Learning at the Speed of Light: A New Type of Optical Neural Network

A. Steven Younger[1] and Emmett Redd[2]

[1] Jordan Valley Innovation Center, Missouri State University, Springfield, MO 65897 USA
[2] Department of Physics, Astronomy and Materials Science, Missouri State University
{SteveYounger,EmmettRedd}@MissouriState.edu

Abstract. Most, if not all, optical hardware-based neural networks are slow during the neural learning phase. This limitation has been not only a speed bottleneck, but it has contributed to the lack of wide-spread use of optical neural systems. We present a novel solution – Optical Fixed-Weight Learning Neural Networks. Standard neural networks learn new function mappings by the changing of their synaptic weights. However, the Fixed-Weight Neural Networks learn new mappings by dynamically changing recurrent neural signals. The (fixed) synaptic weights of the FWL-NN implement a learning "algorithm" which adjusts the recurrent signals toward their proper values.

Keywords: Optical Neural Networks, Optical Computing, Fixed-Weight Learning Neural Networks, Adaptive Neural Networks, Accommodative Neural Networks.

1 Introduction

Optical hardware is probably the fastest method of performing the forward-propagation phase of neural networks. An optical neural computer similar to those presented in [1, 2] should be able to perform over 10^{13} synaptic operations per second using current technology. Optical Neural squashing computations can now be performed on the sub-picoseconds time scale [3].

Most, if not all optical hardware schemes are slow during the neural learning phase. Optical learning has traditionally been done on a separate (non-optical) computer and the results stored on film, or required the use of a relatively slow (and/or expensive) spatial light modulator. This limitation has been not only a speed bottleneck, but it has contributed to the lack of wide-spread use of optical neural systems.

We present a different solution – Optical Fixed-Weight Learning Neural Networks (Optical FWL-NN). Standard neural networks learn new function mappings by the changing of their synaptic weights. However, the FWL-NNs learn new mappings by dynamically changing recurrent neural signals. The (fixed) synaptic weights of the FWL-NN implement learning "algorithm" which adjusts the recurrent signals toward their proper values. That is, instead of encoding a particular mapping, the synaptic weights of a FWL-NN encode how to learn any mapping (within a large, perhaps infinite, set of possible mappings).

We developed an optical hardware neural network to investigate the precision, alignment, calibration, speed, and algorithmic issues associated with Optical FWL-NNs. We report on the hardware design, generation of the synaptic weights, and initial results for some Fixed-Weight Learning tasks.

2 Optical Neural Hardware

Our optical hardware was not designed to be especially fast or to accommodate exceptionally large networks. It serves as a test apparatus for studying Optical Fixed-Weight Learning Neural Networks. Flexibility of use and (relatively) low cost were our main design criteria. With what we have learned, we are in the process of designing a fast, compact and expandable Optical Neural Network platform.

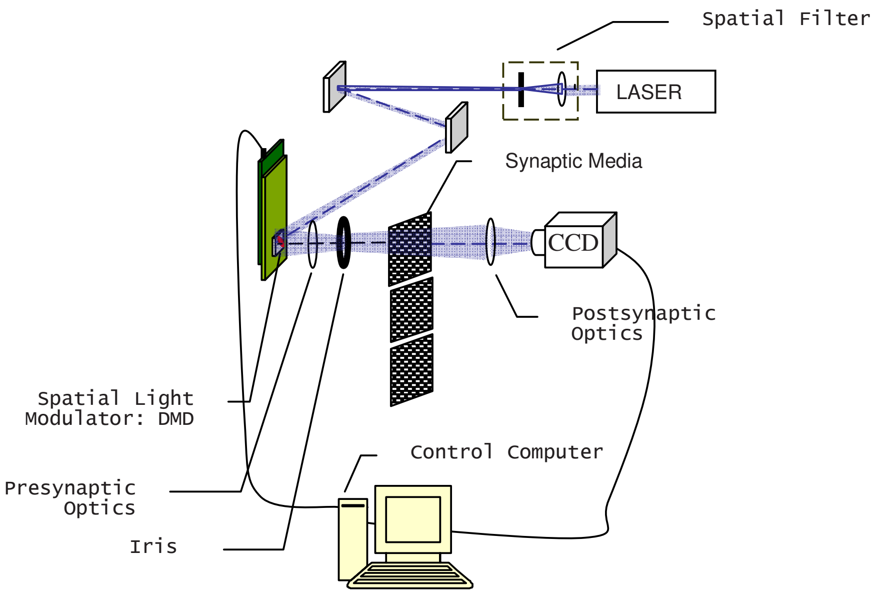

Fig. 1. Optical Neural Hardware

2.1 Hardware Overview

Figure 1 shows the optical neural hardware test apparatus. Light from a laser is expanded and directed toward a Spatial Light Modulator (SLM). The SLM creates the neural signals by modulating the intensity of a set of light beams. We used a Digital Micromirror Device (DMD) for the SLM. The DMD consists of a rectangular array of almost 1 million tiny mirrors along with drive and interfacing electronics. Under software control, each mirror can be individually set to either on (reflecting its beam toward the presynaptic optics) or off (reflecting away). The resulting signal beams pass through pre-synaptic optics and onto the synaptic medium, (35mm slide). The slide has small rectangular areas of various shades of gray that encode the synaptic weights. Synaptic multiplications are performed by the attenuations of the several light beams passing through the medium.

The attenuated optical signals are focused onto a CCD array and sent to the computer. The optical signals are spatially integrated over each region-of-interest. These dendrite signals are then sign-summed and the nonlinear squashing function applied, producing the network outputs. These last functions are currently performed in software.

2.2 Distortion, Alignment and Calibration

The 35 mm film synaptic medium can be generated with high spatial precision, as can the positions of a source neuron on the DMD and terminal neurons on the CCD. However, the pre- and post-synaptic optics generate considerable distortion. One solution would have been to create elaborate optics to eliminate these distortions. We decided to use more flexible software processing to correct the distortions.

There are two distortions to correct. First, the distortion of the DMD image caused by the pre-synaptic optics must be canceled out before it reaches the fixed synaptic medium, where precise registration is critical. Second, we must correlate the CCD positions of the (now attenuated) dendrite images which are further distorted by the post-synaptic optics.

The first distortion is hard to measure because we can't directly view the image projected onto the synaptic media. Instead, we project a rectangular array of dots (called *pegs*) from the DMD through a transparent slide and onto the CCD. By automatically measuring the CCD coordinates of the pegs, and knowing where they are on the DMD image, we computed a transformation matrix $DMD-to-CCD$.

The second distortion can be more directly measured by sending an *all pixels on* signal to the DMD, and projecting the light through a slide of a rectangular array of *holes*. These holes are clear areas on an otherwise opaque slide. They are at the same relative positions on the film as the pegs were on the DMD. By semi-automatically capturing the CCD coordinates of the holes, and knowing where they are on the slide, we computed the transformation matrix $Slide-to-CCD$.

Since $DMD-to-CCD = DMD-to-Slide \times Slide-to-CCD$, we can compute the matrix $DMD-to-Slide$, which transforms from the DMD to the synaptic medium. From this information, we determined how to *pre-distort* the DMD image in such a way that the presynaptic optics *undistorts* it and cause the DMD image to arrive properly aligned with the synaptic slide. That is, to make the pegs match the holes.

We performed the matrix calculations with up-to linear, quadratic and cubic terms in the matrices. The cubic calculation performed the best. It can correct for translations, rotations, stretching, keystoning, pincushioning and barrel distortions. We found that 42 pegs/holes (6 x 7) gave good results. This created over-determined transformation matrices. We used the Moore-Penrose pseudo inverse to find a least-squares solution.

Figure 2 shows a CCD image of 30 synapses projected from the DMD through the slide and onto the CCD. The gray levels differ due to the attenuation by the slide. Notice substantial distortion of the (originally rectangular) areas. The white boxes

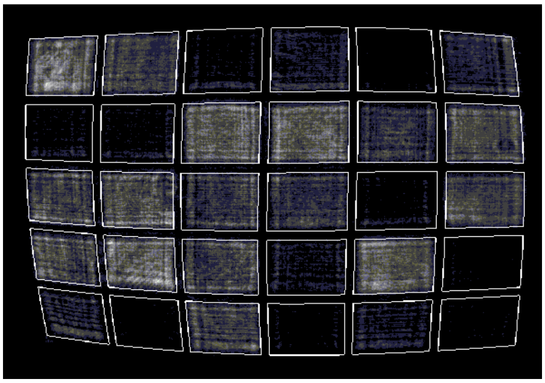

Fig. 2. Actual and Computed Regions-of-Interest

around the gray areas show where the projected synapses are expected to be -- based on the matrix calculations. Note that the computed locations are in excellent agreement with the actual image projections, although they are substantially distorted.

2.3 Encoding the Synaptic Weights

Film has a large storage capacity of 10 MegaPixels for a 35mm slide. A high-quality Holographic Plate can store an order of magnitude more information [2]. To reduce problems associated with film's very non-linear grayscale, we used a binary pixel area encoding for our synapses. A synaptic weight of $W \in [0 \ldots 1]$ will have a portion of W of randomly selected pixels within its area set to clear, and $1-W$ of its pixels set to opaque. This worked out to be over 16 significant bits of gray level precision for the synapses in our networks.

The accuracy of the film medium proved to be more problematic. The lack of reproducibility of the actual gray level from slide-to-slide, and even between different areas of the same slide, was a major difficulty to be solved for fixed-weight learning to be successful.

Our solution this problem was to calibrate each dendritic area individually. Doing this *automatic gain control* once-per-phase also solved problems of the laser light source intensity and beam profile drifting over time.

2.4 Encoding the Neural Signal Intensity

The individual DMD pixels can be in one of two states – 0 or 1, on or off. This limits the type of intensity modulation schemes that can be used with this design. We tested three modulation schemes.

1. Area Pixelation (AP): Because the source areas contain many pixels (several thousand in our examples) a gray level of between zero and one can be produced by turning on the number of pixels proportional to the desired signal intensity. These pixels were uniformly and stochastically selected over each source rectangle.

2. Pulse Width Modulation (PWM) – generate N time slices according to the number of required significant bits of the signal. (e.g. 256 time slices for 8 significant bits.) Turn all the pixels in a rectangular area on a number of time slices proportional to the desired signal value, and then turn them off for the remaining pulse train time slices.
3. Stochastic Pulse (SP) – same as PWM, except the pixels are switched on and off stochastically in the proper portions to create the desired signal value [10].

The AP had the advantage of being much faster because it only used one time slice instead of 256 or more. However, the response of the system was very non-linear.

The PWM and SP methods both have the advantage of being able to adjust the number of required signal significant bits by changing the number of time slices. Speed can be sacrificed for accuracy, or vice versa. The accuracy of both the PWM and SP methods was essentially the same. The PWM was slightly faster on our hardware.

2.5 Wave Effects: Diffraction and Interference

There were two main problems created by to the wave properties of the light. First, the periodic arrangement of the DMD formed a 2-D diffraction grating, creating multiple copies of the DMD image to be formed. This problem proved easy to solve, since the zeroth-order image was clearly brighter and easily identifiable. A converging lens in the presynaptic optics produced clearly separated diffraction images. An iris excluded all images but the zeroth order.

The second problem was created by the diffraction of the light from the individual rectangular source areas on the DMD. These are rectangular apertures which create a diffraction pattern of light, some of which spills outside of the rectangular image on the film and on the detector. This was the major source of crosstalk between synaptic areas. Even a fairly small amount of crosstalk was highly detrimental to the network accuracy. We solved this problem by increasing the dark borders between the source areas to about 20% of the size of the rectangle. This diffraction imposes a limit on the number of synapses that can be handled by this optical neural network design. A goal of our future designs is to eliminate this problem.

2.6 Clock Issues: Cycles, Phases and Pulses

All neurons in our networks were synchronous. Each neuron computes its next state based on the current activations of its source neurons, but does not change its output until all neurons have finished computing their next state. Then all neurons on the same class change their state simultaneously. There are two classes of neurons: *per-pulse-update* and *per-phase-update*.

Our optical hardware network has three levels of timing. External cycle, internal phase and signal pulses. External cycles represent one network input vector being processed to generate a network output vector. That is, one *exemplar* is processed. The network input vector is applied at the beginning of a cycle and remains unchanged until the next cycle. The network's output vector is decided at the end of a cycle.

An internal phase is the time it takes a signal to forward-propagate one layer. Typically, our networks may have three to six internal phases for each external cycle. The per-phase-update neurons change their outputs at the end of an internal phase.

For pulse-based intensity modulation schemes, each internal phase is divided into a number of pulses. The number of pulses depends on the number of significant bits required for the neural signals. It was typically 256 pulses for 8 significant bits. The DMD is updated and a new CCD image acquired each pulse. The per-pulse-update neurons change their output at the end of each pulse.

3 Fixed-Weight Learning Neural Networks

3.1 The Fixed-Weight Learning Theorem

Fixed-Weight Learning (also called Adaptive or Accommodative Neural Networks) has been investigated by several researchers [4-9] However, this is (as far as we know) the first reporting of results from FWL-NNs implemented in special hardware, whether optical or electronic. Previous papers have been mostly concerned with their highly adaptive nature and/or their use in optimizing learning.

In [4] Cotter and Conwell proved the Fixed-Weight Learning Theorem: Given a neural network topology (which learns by changing weights) and its attendant learning algorithm, there exists an equivalent FWL-NN. Any mapping that can be learned by changing the weights of the original network can be learned by the FWL-NN without changing any synaptic weights.

The FWL-NN learns because a learning algorithm is encoded in its (fixed) synaptic weights. The learned function mapping information is dynamically stored in recurrent neural signals. We call these signals *potencies* (also known as *flying weights* [11]) to distinguish them from the standard synaptic weights.

A FWL-NN can learn the full range of mappings that its non-fixed-weight equivalent network can learn. However, there are costs associated with fixed-weight learning. The FWL-NN will (almost always) be larger than the equivalent changing-weight network. This is because it also has to perform the learning computations along with the mapping computations of the equivalent network. Also, FWL-NNs are necessarily recurrent even if the initial equivalent network was not.

All of the FWL-NNs presented here perform *on-line* learning. The target value of the previously presented exemplar (during the last external cycle, *t-1*) is provided to the network. Alternatively, the error of the network output for the previous exemplar could have been provided. In general, on-line learning is not a requirement for FWL-NNs.

3.2 Creating the Fixed-Weight Learning Networks

The method we used for this work assumes that the network can be divided into two main parts: the *planapse* and the *tranapse*. The planapse (from $\pi\lambda\alpha\nu\eta$ meaning *error*) performs the potency update calculations, and the tranapse performs 'the potency signal times the input signal' calculations. In our networks, there is one planapse and one tranapse for each synapse in the equivalent non-fixed-weight equivalent network.

The planapse and tranapse computations are performed by sub-networks that were trained separately and integrated together to form the FWL-NN.

Planapse. The planapse sub-network was trained to learn well-known (on-line) Backpropagation Learning Rule:

$$\begin{array}{ll} \multicolumn{2}{l}{\Delta P(t) = x(t-1) \times y(t-1) \times \left(1 - y(t-1)\right) \times \left(y(t-1) - T(t-1)\right), \text{ where}} \\ t & \text{:current exemplar (external cycle)} \\ t-1 & \text{:exemplar one cycle previous} \\ x & \text{:input to synapse} \\ y & \text{:output of neuron} \\ T & \text{:Target value for neuron} \\ \Delta P & \text{:Change in Potency (flying weight signal)} \end{array} \tag{1}$$

This can be a bit confusing since the Backpropagation learning rule was used to train the planapse on the Backpropagation learning rule. Of course, other learning rules could be used for either.

The training data sets were generated by choosing random values for the inputs, and using the mapping formula to compute the targets. Note that a feedback signal consisting of the target value $T(t-1)$ for the previous data exemplar must be provided to the FWL-NN. Alternatively, an error signal, such as $e(t) = y(t-1) - T(t-1)$ could have been used as the feedback. The ΔP can be either positive or negative (bipolar), but optical intensity signals are unipolar. We scaled the calculations so that zero was represented by light as half intensity, the most negative signal was represented as no light intensity, and the most positive signal was represented by full light intensity.

Tranapse. The tranapse sub-network was trained to perform a scaled version of $s(t) = P(t) \times x(t)$. That is, a *Potency signal* times an *input signal.* We named this sub-network a *tranapse*, because a tranapse is to a (artificial) synapse as a transistor is to a resistor. The same bipolar scaling method was used as with the planapse. In addition, the tranapse output was scaled to effectively extend the potency range to $-\omega \leq P \leq +\omega$, where ω was usually 4.0.

3.3 From Software Synaptic Weights to Optical Attenuations

The (fixed) synaptic weights ranged from -10 to +10. However, attenuation of light physically ranges from 0 to 1. The sign of the synaptic weight was known by our software neurons, so the attenuation needed to only encode the magnitude of the weights. The maximum synaptic weight magnitude was determined for each neuron. Each of the neuron's synaptic weights was divided by this maximum weight magnitude to compute the required attenuation. To compensate for this weight scaling, each neuron has a constant multiplicative scale factor which is equal to the maximum weight magnitude. This "extra" scale factor should require no extra opto-electronic hardware, since the optical signal must be amplified anyway as part of the light detection process.

We tried various ways of dividing the planapse/tranapse pairs into sub-networks. For instance, should the multiplications be separately trained? Should the planapse and tranapse operations be done within the same-subnet with simultaneously training? Each of these sub-network configurations had its own strengths and weaknesses.

One lesson we learned is that the precise numerical addition of two signals that are very different in magnitude (such as the weight update from *t-1* to *t*) is very difficult for a network to learn, and easiest done by "hand" – that is, wiring up a linear neuron with appropriate synaptic weights.

4 Experimental Testing Results

4.1 Sub-network Training

Table I illustrates (software-based simulation) performance data for a sub-network trained to perform unsigned multiplication (for instance, it may be used to perform the $x(t-1)\times y(t-1)$ calculation in the planapse). Training was performed using the MATLAB *nntool.m,* using the automatic step size adjustment option *traingdx*. We trained each of the networks for 100,000 epochs of 10,000 randomly-generated training pairs. The large number of epochs was necessary to reduce the network errors. The relatively large training data set helped reduce overlearning. All of the planapse/tranapse schemes required a similar amount of training.

We believe that the small increase in error for more than 7 hidden neurons could be reduced by more epochs training on these larger networks.

A separately generated data set was used to test the sub-networks after they were trained.

The MSE and SigBits columns were calculated from:

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{n=1}^{N}(y_n - T_n)^2 \\ SigBits &= -\log_2\left(\frac{1}{N}\sum_{n=1}^{N}\left|y_n - T_n\right|\right), \text{ where} \\ N &= \text{Number of Exemplars in Test Set} \end{aligned} \tag{2}$$

As the table shows, the accuracy of the sub-network depends on the number of hidden nodes. This points out a property of FWL-NNs -- the *size* of the neural network required to learn a mapping set depends on the *accuracy* needed to learn the mapping set.

4.2 Testing on Optical Hardware

Table 2 shows the results of testing two neural networks on the optical hardware. The first network is an unsigned multiplication (*uMULT*). This is the same feed-forward network shown in Table 1 and Figure 2. The test data was a set of exemplars with two random inputs $x_1, x_2 \in \{x | 0 \le x < 1 \subset \Re\}$ and one target $T = x_1 \times x_2$.

The second network is *PlanTran*, a FWL-NN that is equivalent to a network with a single changing synaptic weight, with logsig squashing function and trained by Backpropagation. This FWL-NN is made from a single Planapse – Tranapse pair.

Table 1. (Simulated) Unsigned Multiplication. Hidden Layer Size vs. Mean Squared Error (MSE), and number of significant bits of result. Size of Training Set: 10,000. Epochs: 100,000.

Hidden Layer	***MSE***	***Sig Bits***
3	6.5003×10^{-4}	5.3
4	3.6876×10^{-4}	5.7
5	3.0794×10^{-4}	5.8
6	3.1636×10^{-5}	7.5
7	2.1617×10^{-5}	7.7
8	4.0069×10^{-5}	7.3
9	5.4367×10^{-5}	7.1

Generating Test Data for FWL-NNs. The algorithm to generate training/test data for a FWL-NN is:

```
repeat Number-of-Mappings times
  Randomly select a mapping M from a set of mappings S.
    repeat Number-of-Exemplars-per-Mapping times
      Generate a random input vector x
      Use x with mapping M generate target vector T
      Output training pair (x,T)
     end repeat
end repeat
```

For *PlanTran*, **S** was the set of all function mappings $T = \text{logsig}(M \cdot x),\ -4 \le M \le +4$, where the real index *M* specifies the particular mapping. The set **S** represents all mappings that a single-synapse neural network (with logsig squashing function) can (in theory) learn exactly.

There are several important parameters that were measured during the network testing. Size of the network, the number of internal clock phases per external clock cycle (exemplars), the average number of cycles the network required to learn a mapping, the residual error of the FWL-NN after learning has occurred.

For *PlanTran*, the number of SigBits was computed by:

$$SigBits = -\log_2\left(\frac{1}{N-L}\sum_{n=L+1}^{N}\left|y_n - T_n\right|\right),\ \text{where}$$

N = Number of-Exemplars

L = Number of Exemplars Required to Learn Mapping

Table 2. Experimental Results on Optical Hardware. L- Number of Layers, N–number of neurons, W–number of synapses, ϕ – Phases per Exemplar, Pulses – Number of pulse timeslots in one Phase. Learn – Number of Exemplars required to learn mapping (for FWL-NN) , MSE – mean squared error (after learning), SigBits – Number of Significant Bits.

NN	L	N	W	ϕ	Pulses	Learn	MSE	SigBits
uMULT	3	13	30	2	128	n/a	0.0013	~6
PlanTran	4	29	100	6	256	11	0.0083	~4

5 Conclusion and Future Work

The initial *uMULT* results show that the optical hardware can perform the unsigned operation to moderate precision (six bits or more). The *PlanTran* network results demonstrates that Fixed-Weight Learning can work on an optical hardware platforms. However, a more accurate and reproducible method of creating optical attenuation than 35mm film may be necessary for larger networks.

Both of the above neural networks are fundamental "building blocks" on which larger FWL-NNs can be constructed. We are currently performing ongoing measurements and testing to extend these results.

Based on what we have learned while designing, building, and testing this optical hardware, we are designing a new hardware platform to support these new FWL-NNs. The goal of this research and development is to create a practical hardware platform capable of performing large, complex real-world neural computation tasks at very high speeds.

Acknowledgments. This material is based on work supported by the United States National Science Foundation under Grant No. 0725867.

References

1. Keller, P.E., Gmitro, A.F.: Operational Parameters of an Opto-Electronic Neural Network Employing Fixed-Planar Holographic Interconnects. World Congress on Neural Networks (1993)
2. Abu-Mostafa, Y.S., Psaltis, D.: Optical Neural Computers Scientific American (March 1987)
3. Kubler, C., Ehrke, H., Huber, R., Lopez, R., Halabica, A., Haglund, R.F., Leiterstorfer, A.: Coherent Structural Dynamics and Electronic Correlations during an Ultrafast Insulator-to-Matal Phase Transition in VO_2. Physical Review Letters. PRL 99, 116401 14 (September 2007)
4. Cotter, N.E., Conwell, P.R.: Learning algorithms and fixed dynamics. In: Proceedings of the International Conference on Neural Networks 1991, vol. I, pp. 799–804. IEEE, Los Alamitos (1991)
5. Feldkamp, L.A., Prokhorov, D.V., Feldkamp, T.: Conditioned Adaptive Behavior from Kalman Filter Trained Recurrent Networks. IEEE, Los Alamitos (2003)
6. Younger, A.S., Conwell, P.R., Cotter, N.E.: Fixed-Weight On-Line Learning. IEEE Transactions on Neural Networks 10(2), 272–283 (1999)

7. Prokhorov, D.V., Feldkamp, L.A., Tyukin, I.Y.: Adaptive Behavior with Fixed Weights in RNN: An Overview. In: IJCNN 2002. IEEE, Los Alamitos (2002)
8. Lo, J.T., Bassu, D.: Adaptive vs. Accomodative Neural Networks for Adaptive System Identification. In: IJCNN 2001, IEEE, Los Alamitos (2001)
9. Hochreiter, S., Younger, A.S., Conwell, P.R.: Learning To Learn Using Gradient Descent. In: Proceedings of the International Conference on Artificial Neural Networks, Springer, Heidelberg (2001)
10. Bade, S.L., Hutchings, B.L.: FPGA-Based Stochastic Neural Networks. In: Implementation IEEE FPGAs for Custom Computing Machines Workshop, Napa, CA, pp. 189–198 (1994)
11. Werbos, P.: Private Communication (2004)