Application of the ANNA Neural Network Chip to High-Speed Character Recognition

Eduard Säckinger Bernhard E. Boser Jane Bromley
Yann LeCun Larry D. Jackel

March 18, 1991

Abstract

A neural network with 136,000 connections for recognition of handwritten digits has been implemented on a mixed analog/digital neural-network chip. The chip is capable of recognizing 1,000 characters per second with essentially the same error rate (5 %) as a simulation of the network with floating-point precision.

1 Introduction

We have designed, fabricated, and tested a reconfigurable neural network chip, called the ANNA chip (for Analog Neural Network ALU). The chip is optimized specifically for locally connected, weight sharing networks and time-delay neural-networks (TDNN) but can also be used for a wide variety of other topologies like fully connected and recurrent networks. A detailed description of the chip has been published in [1, 2].

A very successful neural network for optical character recognition (OCR) has been developed in our department [3]. The network performs particularly well in recognizing noisy, handwritten characters and in estimating the confidence of the classification. The good performance is due to an advanced architecture featuring local connections and weight sharing. This architecture is well suited for implementation on the ANNA chip.

Many neural network chips have been developed to date, but demonstration of large real-world applications for them is often neglected. The implementation of the OCR network with a total of 136,000 connections on just one ANNA chip clearly shows its practical significance.

Figure 1: Simplified architecture of the ANNA chip.

Important questions, like the speed advantage gained through this special-purpose hardware and the impact of low resolution arithmetic on the classification accuracy are discussed in this text.

2 The ANNA Neural Network Chip Architecture

The ANNA neural network chip is implemented in a $0.9\,\mu$m CMOS technology, contains 180,000 transistors, and its die size is $4.5 \times 7\,\mathrm{mm}^2$. The chip implements 4096 *physical* synapses which can be time multiplexed in order to realize networks with many more than 4096 (shared) connections. The resolution of the synaptic weights is 6 bit and that of the states (input/output of the neurons) is 3 bits; additionally a 4 bit scaler per neuron extends the dynamic range of the weights. Although the chip uses analog computation techniques internally, all input/output of the chip is digital. This combines the advantages of high synaptic density, low power, and easy interfacing to a digital system (e.g. digital signal processor DSP).

In the following a simplified account of the chip's architecture will be presented (see Fig 1) This description leaves out many details but is sufficient to understand the implementation of the network on the chip. More detailed descriptions can be found in [1, 2].

The chip evaluates eight dot-products of state $\boldsymbol{x}$ and weight $\boldsymbol{w}_i$ vectors in parallel. The eight scalar results are passed through a squashing function $f(\cdot)$ yielding the neuron outputs z_i. The whole evaluation process takes 200 ns or four clock cycles. The chip can be reconfigured for weight and state vector sizes of 64, 128, and 256. These figures also correspond to the number of synapses per neuron.

The input state-vector $\boldsymbol{x}$ is provided by a shift register which can be shifted by 1, 2, 3, or 4 positions in each clock cycle (50 ns). Correspondingly 1, 2, 3, or 4 new data values are read into the left end of the shift register in each cycle. This barrel shifter serves two purposes: (i) Because of pin limitations, it is not possible to load the whole state vector (256×3 bits) in parallel onto the chip; therefore sequential loading is imperative. (ii) A barrel shifter is the ideal preprocessor for networks with local receptive fields and weight sharing as well as time-delay neural-networks. This will be clarified in the section about the network implementation. The barrel shifter on the chip has length 64, but can be extended to larger sizes by means of an associated vector-register file.

A total of 4096 weight values are stored on the chip. These values can be grouped into

Figure 2: General structure of the OCR network.

Figure 3: Example for the states of the OCR network.

vectors of size 64, 128, and 256 in a flexible way. It is for instance possible to have 32 weight vectors of size 64, 8 vectors of size 128, and 4 vectors of size 256 on the same chip. Of the many weight vectors stored on chip eight ($\boldsymbol{w}_1 \ldots \boldsymbol{w}_8$) are selected in each calculation cycle and multiplied with the output of the barrel shifter ($\boldsymbol{x}$).

3 Optical Character Recognition (OCR) Network

The general structure of the OCR digit recognizer is that of a five-layer feed-forward network (see Fig 2). Since there is no feed-back and thus no relaxation, the network can be evaluated fast in a single pass. The net has 400 inputs corresponding directly to the 20×20 pixel image, i.e., no preprocessing like feature extraction is done. The ten outputs of the network code the ten digits in a '1 out of 10' code. Since the outputs of the neurons are real valued, the output does not only contain information about the classification result (the most active output) but also about the *certainty* of this classification. Actually, the difference between the most active and second to most active neuron is an accurate measure of certainty and can be used to reject ambiguous digits. An example showing all the states of the network for the case of a handwritten 6 is shown in Fig. 3. The states of the input and the first four layers are represented as grey levels; the states of the output layer are proportional to the size of the black (negative) and the white (positive) squares.

Of the five layer network, only the last layer is fully connected with all weights being independent. The first four layers are carefully constrained to improve the capability of the network to generalize well for patterns the network has not been trained on [3]. These constraints are symbolized by the local receptive fields shown in Fig. 2 and are discussed in more details in the following.

Each neuron in the **first layer** has 25 inputs and connects to a 5×5 environment in the pixel image (see Fig. 4). This environment is called the receptive field of the neuron. Two adjacent neurons, belonging to the same feature map, have local receptive fields that are displaced by *one* pixel in the corresponding direction. The neurons are grouped into four feature maps, each organized as described above. All neurons within one feature map have identical weights (weight sharing) and therefore the whole layer is determined by just 4×25 parameters (plus four bias quantity).

Figure 4: Architecture of the first layer of the OCR network.

Figure 5: Architecture of the second layer of the OCR network.

Another way to look at the operation of the first layer is to view it as a two-dimensional, nonlinear convolution of the pixel image with four kernels.

The **second layer** reduces the spatial resolution of the four feature maps generated by the first layer, resulting in another set of four feature maps of 1/4th the size. The purpose of this layer is to provide some degree of translational and rotational invariance. This operation is implemented by neurons with four synapses; each neuron averaging four inputs. Again, the architecture of weight sharing and local receptive fields is used, but now the local receptive fields of adjacent neurons do *not* overlap, they are displaced by *two* input units (see Fig 5)

The **third layer** is similar to the first one. It performs feature extraction using a 5×5 receptive field. The new aspect of this layer is that input from up to two feature maps are *combined*. A typical neuron of layer three has 50 inputs, 25 of which connect to one feature map, the remaining 25 connect to the same spatial area in another feature map (see Fig 6).

The **fourth layer** performs the same averaging and subsampling function as explained for the second layer.

The **fifth** and **last layer** has 300 inputs and 10 outputs and is fully connected, i.e. it contains 3,000 independent connections. The last layer classifies the patterns by using 10 hyperplanes in a 300 dimensional feature space generated by the first four layers of the network.

4 Implementation of the OCR Network on the ANNA Chip

To demonstrate the practical usefulness of the chip for real-world applications the first four layers of the OCR network have been ported onto the ANNA chip.

The computational precision required for each layer in a multi-layer feed-forward network typically *increases* from the input layer to the output layer. At the same time the number

Figure 6: Architecture of the third layer of the OCR network.

Figure 7: A one-dimensional nonlinear convolution suitable for the ANNA chip.

Figure 8: Implementation of the first layer on the ANNA chip.

of connections and thus the computational load *decreases* from the input to the output. The optimal hardware implementation of the OCR network is therefore to put the first n_c layers on the ANNA chip and the remaining $5 - n_c$ layers on a floating point processor like the DSP 32C. Implementations with both $n_c = 3$ and $n_c = 4$ have been realized and studied. In the first case 96.9 % of the network is evaluated by the ANNA chip, in the second case this figure increases to 97.8 %. In the following the implementation of the first four layers on the ANNA chip will be discussed in detail.

The ANNA chip can directly perform a one-dimensional nonlinear convolution. The corresponding network for this operation is illustrated in Fig 7. All neurons have the same weight vector (indicated by corresponding line types) and are displaced by one input unit. The straight-forward implementation on the ANNA chip is to realize only one neuron on the chip and to time multiplex it between the various locations. As can be seen from Fig. 7, the barrel shifter set to shift-count = 1 performs the correct multiplexing for this network. The evaluation of such a network with n neurons takes n calculation cycles on the ANNA chip, however if the network contains more than one neurons with identical receptive fields then up to eight neurons can be evaluated in parallel.

As explained previously, the **first layer** of the OCR network does a two-dimensional nonlinear convolution. How can this 2D-convolution be mapped to the 1D-convolution suitable for the chip? Figure 8 illustrates the principle: (i) The two-dimensional image data is reformatted into a one-dimensional stream as illustrated by the figure. This reformatting transforms the two-dimensional receptive fields into one-dimensional receptive fields. (ii) Although this reformatting transforms all two-dimensional receptive fields into one-dimensional ones, not all of the one-dimensional receptive fields are used (e.g. the field 5-2-6-3). These fields can easily be skipped by using a shift count larger than one (two and four in the example) or by using multiple shift instructions. In the first case, execution speed is the same as in the simple one-dimensional case.

The **second layer** and **fourth layer** are mapped to a 1D-convolution as illustrated in Fig 9. In contrast to the first layer subsampling causes more of the 1D receptive fields to be skipped. This is taken care of by increasing the shift count or using multiple shift instructions as already mentioned for the first layer.

Figure 9: Implementation of the second layer on the ANNA chip.

Figure 10: Implementation of the third layer on the ANNA chip.

The **third layer** is different from the first one in that it extracts features from two feature maps simultaneously. Figure 10 shows how this type of feature extraction can be mapped to an ordinary 2D-convolution with subsampling factor two. By interleaving the columns of the feature maps as well of the columns of the kernel data, the two disjoint local receptive fields become one contiguous field. The implementation of this 2D convolution on the chip is straight-forward.

The ANNA chip's programmable weight range is $-0.5 \ldots 0.5$; the OCR network, however, contains weights up to 1.0 in layers one and three. These large weight are realized on the chip by using two connections fed by the same input value. Neurons with many synapses typically have small weights which means that larger networks will make this trick unnecessary.

In order to facilitate the implementation of neural networks on the ANNA chip, a LISP program has been developed (called NECTAR) which generates assembly code for the ANNA chip. Convolutional networks with subsampling and extraction from multiple feature maps are covered. Furthermore, configuration of the chip for the appropriate neuron sizes (64, 128, and 256), quantization of the real valued weights (and biases) to 6 bit and selection of the optimal scale factor, as well as realization of large (< 0.5) weight values are supported by NECTAR. The output of NECTAR (symbolic assembly code) is then passed through ANNANAS (ANNA Assembler) which generates microcode that can be run on the ANNA chip.

The percentage of on-chip weight-memory used for layer 1 to 4 is 69 %. The first layer occupies 4 vectors of length 64; the second layer also uses 4 vectors of length 64; the third layer requires 12 vectors of length 128 because each neuron has $2 \times (25 + 25)$ synapses (the factor two is because of weights larger than 0.5); the fourth layer uses 12 vectors of length 64. Note, that many weight values are programmed to zero, for instance in the second layer only four out of the 64 weight values are nonzero. This means that much larger networks can be implemented on a single ANNA chip, if the number of synapses per neuron is larger.

5 Performance

The **execution time** of the network on the chip is an important parameter. High speed is necessary when either the patterns appear at a high rate or if each pattern is classified several times using different scales, translations, and rotations in order to recognize it independent of distortions.

In the following table we compare the speed of the ANNA chip with the corresponding figures of a DSP 32C (40 MHz), and a SUN SPARC:

Layer	ANNA Chip	DSP 32C	SUN SPARC
1	330 μs	29 ms	0.29 s
2	207 μs	1 ms	0.01 s
3	316 μs	18 ms	0.18 s
4	98 μs	0.5 ms	0.005 s
5	–	1 ms	0.01 s
Total	951 μs	$\approx$ 50 ms	$\approx$ 0.5 s

The execution time for the ANNA chip is specified for a continuous 20 MHz clock rate and for the micro-code generated by the NECTAR/ANNANAS system.[1] The total is calculated assuming that the four layers are executed sequentially on the same chip. If a separate chip is assigned to each layer and the chips operate as a pipeline the total would come down to 330 μs.

The average speed in connections per second of the network on the chip is 140 MC/s. The chip, however, is capable of performing 5,000 MC/s. The reason for this low utilization is that only a small fraction of the chip's parallelism is used by this particular network. The chip can multiply vectors of size 256 in one calculation cycle, but layers 2 and 4 of the network require only the multiplication of vectors of size four. Larger networks (larger weight vectors) will execute more efficiently on the ANNA chip than the current one.

These figures show that a classification rate of 1,000 characters/second is possible with the ANNA chip. This rate corresponds to a speed up factor 50 over the DSP and a factor of 500 over the SUN implementation.

The **error rate** measures how many misclassification the network does per 100 patterns. The network is tested on a test-set it has never seen before. Another even more important performance parameter is the **reject rate** which specifies the number of patterns that have to be rejected in order to achieve a desired error rate, typically 1 %. This figure is important since errors are usually related to a cost which must be kept under a certain limit. The rejected patterns can for instance be classified manually.

The network running on the SUN using floating point arithmetic achieves an error rate of 4.93 % and a reject rate of 9.12 % (for an error rate of 1 %). The same performance figures

[1] The HP16500 tester system currently used for running the code on the chip can clock the chip at 20 MHz but *not continuously*. Every few thousand clock cycles the clock is stopped and new data is transferred from the SUN to the tester system As a result the actual execution time on the tester system are much longer than the effective one

have been measured for three ANNA chips and for two implementations one with 3 layers ($n_c = 3$) and one with 4 layers on the chip ($n_c = 4$):

	Chip #1	Chip #2	Chip #3
Error Rate ($n_c = 3$)	5.83 %	5.73 %	5.63 %
Reject Rate ($n_c = 3$)	16.09 %	15.89 %	17.79 %
Error Rate ($n_c = 4$)	6.83 %	6.58 %	6.68 %
Reject Rate ($n_c = 4$)	18.78 %	19.83 %	20.93 %

The degradation with respect to error rate is small and in the case of $n_c = 3$ less than 1 %. The reject rate is affected more seriously by the chips low resolution arithmetic. This can be explained by the fact that the error rate depends only on the most active output of the network while the reject rate depends on the precise value of the most and second to most active output which requires more precision to calculate.

The observed degradation is due to the following chip nonidealities:

- Quantization of the states to 3 bits.
- Quantization of the weights to 6 bits.
- Imprecision in the analog computation.
- Approximation of the tanh-nonlinearity by a piece-wise linear function.

A simple and effective way to improve the performance of the network is to *re-train the last layer* with back-propagation. During re-training, the chip is used for the forward pass, and the backward pass affects only the last layer which is off-chip; the weights on the chip are frozen. This method is very effective since back-propagation is used only for one layer and thus degenerates to the simple perceptron learning rule. Back-propagating further into the networks is difficult because of the weight quantization (large steps in weight space) and state quantization (derivatives are zero or infinite). The results obtained by re-training just the last layer are encouraging:

	Chip #1	Chip #2	Chip #3
Error Rate ($n_c = 3$)	5.58 %	5.03 %	5.23 %
Reject Rate ($n_c = 3$)	11.06 %	10.91 %	11.41 %
Error Rate ($n_c = 4$)	5.58 %	5.23 %	5.63 %
Reject Rate ($n_c = 4$)	13.35 %	13.20 %	12.51 %

Figure 11: How re-training of the last layer improves the performance.

After re-training, not only the error-rate but also the reject rate comes very close to the original performance. It is very important for practical purposes that re-training has *not* to be done for each individual chip. The above figures were obtained by using chip #1 as a distortion model to re-train the last layer. After that, chip #1 has been replaced by other chips without degradation of performance. This is due to the good chip to chip matching.

Figure 11 illustrates why re-training of the last layer yields an improvement. The example assumes a two-dimensional feature space spanned by the outputs of the fourth layer and just two classes (o and x). Figure 11a shows the situation after training the whole network with back-propagation and floating point precision. Figure 11b shows the situation after porting the first four layer onto the chip by just quantizing the weights and the states. Each pattern in the feature space is shifted a little bit but the decision line stays in the same place since it is given by the last layer which retains full precision. Figure 11c shows how changing the location of the decision line (re-training of last layer) improves the classification performance.

6 Conclusions

It has been demonstrated that a large and practical neural-network application can be implemented on a single ANNA chip. A speed advantage of 50 to 500 over conventional hardware is gained, despite the fact that the network has not been specifically tailored to take advantage of the chips resources.

The recognition accuracy achieved with the chip's 6 bit/ 3 bit arithmetic compares well to the accuracy of the network evaluated with floating-point precision. The high accuracy has been achieved by re-training the last layer to adapt to the chips low resolution arithmetic. Chip to chip matching is good enough such that one chip can be replaced with another without decrease in performance.

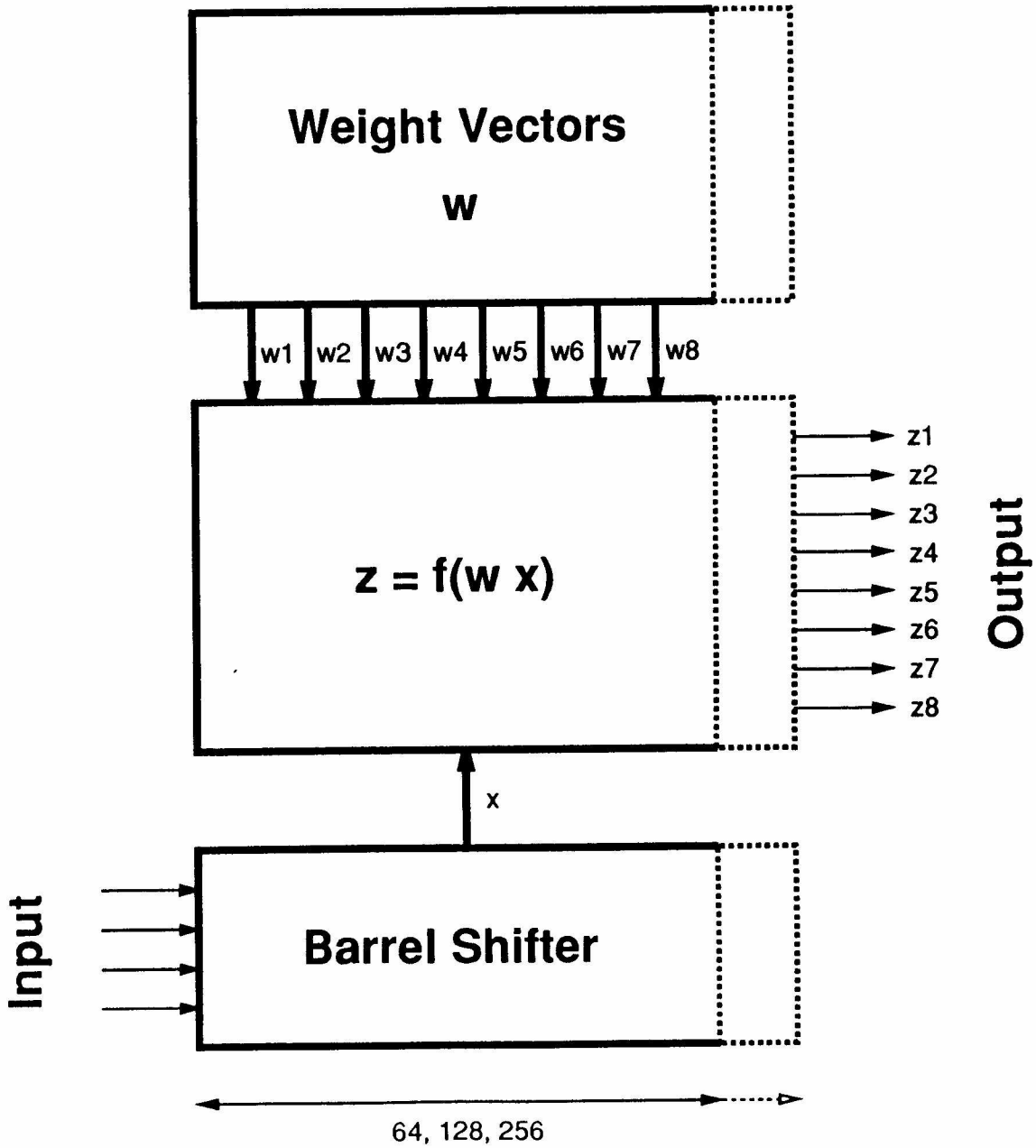
The ANNA chip is well suited for networks larger than the described one. Networks with more synapses per neuron will take even better advantage of the chip: (i) The chip's parallelism is utilized better and more connections per second are evaluated. (ii) The impact of the state quantization will be less because each neuron takes into account, and thus averages the quantization errors of more state values.

References

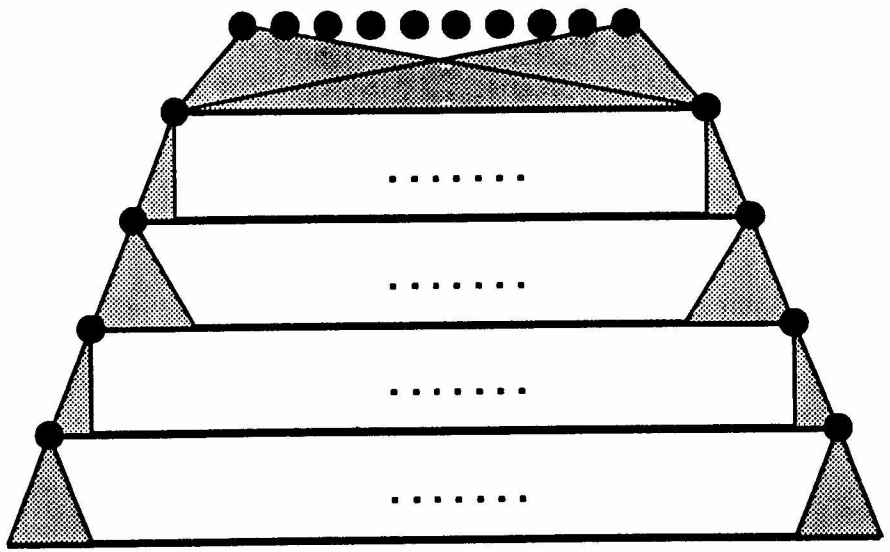
[1] Bernhard Boser and Eduard Säckinger. An analog chip for time delay and feature extraction neural networks. Technical Report TM 11315-900613-17, AT&T Bell Laboratories, 1990.

[2] Bernhard Boser and Eduard Säckinger. An analog neural network processor with programmable network topology. In *ISSCC Dig. Tech. Papers*, pages 184–185. IEEE Int. Solid-State Circuits Conference, 1991.

[3] Yann Le Cun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Larry D. Jackel. Handwritten digit recognition with a back-propagation network. In David S. Touretzky, editor, *Neural Information Processing Systems*, volume 2, pages 396–404. Morgan Kaufmann Publishers, San Mateo, CA, 1990.


Weight Vectors
w
w1
w2
w3
w4
w5
w6
w7
w8
z = f(w x)
z1
z2
z3
z4
z5
z6
z7
z8
Output
x
Input
Barrel Shifter
64, 128, 256

T.J. 1



Layer Number (Neurons/Synapses):

5. Layer (10/3,000)

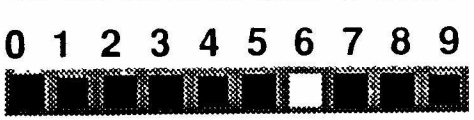
4. Layer (300/1,200)

3. Layer (1,200/50,000)

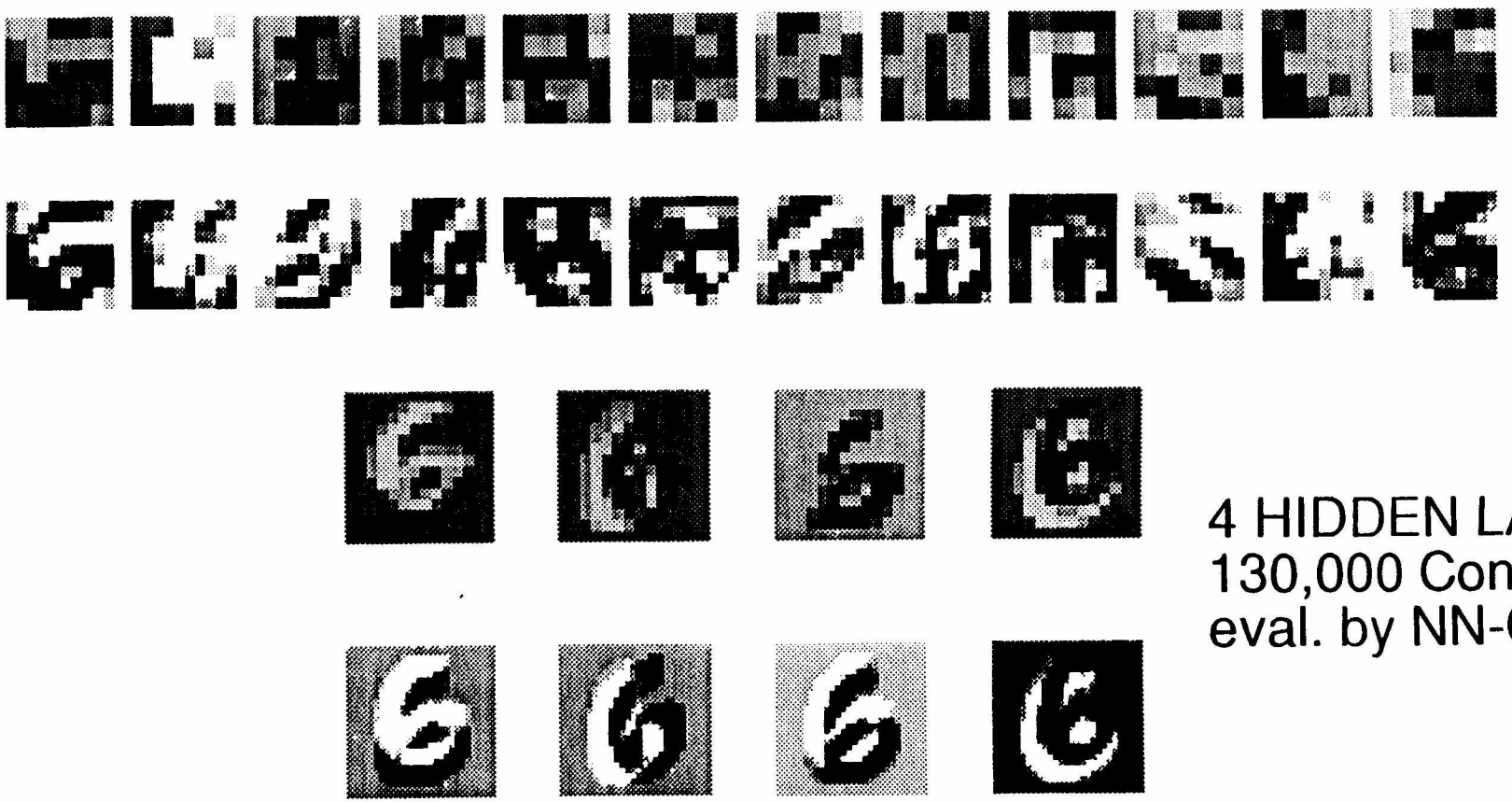
2. Layer (784/3,136)

1. Layer (3,136/78,400)

● Neuron ▒ Receptive Field of Neuron



OUTPUT LAYER
3,000 Conn. eval. by DSP

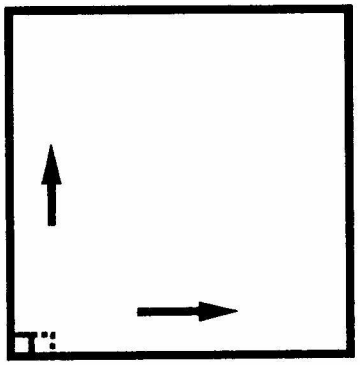


4 HIDDEN LAYERS
130,000 Connections
eval. by NN-Chip

INPUT (20x20 Pixels)

Fig-3

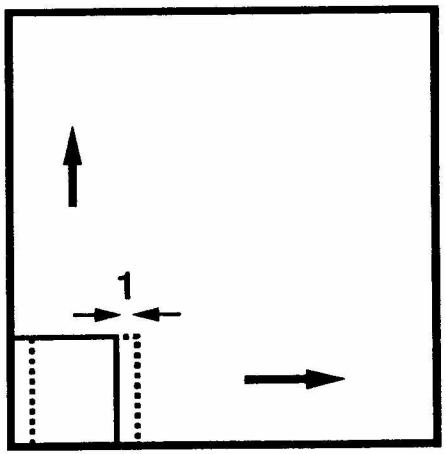
Connectivity of 1. Layer (Feature Extraction):



Feature Map (4 in total)

Neuron #1

Neuron #2

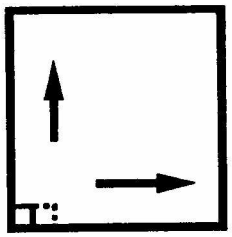


Pixel Image

Receptive Field of Neuron #1

Receptive Field of Neuron #2

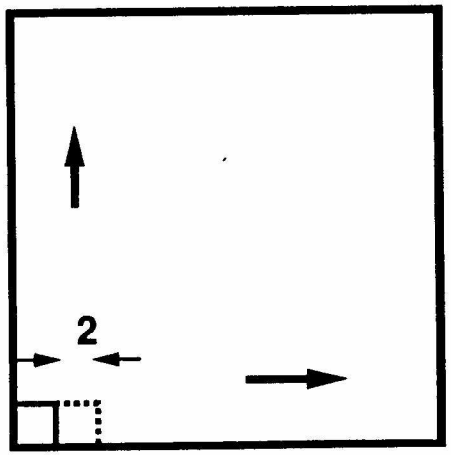
Connectivity of 2. and 4. Layer (Averaging/Subsampling):



Feature Map (4 in total)

Neuron #1

Neuron #2



Feature Map (4 in total)

Receptive Field of Neuron #1

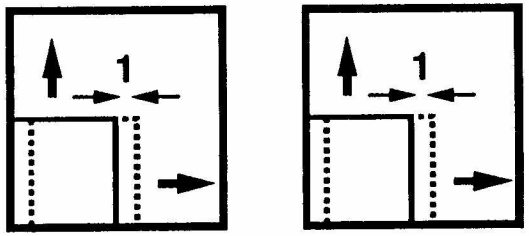
Receptive Field of Neuron #2

Connectivity of 3. Layer (Feature Combination):

Feature Map (12 in total)

Neuron #1

Neuron #2



Feature Map (4 in total)

Receptive Field of Neuron #1

Receptive Field of Neuron #2

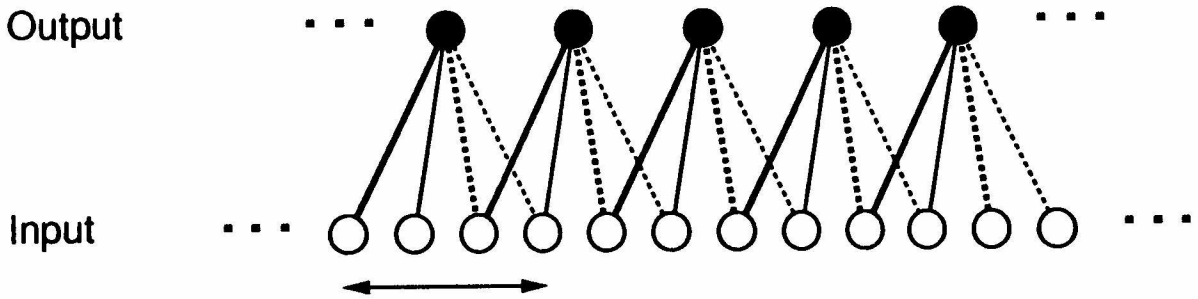


Fig. 7

Implementation of 1. Layer on Chip:

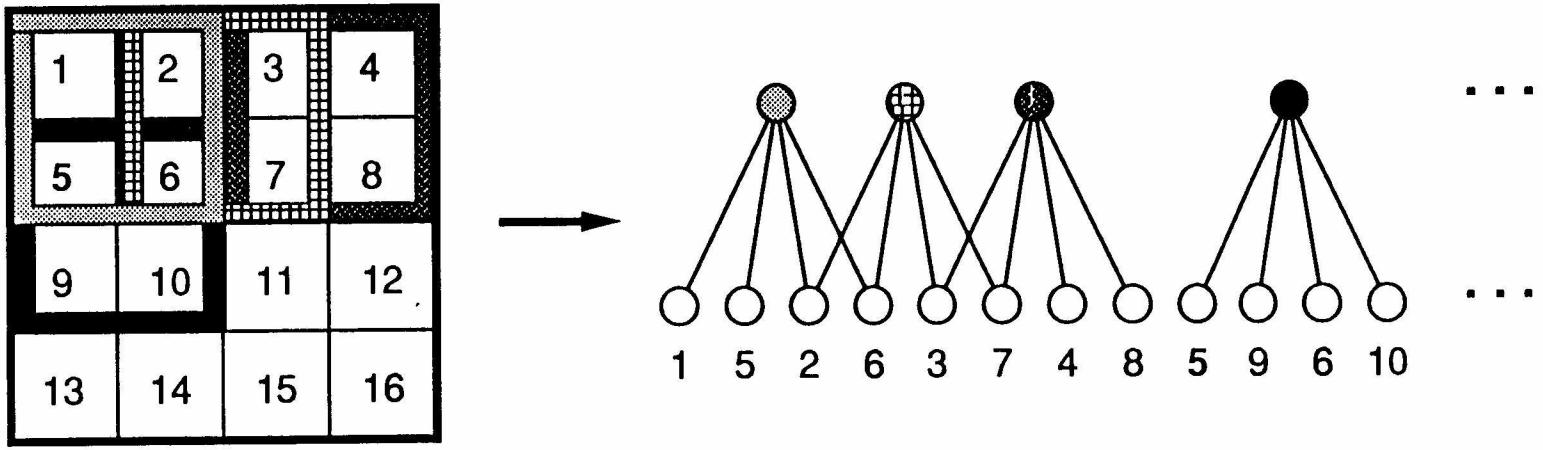
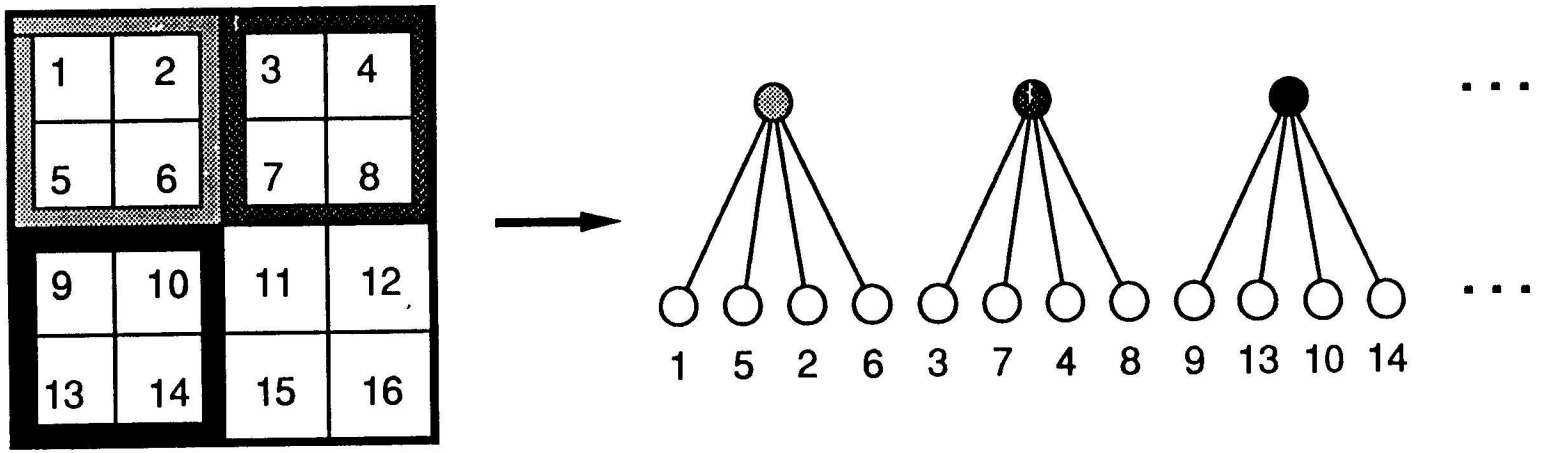


Fig 8

Implementation of 2. Layer on Chip:



Implementation of 3. Layer on Chip:

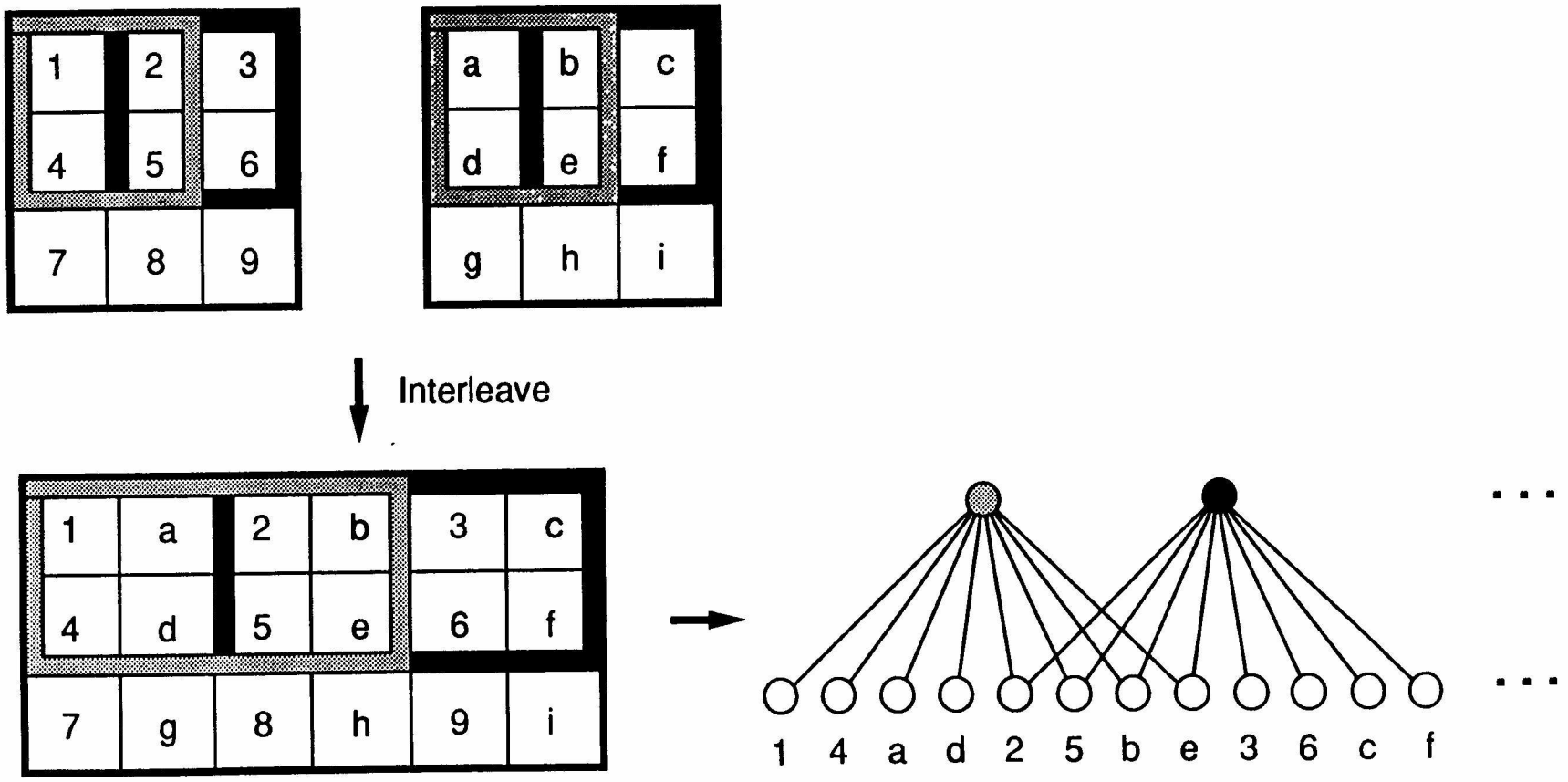
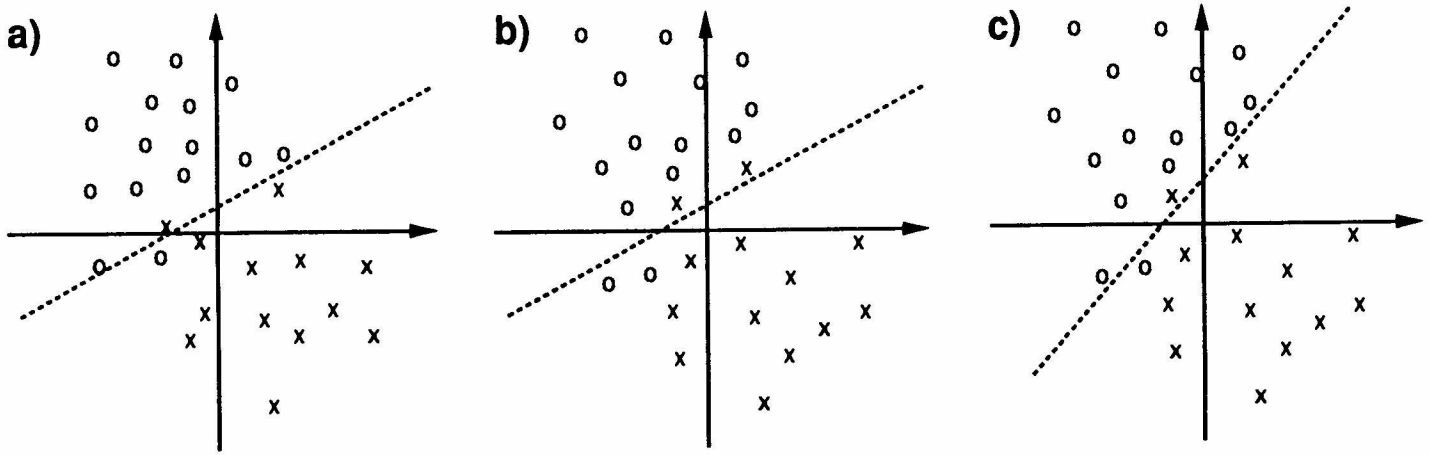


Fig. 10



States for last layer evaluated with floating point precision

------- Decision line for last layer

States for last layer evaluated with 6bit/3bit (chip-) precision

------- Original decision line for last layer

States for last layer evaluated with 6bit/3bit (chip-) precision

------- Decision line after retraining of last layer

Fig. 11