect class is 0, while that of the correct class is a parameter. It was also assumed that postsynaptic sums distributions of both classes have the same standard deviation, which is also a parameter. Finally, the postsynaptic sum distributions can correlate with each other positively or negatively, and their Pearson correlation coefficient is a parameter. Note that when there is no correlation between the classes, we can perfectly use (1)/(9) or (2) for the given mean and standard deviation values. We can also use (1)/(9) or (2) even when there is a correlation, but in this case, the predicted accuracy would differ from the actual accuracy. In order to see the effect of the correlation between classes on the accuracy, we simulated many cases for different values of correlation, mean, and standard deviation. Fig. 14 depicts the results. As we can see, there is a highly nonlinear relation among the predicted accuracy, the Pearson correlation coefficient, and the empirical accuracy. Qualitatively, we expect that in the case of a negative correlation, the accuracy will be lower than the predicted accuracies [see Fig. 1 (red lines)] and vice versa in the case of the positive correlation. However, it seems hard to make more quantitative correction without making a lookup table like structure, which would interpolate the expected accuracy value for the given correlation and predicted accuracy. While this is possible to do for the binary classification problems, this solution becomes inadequate for larger values of D, because we would have to deal with many interactions in the covariance matrix, which cannot be simply stored as a compact lookup table.

REFERENCES

[1] N. A. Smuha, "The EU approach to ethics guidelines for trustworthy artificial intelligence," *Comput. Law Rev. Int.*, vol. 20, no. 4, pp. 97–106, Aug. 2019.

[2] *Preparing for the Future of Artificial Intelligence*, Executive Office of the President National Science and Technology Council Committee on Technology, Washington, DC, USA, 2016.

[3] V. Papyan, X. Y. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 40, pp. 24652–24663, Oct. 2020.

[4] R. Geirhos et al., "Shortcut learning in deep neural networks," *Nature Mach. Intell.*, vol. 2, no. 11, pp. 665–673, Nov. 2020.

[5] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017, *arXiv:1703.00810*.

[6] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

[7] M. Egmont-Petersen, J. L. Talmon, J. Brender, and P. McNair, "On the quality of neural net classifiers," *Artif. Intell. Med.*, vol. 6, no. 5, pp. 359–381, Oct. 1994.

[8] T. Unterthiner, D. Keysers, S. Gelly, O. Bousquet, and I. Tolstikhin, "Predicting neural network accuracy from weights," 2020, *arXiv:2002.11448*.

[9] C. DeChant, S. Han, and H. Lipson, "Predicting the accuracy of neural networks from final and intermediate layer outputs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–6.

[10] H. Jaeger, "Long short-term memory in echo state networks: Details of a simulation study," School Eng. Sci., Jacobs Univ., Bremen, Germany Tech. Rep., 27, 2012.

[11] E. P. Frady, D. Kleyko, and F. T. Sommer, "A theory of sequence indexing and working memory in recurrent neural networks," *Neural Comput.*, vol. 30, no. 6, pp. 1449–1513, 2018.

[12] R. Féraud and F. Clérot, "A methodology to explain neural network classification," *Neural Netw.*, vol. 15, no. 2, pp. 237–246, Mar. 2002.

[13] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller, "How to explain individual classification decisions," *J. Mach. Learn. Res.*, vol. 11, no. 6, pp. 1803–1831, 2010.

[14] B. Deng, J. Yan, and D. Lin, "Peephole: Predicting network performance before training," 2017, *arXiv:1712.03351*.

[15] R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, and A. C. I. Malossi, "TAPAS: Train-less accuracy predictor for architecture search," in *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI)*, vol. 33, 2019, pp. 3927–3934.

[16] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[17] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–19.

[18] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.

[19] A. Dhurandhar, K. Shanmugam, R. Luss, and P. A. Olsen, "Improving simple models with confidence profiles," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 10296–10306.

[20] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digit. Signal Process.*, vol. 73, pp. 1–15, Feb. 2018.

[21] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.

[22] C. H. Martin, T. Peng, and M. W. Mahoney, "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," *Nature Commun.*, vol. 12, no. 1, pp. 1–13, Jul. 2021.

[23] E. Hoffer, I. Hubara, and D. Soudry, "Fix your classifier: The marginal value of training the last weight layer," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–11.

[24] D. Zhou et al., "EcoNAS: Finding proxies for economical neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11396–11404.

[25] M. S. Abdelfattah, A. Mehrotra, L. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–17.

[26] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 7588–7598.

[27] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–15.

[28] J. Turner, E. J. Crowley, M. O'Boyle, A. Storkey, and G. Gray, "Block-Swap: Fisher-guided block substitution for network compression on a budget," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–15.

[29] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–11.

[30] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6377–6389.

[31] F. Rosenblatt, "The perceptron—A perceiving and recognizing automaton," Cornell Aeronaut. Lab., Buffalo, NY, USA, Tech. Rep., 85-460-1, 1957.

[32] F. Rosenblatt, "Principles of neurodynamics. Perceptrons and the theory of brain mechanisms," Cornell Aeronaut. Lab., Buffalo, NY, USA, Tech. Rep. VG-1196-G-8, 1961.

[33] K. Steinbuch and U. A. W. Piske, "Learning matrices and their applications," *IEEE Trans. Electron. Comput.*, vol. EC-12, no. 6, pp. 846–862, Dec. 1963.

[34] S. Amari and M. A. Arbib, "Competition and cooperation in neural nets," in *Systems Neuroscience*. New York, NY, USA: Academic, 1977, pp. 119–165.

[35] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Netw.*, vol. 1, no. 1, pp. 17–61, 1988.

[36] W. Maass, "On the computational power of winner-take-all," *Neural Comput.*, vol. 12, no. 11, pp. 2519–2535, 2000.

[37] W. Peterson, T. Birdsall, and W. Fox, "The theory of signal detectability," *Trans. IRE Prof. Group Inf. Theory*, vol. 4, no. 4, pp. 171–212, Sep. 1954.

[38] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[39] B. Babadi and H. Sompolinsky, "Sparseness and expansion in sensory representations," *Neuron*, vol. 83, no. 5, pp. 1213–1226, Sep. 2014.

[40] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognit. Comput.*, vol. 1, no. 2, pp. 139–159, Oct. 2009.

[41] T. A. Plate, *Holographic Reduced Representations: Distributed Representation for Cognitive Structures*. Stanford, CA, USA: Center for the Study of Language and Information, 2003.

[42] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations," *ACM Comput. Surveys*, vol. 55, no. 6, pp. 1–40, Jul. 2023.

[43] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part II: Applications, cognitive models, and challenges," *ACM Comput. Surveys*, vol. 55, no. 6, pp. 1–40, Aug. 2022.

[44] D. Kleyko et al., "Vector symbolic architectures as a computing framework for emerging hardware," *Proc. IEEE*, vol. 110, no. 10, pp. 1538–1571, Oct. 2022.

[45] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk, "Sparse binary distributed encoding of scalars," *J. Autom. Inf. Sci.*, vol. 37, no. 6, pp. 12–23, 2005.

[46] E. Paxon Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer, "Computing on functions using randomized vector representations," 2021, *arXiv:2109.03429*.

[47] E. P. Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer, "Computing on functions using randomized vector representations (in brief)," in *Proc. Neuro-Inspired Comput. Elements Conf. (NICE)*, 2022, pp. 115–122.

[48] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals," *Proc. IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019.

[49] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5880–5898, Dec. 2018.

[50] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits Syst. Mag.*, vol. 20, no. 2, pp. 30–47, Jun. 2020.

[51] T. A. Plate, "Holographic reduced representations," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 623–641, May 1995.

[52] D. B. Owen, "A table of normal integrals," *Commun. Stat.-Simul. Comput.*, vol. 9, no. 4, pp. 389–419, 1980.

[53] A. Rahimi et al., "High-dimensional computing as a nanoscalable paradigm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.

[54] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Comput.*, vol. 25, no. 8, pp. 2038–2078, 2013.

[55] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Sekercioglu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1250–1262, Jun. 2017.

[56] D. A. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 261–276, Mar. 2001.

[57] E. P. Frady, D. Kleyko, and F. T. Sommer, "Variable binding for sparse distributed representations: Theory and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 3, 2021, doi: 10.1109/TNNLS.2021.3105949.

[58] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2003, pp. 593–600.

[59] M. Lukoševičeus, "A practical guide to applying echo state networks," *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700. Berlin, Germany: Springer, Aug. 2012, pp. 659–686.

[60] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[61] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[62] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, pp. 3133–3181, Jan. 2014.

[63] D. Kleyko, M. Kheffache, E. P. Frady, U. Wiklund, and E. Osipov, "Density encoding enables resource-efficient randomly connected neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3777–3783, Aug. 2021.

[64] E. M. Kussul, L. M. Kasatkina, D. A. Rachkovskij, and D. C. Wunsch, "Application of random threshold neural networks for diagnostics of micro machine tool condition," in *Proc. IEEE Int. Joint Conf. Neural Netw. IEEE World Congr. Comput. Intell. (IJCNN)*, vol. 1, May 1998, pp. 241–244.

[65] A. Goltsev and D. Rachkovskij, "Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings," *Pattern Recognit.*, vol. 38, no. 3, pp. 315–322, Mar. 2005.

[66] O. J. Räsänen and J. P. Saarinen, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1878–1889, Sep. 2015.

[67] D. Kleyko, E. Osipov, and D. A. Rachkovskij, "Modification of holographic graph neuron using sparse distributed representations," *Proc. Comput. Sci.*, vol. 88, pp. 39–45, Oct. 2016.

[68] D. Kleyko, E. Osipov, N. Papakonstantinou, and V. Vyatkin, "Hyperdimensional computing in industrial systems: The use-case of distributed fault isolation in a power plant," *IEEE Access*, vol. 6, pp. 30766–30777, 2018.

[69] C. Diao, D. Kleyko, J. M. Rabaey, and B. A. Olshausen, "Generalized learning vector quantization for classification in randomized neural networks and hyperdimensional computing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–9.

[70] P.-C. Huang, D. Kleyko, J. M. Rabaey, B. A. Olshausen, and P. Kanerva, "Computing with hypervectors for efficient speaker identification," 2022, *arXiv:2208.13285*.

[71] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2012, pp. 1097–1105.

[73] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[75] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[76] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[77] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[78] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[79] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

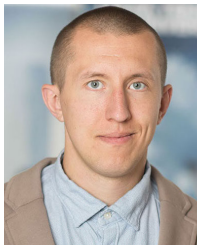
[80] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 4278–4284.

[81] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[82] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[83] D. Kleyko, E. P. Frady, M. Kheffache, and E. Osipov, "Integer echo state networks: Efficient reservoir computing for digital hardware," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1688–1701, Apr. 2022.

[84] O. Yilmaz, "Machine learning using cellular automata based feature expansion and reservoir computing," *J. Cellular Automata*, vol. 10, nos. 5–6, pp. 435–472, 2015.
[85] O. Yilmaz, "Symbolic computation using cellular automata-based hyperdimensional computing," *Neural Comput.*, vol. 27, no. 12, pp. 2661–2692, 2015.
[86] N. McDonald, "Reservoir computing & extreme learning machines using pairs of cellular automata rules," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2429–2436.
[87] D. Kleyko and E. Osipov, "No two brains are alike: Cloning a hyperdimensional associative memory using cellular automata computations," in *Biologically Inspired Cognitive Architectures (BICA)* (Advances in Intelligent Systems and Computing), vol. 636. Cham, Switzerland: Springer, 2017, pp. 91–100.
[88] M. Schmuck, L. Benini, and A. Rahimi, "Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 4, pp. 1–25, Oct. 2019.
[89] D. Kleyko, E. P. Frady, and F. T. Sommer, "Cellular automata can reduce memory requirements of collective-state computing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2701–2713, Jun. 2022.
[90] A. Menon et al., "Efficient emotion recognition using hyperdimensional computing with combinatorial channel encoding and cellular automata," *Brain Informat.*, vol. 9, no. 1, pp. 1–13, Dec. 2022.
[91] M. Hersche, S. Lippuner, M. Korb, L. Benini, and A. Rahimi, "Near-channel classifier: Symbiotic communication and classification in high-dimensional space," *Brain Informat.*, vol. 8, no. 1, pp. 1–15, Dec. 2021.
[92] D. Kleyko et al., "Efficient decoding of compositional structure in holistic representations," *Neural Comput.*, 2023.
[93] D. A. Rachkovskij, "Some approaches to analogical mapping with structure sensitive distributed representations," *J. Exp. Theor. Artif. Intell.*, vol. 16, no. 3, pp. 125–145, 2004.
[94] D. A. Rachkovskij and S. V. Slipchenko, "Similarity-based retrieval with structure-sensitive sparse binary distributed representations," *Comput. Intell.*, vol. 28, no. 1, pp. 106–129, 2012.
[95] E. Osipov, D. Kleyko, and A. Legalov, "Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2017, pp. 3276–3281.
[96] D. Kleyko, A. Rahimi, R. W. Gayler, and E. Osipov, "Autoscaling Bloom filter: Controlling trade-off between true and false positives," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3675–3684, Apr. 2020.
[97] T. Yerxa, A. Anderson, and E. Weiss, "The hyperdimensional stack machine," in *Proc. Cogn. Comput.*, 2018, pp. 1–2.
[98] D. V. Pashchenko, D. A. Trokoz, A. I. Martyshkin, M. P. Sinev, and B. L. Svistunov, "Search for a substring of characters using the theory of non-deterministic finite automata and vector-character architecture," *Bull. Electr. Eng. Informat.*, vol. 9, no. 3, pp. 1238–1250, Jun. 2020.
[99] D. Kleyko, E. Osipov, and R. W. Gayler, "Recognizing permuted words with vector symbolic architectures: A Cambridge test for machines," *Proc. Comput. Sci.*, vol. 88, pp. 169–175, Dec. 2016.
[100] A. Joshi, J. T. Halseth, and P. Kanerva, "Language geometry using random indexing," in *Proc. Int. Symp. Quantum Interact. (QI)*, 2016, pp. 265–274.
[101] D. A. Rachkovskij and D. Kleyko, "Recursive binding for similarity-preserving hypervector representations of sequences," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.
[102] D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, "Dependable MAC layer architecture based on holographic data representation using hyper-dimensional binary spatter codes," in *Multiple Access Communications* (Lecture Notes in Computer Science), vol. 7642. Berlin, Germany: Springer, 2012, pp. 134–145.
[103] H.-S. Kim, "HDM: Hyper-dimensional modulation for robust low-power communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[104] R. Guirado, A. Rahimi, G. Karunaratne, E. Alarcón, A. Sebastian, and S. Abadal, "Wireless on-chip communications for scalable in-memory hyperdimensional computing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.
[105] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, Aug. 2009.
[106] G. Tanaka et al., "Recent advances in physical reservoir computing: A review," *Neural Netw.*, vol. 115, pp. 100–123, Jul. 2019.



Denis Kleyko (Member, IEEE) received the B.S. degree (Hons.) in telecommunication systems and the M.S. degree (Hons.) in information systems from the Siberian State University of Telecommunications and Information Sciences, Novosibirsk, Russia, in 2011 and 2013, respectively, and the Ph.D. degree in computer science from the Luleå University of Technology, Luleå, Sweden, in 2018.

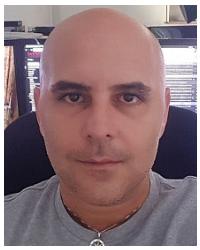
He is currently a Post-Doctoral Researcher on a joint appointment between the Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA, USA, and the Intelligent Systems Laboratory, Research Institutes of Sweden, Kista, Sweden. His current research interests include machine learning, reservoir computing, and vector symbolic architectures/hyperdimensional computing.

Antonello Rosato (Member, IEEE) received the M.S. degree (Hons.) in telecommunication engineering and the Ph.D. degree in information and communications technologies from the University of Rome "La Sapienza," Rome, Italy, in 2015 and 2019, respectively.

He is currently a Research Fellow with the University of Rome "La Sapienza." His current research interests include machine learning techniques for the analysis of complex behaviors, randomized neural networks, distributed training algorithms, and hyperdimensional computing systems.

Edward Paxon Frady received the B.S. degree in computation and neural systems from the California Institute of Technology (Caltech), Pasadena, CA, USA, in 2008, and the Ph.D. degree in neuroscience from the University of California at San Diego, La Jolla, CA, USA, in 2014.

He is currently a Research Lead with the Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA, USA. His research interests include neuromorphic engineering, vector symbolic architectures/hyperdimensional computing, and machine learning.



Massimo Panella (Senior Member, IEEE) received the five-year Laurea degree (Hons.) in electronic engineering and the Ph.D. degree in information and communication engineering from the University of Rome "La Sapienza," Rome, Italy, in 1998 and 2002, respectively.

He is currently a Full Professor of electrical engineering and computational intelligence with the University of Rome "La Sapienza." His research interests include computational intelligence and quantum computing for the modeling, optimization, and control of real-world systems, namely, the use of neural networks, fuzzy logic, evolutionary algorithms, and quantum circuits for solving both supervised and unsupervised learning problems, for time series analysis, and for the general processing of signals and data.

Friedrich T. Sommer received the Diploma degree in physics from the University of Tübingen, Tübingen, Germany, in 1987, the Ph.D. degree from the University of Düsseldorf, Düsseldorf, Germany, in 1993, and the Habilitation degree in computer science from Ulm University, Ulm, Germany, in 2002.

He is currently an Adjunct Professor with the Redwood Center for Theoretical Neuroscience, University of California at Berkeley, Berkeley, CA, USA, and a Researcher with the Neuromorphic Computing Laboratory, Intel Labs, Santa Clara, CA, USA. His research interests include neuromorphic engineering, vector symbolic architectures/hyperdimensional computing, and machine learning.