Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 144 (2018) 203–213

Procedia Computer Science

www.elsevier.com/locate/procedia

INNS Conference on Big Data and Deep Learning 2018

Evolution of Abstraction Across Layers in Deep Learning Neural Networks

Robert Kozma[a,b,*], Roman Ilin[c], Hava T. Siegelmann[a]

[a]Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA
[b]University of Memphis, Department of Mathematics, Memphis, TN 38152, USA
[c] Air Force Research Laboratory, RYAT, Wright Patterson AFB, Dayton, OH 45433, USA

Abstract

Deep learning neural networks produce excellent results in various pattern recognition tasks. It is of great practical importance to answer some open questions regarding model design and parameterization, and to understand how input data are converted into meaningful knowledge at the output. The layer-by-layer evolution of the abstraction level has been proposed previously as a quantitative measure to describe the emergence of knowledge in the network. In this work we systematically evaluate the abstraction level for a variety of image datasets. We observe that there is a general tendency of increasing abstraction from input to output with the exception of a drop of abstraction at some ReLu and Pooling layers. The abstraction level is relatively low and does not change significantly in the first few layers following the input, while it fluctuates around some high saturation value at the layers preceding the output. Finally, the layer-by-layer change in abstraction is not normally distributed, rather it approximates an exponential distribution. These results point to salient local features of deep layers impacting overall (global) classification performance. We compare the results extracted from deep learning neural networks performing image processing tasks with the results obtained by analyzing brain imaging data. Our conclusions may be helpful in future designs of more efficient, compact deep learning neural networks.

Keywords: Deep Learning; Convolutional Neural Networks; Abstraction Level; Image Processing; Knowledge.

1. Introduction

Cutting-edge machine learning tools produce excellent results in automatic pattern recognition [1-4]. In recent decade, deep learning (DL) dominates the field of AI and machine learning due its outstanding performance in many if not most benchmark problems [5-9]. The well-established backpropagation learning is the backbone of most of the DL tools [10]. The reason of these spectacular successes, however, is not fully understood in terms of model

[*] Corresponding author. Tel.: +1-413-577-4282.
E-mail address: rkozma55@gmail.com

10.1016/j.procs.2018.10.520

design and parameterization [11]. There are no known general principles to design a deep learning network that has optimal performance for a specific task and the design is often accomplished on the trial and error basis. To have a more systematic and robust design approach, we need to answer the crucial open question concerning the relationship between information and meaning in intelligent computational systems.

We pose the question about the way input data (e.g., images) are converted into meaningful knowledge (i.e., assigned class labels) at the output of DLNNs. In previous studies, we defined abstraction level as a quantitative measure to describe the emergence of knowledge in the deep network from layer to layer [12, 13]. This work expands on previous results by testing the robustness of this measure for different image dataset, under various experimental conditions and multi-layer structures to improve the learning process. Our results confirmed that the so-called Q matrix is suitable to monitor abstraction level in various image classification tasks. The analysis of the layer-by-layer change of the abstraction in Deep Learning Neural Networks (DLNNs) points to the following key observations: (1) a general tendency of increasing abstraction from input to output; (2) exceptions to rule (1) are some convolutional layers with high abstraction, which drops in consecutive ReLu and pooling layers; (3) abstraction is relatively low and it does not change significantly in the first few layers following the input; (4) there is significant increase of abstraction in some intermediate layers, which reaches a saturation at high level in laters preceding the output; (5) the layer-by-layer increments of the abstraction demonstrate statistically significant deviation from normal distribution, and are consistent with exponential distribution. These results point to salient local features of DLNNs impacting overall (global) classification performance. These conclusions are helpful in future designs of more efficient, compact DL designs without notable sacrifice in overall performance characteristics.

2. Methodology for Abstraction Evaluation

2.1. Q Matrix Algorithm

Here we summarize the basic methodology of abstraction evaluation using Q Matrix approach, for details see [12, 13]. The main goal is to determine how symbolic knowledge evolves in DLNN from input to output. In this approach we observe that:

- There is a minimal symbolic content at the input, as observed in the raw data without class assignments.
- The symbolic content is high at the output in the form of class labels are assigned to each node.

We propose an approach, in which the symbolic representation at the output is measured through the classification performance. Let us consider the overall classification performance of the trained DLNN, which is a positive real number less or equal than 1. If the classification performance is perfect (value is 1 for the correct node an zero for all others), the activation of the output nodes completely represents the symbolic knowledge. Statistically speaking, there is a perfect correlation between the class labels and the activity of the output layer. Based on this observation, we develop an algorithm that determines the classification performance at consecutive deep layers based on the activations of the nodes in the specific layer. The developed approach is called the Q matrix algorithm and it is summarized in Table 1; for details see [13]. In our present pattern recognition task we know the correct class labels, thus the Q matrices can be used to evaluate the correct classification rate based on the activations of a given layer.

For layer n, Q_n is a 3-dimensional matrix whose rows correspond to nodes in layer n, whose columns are the nodes in layer $n + 1$, while its 3rd dimension specifies the classes. If X_n is the activation vector for layer n, and W_n is the weights connecting layer n to layer $n + 1$, for some neuron i in layer n, some neuron j in layer $n + 1$, and some class t (using threshold θ_{n+1}),

$$Q_n^{ijt} = \begin{cases} \text{corr}(X_n, X_{n+1})_{ijt} * W_n^{ij} & \text{if } Q_{n+1}^{jkt} > \theta_{n+1} \text{ for some k} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Table 1. Q Matrix Algorithm for DLNN, based on [13]

Step	Function	Description
1.	Define DLNN	*Build DLNN with N layers* • Layer n is denoted as L_n, the number of nodes in layer n is M_n, where $n = 1, \ldots, N$. • Layers can be Convolutional, Rectified Linear (ReLu), and Pooling according to the specific DLNN design.
2.	Train DLNN	*Implement the desired gradient learning* • Use a fixed number of examples K for each image class $c = 1, \ldots, C$; total of $K \times C$ image examples. • Conduct the training until proper performance is achieved.
3.	Test DLNN	*Test the trained DLNN on test images* • Use T images for C image classes, a total of $T \times C$ test images. • Tests runs produce a sequence of activations at each node of a given layer; e.g., C vectors of length $C \times T$ at the output layer.
4.	Determine Q_N matrix at the output layer L_N.	*Calculate activation correlations* • Determine correlation coefficients between the activations at the output layer L_N and the known class labels. This gives a $C \times C$ correlation matrix • For perfect classification, this is the identity matrix. In realistic cases with errors, the diagonal is not all 1s and there are non-zero off-diagonal elements.
5.	Determine Q_{N-1} matrix at layer $L-1$.	*Extend step 4 to layer L_{N-1} and get Q_{N-1}.* • Determine correlation coefficients between activations of layer L and $L-1$ for each of the classes; this produces c matrices each of size $C \times M_{N-1}$. • Combine these correlations matrices with the specific input-induced activation at layer $N-1$, including weighing and thresholding to infer a classification of the input.
6.	Determine Q_K matrix in deep layer L_K	*Extend step 5 iteratively to deeper layers* • Apply the procedure to consecutive deeper layers K, going backwards from the output towards the input.

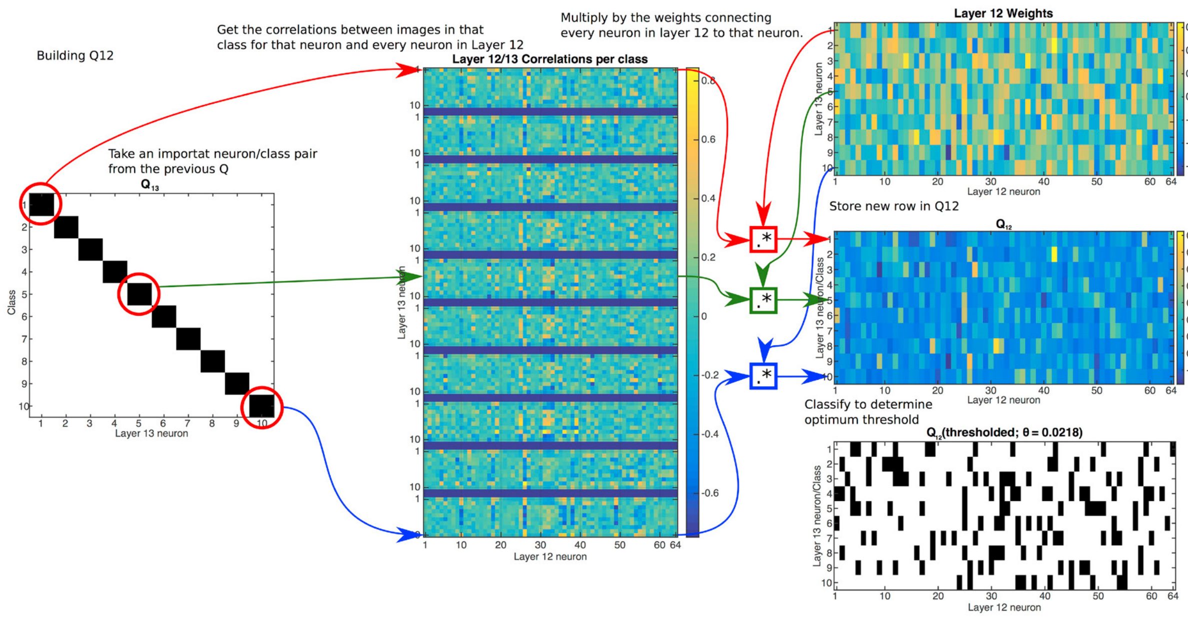
Building Q12
Get the correlations between images in that class for that neuron and every neuron in Layer 12
Multiply by the weights connecting every neuron in layer 12 to that neuron.
Take an importat neuron/class pair from the previous Q
Layer 12/13 Correlations per class
Layer 12 Weights
Store new row in Q12
Classify to determine optimum threshold
Q12(thresholded; θ = 0.0218)
Layer 13 neuron
Layer 12 neuron
Class
Layer 13 neuron/Class

Fig. 1. Illustration of the Q matrix algorithm by layers L_{13} and L_{12} of a 13-layer DLNN; layer L_{13} has 10 nodes (classes), layer L_{12} has 64 nodes.

That is, the correlation between the two neurons i and j is based only on activations sampled from images in class t. We also considered a variant of the Q matrix without the network weights W_n.

Let x_i^n denote the activation of neuron i in layer L_n for a specific input image. Classification using layer L_n activations is then determined as

$$S_t^n = \sum_{i,j} x_i^n * [Q_n^{ijt} \geq \theta_n] \tag{2}$$

$$\text{class} = \text{argmax } S_t^n \tag{3}$$

Threshold θ_n is chosen such a way that provides the highest classification accuracy for layer n. The classification accuracy is measured as the fraction of correct classification. This number for the last layer L_N equals to the overall network classification accuracy. The main results introduced in the next section use the classification performance to estimate the evolution of the generalization across the DLNN. In Fig. 1 we display the example of determining Q_{12} for a DLNN with 13 layers; here layer 1 in the input, and layer 13 corresponds to the output. We calculate the class-wise correlations between all nodes in layer 12 and layer 13, respectively. For a classification problem with 10 classes, we obtain 10 correlation matrices each of size 10 × 64, i.e., one matrix for each class. To get the c-th row in Q_{12}, we select the c-th row from the c-th correlation matrix. To produce the final Q12, we multiply by the weights of layer 13 by the normalized Q_{12} from the last step, and threshold the values using a suitable value θ. The final classification is determined by summing up the values in each row; the row with the highest value gives the classification output.

2.2. *Analyzed Datasets*

2.2.1. *CIFAR-10 Database*

The CIFAR-10 database consists of 60000 32 x 32 color images (3 colors RGB) in 10 classes, with 6000 images per class [5, 13]. Figure 2 provides examples of the analyzed images. The input layer has 32x32x3 = 3072 nodes describing the 32x32 pixel images using RGB 3-color maps. Note that 4 of the image classes describe vehicles (airplane, truck, ship, and car), and 6 classes are for animals (bird, cat, dear, dog, frog, and horse). Accordingly, there are two super-

Fig. 2. Examples of CIFAR-10 images with 10 classes; 10 examples for each class.

classes, one for vehicles and one for animals. The importance of such super-classes has been identified in our analysis and will be discussed later. The dataset is divided into 50,000 training images (5,000 for each of the 10 classes) and 10,000 test set. The test batch contains 1,000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. In the experimental evaluation procedures, we used standard convolutional DLNN architecture with 13 layers, in which all layers are selected based on a specific MatConvNet DLNN design [11]. For input images from class c ($c = 1, \dots C$), output node c is trained to have the value of 1, while all other output nodes should be zero for this image.

2.2.2. *Tiny ImageNet Database*

The ImageNet project is inspired by the desire of the image processing community to have more standard data suitable for benchmarking purposes [14]. ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a ”synonym set” or ”synset”. There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). ImageNet aims to provide on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated. ImageNet aims at offering tens of millions of cleanly sorted images for most of the concepts in the WordNet hierarchy. At the present stage, ImageNet data set consists of 1000 different categories for image classification. It has a total of 1,200,000 labeled images in the training set and 150,000 labeled images in the validation and test set. ImageNet's dataset size is 256x256 pixels of colored images. Tiny ImageNet Data (TIND) is similar to the ImageNet, but it contains only 200 image classes from the original 1000 classes in ImageNet. TIND contains a training set of 100,000 images, a validation set of 10,000 images, and a test set of 10,000 images, in which each class has 500 images. The images are downscaled from the original ImageNet's dataset size of 256x256 to 64x64 colored ones. For the purposes of the present analysis, we further down sampled the images to 32x32 pixels for comparison purposes. 3 displays examples of the TIN data for 10 classes and 10 examples for each class.

2.3. *Summary of Previous Results on the Abstraction Increments using CIFAR-10 Data*

To study the nature of the layer-by-layer change in classification performance, we determined statistical properties of these increments. Previous results are shown in Fig. 4, which depicts the distribution function of the layer-by-layer increment of the normalized classification performance (Rho) over all available experiments [13]. We observe that the distribution function has a maximum around zero, however, it is significantly skewed with a long tail for large positive increments up to 0.6, while the negative values extend to -0.2. The corresponding candidate outliers

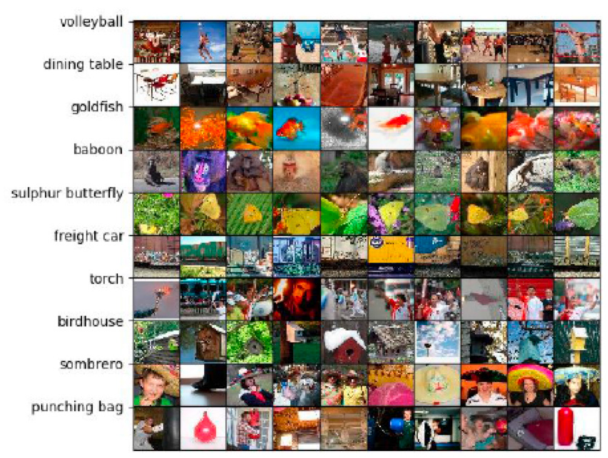

Fig. 3. Illustration of the Tiny ImageNet data using 10 classes; 10 examples for each class.

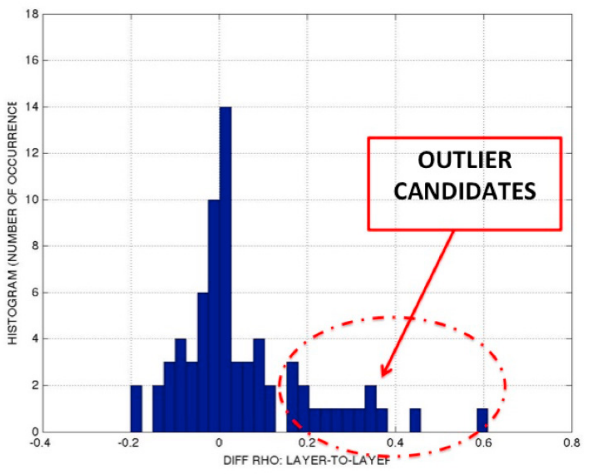

Fig. 4. Distribution function of the layer-by-layer increment of the normalized abstraction performance over all available experiments using CIFAR-10 database, see [12].

from normal distribution at large positive values are indicated by red circle in Fig.4. The distribution is apparently non-Gaussian, but more studies are needed to evaluate this hypothesis. In the present work, we study the statistical distribution of of the layer-by-layer increments of the generalization, using additional image examples from the Tiny ImageNet database. The main goal is to test the validity of the previous hypothesis on the non-Gaussian nature of the generalization increments.

3. Results and Discussions

3.1. Evolution of the Abstraction Across Layers of DLNNs

To test the statistical significance of our initial results with CIFAR-10, we chose Tiny ImageNet Dataset (TIND). In order to allow direct comparison with CIFAR-10, we further down-sampled the 64x64 TIND color images to 32x32 pixel size and use the same 13-layer DLNN what has been employed previously. for CIFAR-10. The dataset has been subsampled also in the context of number of classes; accordingly, we had experiments not only with 200 classes, but

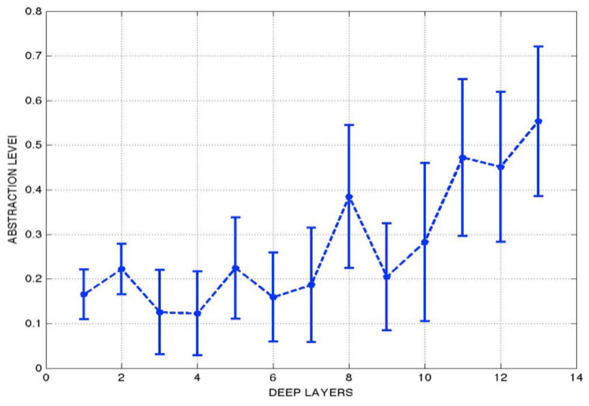

Fig. 5. Evolution of the abstraction level over deep layers; average value and standard deviation determined over 50 networks.

also with 50, 20, and 10 classes. In the case of 10 and 20 classes, an overall testing performance of 68-70% has been reached. This performance level is reasonable considering the low pixel resolution of 32x32.

Extensive experiments have been conducted testing various versions of the Q matrix approach, including weighted and unweighted options. Here we describe results with weighted Q matrix for 10-class classification task. There are 500 training samples in each class, in total 5,000 training examples, while the number of validation and testing examples for each class was 50. We have designed 50 different training sets with randomly selected 10 classes in each experiment. Each of the 50 trained and validated 13-layer DLNNs has been tested on all 10 classes, resulting in a total of 500 sequences of abstraction levels determined using Q matrices.

There is a general tendency of increasing abstraction level from input (L_1) towards output ($L_1 3$), with the exception of L_8, and sometimes L_5, which exhibit some peak values. Note that L_8 and L_5 are convolutional layers. This observation pointing to the special role of convolution layers is in line with earlier conclusions based on CIFAR-10 data [13]. To provide an overall view on the results on abstraction evolution in deep layers, we show in Fig.5 the abstraction level averaged over all 50 networks, as well as standard deviations. We have the following observations:

1. There are some fluctuations in abstraction level in the first few layers, L_1 to L_7, with mean values in the range of about 0.1 to 0.2. The displayed standard deviations have magnitudes in the range of the mean fluctuations.
2. Layer L_8 shows a significant increase in the mean, which drops back almost to the pre-peak level in L_9.
3. In layers $L_{10} - L_{11}$, there is a significant increase in abstraction level, overall exceeding the drop observed in L_9.
4. The abstraction level ultimately settles at some higher values at at the last two layers (L_{12} and L_{13}).
5. Layer L_{13} (the output layer) has the highest value that provides the classification level averaged over the 50 networks.

Next, we sort the 50 DLNNs according to their overall classification performance (13th layer value) at the output. The complete results with all the tests are shown in Fig.6 as a surface plot; we have 500 tests sets (10 pattern sets for each of the 50 DLNNs), which give the 500x13 abstraction matrix. The columns shown in Fig.6 have been sorted by the value in the 13th layer, i.e., overall classification performance.

3.2. Statistical Evaluation of the Layer-by-layer Increments of the Abstraction Level

Here we evaluate the distribution of the step-by-step change of the abstraction level. Figure 7 depicts the distribution of the increments using 16 bins. There is a clear maximum at 0, and we observe an asymmetry of the distribution with stronger tail for positive increments. Just as we observed earlier in the CIFAR-10 database, see Fig. 4, these distributions do not seem to be normal. The thorough evaluation of the statistics is an important task performed in this study. In [12] a hypothesis has been made that the distribution of the increments might be a superposition of normal and Poisson statistics. This hypothesis has been made based on a small sample, i.e., a total of a few 100 values. Our

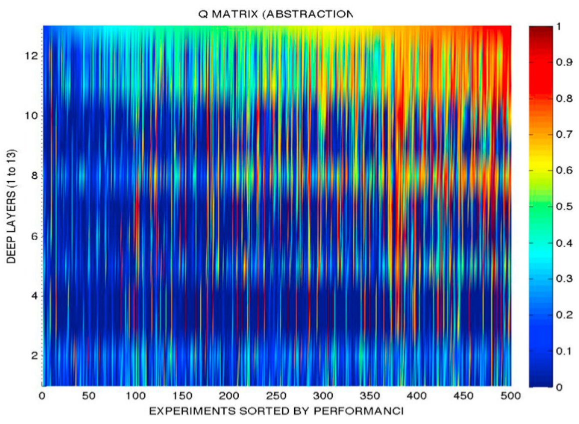

Fig. 6. Abstraction levels in deep layers (1 to 13) over 500 tests; the color bar indicates the levels from 0 (deep blue), to 1 (red).

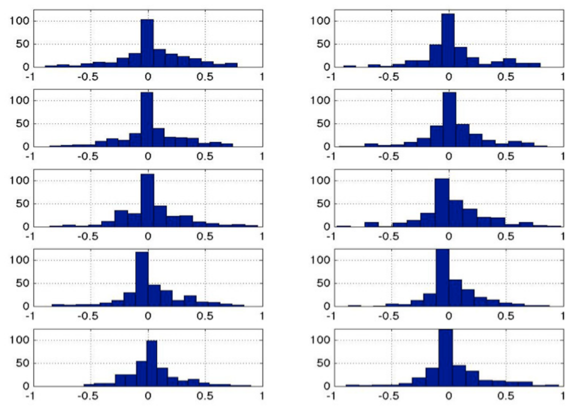

Fig. 7. Distribution of the abstraction increment in 10 networks.

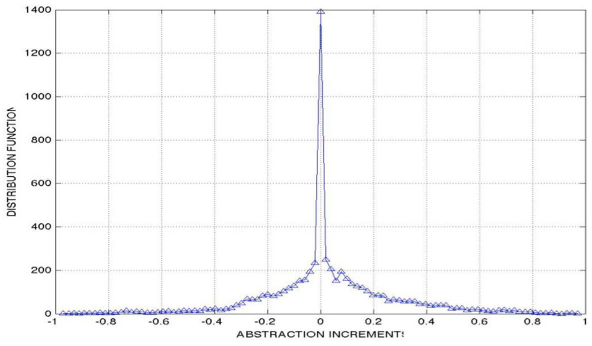

Fig. 8. Distribution of the abstraction layer-by-layer increment in all 50 networks, each tested with 10 class examples.

present study using a much broader database from Tiny ImageNet shows that exponential distribution is better suited to describe the experimentally observed statistics.

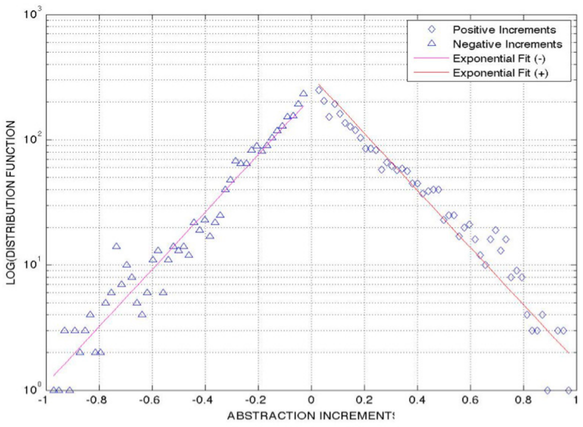

Fig. 9. Exponential fit of the increment distribution; see red and pink lines; the data points are marked by triangles and diamonds, respectively.

Next, we study the increments including all of the 500 tests. Considering that each test involves a DLNN with 12 increments between the 13 layers, we have 6,000 increment values, which will be used in the systematic evaluation of the distribution. We determine the distribution of the abstraction layer-by-layer increment in all 50 networks, each tested with 10 class examples; 100-bin resolution is used over the range of increments [-1, 1]. There is a prominent peak at the bin that includes zero increments, see the zero bin in Fig. 8. Thorough analysis points to the potential reason of this peak. We observe that the abstraction level is often zero at some deep layers in a single test. Namely, in 2051 cases the generalization value is zero, which constitutes about 1/3 of all values. Moreover, the abstraction remains zero in 1344 consecutive experiments; see deep blue regions in Fig. 6. This means that the Q matrix algorithm hits a low threshold, and it stays at the zero level for several consecutive layers, which may reflect a truncation effect. As the input layers have an initial abstraction level around 0.1 corresponding to random selection, a zero level in consecutive deeper layers has an unclear interpretation. It may be an artifact of the Q matrix algorithm, which requires further analysis.

In the present work, we do not include these zero increment values in the statistical analysis. The resulting probability distribution of the increments is not normal, as it will be analyzed next. Figure 9 shows fitting the experimental data using exponential distribution. We use log coordinates in the vertical axis to visualize the exponential effect. Data have been fitted using the functional relationship between variables y and x, which stand for the magnitude of the distribution function (y coordinate) and abstraction increment (x coordinate):

$$y = aa * 10^{(bb*x)}. \tag{4}$$

The parameters specified in Eq.4 are obtained using least-squared estimation (LSE) statistics, giving $bb = 2.29 \pm 0.14$ and $aa = 218$ for region [-1,0], and $bb = -2.28 \pm 0.13$ and $aa = 323$ for region [0,1]. The slopes of the exponentials are close in the positive and negative regions, with the proper signs. The LSE show that exponential fit is a possible statistics to interpret the obtained results. The exponential statistics signifies that the increments occur randomly in consecutive layers. These results are of significance for efficient design and operation of DLNNs, as they indicate that some layers may be superficial for the performance, while other layers play critical role. The layer-by-layer abstraction level has been investigated in cognitive and brain networks [15, 16], and our present computational results in DLNNs may be beneficial to interpret those experiments in future comparative studies.

4. Conclusions

The main goal of the present work has been to conduct statistical testing of the preliminary result on the evolution of the abstraction level determined based on the Q matrix. This work involved comparison of CIFAR-10 and Tiny ImageNet data results while determining the Q matrix in deep layers of DLNNs. Our work achieved its main goal and confirmed previous results on the usefulness of Q matrix. We have the following major observations:

- There are base-level fluctuations in abstraction at the first few layers (L_1 to L_5), in some cases until L_7. The mean values of the abstraction level fluctuate in the range of 0.1 to 0.2, corresponding to random assignment.
- We observe a significant increase in the mean of abstraction level at L_8 (sometimes from L_5), followed by significant drops in L_9 to a value close to the pre-peak level in L_7.
- There is a significant increase in abstraction level in $L_{10} - L_{11}$, which generally exceed the drop observed in L_9.
- The abstraction level ultimately settles at some higher values at the last two layers (L_{12} and L_{13}).
- Layer 13 (the output layer) has the highest value that provides the classification level averaged over the 50 networks.

The previously stated hypotheses have been confirmed regarding the 3 stages of the evolution of abstraction across layers in DLNNs. These observations are valid especially for well-trained networks, top 2/3 of all DLNNs. The bottom 1/3 (worst performers) have little change in abstraction until L_{10}, and pick up performance above chance only at L_{11} to L_{13}. The *Initial Stage* with low abstraction is observed in layers close to the input, whereas we see base level of oscillations without significant increase in abstraction levels. The *Intermediate Stage* is characterized by drastic increase of abstraction in Convolution Layers, and significant oscillations in other layers. The *Final Stage* at layers close to the output produces a saturation of abstraction at high levels. These results point to salient local features of deep learning neural networks impacting overall classification performance and indicate ways to produce a powerful tool to support deep learning network designs.

In a parallel development, abstraction levels have been studied in multi-layer architectures abstracted from massive brain imaging databases [15]. The present results based on analyzing DLNN in machine learning setting may provide a support to interpret those results from brain imaging. Moreover, the conclusions on the evolution of abstraction in DLNN, especially the observed transition region at intermediate layers may provide a useful link to the recently identified phase transitions in cognitive processing and in brain dynamics [16]. The corresponding research is the objective of ongoing studies.

Acknowledgements

This work has been supported in part by AFRL RY CERFER project funded via Matrix Research contract WP-TR-2017-0210, and by DARPA MTO Superior AI grant HR0011-16-l-0006. Results include computations conducted using WPAFB HPC facility by Thomas Watson during his 2016 AFRL summer internship, and by Ryan McCormick and Dan Saunders during their 2017 summer internships with AFRL/ATR.

References

1. Cortes, C., V. Vapnik. Support-vector networks. *Machine Learning*, 20(3), pp. 273-297, 1995.
2. Mitchell, T., *Machine Learning*. McGraw Hill, 1997.
3. Russell, S., J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (1995). *Artificial Intelligence: A Modern Approach*, Englewood Cliffs: Prentice Hall.
4. Bengio, Y. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1), pp. 1-127, 2009.
5. Bengio, Y., A. Courville, P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. on pattern analysis and machine intelligence*, 35(8), pp. 1798-1828, 2013.
6. Jordan, M. I., M. T. Mitchell, Machine learning: Trends, Perspectives, and Prospects, *Science*, 349, Issue 6245, pp. 255-260, 2015.
7. Krizhevsky, A. *Learning multiple layers of features from tiny images.* MSc. Thesis, U Toronto, 2009.
8. LeCun, Y., Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, No. 7553, pp. 436-444, 2015.
9. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61, pp. 85-117, 2015.

10. Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Thesis, Harvard University. 1975.
11. Vedaldi, A., K. Lenc, MatConvNet - Convolutional Neural Networks for MATLAB, *Proc. of the ACM Int. Conf. on Multimedia*, 2015.
12. Kozma, R., Analysis of Deep Learning Tools in Dynamical Problem Domain, *Report CRFR-036-002*, Matrix Research Inc. 2017.
13. Ilin, R., T. Watson, R. Kozma Representations in Deep Neural Networks for Image Processing, *Proc. IEEE/INNS Int. Joint Conf, Neur. Netw. (IJCNN2017)*, May 14-19, 2017, Anchorage, AK, USA, pp. 768-774, 2017.
14. ImageNet Database *http://www.image-net.org*. Accessed August 30, 2017.
15. P. Taylor, J. N. Hobbs, J. Burroni, H. T. Siegelmann, The global landscape of cognition: hierarchical aggregation as an organizational principle of human cortical networks and functions, *Scientific Reports*, 5, 18112, 2015.
16. Kozma, R., W. J. Freeman. *Cognitive Phase Transitions in the Cerebral Cortex - Enhancing the Neuron Doctrine by Modeling Neural Fields.* Springer Verlag, 2016.