Efficient shallow learning mechanism as an alternative to deep learning

Ofek Tevet[a], Ronit D. Gross[a], Shiri Hodassman[a], Tal Rogachevsky[a], Yarden Tzach[a], Yuval Meir[a] and Ido Kanter[a,b*]

[a] Department of Physics, Bar-Ilan University, Ramat-Gan, 52900, Israel.
[b] Gonda Interdisciplinary Brain Research Center, Bar-Ilan University, Ramat-Gan, 52900, Israel.

[*] Corresponding author at: Department of Physics, Bar-Ilan University, Ramat-Gan, 52900, Israel. E-mail address: ido.kanter@biu.ac.il (I. Kanter).

Abstract

Deep learning architectures comprising tens or even hundreds of convolutional and fully-connected hidden layers differ greatly from the shallow architecture of the brain. Here, we demonstrate that by increasing the relative number of filters per layer of a generalized shallow architecture, the error rates decay as a power law to zero. Additionally, a quantitative method to measure the performance of a single filter, shows that each filter identifies small clusters of possible output labels, with additional noise selected as labels outside the clusters. This average noise per filter also decays for a given generalized architecture as a power law with an increasing number of filters per layer, forming the underlying mechanism of efficient shallow learning. The results are supported by the training of the generalized LeNet-3, VGG-5, and VGG-16 on CIFAR-100 and suggest an increase in the noise power law exponent for deeper architectures. The presented underlying shallow learning mechanism calls for its further quantitative examination using various databases and shallow architectures.

1. Introduction

Traditionally, neural network learning techniques stem from the dynamics of the brain[1, 2], however, these two scenarios are intrinsically different. One of the most prominent differences is the number of feedforward layers. Deep learning (DL) architectures typically consist of numerous convolutional and fully-connected (FC) hidden layers that can be increased to hundreds[3, 4]. These deep architectures enable the efficient supervised learning of complex classification tasks[5, 6], made possible by the advancement of powerful GPU technology. Contrastingly, the brain differs significantly from DL architectures and consists of very few feedforward layers[7-9], only one of which approximates convolutional wiring, mainly from the retinal input to the first hidden layer[7, 10]. Despite the shallow architecture and noisy and slow dynamics of the brain, it can efficiently perform complex classification tasks. The key objective of our research is to reveal the mechanism underlying efficient shallow learning that enables to achieve nontrivial classification tasks with the same accuracy as DL.

Until recently, the terminology of how DL works was based on features that are progressively revealed with the layers[5, 11-14]. The first convolutional layer (CL) reveals the local features of an input object, where large-scale features, features of features that characterize a class of inputs are progressively revealed in subsequent CLs[5, 15-17].

Recently, a quantitative method for explaining the underlying mechanism of successful DL has been presented[18, 19]. This enables quantifying the progressing accuracies with the layers and the functionality of each filter, consisting of the following three main stages. In the first stage, the entire deep architecture is trained using optimized parameters to minimize the loss function. In the second stage, the weights of the first selected number of trained layers remain unchanged, and their outputs are FC with random initial weights assigned to the output layer, representing the labels. Training only the FC layer indicates that the test accuracy progressively increases with the number of layers toward the output. In the last stage, the single-filter performance is estimated, where all the weights of the FC layer are silenced, except for the specific weights that emerge from a single filter. The test inputs are then

presented and influence the output units only through the small aperture of a single filter.

The results indicate that each filter identifies small clusters of possible output labels, with additional noise selected as labels outside the clusters. This noise per filter progressively decays with the layers, thereby resulting in enhanced signal-to-noise ratios and accuracies[19], which represents the mechanism underlying successful DL. In the following, we demonstrate that the mechanism underlying efficient shallow learning is similar to that of DL. However, the progressive effect with the layers is interchanged with increasing the number of filters in generalized shallow architectures.

2. Results

2.1. Generalized LeNet-3 trained on CIFAR-100

The generalized shallow architecture LeNet-3 consists of two 5×5 CLs, with d and $\frac{16}{6} \cdot d$ filters, respectively, preserving the LeNet-5 ratio ($d = 6$)[20] (Fig. 1(a)). The $25 \cdot \frac{16d}{6}$ outputs of the 2×2 max-pooling of the second CL, are FC to the 100 outputs, representing the CIFAR-100 labels (Fig. 1(a)).

The generalized LeNet-3 was first trained on CIFAR-100 using optimized parameters to minimize the loss function. The test error, ε, was observed to follow a power law as a function of the number of filters in the first CL, d (Fig. 2(a)), which was previously obtained for the generalized LeNet-5 trained on CIFAR-10[21]. For a given d and weights that minimized the loss function, the performance of each of the $\frac{16d}{6}$ filters was estimated using the following procedure: All the weights of the FC layer were silenced except for the specific 25 weights that emerged from a single filter (Fig. 1(b)). The CIFAR-100 test inputs were presented to the silenced LeNet-3 (Fig. 1(b)) and the results were represented by a 100×100 matrix M (Fig. 1(c), top). Element $M(i,j)$ represents the average fields generated by label i test inputs on output j, where the matrix elements are normalized by their maximal element. Next, a three valued (colors) clipped version of the matrix was calculated following a given threshold (Fig. 1(c), middle); $M(i,j) \geq threshold$ (white), $M(i,j) \leq -threshold$ (green)

and $|M(i,j)| < threshold$ (black). Permutations of these clipped matrix labels (Fig. 1(c), bottom) resulted in above-threshold diagonal clusters (white) of $1 \times 1, 2 \times 2$ and 4×4 (magnified upper-left corner red box). The above-threshold elements outside the diagonal clusters were defined as filter noise n (yellow elements in Fig. 1(c), bottom). The results indicate that the average noise per filter n follows an approximate power law as a function of d (Fig. 2(b)).

The extrapolation of the approximated power law of the test error, ε, as a function of d (Fig. 2(a)) indicates that using LeNet-3 with a large enough d, any small ε can be achieved. The mechanism underlying efficient shallow learning is the decrease of the average noise per filter, n, as a power law with increasing d.

The necessary condition of reduced n to enhance the accuracy was quantitatively formulated based on the detailed statistical features of the filters[18]. Here, we present only the following qualitative argument. The emergence of small clusters in a given filter, represented by the diagonal blocks of the permuted matrix (e.g., $1, 2,$ and 4 in Fig. 1(c), bottom), strongly enhances the knowledge of the input label. For example, an input with a strong field on an output unit belonging to a cluster of size 2 strongly indicates that the input belongs to one of these two labels. Each cluster identifies a small subset of possible output labels, whereas the noise n selects labels outside the cluster, thereby reducing the certainty of the input label. This quantitative argument is based on the single-filter performance, and the achievement of high accuracy emerges from the statistical selection of all filters[18], where many of their clusters select the correct label and induce low noise on the others.

A unique feature of the filters in LeNet-3 that is absent in deeper architectures, is the emergence of many negative matrix elements smaller than $-threshold$ (Fig. 1(c), green). A green off-diagonal element, $M(i,j)$, represents a strong repulsion, where the average fields generated by the label i test inputs on output j $(\neq i)$, is significantly negative. This type of repulsion may decrease the probability of error in the predicted output label, further enhancing the accuracy and balancing some of the noise, n. Indeed, the probability of green matrix elements in rows associated with the clusters (e.g., the seven upper rows in Fig. 1(c), bottom) is significantly higher than for those in other rows. For

example, the probability of green elements in a row associated with clusters, averaged over all 112 filters ($d \cdot \frac{16}{6}, d = 42$), is approximately five times higher than for other rows. This higher concentration of repulsive elements in the cluster rows enhances the probability of selecting an input label belonging to the cluster, thereby partially balancing the noise and enhancing the accuracy.

Simulations of LeNet-3 with various d values (Fig. 2) indicate that the average number of green elements below $-threshold$ decreases with d (not shown). The possible power-law scaling of the number of green elements with d is unclear and requires further examination. However, the conjugate trend of decreasing noise and repulsive elements with d suggests an efficient learning mechanism in highly noisy filter environments having a shallow architecture.

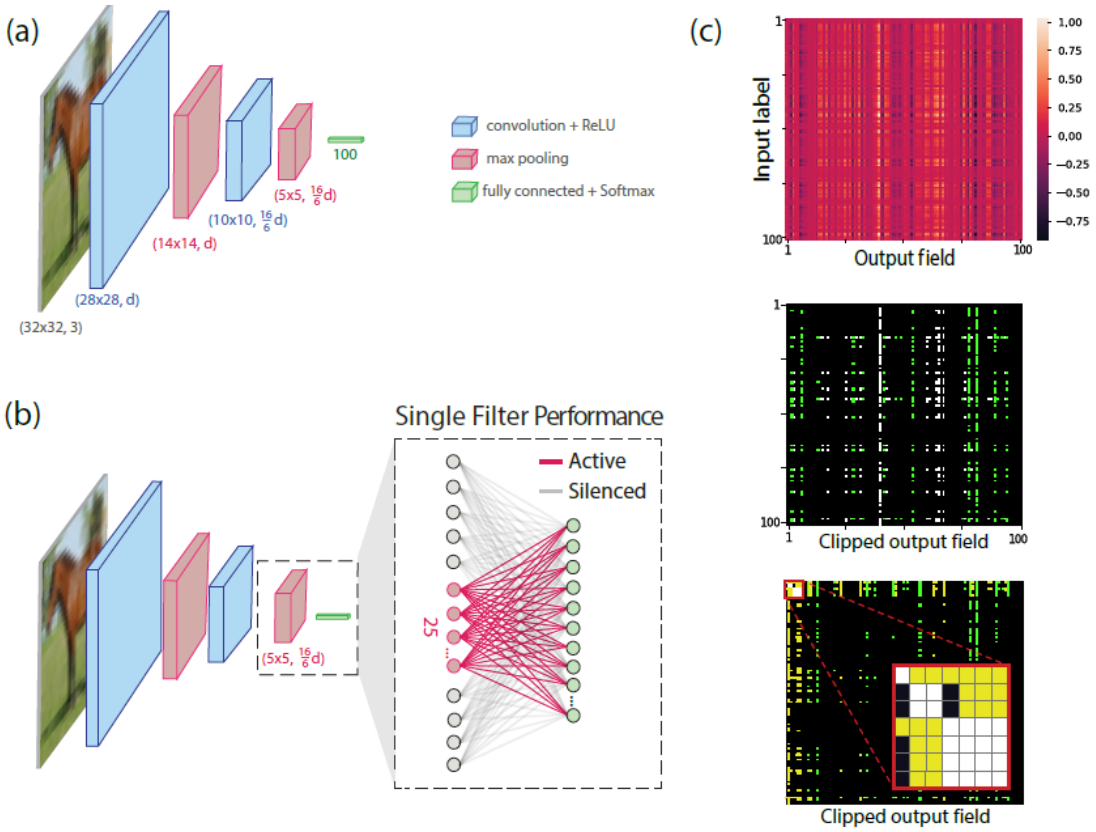

Fig. 1. Generalized LeNet-3 architecture and its single-filter performance. (a) Generalized LeNet-3 architecture for CIFAR-100, consisting of two 5×5 CLs

(light blue) with adjacent 2×2 max-pooling operators (light red). (b) Scheme of a single-filter performance consisting of 25 output units (red) that are FC to the 100 outputs (green), where the other weights are silenced (gray). (c) Matrix element $M(i,j)$ of a filter, as in panel b ($d = 42$), denotes the averaged fields generated by label i test inputs on an output j, where the matrix elements are normalized by their maximal element (top). The clipped matrix following a given threshold (middle), $M(i,j) \geq threshold$ (white), $M(i,j) \leq -threshold$ (green) and $|M(i,j)| < threshold$ (black). Permutations of the clipped matrix labels result in diagonal blocks (white, bottom), $1 \times 1, 2 \times 2,$ and 4×4 (magnified upper-left corner red box), where the above-threshold elements, $n \sim 280$, out of the clusters are noise elements (yellow).

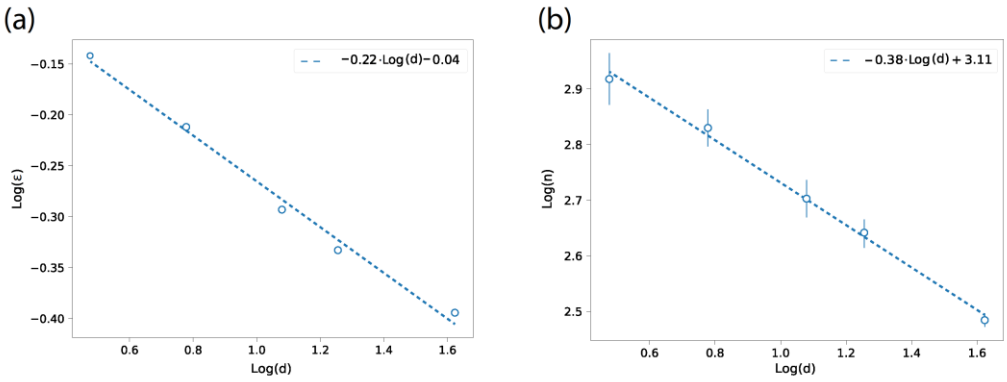

Fig. 2. Underlying mechanism of efficient learning of the generalized LeNet-3. (a) Power law of the error rate, ε, as a function of the number of filters in the first CL, d for generalized LeNet-3 trained on CIFAR-100 (Fig. 1(a)), where the standard deviations are comparable to the symbol size. (b) Power law of the average noise, n, as a function of the number of filters in the first CL, d, and their standard deviations.

2.2 Generalized VGG-5 trained on CIFAR-100

The second relatively shallow architecture examined is the generalized VGG-5, consisting of four consecutive pairs of 3×3 CLs and 2×2 max-pooling, terminating with $2 \times 2 \times 8 \times d$ input units which are FC to the 100 output units (Fig. 3(a)). The number of filters in the four CLs is $d \cdot 2^k, k = 0,\ 1, 2,$ and 3 (Fig. 3(a)). The test error decreases with d and can be approximated by a power law

as a function of d, with a similar small exponent, ~ -0.23, as for the generalized LeNet-3 (Figs. 2(a) and 3(b)). The average noise per filter measured at the FC layer also decays approximately as a power law with d, with an exponent ~ -0.61. Note that the clipped matrices associated with the $8 \cdot d$ filters (not shown) do not include green negative elements below $-threshold$.

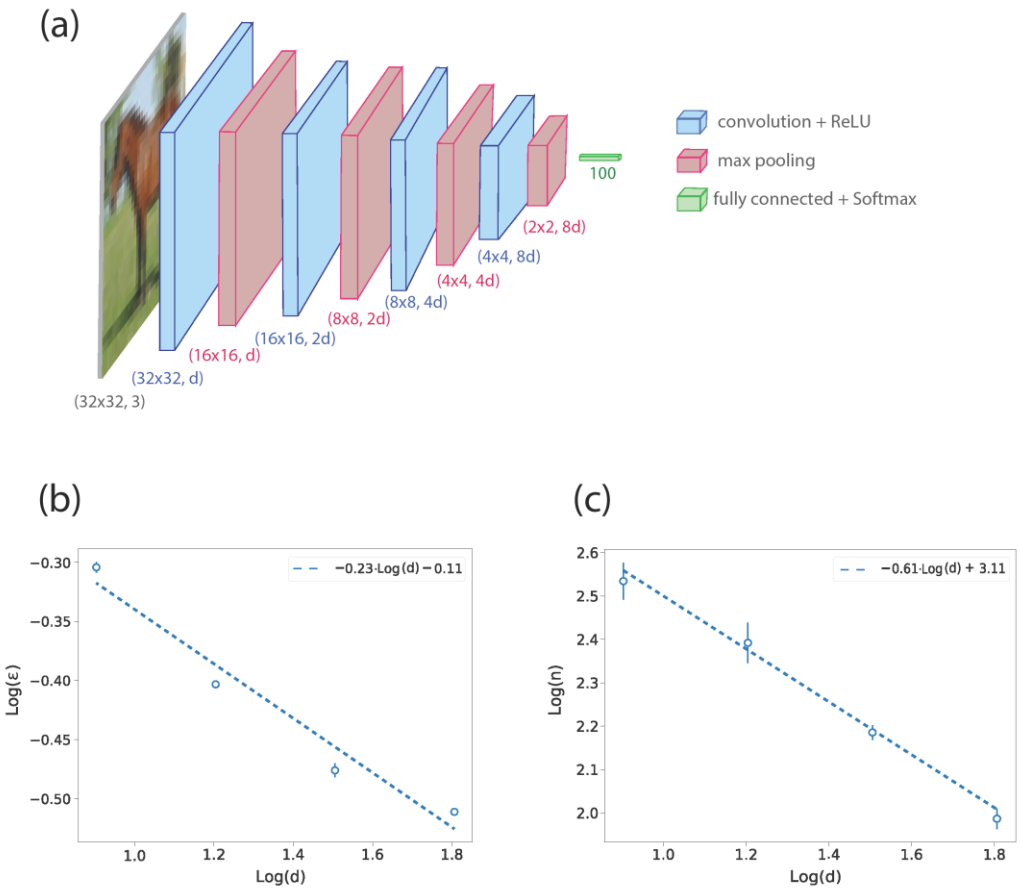

Fig. 3. Underlying mechanism of efficient learning on the generalized VGG-5. (a) Generalized VGG-5 architecture for CIFAR-100, consisting of four pairs of 3×3 CLs (light blue) and 2×2 max-pooling (light red), terminating with $2 \times 2 \times 8 \times d$ input units that are FC to the 100 output units (green). (b) Power law of the error rate, ε, as a function of d for generalized VGG-5 trained on CIFAR-100. (c) Power law of the average noise per filter, n, as a function of d, and their standard deviations.

2.3 Generalized VGG-16 trained on CIFAR-100

A comparison of the presented results for the generalized LeNet-3 and VGG-5 suggests that the power-law exponent of the test error is almost independent of the number of CLs (Figs. 2(a) and 3(b)). However, the power-law exponent of the average noise per filter increases for deeper architectures (Figs. 2(b) and 3(c)). To further examine this hypothesis, the results were extended to the generalized VGG-16, where the number of filters in CLs $(1-2),\ (3-4),\ (5-7),\ (8-13)$ are $d, 2\cdot d, 2^2\cdot d, 2^3\cdot d$, respectively[20].

In the first step, the generalized VGG-16 was trained on CIFAR-100 with optimized parameters[18]. Next, the weights of the first 10 trained layers were held unchanged, and their $2\times 2\cdot 8\cdot d$ outputs were FC to the output layer. Training this FC layer to minimize the loss function indicates that the test error, ε, is already saturating at the 10th layer[18]. In addition, ε as a function of d approximately scales as a power law with an exponent, ~ -0.29 (Fig. 4(a)), which slightly increases in comparison to VGG-5 (Fig. 3(a)). This power-law behavior is consistent with the recently obtained results for the generalized VGG-16 trained on CIFAR-10[21]. The average noise per filter, n, as a function of d, also scales approximately as a power law with an exponent of ~ -1.2 (Fig. 4(b)). A comparison of this result with the power-law noise exponents ~ -0.38 and ~ -0.61 for the generalized LeNet-3 and VGG-5 (Figs. 2(b) and 3(c)), respectively, supports the hypothesis that the power-law noise exponent decreases for deeper architectures consisting of increasing number of CLs. We note that below threshold elements (green in Fig. 1(c)) are absent in the clipped matrices of the 10th layer.

The decrease in the average noise per filter, n, as a function of d (Fig. 4(b)) also affects the noise distribution among the 512 filters of the 10th layer (Fig. 4(c)). The distribution became narrower as d increased from 8 to 64, where its standard deviation appears to scale linearly with the mean.

The decrease in n as a function of d in the 10th layer (Fig. 4(b)), did not reveal the characteristic features of noise in the much lower layers. Recently, the training of VGG-16 $(d=64)$ on CIFAR-100 indicated that the accuracy increases progressively from the 2nd layer to the 10th layer, where the average noise per filter n decreases[18]. In addition, the common features of the filter

noise in the 2nd layer of VGG-16 (Figs. 4(d) and (e)) resembled the clipped matrices of the filters in the second CL of LeNet-3 (Fig. 1(c)). Each filter selects several small clusters among the 100 labels. These clusters are accompanied by a large number of noise elements, mainly along their columns (Fig. 4(d), yellow) and with additional repulsive elements below $-threshold$ (Fig. 4(d), green), which mainly form several columns.

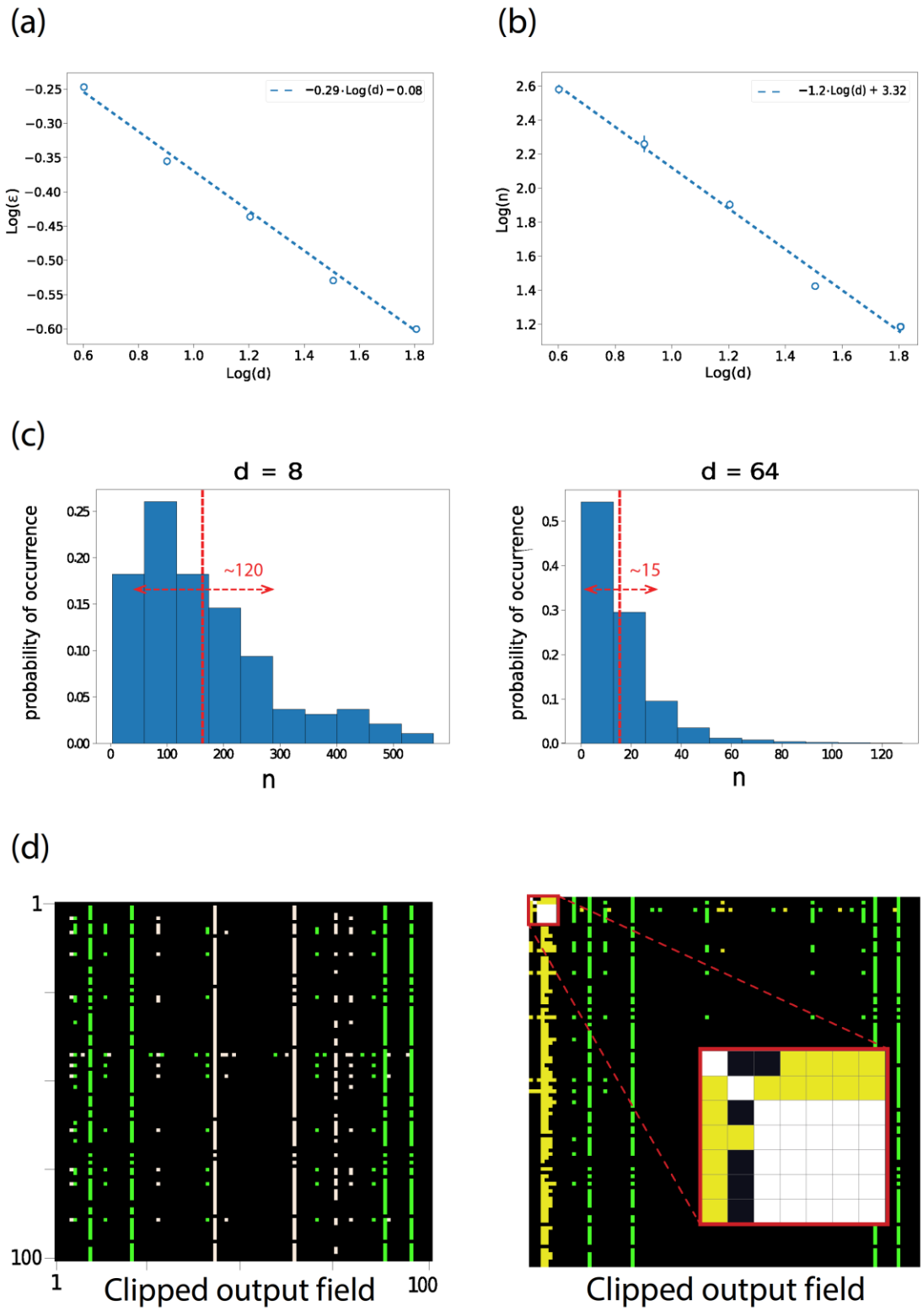

Fig. 4. Underlying mechanism of efficient learning of generalized VGG-16. (a) Power law of the error rate, ε, as a function of the number of filters in the first CL, d, measured at the output of the 10th layer (see text) for generalized VGG-

16 trained on CIFAR-100. (b) Power law of the average noise per filter, n, as a function of d, and their standard deviations. (c) Noise distribution of the 512 filters (averaged over five samples) at the 10th layer for $d = 8$ (left) and $d = 64$ (right). Average/standard deviation of the distributions ($\sim 162/120$ for $d = 8$ and $\sim 15.3/15$ for $d = 64$) are denoted by vertical/horizontal dashed-red lines, respectively. (d) The clipped matrix, M, of a filter at the 2nd layer of VGG-16 with $d = 64$, where $M(i,j) \geq threshold$ (white), $M(i,j) \leq -threshold$ (green) and $|M(i,j)| < threshold$ (black), and its permuted matrix with an element of the clusters (white) (similar to Fig. 1(c), middle and bottom). Above threshold elements outside the clusters are denoted as noise elements (yellow).

3. Discussion

The underlying mechanism of shallow learning was quantitatively examined for generalized LeNet-3 and VGG-5 architectures trained on CIFAR-100. A generalized architecture comprises a family of architectures with varying number of filters, d, in the first CL; while the ratio between the number of filters in consecutive CLs remains fixed. The discovery of the shallow-learning mechanism consists of the following two-step procedure: First, the generalized architecture is trained to minimize the loss function. Second, the single filter performance is estimated for the CL adjacent to the outputs, where all weights of the FC layer are silenced, except for the specific weights that emerge from a single filter.

The test inputs are then presented, which influence the output units only through the small aperture of a single filter. The results indicate that both test error, ε, and the average noise per filter, n, approximately decay as a power law with an increase in the number of filters in the first CL, d.

The efficient learning mechanism, which stems from the decrease in n, is common to both shallow and deep architectures. In deep architectures, n decreases progressively with the number of layers, whereas in generalized shallow architectures the decrease stems from the increasing the number of filters in the first CL, d. The tunable parameter d enables the observation of its approximated power-law scaling with n, whereas in a deep architecture, the tunable parameter is currently unclear. The preliminary results indicate a

decrease in the power-law exponent of n with d, for generalized shallow or deep architectures with increasing number of CLs. This result calls for its further quantitative examination using various databases and generalized shallow and deep architectures.

For the very shallow architecture, LeNet-3, the noise n is found to be partially balanced by another mechanism, the emergence of very negative elements. Preliminary results indicate that the number of those negative repulsive elements (Fig. 1(c)) decreases with d, however its scaling, similar to n, deserves further research. This mechanism was also observed in the second CL of VGG-16, but completely disappeared in the last CL of VGG-5 and VGG-16. These results suggest that repulsive elements are expected to be an additional common learning mechanism for the very first CLs in both shallow and deep architectures, however, their quantitative and statistical features deserve further research.

Finally, the efficient realization of shallow architectures with a large d requires a shift in the properties of advanced GPU technology. Above a critical number of filters, depending on the GPU properties and properties of the dataset, the running time of an epoch is significantly slowed down. The GPU based architectures are capable of efficiently calculate forward and backward steps for a hundred times more layers, but not for a hundred times more filters per layer. Therefore, the realization of efficient learning in shallow architectures, mimicking the brain's dynamics, requires a new type of hardware.

4. Methods

4.1 Architectures and training of the fully-connected layers

In this study, three architectures were examined: LeNet-3, a modified version of LeNet-5[22], constituting only one FC layer instead of three; VGG-5, a modification of VGG-6[11] constituting only four CLs instead of five; and VGG-16[11]. The architectures were trained to classify the CIFAR-100 dataset[15], with no biases on the output units to ensure that each filter’s effect on the output fields would be exemplified and not overshadowed by the much larger biases.

Removing the biases of the output layer did not affect the architectures' average accuracies, in comparison to architectures trained with output biases.

The evaluation was performed by taking an architecture trained on the entire dataset, cutting it at designated layers and training a new FC layer between the output of that specific layer and the output layer. During this training process, only the FC layer was trained, whereas the weights and biases of the rest of the architecture remained fixed. For VGG-16, the evaluation was performed on layer 10. For VGG-5 layer 4 was evaluated. For LeNet-3 the evaluation was performed without training a new FC layer.

4.2 Optimization

The cross-entropy cost function was selected for the classification task and minimized using the stochastic gradient descent algorithm[6, 13]. The maximal accuracy was determined by searching through the hyper-parameters (see below). Cross-validation was performed using several validation databases, each consisting of a randomly selected fifth of the training set examples. The average results were within the same standard deviation (Std) as the reported average success rates. The Nesterov momentum[23] and L2 regularization method[24] were applied.

4.3 Dataset and preprocessing

The image pixel in the CIFAR-100 dataset[15] were normalized to the range $[-1,1]$ by dividing by 255 (the maximal pixel value), multiplying by 2, and subtracting 1. In all simulations, data augmentation derived from the original images was performed, by random horizontal flipping and translating up to four pixels in each direction.

4.4 Hyper-parameters

The hyper-parameters η (learning rate), μ (momentum constant[23]), and α (L2 regularization[24]) were optimized for offline learning, using a mini-batch size of 100 inputs. The learning-rate decay schedule[25] was also optimized. A linear scheduler was applied such that it was multiplied by the decay factor, q,

every Δt epochs, and is denoted below as $(q, \Delta t)$. Different hyper-parameters were used for each architecture.

4.5 LeNet-3 hyper-parameters

LeNet-3 with d number of filters in the first CL, was trained for at least 240 epochs using the following hyper-parameters to achieve maximal accuracy on CIFAR-100:

LeNet-3 on CIFAR-100			
d	η	μ	α
3	0.016	0.9	5e-4
6	0.021	0.905	9.1e-4
12	0.06	0.85	8e-4
18	0.038	0.93	1.1e-3
42	0.08	0.86	8e-4

Table 1. Hyper-parameters for LeNet-3 trained on CIFAR-100.

where the decay schedule for the learning rate for $d = 3$ is:

$$(q, \Delta t) = \begin{cases} (0.85, 10), & epoch < 120 \\ (0.75, 10), & epoch \geq 120 \end{cases}$$

and for any different d:

$$(q, \Delta t) = \begin{cases} (0.8, 10), & epoch < 120 \\ (0.7, 10), & epoch \geq 120 \end{cases}$$

In this case, there was no separate training of the FC layer.

4.6 VGG-5 hyper-parameters

VGG-5 was trained over 300 epochs using the following hyper-parameters to achieve maximal accuracy on CIFAR-100:

VGG-5 on CIFAR-100			
d	η	μ	α
8	2e-2	0.976	3.75e-3
16	2.2e-3	0.976	3.8e-3
32	2e-3	0.976	3.75e-3

64	2.2e-3	0.976	3.75e-3

Table 2. Hyper-parameters for VGG-5 trained on CIFAR-100.

The decay schedule for the learning rate during training of the entire system is defined as follows:

$$(\mathrm{q}, \Delta \mathrm{t}) = \begin{cases} (0.65,20) & 160 > epoch \\ (0.55,20) & 160 \leq epoch \end{cases}$$

For the training of the FC layer, $\eta = 0.001$, $\mu = 0.975$, $\alpha = 4e-3$, with a learning rate scheduler of $q = 0.65$ every 20 epochs, and the other weight values and biases of the architecture remained fixed.

4.7 VGG-16 hyper-parameters

VGG-16 was trained over 300 epochs using the following hyper-parameters to achieve maximal accuracy on CIFAR-100:

VGG-16 on CIFAR-100			
d	η	μ	α
4	1.5e-3	0.977	4.1e-3
8	1.5e-3	0.977	4.1e-3
16	3e-3	0.968	4e-3
32	1.5e-3	0.977	4.1e-3
64	2e-3	0.975	4e-3

Table 3. Hyper-parameters for VGG-16 trained on CIFAR-100.

The decay schedule for the learning rate during training of the entire system is defined as follows:

$$(\mathrm{q}, \Delta \mathrm{t}) = \begin{cases} (0.65,20) & 160 > epoch \\ (0.55,20) & 160 \leq epoch \end{cases}$$

For the training of the FC layer, $\eta = 0.004$, $\mu = 0.975$, $\alpha = 1.5e-3$, with a learning rate scheduler of $q = 0.65$ every 20 epochs, and the other weight values and biases remained fixed.

4.8 Raw data of the figures

The raw data of the error rate ε and the average noise n of the figures are presented in the following tables.

Results for Fig 2. LeNet-3 on CIFAR-100				
d	ε	$Std(\varepsilon)$	n	$Std(n)$
3	0.720	0.0048	827.8	88.6
6	0.613	0.0049	676.1	52.0
12	0.509	0.0036	504.4	39.0
18	0.464	0.0034	438.2	27.3
42	0.403	0.0021	305.0	8.3

Table 4. The results for Fig. 2, LeNet-3 trained on CIFAR-100.

Results for Fig 3. VGG-5 on CIFAR-100				
d	ε	$Std(\varepsilon)$	n	$Std(n)$
8	0.496	0.0051	342.2	33.3
16	0.395	0.0029	246.6	26.6
32	0.334	0.0045	153.2	5.9
64	0.308	0.0015	96.9	5.2

Table 5. The results for Fig. 3, VGG-5 trained on CIFAR-100.

Results for Fig 4. VGG-16 on CIFAR-100				
d	ε	$Std(\varepsilon)$	n	$Std(n)$
4	0.566	0.0022	380.1	22.7
8	0.441	0.0035	181.7	20.0
16	0.366	0.0048	80.0	4.3
32	0.295	0.0057	23.5	1.3
64	0.250	0.0034	15.3	0.7

Table 6. The results for Fig. 4, VGG-16 trained on CIFAR-100.

For LeNet-3 the accuracy is 0.387 for $d = 6$ filters in the first CL (Table. 4), whereas the achievable accuracy for LeNet-5 is ~ 0.43. For VGG-5, the

accuracy is 0.692 for $d = 64$ filters (Table. 5) whereas the achievable accuracy for VGG-6 is ~ 0.71.

4.9 Calculation of clusters and noise

For CIFAR-100, the 100 output fields of each filter were summed over all $10{,}000$ inputs of the test set, resulting in a 100×100 matrix, where each cell (i, j) represents the summed field of output field j for all test set inputs of label i. The matrix was normalized by dividing it by its maximal value, resulting in each matrix having a maximum value of 1. The clipped Boolean output field matrix was calculated, where each element whose value was above a threshold (0.3) was set to 1, each element below the negative threshold (-0.3) was set to -1, and all others were zeroed. Similar qualitative results were obtained for different thresholds $[0.1, 0.6]$, where 0.3 represents the matrix's distribution cutoff between high and low values.

The axes were then permuted such that all labels belonging to a cluster were grouped consecutively, thereby displaying the clusters in an adjacent fashion, where they were shown as a diagonal block of elements with value 1. Each cluster was defined as a subset of n indices, where for each $i, j \in n$ elements (i, j) have the value of 1. The size of the cluster was defined as n, where n is the number of labels whose all pairs (and their permutation) are equal to 1, thereby forming a cluster. The minimal cluster size is 1, that is one element on the diagonal, or 100, the entire matrix. The elements equal to 1 were then colored white, representing that they belong to a cluster in the filter, while non-cluster cells with a value of 1 were classified as above-threshold external noise and were colored yellow; cells with a value of -1 were colored green and the rest were zero and colored black.

The clusters were calculated by running along the diagonal from index $(0{,}0)$ to $(99{,}99)$, where the first (i, i) element with a value of 1 was initially designated as a cluster of size 1×1. The next (j, j) element, where $j \neq i$, with a value of 1 was then checked to determine if it can complete a cluster with (i, i); if yes, it was added to the cluster and the next diagonal element with a value of 1 was checked. This process was repeated for all value 1 cells in the diagonal, as long as there were elements that did not belong to a cluster. This process is not

uniquely defined, that is, the order by which the indices are iterated can change the outcome of the clustering process. For example, a filter with two clusters of sizes 3×3 and 1×1 retrieved by iterating from 0 to 99 can yield, in some very rare scenarios, two clusters of size 2×2. While possibly alternating the results of a single filter, the overall obtained averaged results remain the same when performing cluster creation while iterating in reverse order, because these scenarios are very rare and occur in a negligible number of filters.

4.10 Negative matrix elements - repulsive elements

The probabilities of the negative green elements (elements below $-threshold$) in rows associated with clusters for LeNet-3 and $d = 42$ (Fig. 1(b)) were calculated as follow. By averaging all samples, the averaged cluster size was found to be $C_s\ (d = 42) = 2.62$, and the number of clusters per filter was $N_c\ (d = 6) = 2.88$.

An approximation of the average number of rows associated with a cluster in each filter was $N_r\ (d = 42) = C_s\ \times N_c\ = 2.62 \times 2.88 = 7.55$. Hence, there were 755 values in the clusters' rows. In addition, the average number of below $-threshold$ elements per filter was $N_{neg}\ (d = 42) = 280.25$, and that the proportion of those elements, that shared a row with a cluster was 0.31. Thus, the probability of an element that shared a row with a cluster, to have a value below $-threshold$ was $P_{cluster} = \frac{0.31 \cdot 280.25}{755} = 0.112$. The probability of an element which did not share a row with a cluster, to have a value below $-threshold$ was $P_{no\ cluster}\ (d = 42) = \frac{(1-0.31) \cdot 280.25}{(100-7.55) \cdot 100} = 0.021$, thus $\frac{P_{cluster}\ (d=42)}{P_{no\ cluster}\ (d=42)} = 5.33$

4.11 Statistics

Statistics for all results were obtained using at least five samples.

References

[1]T.L. Watkin, A. Rau, M. Biehl, The statistical mechanics of learning a rule, Reviews of Modern Physics, 65 (1993) 499.

[2]W. Kinzel, Physics of neural networks, Europhysics News, 21 (1990) 108-110.

[3]G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700-4708.

[4]D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5927-5935.

[5]Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 521 (2015) 436-444.

[6]J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, 61 (2015) 85-117.

[7]K. Fukushima, Cognitron: A self-organizing multilayered neural network, Biol Cybern, 20 (1975) 121-136.

[8]Y. LeCun, Y. Bengio, Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks, 3361 (1995) 1995.

[9]T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, T. Poggio, A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex, in, MASSACHUSETTS INST OF TECH CAMBRIDGE MA CENTER FOR BIOLOGICAL AND …, 2005.

[10]D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, The Journal of physiology, 160 (1962) 106.

[11]K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.(2014) ,

[12]S. Zagoruyko, N. Komodakis, Wide residual networks, arXiv preprint arXiv:1605.07146 , .(2016)

[13]K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition ,in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.

[14]C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI conference on artificial intelligence, 2017.

[15]A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images.(2009) ,

[16]A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM, 60 (2017) 84-90.

[17]L. Hertel, E. Barth, T. Käster, T. Martinetz, Deep convolutional neural networks as generic feature extractors, in: 2015 International Joint Conference on Neural Networks (IJCNN), IEEE, 2015, pp.1-4 .

[18]Y. Meir, Y. Tzach, S. Hodassman, O. Tevet, I. Kanter, Universality of underlying mechanism for successful deep learning, arXiv preprint arXiv:2309.07537.(2023) ,

[19]Y. Tzach, Y. Meir, O. Tevet, R.D. Gross, S. Hodassman, R. Vardi, I. Kanter, The mechanism underlying successful deep learning, arXiv preprint arXiv:2305.18078.(2023) ,

[20]Y. Meir, O. Tevet, Y. Tzach, S. Hodassman, R.D. Gross, I. Kanter, Efficient shallow learning as an alternative to deep learning, Scientific Reports, 13 (2023) 5.423

[21]Y. Meir, S. Sardi, S. Hodassman, K. Kisos, I. Ben-Noam, A. Goldental, I. Kanter, Power-law scaling to assist with key challenges in artificial intelligence, Scientific reports, 10 (2020) 19628.

[22]Y. Le Cun, L.D. Jackel, B. Boser, J.S. Denker ,H.P. Graf, I. Guyon, D. Henderson, R.E. Howard, W. Hubbard, Handwritten digit recognition: Applications of neural network chips and automatic learning, Ieee Commun Mag, 27 (1989) 41-46.

[23]A. Botev, G. Lever, D. Barber, Nesterov's accelerated gradient and momentum as approximations to regularised update descent, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1899-1903.
[24]C. Cortes, M. Mohri, A. Rostamizadeh, L2 regularization for learning kernels, arXiv preprint arXiv:1205.2653.(2012) ,
[25]K. You, M. Long, J. Wang, M.I. Jordan, How does learning rate decay help modern neural networks?, arXiv preprint arXiv:1908.01878.(2019) ,