Neural Networks as Computationally Exact Logical Sensor Systems

Thomas C. Henderson and William Raley

UUCS-25-006

Kahlert School of Computing
University of Utah
Salt Lake City, UT 84112 USA

15 May 2025

Abstract

Neural networks can be viewed as a set of computational nodes (neurons) each of which computes some function over its inputs. This is a restricted form of a Logical Sensor System (LSS) where the neurons are logical sensors. Taking this viewpoint exposes several structural aspects of neural nets which can be augmented by the capabilities of LSS modules; this includes: semantic analysis of the system becomes possible, tests can be added to check for input correctness, alternate subnets can be specified to accommodate user requirements. Moreover, it becomes possible to replace a given neuron function with a computationally exact neural subnet.

As an example application of this approach, we demonstrate its effectiveness in dealing with convolutional neural networks. Convolutional neural networks trained on some specific image classification problem learn sets of convolution kernels which are applied to the input images to extract features relevant to the solution. It is generally believed that these learned kernels are tuned to the training data, and thus, should outperform standard image processing convolution filters. The study conducted here indicates that this is not always the case, and that it may be beneficial to either use standard convolution filters from the start, or to replace the learned kernels with the best matching computationally exact standard filter (e.g., a Gabor filter).

Neural Networks as Computationally Exact Logical Sensor Systems

Thomas C. Henderson* and William Raley*

Abstract— **Neural networks can be viewed as a set of computational nodes (neurons) each of which computes some function over its inputs. This is a restricted form of a Logical Sensor System (LSS) where the neurons are logical sensors. Taking this viewpoint exposes several structural aspects of neural nets which can be augmented by the capabilities of LSS modules; this includes: semantic analysis of the system becomes possible, tests can be added to check for input correctness, alternate subnets can be specified to accommodate user requirements. Moreover, it becomes possible to replace a given neuron function with a computationally exact neural subnet.**

As an example application of this approach, we demonstrate its effectiveness in dealing with convolutional neural networks. Convolutional neural networks trained on some specific image classification problem learn sets of convolution kernels which are applied to the input images to extract features relevant to the solution. It is generally believed that these learned kernels are tuned to the training data, and thus, should outperform standard image processing convolution filters. The study conducted here indicates that this is not always the case, and that it may be beneficial to either use standard convolution filters from the start, or to replace the learned kernels with the best matching computationally exact standard filter (e.g., a Gabor filter).

I. INTRODUCTION

Logical sensor systems [1]–[12] provide a clear computational framework for the specification and implementation of sensor-actuator systems. Figure 1 shows the organization of each logical sensor. First notice that a Logical Sensor (LS) component is a pull model in that a command arrives which invokes it to action and it must deliver a response to the caller whereas neural nets may be considered a *push model* in that they are viewed as feedforward functions. Each component has a unique name and a Command Control Interpreter which interprets the parameters of the command and then the Select Function chooses a subnet to produce the requested result. The Select Function can be viewed as a local knowledge base that understands the interface between calling LS components and called components. Monitors check the validity of the vlaues returned from the component and taps allow each element of the output to be checked for correctness. We have shown that an operational semantics of LSS networks allows an analysis of the network.

The ILLS methodology can be applied at both a macro scale where a neural net is one of the alternate subnets in an LS component and at the micro scale where each neuron of the neural network is an LS component. One possibility is to augment neural networks so as to incorporate the various aspects of an LS component; e.g., Command Control Interpreter, Select Function, Monitors, Taps, and subnets as connections to other neurons). This would require a neural network that allows connections between any neurons in the network, and thus, would not be feedforward. On the hone hand this complicate the neural network structure, but on the other is would also more closely emulate biological neural systems.

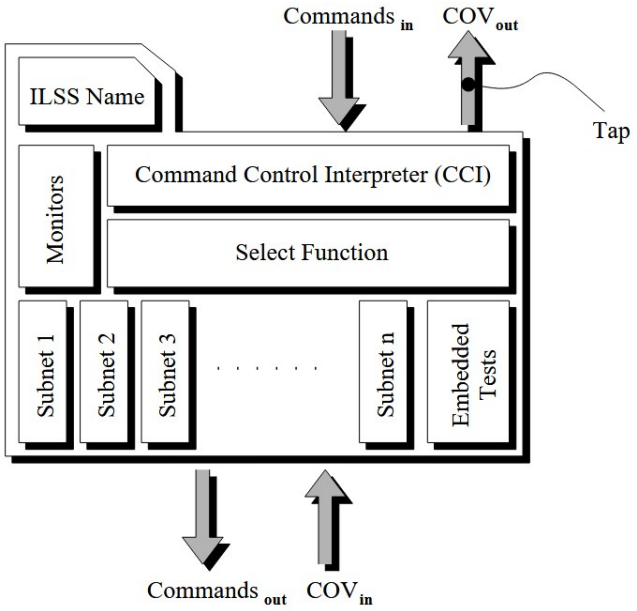

Fig. 1: Layout of Instrumented Logical Sensor.

The Instrumented Logical Sensor Systems framework is shown in Figure 2. The left of the figure indicates the mod-

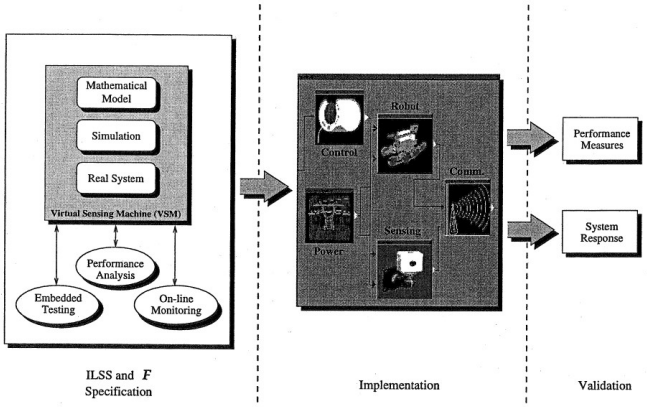

Fig. 2: The Instrumented Logical Sensor System Methodology.

*T. Henderson and W. Raley are with the Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA tch@cs.utah.edu

eling, simulation and real system design and specification part of the framework. Not only is the ILSS system (i.e., the LS components and their connections) specified, but a set of functions (called F in the figure) are given which are intended to measure system properties of interest. Next, the center of the figure indicates an implementation that is created either by hand or automatically. Finally, the right hand figure indicates the validation process by which the monitored and tapped values are compared with valid ranges which allows the determination of the robustness of and uncertainty in the system.

Given a set of such LS components, the ILLS approach will greatly enhance the computational capability and robustness of the network, but also allows each component to be analyzed as to its purpose (i.e., determine what a particular neuron function computes). Any neuron which is determined to compute (or approximate) a known analytic function could then be replaced with a computationally exact LS component. For example, if a neuron is determined to be approximating the atan2 function, then that neuron's input connections can be deleted and a new set of inputs from the actual arguments of the analytic function provided to the neuron. It may be necessary to create subnets that compute these inputs in a computationally exact way.

A. Computationally Exact Convolutional Neural Networks

Here, we explore a more restricted application of the ILLS methodology by analyzing individual neuron convolution functions and replacing them with computationally exact networks. That is, we propose to determine what function is being computed and replace the learned neural net computation with an exact computation. This problem is selected because the networks already have a reduced input structure to convolution neurons linking just the required sub-window of an image to produce pixel values in the output of the neuron. This corresponds to the notion described above of tailoring the inputs to a neuron to be just those required in an exact computation. The only thing up in the air is the values of the convolution kernel which are learned by the CNN.

The thesis is that that standard image processing convolution filters, e.g., Gaussian blur, anisotropic edge detectors like Sobel, isotropic edge detectors like the Laplacian of Gaussian, or various basis functions like Fourier or Gabor, result in higher accuracy for convolutional neural network image classifiers than the filters produced by the CNN learning phase.

This topic has been studied previously; [13] examined the relationship between learned kernels and level set computation and show that "neural networks that use strictly mean zero finite difference stencils as convolutional kernels can be treated as upwind discretizations of differential equations." Their work touches on an issue of interest to us -– the mathematical basis for these learned kernels. If there is a strong relation, then the goal is to replace the learned kernel with the exact computation it is trying to discover.

The goal of our work, however, is to determine which, if any, standard filter a learned filter is approximating, and then to replace it with that filter and see if the network classification performance improves.

II. METHOD

The method is as follows:

1) select for analysis a neuron in a trained neural net
2) determine, if possible, what function the neuron approximates
3) replace, if necessary, the current predecessor neurons with appropriate neurons that provide the computationally exact arguments to the neuron under analysis
4) provide a sub-network that provides a computationally exact function to replace the neuron under analysis
5) re-train the network while holding fixed the newly spliced in neurons and the weights across their connections.

The study of convolution neurons in a CNN neural reduces the method to (1) selecting a convolution neuron (and we restrict this to the first convolutional layer), (2) determining which standard kernel the neuron is trying to approximate, (3) replacing the convolution kernel values with the best matching computationally exact standard kernel, (4) leaving the network structure intact, and (5) re-training the network (with the newly splice convolution kernels not learned). In the domain studied here, step one will select only neurons in the first convolution layer.

In order to examine this hypothesis, two CNNs are studied: (1) a simple digit classifier called DigitNet, and (2) AlexNet, an image classifier trained on over 1 million images. DigitNet allows detailed analysis of the learned convolution kernels since the inputs are one-channel gray level images, and there are only four learned kernels in the first covolution layer. Moreover, the problem is small enough to allow direct re-training of the network on the original training dataset. AlexNet offers a more complex example with three-channel input and 96 convolution kernels in layer 1. To study the impact of replacing the kernels, a transfer learning problem is studied involving Coke bottle inspection.

1) A Standard Image Processing Filter Dataset: A set of 106 MxM filters is created with seven basic types (see Appendix A for Matlab code):

1) average
2) Sobel edge detector (4 orientations)
3) disk (average of circular area)
4) Gaussian (a range of variances are used)
5) Laplacian of Gaussian (a variety of variances are used)
6) Fourier basis functions
7) Gabor basis functions.

These filters are all created as NxN arrays and bicubic interpolation is used to generate them. Note that DigitNet uses 5x5 kernels while AlexNet uses 11x11 kernels.

A. DigitNet

A CNN digit classifier is created which results in a set of four learned convolution kernels in the first convolution layer. This net sets the baseline accuracy.

Filters are compared to the learned kernels in four ways:

1) The exact values of the matched kernel substituted into the baseline DigitNet with no retraining.
2) The exact matched kernels are substituted and held fixed while the net is re-trained.
3) Each standard kernel is scaled to the same range as the learned kernel and used without retraining.
4) Each standard kernel is scaled to the same range as the learned kernel and held fixed while the net is re-trained.

When re-training is performed, 10 trials are run and the mean statistics are used for comparison to the baseline accuracy.

B. AlexNet

A different approach is used with AlexNet. A transfer learning task is performed. A set of faults are defined for a Coke bottle, and bottles are to be categorized according to these faults (see Figure 3 for a set of example inspection images). These images are from a project from the text by Solomon and Breckon [14].

Fig. 3: A set of inspection images demonstrating the types of faults.

AlexNet is loaded into Matlab and is trained on the bottle fault classification problem. This serves as the baseline accuracy. The faults include:

- row 1, col 1: crooked label, overfilled, no cap
- row 1, col 2: deformed, overfilled
- row 2, col 1: no label
- row 2, col 2: missing bottle
- row 3, col 1: no cap, overfilled
- row 3, col 2: overfilled
- row 4, col 1: no cap, underfilled
- row 4, col 2: white lablel

If no fault is found, the bottle is OK and passes inspection.

Next, the 288 (i.e., 96*3) kernels are matched and replaced, and the resulting accuracies are compared to the baseline just as described for DigitNet.

III. EXPERIMENTS

A. DigitNet

DigitNet is a twelve-layer CNN (see Matlab help for exact code):

Layer	*Name*	*Type*	*Size*
1	imageinput	Image Input	28x28x1
2	conv_1	2D Conv	4 5x5x1
3	batchnorm_1	Batch Norm	Batch Norm
4	relu_1	Relu	Relu
5	maxpool_1	2D Max Pool	2x2 Pooling
6	conv_2	2D Conv	16 5x5x1
7	batchnorm_2	Batch Norm	Batch Norm
8	relu_2	Relu	Relu
9	maxpool_2	2D Max Pool	2x2 Pooling
10	fx	fully connected	10 fully connected
11	softmax	Soft Max	soft max
12	classouptput	Class Output	cross entropy

Here we focus on the first convolution layer (Layer 2) which has four 5x5 convolution kernels.

The base DigitNet is created using stochastic gradient descent with a learning rate of 0.01 and 8 epochs. The number of training samples ranges from 750 down to 150 in decrements of 100. Table 1 gives the accuracy results for these for the baseline and four substitution methods.

TABLE I: Accuracy Improvement of Kernel Matching Methods

Num Samps	Baseline	Exact	Exact/Learn	Scale	Scale/Learn
750	91.6	89.4	93.3	84.1	**95.0**
650	90.0	88.3	92.9	88.0	**93.1**
550	88.2	84.9	**89.5**	85.4	89.3
450	83.0	81.7	87.1	82.4	**87.4**
350	80.3	76.9	**85.2**	81.5	**85.2**
250	72.1	70.9	**77.1**	69.5	75.8
150	59.8	63.3	**67.0**	62.0	64.9

B. AlexNet

AlexNet [15] is an eight-layer convolutional neural network with five convolutional layers and three fully connected. Here we simply find the best match for each convolutional kernel in the first layer, and then retrain on the Coke bottle inspection problem with the standard filters replacing the learned AlexNet filters.

Figure 4 shows the training data for AlexNet learning the Coke Bottle Inspection data. Figure 5 shows the training data for the best matching standard filters which replaced the AlexNet filters.

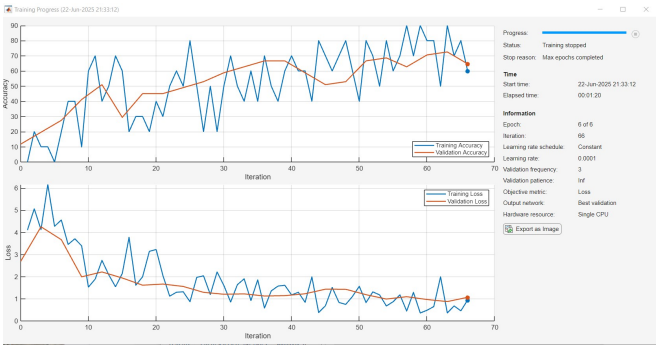

Fig. 4: The Matlab Training Session for AlexNet learning on the Coke Bottle inspection problem.

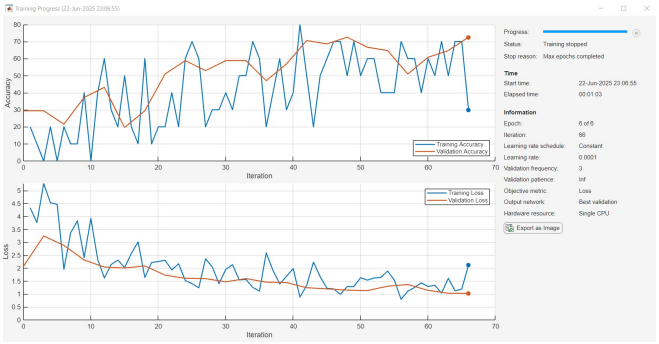

Fig. 5: The Matlab Training Session for the modified AlexNet learning on the Coke Bottle inspection problem.

The accuracy of both networks was 72.55%.

IV. CONCLUSIONS

The general methodology proposed here is to change the viewpoint of a neural net from that of a simple feedforward function to a more general paradigm by viewing each neuron as a computational component (function) interacting with other computational components (functions). If it is possible to determine what these learned functions are attempting to approximate, then they can be spliced out of the network and replaced by computationally exact sub-networks. This provides several advantages:

- the functions become explainable and their role in the overall neural net process can be considered.
- the functions can be replaced with computationally exact functions which allow for possible improvement in performance of the network.
- if each neuron has the features of a Logical Sensor component, then even more power is in the hands of the designer to include monitoring of e.g., the input and output data of the component as well as tap lines to allow insight into the operation of the component.

The experiments show several percentage points in accuracy improvement in the DigitNet case, and no loss of accuracy in the AlexNet example. Moreover, the DigitNet results indicate that it is possible to train on less data and still get accurate results, making it possible to lower the costs of training.

Future work includes:

- deeper set of experiments on large CNNs
- development of a neural network architecture that incorporates the Logical Sensor structure at the neuron level.
- study of more abstract functions in a neural network; that is, convolutional layers further downstream in the network, as well as neurons in the full connected layers of the network.

ACKNOWLEDGMENT

This work was supported in part by the Utah State Higher Education award for the Deep Learning in AI and Robotics prorgram.

REFERENCES

[1] M. Dekhil and T. C. Henderson, "Instrumented Sensor System Architecture," *Journal of Robotics Research*, vol. 17, no. 4, pp. 402–417, 1998.

[2] T. C. Henderson, C. Hansen, E. Shilcrat, and W. S. Fai, "Logical Sensor Specification," in *Proc of the SPIE Conference on Intelligent Robots*, Cambridge, MA, 1983, pp. 578–583.

[3] T. C. Henderson, E. Shilcrat, and C. Hansen, "A Fault Tolerant Sensor Scheme," in *International Conference on Pattern Recognition*, Montreal, CA, 1984, pp. 663–665.

[4] T. C. Henderson and E. Shilcrat, "Logical Sensor Systems," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 169–193, 1984.

[5] T. C. Henderson, W. S. Fai, and C. Hansen, "MKS: A Multi-sensor Kernel System," *IEEE-T Systems, Man and Cybernetics*, vol. 14, no. 5, pp. 784–791, 1984.

[6] T. C. Henderson, C. Hansen, and B. Bhanu, "A Framework for Distributed Sensing and Control," in *International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985, pp. 110–1109.

[7] ——, "The Synthesis of Logical Sensor Specifications," in *SPIE Conference on Intelligent Robots*, Cambridge, MA, 1985.

[8] ——, "The Specification of Distributed Sensing and COntrol," *Journal of Robotic Systems*, vol. 2, no. 14, pp. 387–396, 1985.

[9] T. C. Henderson and E. Weitz, "Multisensor Integration in a Multiprocessor Environment," in *ACME Conference on Concurrent and Supercomputing*, New York, NY, 1987.

[10] T. C. Henderson, A. Mitiche, E. Weitz, and C. Hansen, "Multisensor Knowledge Systems: Interpreting 3-D Structure," *Journal of Robotics Research*, vol. 7, no. 6, pp. 114–137, 1988.

[11] T. C. Henderson and R. Grupen, "Logical Behaviors," *Journal of Robotics Systems*, vol. 7, no. 3, pp. 309–336, 1990.

[12] T. C. Henderson and M. Dekhil, "Instrumented Logical Sensor Systems – Practice," in *IEEE Conference on Robotics and Automation*, Leuven, Belgium, May 1998.

[13] J. Actor, D. Fuentes, and B. Riviere, "Identification of Kernels in a Convolutional Neural Network: Connections between Leve l Set Equaton and Deep learning for Image Segmentation," in *SPIE Int. Soc. Opt. Eng.*, February 2020.

[14] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing*. Oxford, UK: John Wiley and Sons, 2011.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing 2012*, 2012.