

The Anatomy of a Neural Network

James P. LaRue, PhD, *Jadco Signals*, Richard L. Tutwiler, PhD, *Professor Emeritus ARL / Penn State University/CEO LiveMotion3D LLC*,
Dennison J. LaRue, *Juris Doctor Candidate 2020, South Carolina School of Law*
James@JadcoSignals.com rlt1@psu.edu larued@email.sc.edu

Abstract— It is true there have been great improvements with the effectiveness of utilizing Neural Networks. However, these improvements are, for the most part, relegated to improved clock speeds, leveraging increase in memory, and GPU enabled parallelization of up-front processing. However, what has been seemingly forgotten over the last twenty or so years is the understanding of how the internal layers are reacting with respect to convergence in training, and information transformation across layers during test, which in turn may account for a common perception that the internal neural layers are opaque black boxes. This paper will show in two parts that in fact, this is not true. Part one will demonstrate, through matrix visualization, the feed-forward processing throughout a multi-layer convolutional neural network. Part 2 will discuss our unique derivative application of Kohonen's and Kosko's correlation matrix memory methods to the consecutive pairs of layers within the network in order to form stabilized and compressible associative memory matrices. The subtlety of Part 2 is that our stabilized matrices can be simply multiplied together, thus forming a single layer, and therefore realizing The Universal Approximation Theorem of Cybenko and Hornik. In effect, the anatomy of the neural network will reveal how to open up the black box and take advantage of its inner workings.

Index Terms—Convolutional Neural Network, Bidirectional Associative Memory, Deep Belief, Pattern Recognition,

I. INTRODUCTION

This idea for this paper grew out of an observation that there is a prevailing belief that neural networks (NNs) are so-called 'black boxes' and that most analysis of neural networks is relegated to the first and last stages of the NNs but not by any means, to the internal hidden layers. Certainly there have been attempts to explain what is occurring within the hidden layers. We list two very different attempts in the reference section for the interested reader to explore. The first reference focuses on the first two or three layers where it is simplistically shown how the combinations of Gabor functions 'correlate' with the incoming images [1]. The second reference is from a blog where the explanation is given in a very abstract way using terms such as Topology and Manifolds [2]. This paper may lie somewhere in between. We believe our time domain matrix-based approach is a practical depiction of how the artificial neurons evolve into coherent and convergent patterns throughout the training process and across each hidden layer. In effect, the first part of this paper will look at the anatomy of the inside layers making up the 'black boxes'.

The second part of this paper will leverage our detailed 'anatomical' study as an opportunity to present, in a different light, some previously reported results involving the conver-

sion of a convolutional neural network (CNN) into a chain of bidirectional associative memory matrices [3]. We note that in making this conversion, from a feed-forward process to a matrix based process, the chain of matrices, once stabilized, may be multiplied together to form a single matrix and hence offer a path towards realizing the claims associated with the Universal Approximation Theorem (UAT).

This paper is focused on matrix imaging and pattern recognition; there is little in the way of mathematical formulation. Indeed, the well published formulation of the Werbos back-propagation, along with the LeCun style of CNN formulation is taken as the basis for our portrayal of convergent vectors [4, 5]. This paper simply exposes the 'deeper' inner workings of the inner layers, i.e., a micro-anatomy of a neural network.

II. CONVERTING CONVOLUTIONAL PATCHES INTO VECTORS

A CNN, built for image recognition, may be thought of as a two part layering system comprising of correlators that comb the input for features followed by a decision maker. Fig. 1 shows an outline of such a system which is built to decide which MNIST numeral is presented to the system.

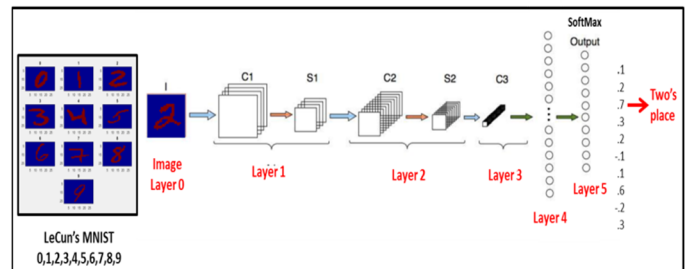


Fig.1. Sketch of a CNN system utilizing the MNIST data set. Layers 1-3 represent the correlation and combinations thereof of Gabor functions with the numeral 2 and Layers 4-5 represent the decision layers.

Fig. 2 is a depiction of layer 1 processing where the numeral 2 is correlated (a more accurate portrayal than stating 'convolved') against 12 Gabor functions to extract features of orientation and scale. Leaving out technical details, these correlations initially yield 28x28 images in sub-layer C1 layer which are then down-sampled to produce twelve 14x14 images in sub-layer S1. The numeral 2 is observable in some of the images in C1 and S1 while others it is not. A more colorful example can be seen in the paper 'Visualization and Understanding Convolutional Neural Networks' by Zeiler and Fergus where their images involve cute puppy dogs correlated against a subset of Gabor-like functions used by Krizhevsky et. al. for the 2012 ImageNet contest [1,6]. However, whereas the research of Zeiler and Fergus focused on showing how far into a

CNN the image remains coherent as to the actual object (a numeral or a puppy dog) being passed through the first couple of layers, we have taken the view on the other hand, that these images should be analyzed by imagining each pixel in the image as an individual neuron. Hence, a 14x14 image will be visualized as a string of 196 individual neurons; the motivation for us doing so being made clear as we proceed with this paper.

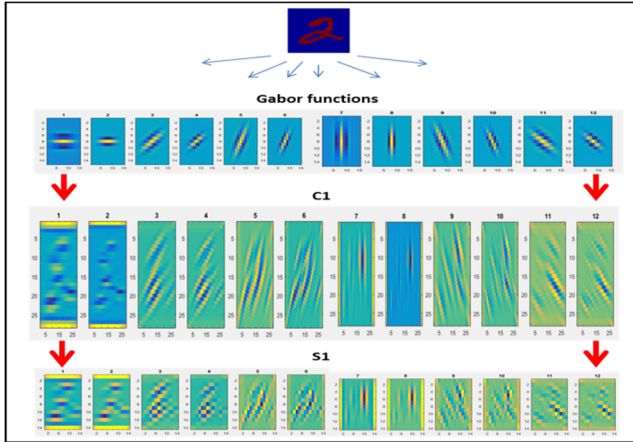


Fig. 2. The numeral 2 is correlated across these 12 Gabor and down-sampled to produce C1 and S2 images.

Our first example of converting an image into a set of neurons begins with Fig. 3. The top box shows how a 28x28 matrix of pixels representing the visual numeral 2 is converted into a vector; a 1x784 set of points. We do this simply by concatenating the columns of the matrix (left to right). The middle and bottom boxes show how we likewise convert the 12 sets of 14x14 matrices into a single vector of 2352 points.

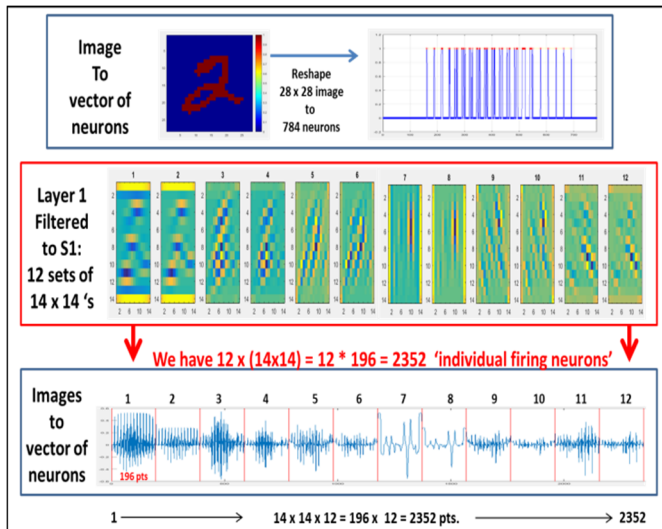


Fig. 3. Shows the appearance of a matrix and its reappearance in vector form for both a single matrix and a series of matrices from layer 1.

We now show a second example of matrix-to-vector conversion using layer 2. Fig. 4 utilizes the matrix form from layer 1, and not the vector we formed, since the CNN operations

in layer 2 require matrices. We show some pseudo-code that takes the layer 1 output and applies the 2-D convolution filter from Matlab (conv2) along with sums of layer 1 matrices, weights, biasing, and activation to produce the layer 2 output of 15 5x5 matrices. To produce a layer 2 output in vector form we repeat the procedure we used for layer 1.

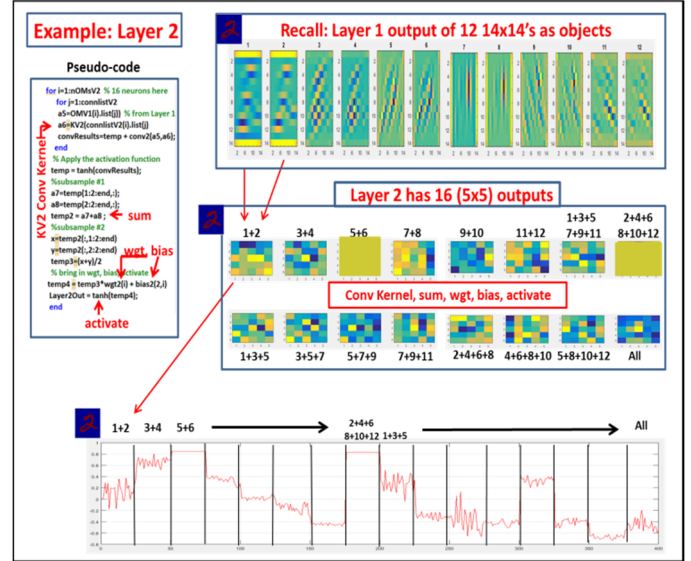


Fig. 4. Shows a second illustration of converting matrix images into vector form with layer 2 processing. Note the pseudo-code that gives an idea of the feed-forward processing the CNN utilizes as it moves from Layer 1 matrix form to layer 2 matrix form. Once the 16 5x5 matrices are formed, they are converted into a 400 dimensional vector.

So far we have shown how to convert the initial layers of a CNN into vector form. The question now arises as to “what are we gaining from moving away from a concrete form of visualization of the numeral 2 as humans perceive it, into this abstract vector form?”. Using figures 5-9 we plan to demonstrate that these abstract pieces, when reassembled as another matrix but in a time domain view, shed light on the evolutionary learning of each and every individual pixel, i.e., individual neuron, in a concrete way.

Fig. 5 is the cornerstone for figures 6-9 and for that matter, for the entire paper as well, as we expose the inner workings of CNN processes within the so-called black-box. Fig. 5 tracks the changes of layer 2 using our matrix-to-vector formatting followed by, surprisingly enough, a vector-to-(time domain) matrix formatting. We do this in four steps. In our first step we are placing the context of this paper into the CNN training process where we randomly introduce the MNIST numerals 0-9 into the process as we train the CNN until we reach 5000 iterations. In Fig. 5 under Step 1 we have assembled some layer 2 outputs between the 1st and 5000th iterations. We can see plots move from a smooth plot in iteration # 1 to a more refined and not as smooth plot in iteration # 5000. In Step 2 we play a trick in replotting the five vectors in vertical form using rotation and reflection. In Step 3 we use another trick of color-coding the amplitudes of the five plots and converting them into the adjacent 400x1 matrices. Finally, in Step 4 we

take these (thin) columns of matrices and we add in between, the vectors obtained from intermediate iterations to form a full 400x5000 matrix. From this matrix we observe a time evolution (over the 5000 iterations) of the columns of vectors which on the left side appear smooth but as we visually move to the right we notice sections of the vectors take on a rougher appearance due to the benefits of training and learning.

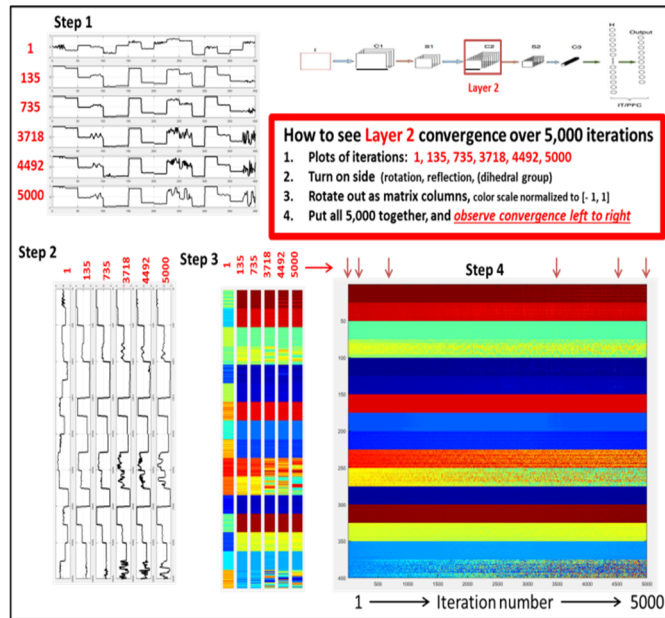


Fig. 5. Shows a four step process of converting output vectors from a CNN training process into a matrix that reveals an evolution of structure in neuron outputs across 5000 iterations.

It is noticeable the convergence is carried out in horizontal sections. A closer observation reveals 16 horizontal sections each with a vertical height of 25 rows which directly correspond to the 25 pixels (neurons) from the 5x5 patches in Fig. 4. An even closer look is given in Fig. 6 where we have focused our attention on the last two sections, 15 and 16.

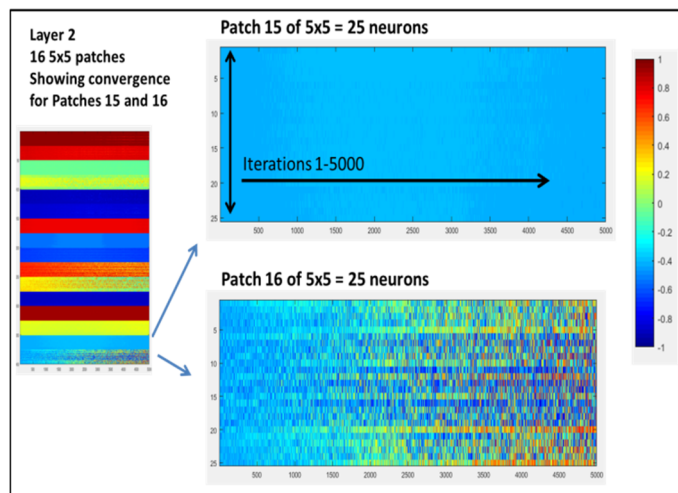


Fig. 6. Zooming in on sectional patches 15 and 16 from the matrix in Fig. 5 to demonstrate the contrast between neurons that are not learning (section 15) from neurons are learning (section 16).

We interpret the uniform coloring in section 15 as a section of neurons that are not learning in contrast to how the neurons are evolving in section 16 with a very visible diversity of colors representing the level of neuron activation. From an observational point of view, we ask the question, “May one assert that the neurons in section 15 are virtually useless and hence be a waste of computational resources?”

Figure 7 demonstrates that we may form matrices from each layer in the CNN. For this CNN, layers 1-3 require reshaping of image patches into vector form while layers 4 and 5 are already in vector form. Except for layer 1, the remaining layers exhibit learning as we follow the columns left to right. Worth noting is the fact that the matrix for layer 5 has only 10 rows. These 10 rows correspond to the ten MNIST numeral 0-9. We would like to point out that although we mentioned that the numerals were introduced randomly, more exact is the fact that we chose 500 each of the numeral 0-9 and it was this collection that the numerals were chosen randomly until all 5000 were exhausted. This was our basis for Fig. 8.

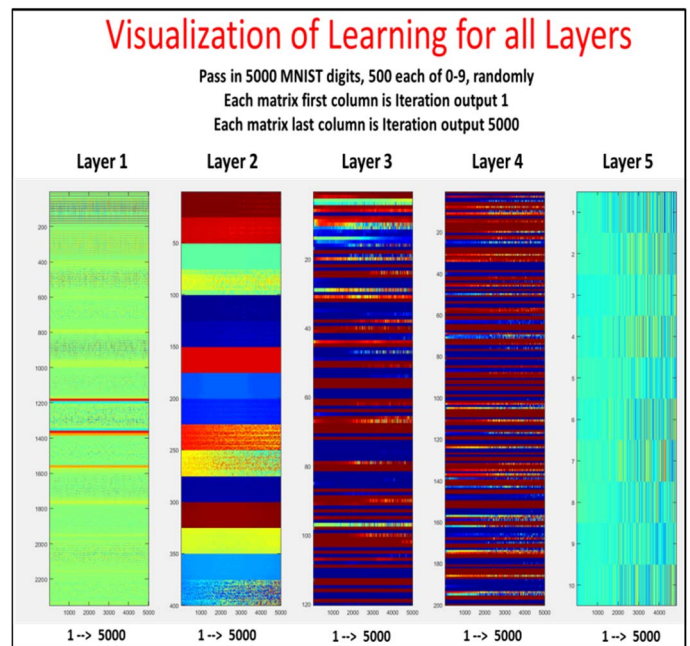


Fig. 7. Shows the learning matrix for each of the five layers. We note that over the course of 5000 iterations the numerals were introduced randomly.

Fig. 8 uses the grouping of the 10 sets of 500 0-9 numerals to examining the convergence process, per numeral. We collected the vectors from the matrix in Fig. 5 into groups according to the digit number. The top plots in Fig.8 show the ‘before grouping’ for layers 4 and 5, and the bottom plots show the ‘after grouping’. We will focus our analysis with layer 5 leaving the reader to back-track over to layer 4. Starting with the top figure for layer 5 we observe the beginning of a list of randomly selected numerals, (2,1,3,4,...), used in the training process. We note that there are 10 rows corresponding to the 10 MNIST numerals. For the visual effect to help in analysis each row is stretched vertically, but technically there are only

10 points in each vertical column. Moving to the bottom of Fig. 5 we note that the horizontal axis consists of the 10 collections of 500 numerals for each digit. Hence, each digit corresponds to a 10x500 sub-matrix of values which are partitioned from the 5000 iterations. In this way we may observe how each digit, on its own, is converging and we can also observe a diagonal effect of digit convergence by virtue of this organization we have imposed.

We now focus our attention on the numeral 6 in the bottom right of Fig. 8 where we have boxed in the part of 6's column that pertains to the place where we expect the highest value assigned would be assigned. Note this is the seventh row representing the seventh number in the sequence 0 to 9. We can see how the color in the row turns red which indicates improved decision making as we move through the iteration process. [The values are scaled from 0 to 1, where blue corresponds to 0 and red corresponds to 1.] We note that if we look at the column for the 4's digit we see high values for the 4 and 9 thus the numeral 9 poses a challenge for the correct labeling of numeral 4. On the other hand an observation of the column for the 1's digit shows little contention. The same type of conclusions may be made for layer 4 albeit it is a little more abstract, yet there is pattern to be exploited even with complex combinations of neurons.

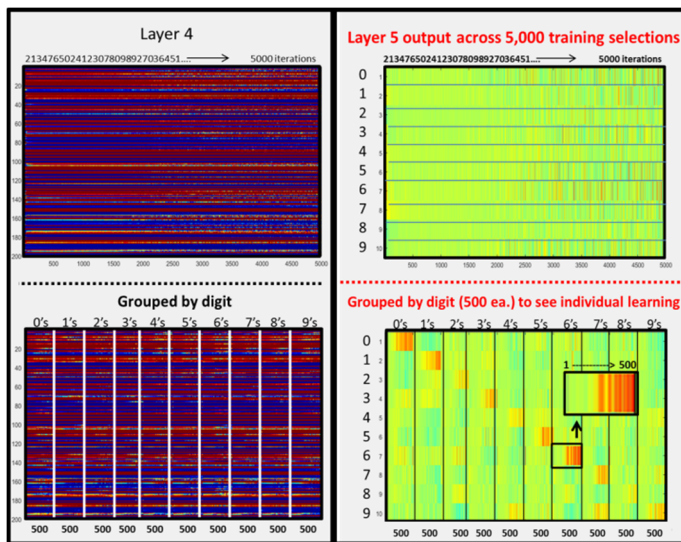


Fig. 8. We take the matrices from Fig. 7 and reorder the columns according to digit number. This enables an analysis of convergence of and contention among the various MNIST numerals.

In Fig. 9 we will take a look at the CNN end-processing stage across all the layers. We repeated our grouping of the digits like that done for layers 4 and 5, but for the remaining layers 1-3. We now take the right most column vector from each layer for each digit. This corresponds to taking the right most column vector from the 10 sections shown in Fig. 8. Consider these vectors the best trained outputs after 5000 iterations. In all, we end up with five sets of matrices where each have 10 columns, one for each digit, from each layer, and each having a different length (number of rows) according to the

CNN processing output at each layer. The top of Fig. 9 shows the matrices just described. The matrix for layer 5 exhibits signs of being diagonal and would only be more pronounced with more training or a better luck of the draw as to the quality of the digit which was passed through last. Upon close observation we can see differences in the individual columns making up a particular matrix (from a particular layer) as expected. However, we would like to place a stronger measure on these matrices by forming a cross-correlation matrix for each layer of vectors and this is what was done to form the matrices at the bottom of Fig. 9. Each of these bottom matrices is a correlation matrix corresponding to the matrix above it. With these correlation matrices we would expect a perfect correlation of 1 on the diagonal and low off-center values. For our example we see a strong diagonal trend with fringe layers 1 and 5 but we do not see such a contrast between the diagonal and off diagonal responses among the internal layers 2-4. The color bar on the far right has been set to range from -1 to +1. The large presence of red in layers 2-4 indicates a correlation coefficient between .8 and 1.0 for each set of vectors. This suggests that the processing being carried out in those layers is having difficulty in separating the ten different input objects for this CNN's instance of choice of neuronal interaction mappings.

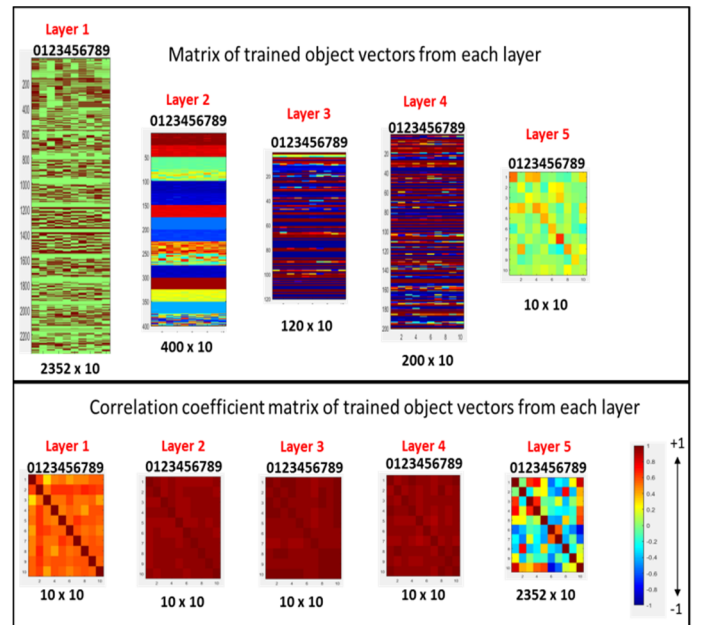


Fig. 9. Top: Shows the matrix of 10 convergent vectors for each layer, for each numeral 0-9. Bottom: Corresponding correlation coefficient matrices.

In this section we have shown that the inner layers to a CNN are not opaque, i.e., a CNN is not a black box. We have shown matrix o vector to matrix transformations that expose how neurons are learning, if they are learning at all. By these simple transformations one may fully observe a CNN process and in doing so come away with alternative mappings of neurons to make the most of computational resources. And now, that the layers have been exposed, we will turn to the next section of this paper where we use the top plots in Fig. 9 to

convert this feedforward-only CNN into a bidirectional matrix framework and in doing so, unintendedly, satisfying the UAT.

III. CALLING ATTENTION TO: A FULLY BI-DIRECTIONAL NEURAL NETWORK

As we stated previously, the main motivation for writing this paper was to shed light on the inner layers of the CNN. In doing so we realized that this is an opportunistic time to quickly show an application that works with the inner layers; one that goes beyond the pattern analysis of convergence and cross-correlation of the layers in the previous section; to one that includes an application that utilizes the inner layers to optimize the CNN. In this section we will show how to form a bidirectional neural network, within each layer, thus making a fully bi-directional neural network (BNN) out of a CNN.

Inspired by the work of Kohonen and after discussions with Stephen Grossberg, Kosko derived his version of a Bi-directional Associative Memory (BAM) [7, 8]. His formulation was a direct input-to-output process where both the input and output were distinct sets of objects to be associated numerically; there were no inner layers in his formulation, nor were there in subsequent similar formations by other researchers [9]. We state this to differentiate our application from the Kohonen and Kosko single-layer applications to the more germane, and more general, application of multilayer neural networks [3].

As an example we turn to Fig. 9 where the objects take on the form of the MNIST digits. Here we set up association matrices by pairing numerals, namely 1-2, 3-4, 5-6, and 7-8, as input/output pairs. We construct the four center outer product matrices by multiplying the vector on the left by the transpose of the vector on the right. Then, by adding the four outer product matrices together, we get what Kosko referred to as an additive matrix. Unfortunately, this additive matrix is very dependent on choosing a threshold to make the process of object association viable. In our work, we mitigated this problem by applying a series of PCA, ICA, and nullspace transformations to stabilize the matrix.

As an example of the utilization of the stabilized additive matrix we turn our attention to the red box in lower right side of Fig. 10. In the top left corner we see a random looking matrix of numbers. Actually this matrix is a noisy numeral 3 which is reshape into a vector that is passed through our stabilized additive matrix which in turn produces an output vector which is then reshaped into a matrix for observation. This process is now repeated in a reverse direction with the goal of finding which pair of numerals ‘resonate out’; in this case it’s a 3 and 4, as expected.

Now we apply this additive matrix formulation to the CNN vectors in the top of Fig. 9. Fig. 11 gives a pictorial description of how we form such matrices between two consecutive layers in the CNN. In this case, instead of relating different object numerals, we use the same numeral across different layers. In the dashed box we show how we form a 784x2352 matrix by:

- 1) Form 10 outer product matrices where the numeral 0 from L0 is associated with the output in L1 as input to form the first matrix and then repeating this process for the remaining numerals to form nine additional matrices.
- 2) Add the 10 matrices to form the single additive matrix, which is then stabilized by the PCA/ICA/Nullspace approach.

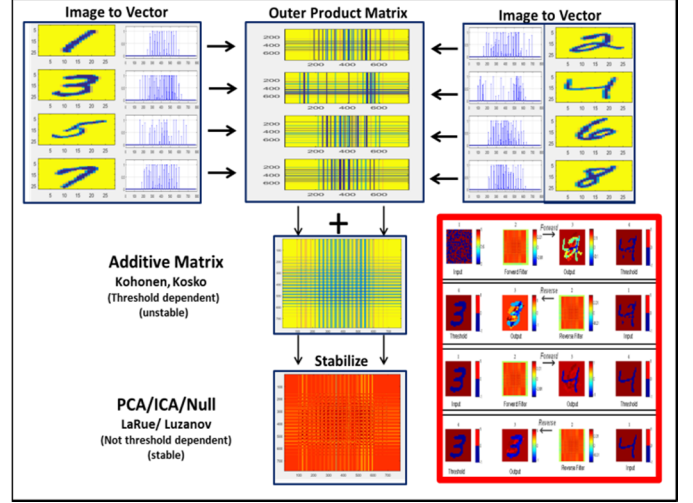


Fig. 10. Overview of how to form an additive matrix; the stabilization approach is described in reference 3. The bidirectional processing within the red box shows the resonance of a noisy numeral 3 and its associate, a 4.

We repeat this procedure for L1-L2, L2-L3, L3-L4, and L4-L5. The matrices, along with their corresponding dimensions are given. There is a note attached to the figure where we show how we can multiply the matrices together to arrive at a single matrix of dimension 784x10. Thus, we have shown a derivation of a process to achieve the results called for in the Universal Approximation Theorem of Hornik and Cybenko [10,11].

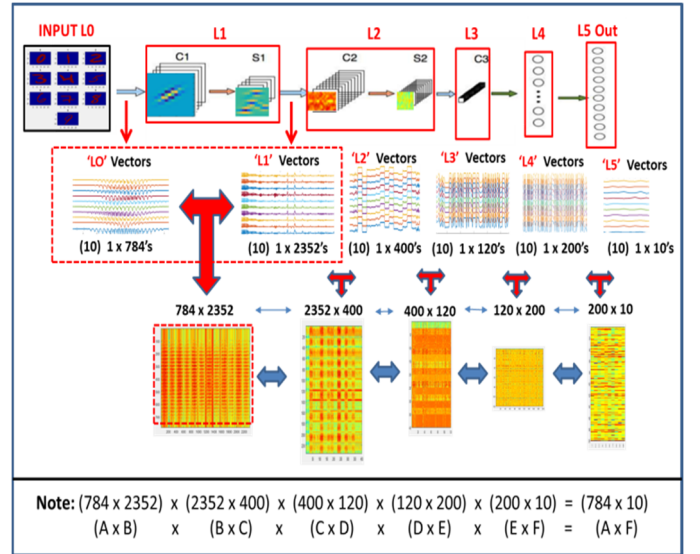


Fig. 11. Diagram of additive matrices made from consecutive layers of a convolutional neural network. These matrices may be multiplied together to form a single matrix and thus achieve a result from the UAT.

IV. SUMMARY

We had a simple goal in mind with this paper; to bring to light the deeper internal workings of a neural network and to show that the additional effort may bring a higher level of enlightenment as to how the internal parts work together, or for that matter, how they don't work. The fact that our unique process could in some way realize the Universal Approximation Theorem was ancillary to, and a direct outcome of, understanding how the system works; and one may call it lagniappe.

This paper is actually only a partial description of the NN anatomy. As discussed during the AIPR 2017 workshop, another set of vectors that may be beneficial to study are the weighting vectors that evolve during back propagation. We understand that a lot of detail has been left out but we believe our central point has been argued, that a neural network, no matter how deep it may be, is not a black box.

REFERENCES

- [1] Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", Computer Vision ECCV 2014, LNCS 8689, pp. 818-833.
- [2] Christopher Olah, Blog: <http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>
- [3] LaRue, J.P. and Luzanov, Y., "Stabilizing bidirectional associative memory with principles in independent component analysis and null space", SPIE Defense, Security, and Sensing, Paper 8745-76, 29 April - 3 May 2013, Baltimore, Maryland.
- [4] Werbos P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, (Doctoral Dissertation), Harvard University, Cambridge, MA, 1974.
- [5] LeCun Y., Bottou L., Bengio Y., and Haffner P., "Gradient-Based Learning Applied to Document Recognition," *Proc. of the IEEE*, pp. 2278-2324, 1998.
- [6] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in neural information processing systems*, 2012, pp. 1097-1105.
- [7] Kohonen T., "Correlation matrix memories," *IEEE Trans. Comput.*, pp. 353-359, 1972.
- [8] Kosko B., "Bidirectional Associative Memories," *IEEE Transactions On Systems, Man, and Cybernetics*, January/February, 49-60, 1988.
- [9] Acevedo-Mosqueda et. al., "Bidirectional Associative Memories: Different Approaches," *ACM Computing Surveys*, pp. 18.1-18.30, 2013.
- [10] Hornik K., "Approximation Capabilities of Multilayer Feedforward Networks," *Neural Networks*, pp. 251-257, 1991.
- [11] Cybenko G., "Approximations by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, pp. 303-313, 1989.