Scaling in Deep and Shallow Learning Architectures

Ella Koresh[a], Tal Halevi[a], Yuval Meir[a], Dolev Dilmoney[a], Tamar Dror[a], Ronit Gross[a], Ofek Tevet[a], Shiri Hodassman[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

The realization of classification tasks using deep learning is a primary goal of artificial intelligence; however, its possible universal behavior remains unexplored. Herein, we demonstrate a scaling behavior for the test error, ϵ, as a function of the number of classified labels, K. For trained utmost deep architectures on CIFAR-100 $\epsilon(K) \propto K^{\rho}$ with $\rho \sim 1$, and in case of reduced deep architectures, ρ continuously decreases until a crossover to $\epsilon(K) \propto log(K)$ is observed for shallow architectures. A similar crossover is observed for shallow architectures, where the number of filters in the convolutional layers is proportionally increased. This unified the scaling behavior of deep and shallow architectures, which yields a reduced latency method. The dependence of $\Delta\epsilon/\Delta K$ on the trained architecture is expected to be crucial in learning scenarios involving dynamic number of labels.

Keywords: Deep learning, Machine learning, Shallow learning

1. Introduction

Deep learning is a key development tool for the advancement of technologies and in experimental and theoretical research fields[1, 2]. A typical supervised learning task involves the classification of input objects into multiple labels. This is realized using deep architectures[2, 3] comprising hundreds of convolutional layers (CLs)[4, 5], each of which comprises of tens or hundreds of filters followed by several fully connected (FC) layers. The output layer comprises several output units, each representing a distinct label that characterizes a subset of the inputs.

Several deep architectures have been examined, differing in many aspects, including the number of learning parameters, learning time, training and test latency, and test accuracy. The results suggest several qualitative trends, such that for a given classification, the task accuracy is enhanced with deeper architectures simultaneously with an increase in the learning process complexity, the number of tunable parameters and test latency. Despite the widespread use of deep learning architectures, the quantitative trends that characterize their behavior remain largely unexplored, except for instance, the power law decay of the test error as a function of the training dataset size[6-9]. This power law was found to govern several shallow architectures; however, the interplay between the features of the architecture and the power-law exponent remains unknown. The possible existence of a universal scaling behavior that governs both the deep and shallow learning architectures lies at the core of this study and is a prerequisite for the establishment of a theoretical quantitative foundation for deep and shallow learning.

In this work, we demonstrate that for a given architecture trained on CIFAR-100[10], the test error ϵ as a function of the K classified labels, representing by the output units, follows a power law

$$\epsilon(K) = K^{\rho} \quad (1).$$

The exponent ρ for the utmost deep architecture is 1, which continuously decreases for reduced deep architectures until a crossover to a logarithmic scaling

$$\epsilon(K) = \log(K) \quad (2)$$

is observed for shallow architectures. A similar crossover from the power law to logarithmic scaling (Eqs. (1)-(2)), is observed for shallow architectures, where the number of filters in the CLs decreases proportionally. This similarity unifies the scaling behaviors of deep and shallow architectures.

2. Materials and methods

2.1. Architectures and Datasets

Several different architectures were examined; Tree-3[11], LeNet-5[12], VGG-6[13], VGG-16[13], EfficientNet-B0 and EfficientNet-B3[14]. All architectures were trained to classify CIFAR-100[10], and CIFAR-K/100[15, 16] where K is the selected number of labels trained. The selected K values are $20, 40, 60, 80,$ and 100, and to reduce fluctuations in the measured accuracies each group of K labels contains the selected $K-20$ labels, and consists of the same number of subclasses from each one of the 20 super-classes.

2.2. Data preprocessing

Each input pixel of an image (32×32) from the CIFAR-K/100 databases was divided by the maximal pixel value, 255, multiplied by 2, and subtracted by 1, such that its range was $[-1,1]$. During the training phase, data augmentation was used, derived from the original images, by random horizontally flipping and translating up to two pixels for Tree-3 and up to four pixels in each direction for and LeNet-5, VGG-6 and VGG-16.

For EfficientNet-B0, the images were normalized by subtracting the average value of each color and dividing by its standard deviation. The images were also expanded from their initial size of (32×32) to (224×224)[17]. Data augmentation was also used during the training phase, which included a random horizontal flip, a random rotation of up to two degrees, a random translation of the image of up to four pixels in each direction and a shear of up to two degrees.

2.3. Optimization

The cross-entropy cost function was selected for the classification task and was minimized using the stochastic gradient descent algorithm[18]. The maximal accuracy was determined by searching through the hyper-parameters (see below). Cross-validation was confirmed using several validation databases, each consisting a fifth of the training set examples, randomly selected. The Nesterov momentum[19] and L2 regularization method[20] were applied.

2.4. Hyper-parameters

The hyper-parameters η (learning rate), μ (momentum constant[19]), and α (regularization L2[20]) were optimized for offline learning, using a mini-batch size of 100 inputs. The learning rate decay schedule[21] was also optimized. A linear scheduler was used 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 one of the architectures on each classification task.

2.4.1. The hyper-parameters for the data presented in Fig. 1

LeNet-5 $(d = 6)$ was trained using the following hyper-parameters to reach maximal accuracies on CIFAR-K/100:

LeNet-5 on CIFAR-K/100			
η	μ	α	epochs
0.028	0.92	9.5e-4	300

Table 1. LeNet-5 hyper-parameters.

The decay schedule for the learning rate is:

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

VGG-6 $(d = 64)$ was trained using the following hyper-parameters to reach maximal accuracies on CIFAR-K/100:

VGG-6					
K	Layer	η	μ	α	epochs
20/40/60/80/100	CLs	2.2e-3	0.976	3.74e-3	280
	FC	1e-3	0.975	4e-3	280

Table 2. VGG-6 hyper-parameters.

The decay schedule for the learning rate is:

For convolutional layers (CLs):

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

For fully-connected layers (FCs):

$$(\mathrm{q}, \Delta\mathrm{t}) = \begin{cases} (0.65{,}20) & epoch < 160 \\ (065{,}20) & epoch \geq 160 \end{cases}$$

VGG-16 $(d = 64)$ was trained using the following hyper-parameters to reach maximal accuracies on CIFAR-K/100:

VGG-16				
K	η	μ	α	epochs
20	0.004	0.965	3e-3	300
40/60/80/100	0.002	0.975	4e-3	300

Table 3. VGG-16 hyper-parameters.

The decay schedule for the learning rate is:

$$(\mathrm{q}, \Delta\mathrm{t}) = (0.65, 20)$$

EfficientNet-B0 and EfficientNet-B3 were trained on CIFAR-K/100 datasets using transfer learning[22] on the pre-trained EfficientNet architectures on

ImageNet dataset. The transfer learning was done using the following hyper-parameters and learning rate scheduler:

EfficientNet-B0/B3			
CIFAR-K/100			
η	μ	α	epochs
0.01	0.9	0.001	200

Table 4. Hyper-parameters for EfficientNet architectures.

The decay schedule for the learning rate is:

$$(\mathrm{q}, \Delta \mathrm{t}) = (0.975, 1)$$

For the first seven stages, the learning rate η was multiplied by a factor 0.1, and for the last stages by 0.2.

2.4.2. The hyper-parameters for the data presented in Fig. 2

LeNet-5, d=240 on CIFAR-K/100				
K	η	μ	α	epochs
20/40	0.005	0.92	5e-3	300
60	0.007	0.92	3e-3	300
80/100	0.005	0.92	2e-3	300

Table 5. LeNet-5 with $d = 240$ hyper-parameters. The decay schedule for the learning rate is:

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

VGG-6, d=4					
K	Layer	η	μ	α	epochs
20	CLs	0.008	0.97	4e-3	300
	FC	0.001	0.975	4e-3	
40	CLs	0.006	0.972	4e-3	300
	FC	0.001	0.975	4e-3	
60	CLs	0.002	0.976	3.75e-3	300
	FC	0.001	0.975	4e-3	
80	CLs	0.002	0.976	3.75e-3	300
	FC	0.001	0.975	4e-3	
100	CLs	0.004	0.975	4e-3	300
	FC	0.001	0.975	4e-3	
VGG-6, d=8					
K	layer	η	μ	α	epochs
20	CLs	0.002	0.976	3.75e-3	280
	FC	0.001	0.975	4e-3	
40	CLs	0.002	0.976	3.75e-3	280
	FC	0.001	0.975	4e-3	
60	CLs	0.002	0.976	3.75e-3	280
	FC	0.001	0.975	4e-3	
80	CLs	0.002	0.976	3.75e-3	280
	FC	0.001	0.975	4e-3	
100	CLs	0.002	0.976	3.75e-3	280
	FC	0.001	0.975	4e-3	
VGG-6, d=160					
K	layer	η	μ	α	epochs
20	CLs	0.008	0.97	4e-3	300
	FC	0.004	0.975	1.5e-3	
40	CLs	0.008	0.97	4e-3	300
	FC	0.004	0.975	1.5e-3	
60	CLs	0.0022	0.976	4e-3	300
	FC	0.004	0.975	1.5e-3	

80	CLs	0.0022	0.976	4e-3	300
	FC	0.004	0.975	1.5e-3	
100	CLs	0.0022	0.976	4e-3	300

Table 6. VGG-6 with different d hyper-parameters.The decay schedule for the learning rate is:

For CLs layers:

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

For FCs layers:

$$(\mathrm{q}, \Delta \mathrm{t}) = (0.65, 20)$$

VGG-16, d=4				
K	η	μ	α	epochs
20/40/60/80	0.008	0.975	4e-3	300
100	0.003	0.975	4e-3	350

Table 7. VGG-16 with $d = 4$ hyper-parameters.

The decay schedule for the learning rate is:

For $K = 20/40/60/80$:

$$(\mathrm{q}, \Delta \mathrm{t}) = \begin{cases} (0.65{,}20) & epoch \leq 200 \\ (0.6{,}20) & 200 < epoch \end{cases}$$

For $K = 100$:

$$(\mathrm{q}, \Delta \mathrm{t}) = \begin{cases} (0.67{,}20) & 0 < epoch \leq 200 \\ (0.65{,}20) & 200 < epoch < 300 \\ (0.6{,}20) & 300 \leq epoch < 350 \end{cases}$$

VGG-16, d=16			
η	μ	α	epochs
0.002	0.975	4e-3	300

Table 8. VGG-16 with $d = 16$ hyper-parameters.

The decay schedule for the learning rate is:

$$(\mathrm{q}, \Delta\mathrm{t}) = (0.65, 20)$$

2.4.3. The hyper-parameters for the data presented in Fig. 3

Tree-3 d=6 (M=16) on CIFAR-K/100			
η	μ	α	epochs
0.07	0.96	5e-5	300

Table 9. Tree-3 with $d = 6$ hyper-parameters.

The decay schedule for the learning rate is:

$$(\mathrm{q}, \Delta\mathrm{t}) = (0.6, 20)$$

2.5. Hardware and software

We used Google Colab Pro and its available GPUs. We used Pytorch for all the programming processes. The power law and the logarithmic fits were obtained using the standard regression method supported by Microsoft Excel and Python.

3. Results

3.1. Scaling in deep learning architectures

The first architectures examined are EfficientNet-B0 and EfficientNet-B3 which comprised approximately 180 and 320 layers, respectively, organized in 9 blocks[23]. It was trained on the CIFAR-100 dataset[10] comprising 20 super-classes, each of which is composed of 5 subclasses, that is, $K = 100$ labels in total. The selected K values are $20, 40, 60, 80,$ and 100, and to reduce fluctuations in the measured ϵ, each group of K labels contains the selected $K - 20$ labels[24]. The results indicate a linear scale, $\epsilon(K) \propto K$ with small pre-factors, ~ 0.001 (Fig. 1(a)), hence ϵ almost vanishes when extrapolated to a small K, as expected.

The second examined case is the standard VGG-16[13] with $d = 64$ filters in the first CL, representing a reduced deep architecture compared with EfficientNet-B0. The error rate follows a power law, $\epsilon(K) \propto K^{\rho}$ where $\rho \sim 0.65$ and with a small pre-factor 0.012, as expected, for almost vanishing ϵ for small K (Fig. 1(b)). Similarly, for VGG-6[13] with $d = 64$, ρ is further slightly reduced to 0.62. Here, the pre-factor is slightly enhanced to 0.016 as ϵ of VGG-6 is expected to be higher than for VGG-16 (Fig. 1(c)).

For LeNet-5[25] a crossover to a logarithmic scale is observed (Fig. 1(d)), where ϵ vanishes at $K \sim 3.5$. Here, $\epsilon(K)$ could also fit to a power law with $\rho \sim 0.25$; however, the pre-factor is ~ 0.16, indicting a large ϵ even for $K = 1$, which is unreasonable.

The improved accuracy of the trained architecture when the number of trained labels is reduced from K to $K_1 = K - 20$ can be deduced from the following three measured quantities (Table 10). The third column in Table 10, $T(K_1|K) \wedge F(K_1|K_1)$, denotes the number of correctly classified test inputs belonging to the K_1 labels for the architecture trained with K labels, which are incorrectly classified on this architecture trained with K_1 labels. This type of inputs decreases the accuracy of the trained network with K_1 labels compared with K. The fourth column, $F(K_1|K) \wedge T(K_1|K_1)$, denotes the number of incorrectly classified test inputs belonging to the K_1 labels for the architecture trained with K labels, which are incorrectly classified on this architecture trained with K_1 labels. For a given K, the numbers of events in the third and fourth columns are relatively similar; hence, their effect on the enhanced accuracy with K_1 labels is minimal. The primary contribution to the enhanced accuracy while reducing the number of labels from K to K_1 is owing to the events belonging to the fifth column, $F((K_1, K]|K) \wedge T(K_1|K_1)$. The term $F((K_1, K]|K)$ denotes events such that, for K trained labels, an input with a label in the range $[1, K_1]$ selects an incorrect output label belonging to $(K_1, K]$, whereas with K_1 trained labels they are correctly classified, $T(K_1|K_1)$. Based on simulation results presented in Table 10, one can verify the following inequality

$$F((K_1, K]|K) \wedge T(K_1|K_1) \gg |T(K_1|K) \wedge F(K_1\,|K_1) - F(K_1|K) \wedge T(K_1|K_1)\,| \qquad (3).$$

For EfficientNet-B0, zeroing the FC weights to the $(K_1,\ K]$ output labels, for a network trained on K labels, results in approximately 80% correct classification of such inputs, $F((K_1, K]|K)$. This indicates that the top-2 output labels are correctly classified with high probability, even without retraining the network with a smaller number of labels, K_1.

For the LeNet-5 case (Table 10(c)), the fifth column, $F((K_1, K]|K) \wedge\ T(K_1|K_1)$, is a constant almost independent of K, and the third and the fourth columns are approximately identical. Hence, the difference in the test error is expressed as:

$$\epsilon_K = \epsilon_{K_1} + \frac{const}{K \cdot 100} \qquad (4)$$

where $K \cdot 100$ denotes the dataset test size for CIFAR-100 with K labels. After rescaling $n = \frac{K}{20}$, one finds that

$$\epsilon_n = \epsilon_{n-1} + \frac{const}{n \cdot (20 \cdot 100)} \qquad (5)$$

where $n = 1, 2, \ldots, 5$. This recursion relation yields

$$\epsilon_n = \epsilon_1 + \frac{const}{20 \cdot 100} \sum_{k=2}^{n} \frac{1}{k} \qquad (6)$$

which can be approximated by

$$\epsilon_n = A \cdot log(n) + B \qquad (7)$$

where A and B are constants, and its form supports the logarithmic scaling behavior (Fig. 1(d)).

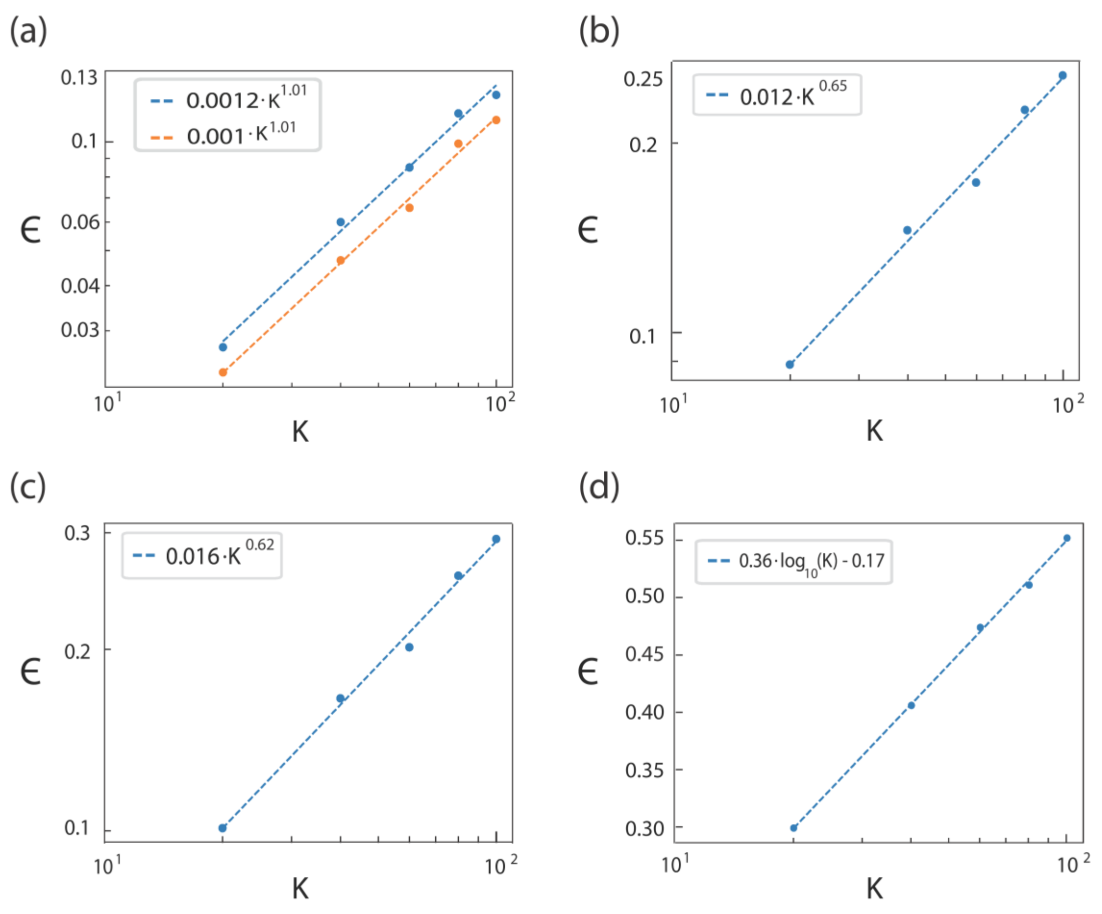

Fig. 1. Scaling for the error rate, ϵ, as a function of the number of classified labels, K, for CIFAR-100. (a) EfficientNet-B0 (blue) and EfficientNet-B3 (orange) with power law fits on a log-log plot (dashed lines). (b) VGG-16 with $d = 64$ and the power law fit on a log-log plot (dashed blue line). (c) Similar to panel (b) VGG-6. (d) LeNet-5 with $d = 6$ and the logarithmic fit on a semi-log plot (dashed blue line).

(a)

EfficientNet-B0				
K	K_1	$T(K_1\|K) \wedge F(K_1\|K_1)$	$F(K_1\|K) \wedge T(K_1\|K_1)$	$F((K_1,K]\|K) \wedge T(K_1\|K_1)$
40	20	12	5	68
60	40	49	26	97
80	60	90	76	160
100	80	136	102	149

(b)

VGG-16, $d = 64$				
K	K_1	$T(K_1\|K) \wedge F(K_1\|K_1)$	$F(K_1\|K) \wedge T(K_1\|K_1)$	$F((K_1,K]\|K) \wedge T(K_1\|K_1)$
40	20	56	38	101
60	40	163	136	168
80	60	284	244	235
100	80	444	410	219

(c)

LeNet-5, $d = 6$				
K	K_1	$T(K_1\|K) \wedge F(K_1\|K_1)$	$F(K_1\|K) \wedge T(K_1\|K_1)$	$F((K_1,K]\|K) \wedge T(K_1\|K_1)$
40	20	137	90	225
60	40	323	310	288
80	60	585	410	294
100	80	631	633	254

Table 10. Three measured quantities while reducing the trained network from K to $K_1 = K - 20$ output labels. $T(K_1|K)/F(K_1|K)$ represents the correct/incorrect test outputs among the K_1 labels, for the trained network on K labels, respectively. Similarly, $T(K_1|K_1)/F(K_1|K_1)$ represents the correct/incorrect outputs for the trained network on K_1 labels, respectively. $F((K_1,K]|K)$ denotes the input events with labels in the range $[1, K_1]$ selecting incorrect output labels belonging to $(K_1, K]$. (a) EfficientNet-B0, (b) VGG-16 with $d = 64$, and (c) LeNet-5 with $d = 6$.

3.2. Scaling in shallow learning architectures

Recent work has indicated that shallow architectures can achieve accuracies similar to those of deep architectures by using an increasing number of filters, d, in the first CL, and accordingly in the entire CLs[7, 26]. This similarity between wide shallow and narrow deep architectures[26] is extended to the scaling behavior of $\epsilon(K)$. A crossover is observed from a logarithmic scaling of $\epsilon(K)$ for Lenet-5 with $d = 6$ to a power law with $\rho \sim 0.54$ for $d = 240$ (Fig. 2(a)). A similar trend was obtained for VGG-16 where for $d = 4,\ 16, 64,\ \rho$ increases to $0.46, 0.56, 0.66$, respectively (Fig. 2(b)). Quantitatively, ρ and ϵ of LeNet-5 with $d = 240$ is very similar to that of VGG-16 with $d = 16$. This demonstrates an example of the equivalence between wide shallow and narrow deep architectures (Fig. 2(c)). Similarly, VGG-6 with $d = 160$ and VGG-16 with $d = 64$ yield the same ρ and ϵ (Fig. 2(d)). These results demonstrate a reduced latency method using wide shallow architectures while maintaining ϵ.

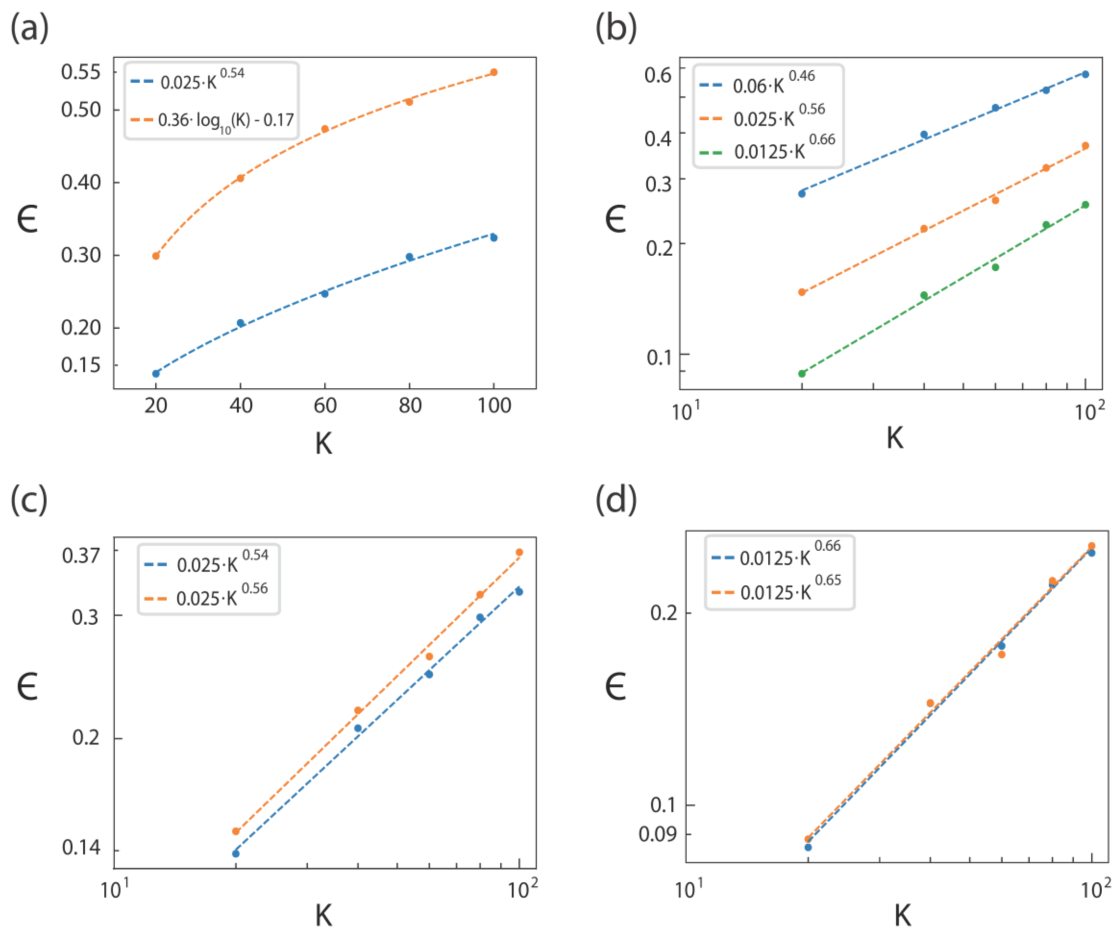

Fig. 2. Crossover of the $\epsilon(K)$ scaling while increasing d . (a) LeNet-5 with $d = 6$ (orange) and $d = 240$ (blue) and their logarithmic and power law fits (dashed lines). (b) VGG-16 with $d = 4$ (blue), 16 (orange), 64 (green) and their power law fits on a log-log scale (dashed lines) (c) Similar to panel (b), LeNet-5 with $d =$

240 (orange) and VGG-16 with $d = 16$ (blue). (d) Similar to panel (b), VGG-6 with $d = 160$ (orange) and VGG-16 with $d = 64$ (blue).

The accuracy achieved by LeNet-5 on CIFAR-10[10] was recently imitated using a Tree-3 architecture comprising only of three layers, where each weight was connected to an output unit via a single route[27]. The equivalence between the LeNet-5 and Tree-3 architectures is extended to CIFAR-100, where the accuracy of both architectures is approximately 0.44 (Fig. 3). Moreover, $\epsilon(K)$ for both architectures follows a logarithmic scale with almost an identical form (Fig. 3). The results may indicate that different shallow architectures with the same ϵ for a given K have also a similar $\epsilon(K)$.

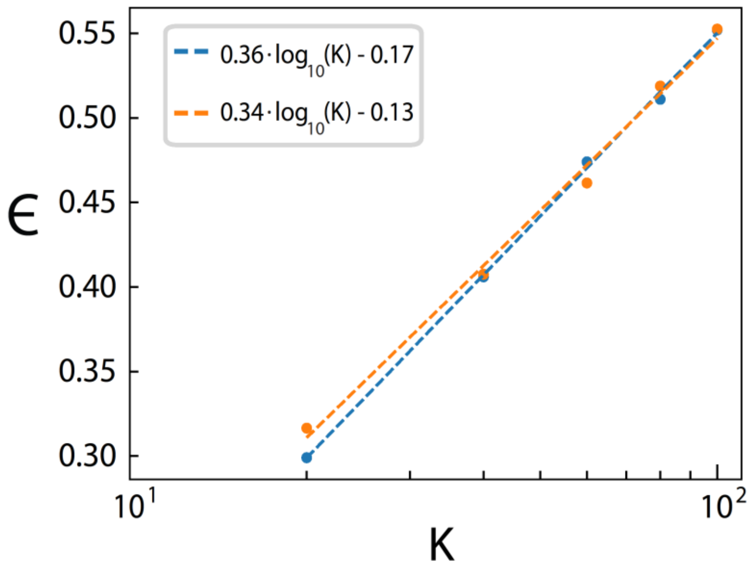

Fig. 3. Semi-log scale of $\epsilon(K)$ for LeNet-5 with $d = 6$ (blue) and the Tree-3 architecture[27] with $M = 16$ (orange) and their logarithmic scale fits (dashed lines).

4. Conclusions

Power law scaling is a central universal feature characterizing critical physical phenomena[28-33] and, in particular, second-order phase transitions[34, 35]. This study demonstrated its applicability to deep learning, where the test error for a given architecture decayed according to a power law with a decreasing number of classified labels (Fig. 1). The power law exponent, ρ (Eq. (1)), decreased as the architecture became less deep. Finally, a crossover to a logarithmic scaling, that is, $\rho \to 0$, was observed for shallow architectures, which was supported by a theoretical argument (Eqs. (3)–(7)). A similar power-law behavior was observed for several examined shallow architectures with a tunable number of filters d in the first CL, and accordingly in all CLs (Figs. 2 and 3). The similarity between wide shallow and narrow deep architectures was explicitly quantitatively demonstrated. Further, it was extended to the scaling behavior of $\epsilon(K)$.

The results of this study have several useful applications, including the following. First, the result that a relatively shallow but wide architecture can achieve the same $\epsilon(K)$ as a narrow deep architecture (Fig. 2(c)-(d)), significantly decreases the latency for an output decision on a test input. Currently, reducing latency is a crucial goal in the implementation of artificial intelligence[7, 24, 36, 37]. However, the extension of shallow architectures to numerous filters d, results in a significant slowdown of training and test procedures. Thus, its efficient utilization requires a new type of graphical processing units. Second, the results obtained are valuable in common realities of the dynamic number of output labels; for instance, new species are dynamically added or subtracted from the classification task. In such realizations, a prior knowledge, before retraining, of the expected change in test error rates, $\frac{\Delta\epsilon}{\Delta K}$, is a valuable information. The results indicated that for shallow architectures, $\frac{\Delta\epsilon}{\Delta K} \propto \frac{1}{K}$ decayed with K and was maximal for a very deep architecture, $\frac{\Delta\epsilon}{\Delta K} = const$, (Fig. 1). Apparently, the addition or subtraction of a label is favored in shallow architectures. However, ϵ and $\frac{\Delta\epsilon}{\Delta K}$ of deep architectures (Fig. 1(a)) were significantly lower than for shallow architectures (Fig. 1(d)), as their scaling pre-factor was considerably smaller.

Results are presented for CIFAR-100 database only, nevertheless preliminary results indicate similar trends for Sports-100 dataset[38, 39] consisting of 100 labels. A crossover from a power law behavior (Eq. (1)) for VGG-16 to a logarithmic scale for LeNet-5 is observed. In addition, for EMNIST dataset[40] consisting of 47 labels and only 50 training examples per label, ρ decreases from ~ 1 for VGG-16 to ~ 0.5 for LeNet-5. The lack of logarithmic behavior, even for a single fully connected network, is attributed to the simplicity of the dataset consisting of gray scale centered images.

The results may suggest that two architectures demonstrating the same ϵ for a given K will have the same quantitative form of $\epsilon(K)$ (Fig. 3). A quantitative exploration of this claim necessitates an extension of the presented work to other datasets, particularly datasets comprising numerous labels covering several power law decades. However, this mission is currently challenging for several reasons. The selected ΔK for measuring $\Delta\epsilon$ must be sufficiently large to neglect fluctuations as a function of the selected subset, $K,$ of labels. Consequently, the power law scaling of $\epsilon(K)$ was not extended to $K \leq 10$ (Figs. 1–3). In addition, an extension to datasets comprising $K > 100$ labels, for example, ImageNet[41, 42], requires the same number of input examples and accuracy for each label, which is currently difficult to realize. Another possible extension of the presented work is to other types of shallow architectures such as Boltzmann machine[43-46].

Finally, an interesting theoretical question is the establishment of lower and upper bounds for $\frac{\Delta\epsilon}{\Delta K}$. It may be possible that $\epsilon(K) \propto \log(log(K))$ will be obtained for the shallowest architecture without hidden layers, and an open question is the reality of a deep architecture where $\frac{\Delta\epsilon}{\Delta K}$ increases with K.

[1]E. Agliari, A. Barra, P. Sollich, L. Zdeborová, Machine learning and statistical physics: theory, inspiration, application, J. Phys. A.(2020) ,

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

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

[4]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.

[5]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.

[6]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.

[7]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) 5423.

[8]Y. Bahri, E. Dyer, J. Kaplan, J. Lee, U. Sharma, Explaining neural scaling laws, arXiv preprint arXiv:2102.06701.(2021) ,

[9]J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M.M.A .Patwary, Y. Yang, Y. Zhou, Deep learning scaling is predictable, empirically, arXiv preprint arXiv:1712.00409.(2017) ,

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

[11]Y. Meir, I. Ben-Noam, Y. Tzach, S. Hodassman, I. Kanter, Learning on tree architectures outperforms a convolutional feedforward network, Scientific Reports, 13 (2023) 962 %@ 2045-2322.

[12]Y. Lecun, 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 net chips and automatic learning, in: Artificial neural networks, IEEE Press, 1992, pp. 463-468.

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

[14]M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in, PMLR, 2019, pp. 6105-6114 %@ 2640-3498.

[15]Y. Meir, Y. Tzach, S. Hodassman, O. Tevet, I. Kanter, Towards a universal mechanism for successful deep learning, Scientific Reports, 14 (2024) 5881 %@ 2045-2322.

[16]O. Tevet, R.D. Gross, S. Hodassman, T. Rogachevsky, Y. Tzach, Y. Meir, I. Kanter, Efficient shallow learning mechanism as an alternative to deep learning, Physica A: Statistical Mechanics and its Applications, 635 (2024) 129513 %@ 120378-124371.

[17]R. Keys, Cubic convolution interpolation for digital image processing, IEEE transactions on acoustics, speech, and signal processing, 29 (1981) 1153-1160.

[18]J .Schmidhuber, Deep learning in neural networks: An overview, Neural networks, 61 (2015) 85-117 %@ 0893-6080.

[19]A. Botev, G. Lever, D. Barber, Nesterov's accelerated gradient and momentum as approximations to regularised update descent, in, IEEE, 2017 ,pp. 1899-1903 %@ 1509061827.

[20]C. Cortes, M. Mohri, A. Rostamizadeh, L2 regularization for learning kernels, arXiv preprint arXiv:1205.2653.(2012) ,

[21]K. You, M. Long, J. Wang, M.I. Jordan, How does learning rate decay help modern neural networks ,?arXiv preprint arXiv:1908.01878.(2019) ,

[22]J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, Advances in neural information processing systems, 27.(2014)

[23]M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105-6114.

[24]Y. Meir, Y. Tzach, S. Hodassman, O. Tevet, I. Kanter, Towards a universal mechanism for successful deep learning, Scientific Reports, 14 (2024) 5881.
[25]Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86 (1998) 2278-2324.
[26]O. Tevet, R.D. Gross, S. Hodassman, T. Rogachevsky, Y. Tzach, Y. Meir ,I. Kanter, Efficient shallow learning mechanism as an alternative to deep learning, Physica A: Statistical Mechanics and its Applications, 635 (2024) 129513.
[27]Y. Meir, I. Ben-Noam, Y. Tzach, S. Hodassman, I. Kanter, Learning on tree architectures outperforms a convolutional feedforward network, Scientific Reports, 13 (2023) 962.
[28]A. Barra, G. Genovese, P. Sollich, D. Tantari, Phase diagram of restricted Boltzmann machines and generalized Hopfield networks with arbitrary priors, Physical Review E 97 , .022310 (2018)
[29]P. Bak, K. Christensen, L. Danon, T. Scanlon, Unified scaling law for earthquakes, Phys Rev Lett, 88 (2002) 178501.
[30]Z.-S. She, E. Leveque, Universal scaling laws in fully developed turbulence, Phys Rev Lett, 72 (1994) 336.
[31]R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of modern physics, 74 (2002) 47.
[32]M. Levy, S. Solomon, New evidence for the power-law distribution of wealth, Physica A: Statistical Mechanics and its Applications, 242 (19.90-94 (97
[33]A. Blank, S. Solomon, Power laws in cities population, financial markets and internet sites (scaling in systems with a variable number of components), Physica A: Statistical Mechanics and its Applications, 287 (2000) 279-288.
[34]K.G. Wilson, The renormalization group: Critical phenomena and the Kondo problem, Reviews of modern physics, 47 (1975) 773.
[35]S.-K. Ma, Modern theory of critical phenomena, Routledge, 2018.
[36]S. Naveen, M.R. Kounte, M.R. Ahmed, Low latency deep learning inference model for distributed intelligent IoT edge clusters, IEEE Access, 9 (2021) 160607-160621.
[37]M.M.H. Shuvo, S.K. Islam, J. Cheng, B.I. Morshed, Efficient acceleration of deep learning inference on resource-constrained edge devices: A review, Proceedings of the IEEE, 111 (2022) 42-91.
[38]J. Cui, Research on application of model ensemble in sports image classification based on environmental information, in: Journal of Physics: Conference Series, IOP Publishing, 2023, pp. 012035.
[39]Gerry, 100 Sports Image Classification, Version 9. https://www.kaggle.com/datasets/gpiosenka/sports-classification/, in, May 2023.
[40]G. Cohen, S. Afshar, J. Tapson, A. Van Schaik, EMNIST: Extending MNIST to handwritten letters, in: 2017 international joint conference on neural networks (IJCNN), IEEE, 2017, pp. 2921-2926.
[41]A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, 25.(2012)
[42]J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248-255.
[43]A. Fachechi, A. Barra, E. Agliari, F. Alemanno, Outperforming RBM feature-extraction capabilities by “dreaming” mechanism, IEEE transactions on neural networks and learning systems.(2022) ,
[44]E. Agliari, F. Alemanno, A. Barra, G. De Marzo, The emergence of a concept in shallow neural networks, Neural Networks.232-253 (2022) 148 ,
[45]A. Barra, G. Genovese, P. Sollich, D. Tantari, Phase transitions in restricted Boltzmann machines with generic priors, Physical Review E, 96 (2017) 042156.

[46]G.E. Hinton, R.R. Salakhutdinov, A better way to pretrain deep boltzmann machines, Advances in Neural Information Processing Systems, 25.(2012)