Application of a Neural Network Demonstrator to optimize the positioning of Kings, Knights or Queens in a 8x8 grid

Ketil Karstad
Heriot-Watt University 2003

Supervisor: Mohamad R Taghizadeh

Abstract

We reveal how one can use an optical neural network to optimize the positioning of Kings, Queens or Knights in such a way that no one are able to take any one in a single move. This project also demonstrates how easy it is to change the interconnects in an optical neural network and prepare it for new tasks. It was expected that this kind of task would not be a problem to solve for this kind of neural network and we demonstrated that this is indeed true.

The performance of the optical neural network depends on a set of neural network parameters, labeled A, b and β, which depend on the neural network itself and the DOE used to define it's interconnects. We found that the ratio A/b is a constant, whose value depends on the DOE, and most values of A and b satisfying that, produce optimal results. It was also established that neuron network performance is independent of β.

Safety

Safety is of chief concern when working in a lab where irresponsible or incorrect use of the equipment available may cause immediate and permanent damage to the operator or to any one that might be in lab. I did not work with any equipment that would be classified as such. The only lasers used were a very low power VCSEL array operating in the near infrared, enclosed in a transparent Perspex enclosure. I did however share lab space with some one using a class 3B HeNe laser and to prevent any scatter or reflections from getting to the area where I was working, I put up a shield of black plastic that divided the lab space in such a way that I was safe from scatter or reflections from the laser.

Introduction

In order to best understand this report one should read the report by Yves Randal and Thesis by Keith J Symington.

This introduction is divided into two sections. First we introduce the reader to the problem we have been solving during this project in order to provide the reader with the best possible start so the rest of the report is easier understood. Thereafter the project and equipment is explained in greater detail.

Imagine the following situation. You are asked to place as many Kings, Queens or Knights as possible on a chess board in such a way that no one are able to take each other in one move. This is a classic optimization problem, and one way of solving this is to sit down with a pencil and paper and try different configurations until you believe you have reached a result that is the best you can do. In figures 1, 2 and 3 we illustrate examples of how you can put the maximum number of Kings, Queens or Knights on a chess board.

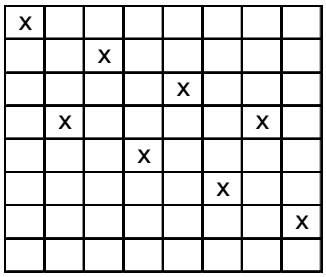

Figure 1: *8 Queens in an 8x8 grid, no one can take each other in one move.*

Figure 1 shows one configuration with 8 Queens. This is the maximum number of Queens that can go in a 8x8 grid without anyone being within the range of each other, but note that the configuration shown here is not the only one that is valid. This is also true for figure 2.

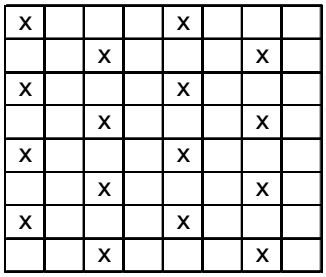

Figure 2: *16 Kings in an 8x8 grid, no one can take each other in one move.*

Figure 2 shows one configuration with 16 Kings. This is the maximum number of Kings that can go in a 8x8 grid without anyone being within the range of each other.

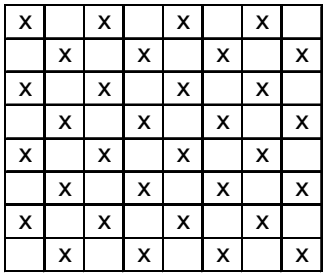

Figure 3: *24 Knights in an 8x8 grid, no one can take each other in one move.*

Figure 3, one configuration with 32 Knights. This is the maximum number of Knights that can go in a 8x8 grid without anyone being within the range of each other and this can be achieved in only two ways.

We have now covered the basic principle of what this project is about and will go on to explain the equipment that was used and how we used it to solve this kind of optimization problem.

In the pursuit of greater bandwidth and reliability for long and medium distance communications, the telecoms industry has for a long time been using optical interconnects between exchanges around the world in stead of electronic. Photons carry no mass or charge and will therefore not interact with each other and are immune to magnetic interference from any man made or naturally occurring phenomena, a great advantage over electrons. Up until now it has been cheaper bit for bit, to use electronic interconnects for short range communication, e.g. for chip to chip and board to board communication, but as bandwidth is increased and research in optical computing is breaking through old barriers, optical interconnects are becoming more viable for use in short distance communications. This project is based on a PhD project held by Keith J Symington, who created an optical neural network that demonstrated the use of optical interconnects for packet switching.

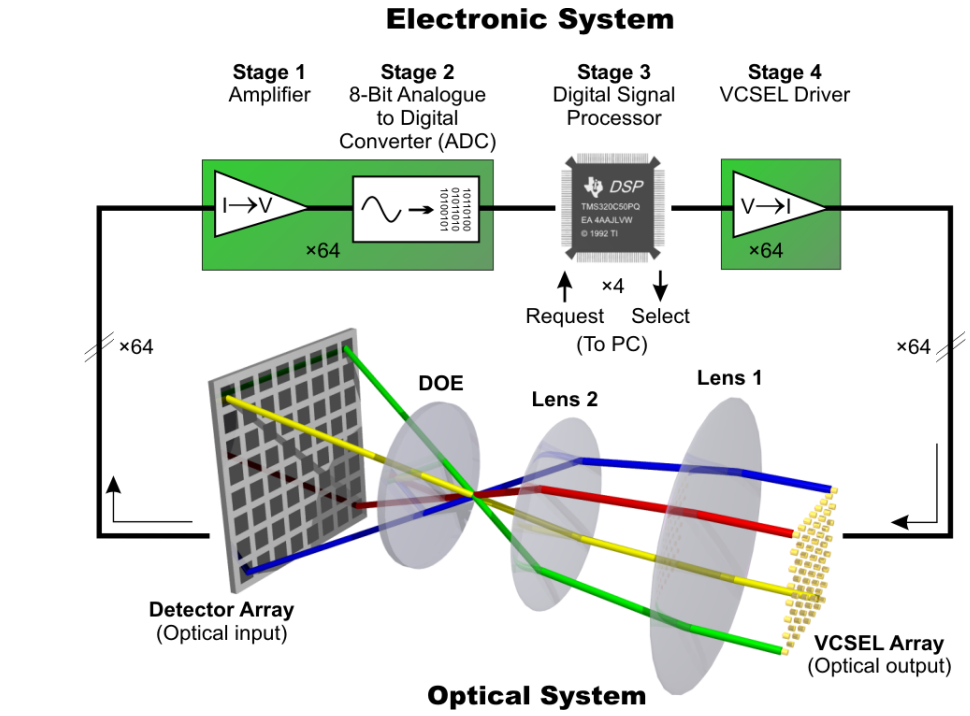

Figure 4.1: *Outline of Optical Neural Network as used in the project.*

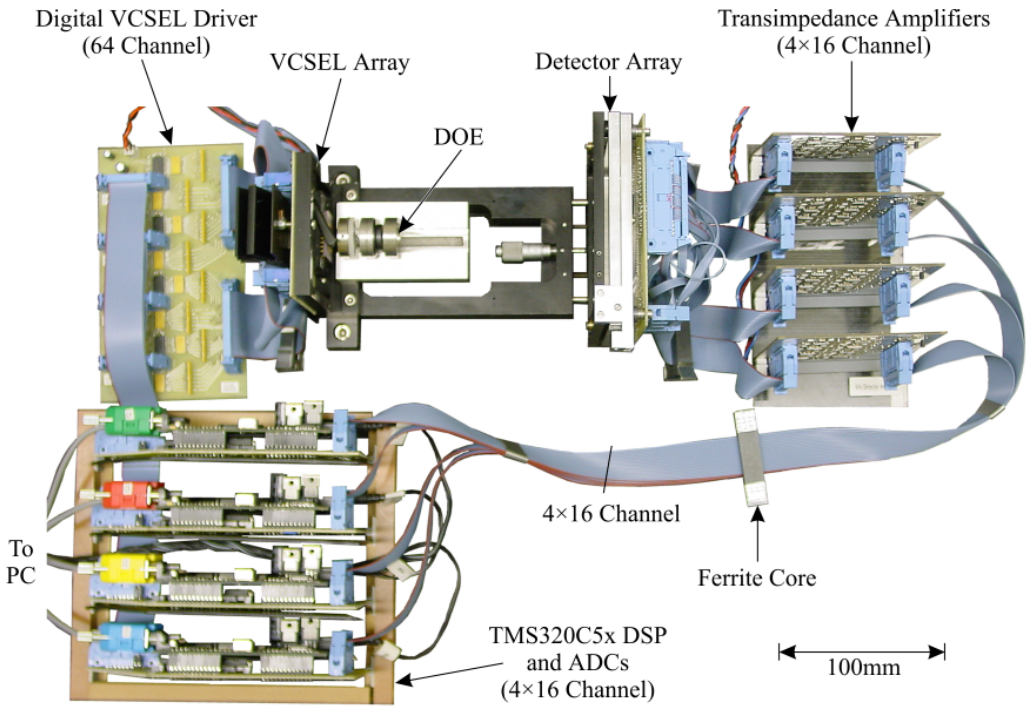

Figure 4.2: *Aeriel view of Optical Neural Network Demonstrator.*

Figure 4.1 is a diagram illustrating the basic building blocks of the optical neural network and 4.2 is an aerial view of the experimental setup.

All together, this is basically a single layer array of 8x8 neurons where the input of the neurons is a 8x8 Silicon detector array and the out put is a 8x8 VCSEL (vertical cavity surface emitting laser) array. The state of each neuron is determined by the value of its sigmoid function and a predetermined threshold value.

At the beginning of a computation a request value is downloaded from a PC to 4 DSPs via the PCs RS232 ports at 115k Baud. Each DSP is remotely programmable and handles 16 out of 64 individual neurons. The VCSEL array is controlled by the DSPs via a digital driver module that controls the current supply to the VCSEL array. The neuron interconnects are in the optical domain and a DOE sets the interconnect pattern depending on the application. A photodiode detector array produces a photocurrent that is converted to a voltage and amplified by 64 dual element transimpedance amplifiers across 4 modules. At the receiving end of each DSP is 2 octal ADCs that convert the analogue signal from the amplifiers.

The DSPs have memory, meaning thousands of problems can be downloaded by the user and solved without intervention. Synchronization of the DSPs is taken care of by an optical signal that indicates the readiness of each DSP. A computation is only started when all DSPs are ready. The VCSEL array used here was produced by CSEM Zurich.

It is an oxide confined device and it operates at 960nm. The Photodiode detector array used is a Centronic MD100-5T, and consists of 10x10 individually addressable photodiodes. The DSPs are 40MHz TMS320C5x's from Texas Instruments.

When running a neuron optimization the sigmoid function for all the neurons is set to be equal to the threshold value and the user sends a request to the neural network that tells it what neurons that is requested to be on at the end of the optimization.

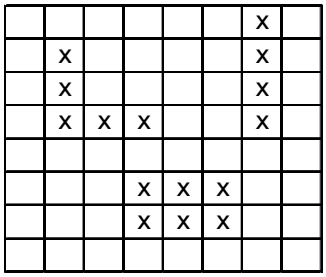

Figure 5: *A neural network request.*

Figure 5 shows an example of a possible request sent to the neural network for optimization. At the start of the optimization these VCSEL's will be switched on.

The neuron optimization consists of a number of neuron calculations or iterations, defined by the user. For each iteration, the neural network takes the input of each neuron, multiplies it with $-A$ and adds b. If we call this temporary number x, it can be written as

$$x = -A \cdot input + b \tag{1.1}$$

Where A and b are neural network parameters whose value depend on the application of the neural network. A neurons sigmoid function is a function of an other temporary

variable that we call *mem*. For the first iteration, the value of *mem* is *x,* but in the following iterations *mem* is given the value

$$mem = previous\ value\ of\ mem + x \tag{1.2}$$

After completing the given number of iterations the sigmoid function is calculated as

$$f(mem) = \frac{1}{1 + e^{-\beta \cdot mem}} \tag{1.3}$$

where β is an other neural network parameter. If *f(mem)*>*threshold,* then the respective neuron is on.

This covers the basics of how the neural network carries out an optimization, but we need to confer how we impose the rules of the optimization.

The optimization rules are imposed on the network by a diffractive optical element (DOE). The DOE used to optimize the King for example, takes the input from one VCSEL and diffracts it into a set of allowed orders that correspond to all possible moves a king can make from the position of the VCSEL. This is illustrated in figure 6. We refer to orders other than the allow orders as suppressed orders.

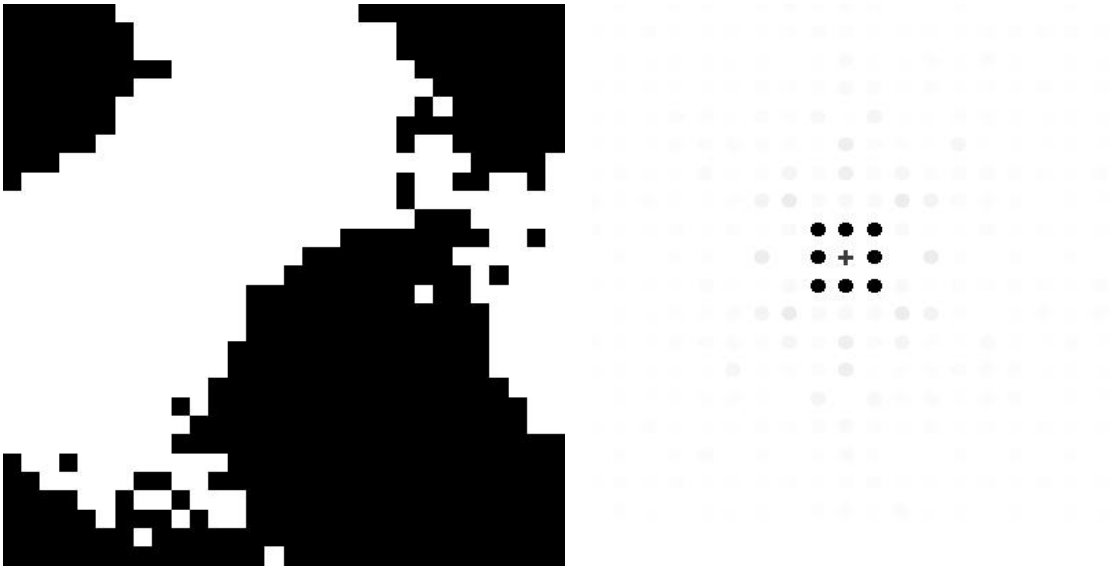

Figure 6: *King DOE mask on left and DOE out put on the right.*

Equations 1.1, 1.2 and 1.3 tell us that an input to a neuron has the effect of switching it off, so in the case for the King, when a neuron is requested to be on, it has the effect of turning off all neurons that represent the position of a King that could be taken in one move from the position that is requested on. E.g. if we request to Kings to be beside each other, the neural network will suppress one of them, but which one of them, is not know at the outset of the optimization.

In figures 6, 7 and 8 we illustrate the DOE mask and output used to optimize the positioning of Kings, Queens and Knights.

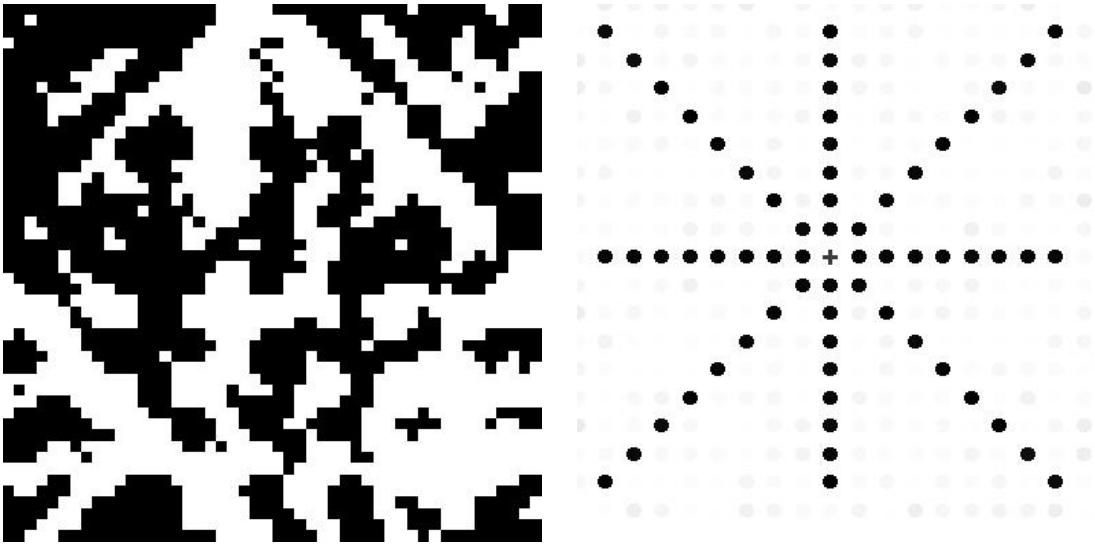

Figure 7: *Queen DOE mask on left and DOE out put on the right.*

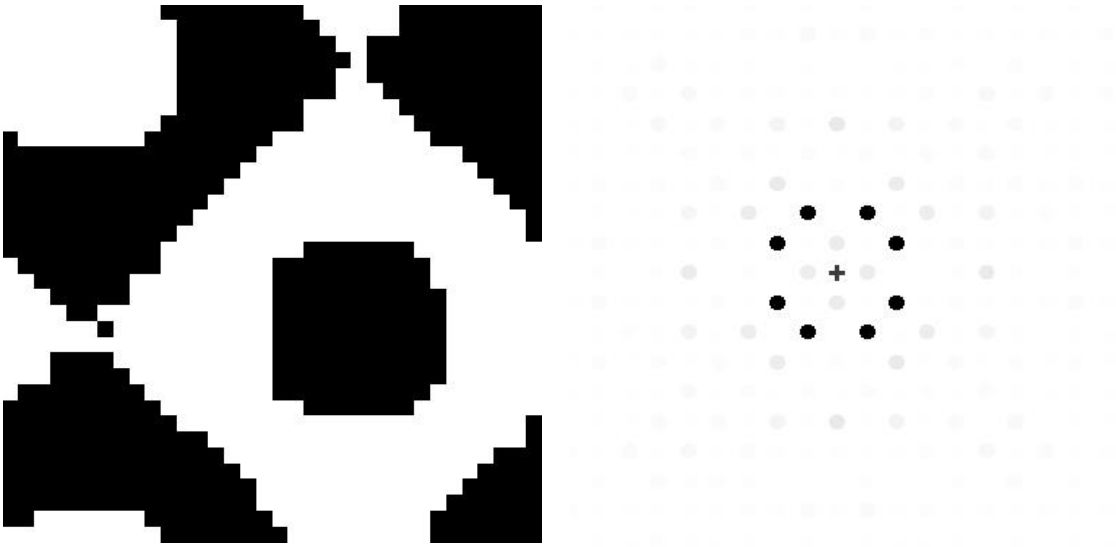

Figure 8: *Knight DOE mask on left and DOE out put on the right.*

The DOE's used in the neural network have to be space invariant, so the diffraction pattern is the same for all VCSEL's.

This can be used to solve the optimization we discussed in the start of this section and the objective of this project was indeed to verify that it would work and how well it would work. It was previously demonstrated that the neural network demonstrator worked very well for packed scheduling optimization, so it was expected that what we wanted to do would not be a problem for the neural network, since optimizing the positioning of Kings, Queens or Knights in a 8x8 grid is essentially the same action as packet scheduling. The only difference is what kind of grating that is used and some very minor software adjustments related to the verification of results.

1. Neural Network as Packet Scheduler

In the start of this project we verified previous results to assess stability and repeatability of the neural network demonstrator. In a switching network, it is important that the traffic passing through the network is optimum and that means in the case of an 8x8 switch, that optimum result is 8 neurons on for all requests, assuming the load is maximum. We call this the optimality of the switch and it is defined as:

$$Optimality = Average\ no.\ of\ neurons\ on\ /\ 8 \tag{1.1}$$

One other important quality for a switching network is the validity of the results, since invalid results can not be allowed. Using the Crossbar it was established that the best values of A and B was 1.05 and 16 respectively, and that an average of 7.67 neurons was all for all requests and validity was 99.9% giving an optimality of 95.9%. Using the Banyan it was established that the best values for A and B was 1.05 and 9 respectively, and that an average of 5.11 neurons was on for all requests giving an optimality of 63.9%.

The optimality using the Banyan was a lot lower than when using the Crossbar and it was questioned whether it was caused by a poor quality DOE. We wanted to investigate this further, so we had a new Banyan DOE manufactured and repeated the measurements.

Experimental results using the Crossbar DOE were the same as was established in previous measurements, just as one would expect, but surprisingly this was also the case when using the new Banyan DOE. This implies that the lower optimality achieved when using the Banyan is not an artifact of a poor quality DOE but indeed an inherent property of how the Banyan works as a switch.

1.1 Stable States

When we repeated previous measurements we had a closer look at the individual results and not just the statistics and discovered that there were some results that repeated them self more than one would expect. E.g. For an 8x8 switch there are 8! = 40320 possible results, so for a run with 100 requests one would expect very few repeated results. We chose to call this phenomenon where results repeat them self in excess of 10% of the total number of results for stable states.

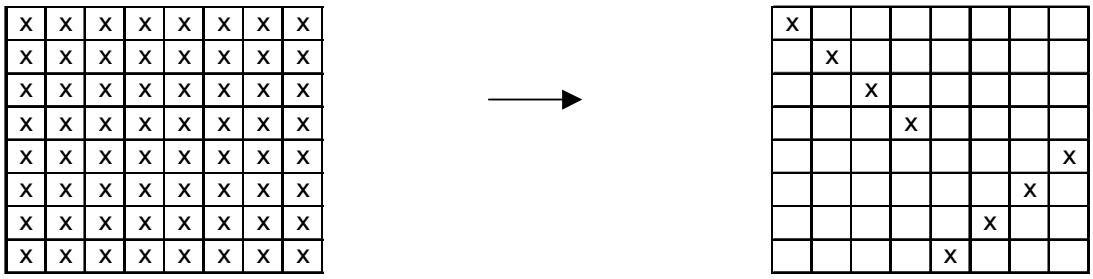

Figure 9: *Array on right shows the requested array and the array on the left illustrates one of the observed stable states.*

A possible explanation of this is that it is caused by light leaking into suppressed orders and that the light does not do so in a uniform manner, causing some result patterns to be more probable than others. Figure 9 illustrates one of the observed stable states with maximum, 8x8 load. We observed that the stable states tend to be symmetrical and have an axis of symmetry going from upper left-hand corner to the lower right-hand corner.

In a set of measurements we requested all neurons to be on in a 3x3 sub-array, whose position within the 8x8 array could be defined by the user. Requesting all neurons in a 3x3 array to be on, resulted in one stable state for each run depending on the position of the 3x3 sub-array within the 8x8 array of the neuron network. Figure 10 illustrates two stable state observed for that particular position of the 3x3 sub-array.

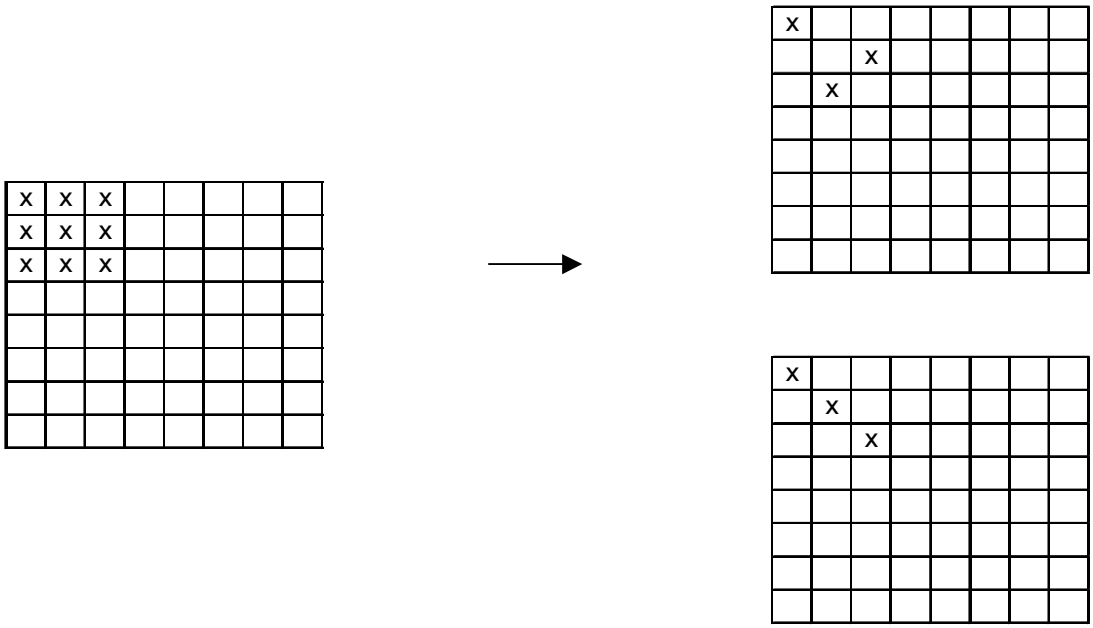

Figure 10: *The array on the left is what was requested and the arrays on the right are the resulting stable states.*

If it was the case that light diffracted into suppressed orders was causing the stable states, then one would expect the resulting stable states to depend on the orientation of the DOE. We found that rotating the DOE through 90 degrees, had no effect on the stable states, they were always the same and would only change depending on the position of the 3x3 sub-array. This indicates that the stable states are not a problem related to poor quality DOEs but rather links it to the detector part or the light source in the network.

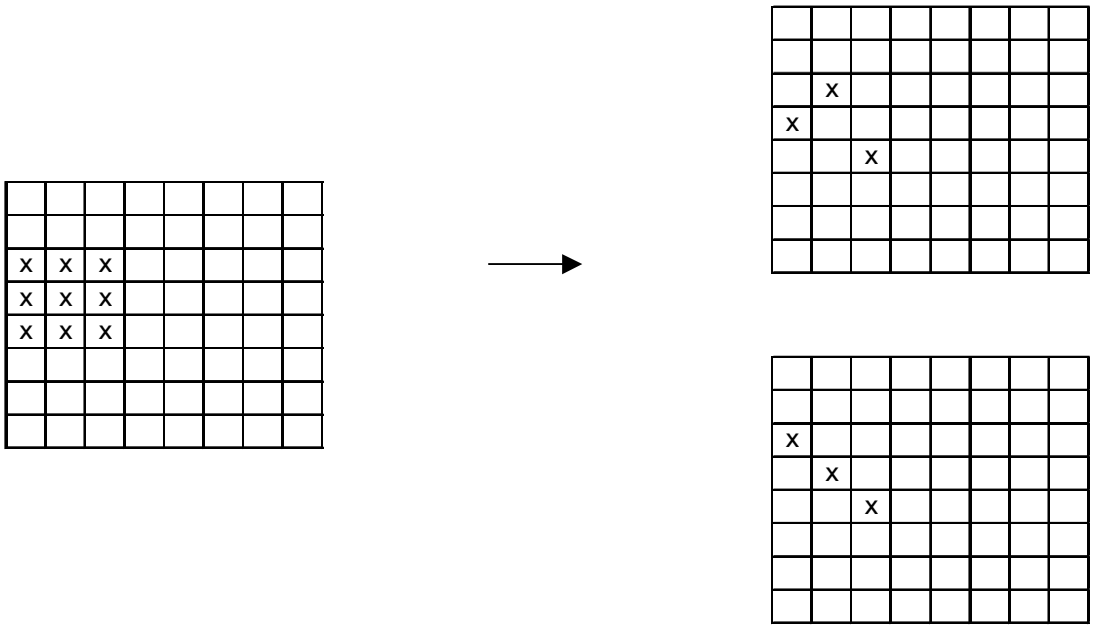

Figure 11: *The array on the left is what was requested and the arrays on the right are the resulting stable states.*

Figure 11 shows the stable states resulting from moving the request array within the 8x8 array. This tells us that whatever is causing the stable states is not space invariant.

Stable states were not observed when carrying out measurements with the Banyan or the Queen DOE, but we did observe stable states when using the Knight and King DOE. We made a new Knight DOE and exposed it for a longer time, making it a low quality DOE to see if it had any effect on the stable states. The effect of overexposing the DOE is that it will work for a different wavelength and when using it with the wrong wavelength, light will escape into suppressed orders, especially the zeroth order. This had no effect on the stable patterns. In fact it did not seem to have any effect at all, and the network performed just as well as with the DOE that was made to work for the VCSEL

wavelength, which is a rather remarkable discovery. We also tried changing the value of β, but that didn't have any effect on the results.

The best explanation of the occurrence of stable states is that they are caused by individual VCSELs with a higher than average output power and/or individual detectors with a lower than average response due manufacturing tolerances [2]. This will make a particular neuron more likely to win over others and hence create stable states. For DOEs with a larger fan-out the signal to noise ratio in the network is reduced, noise becomes more dominant and it eventually drowns out any advantage a particular neuron might have. If this is true, we would expect to observe stable states only for DOEs with a low fan-out and not for DOEs with a large fan-put. This fits with observations. The Banyan and Queen DOEs have relatively large fan-outs and we do not observe stable states when using them. The King, Knight and Crossbar have relatively small fan-outs and we observe stable states when using them.

2 Network operation with King, Queen and Knight

In this section we describe how we used the neural network to solve the optimization problem described in the introduction.

As we mentioned when a neuron is requested on, it will try and suppress all neurons that are in its range. In this application we define a neurons range to be all neurons representing the position of a valid move from the requested neuron.

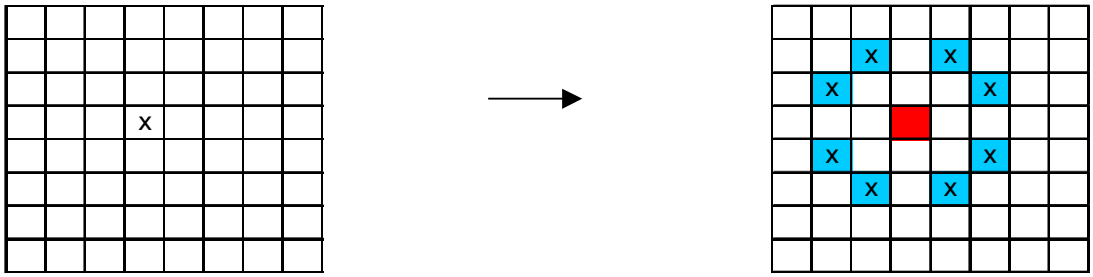

Figure 12: *A neurons range, here in the case of a Knight DOE, marked in blue and requested neuron marked in red.*

Figure 12 illustrated the range of a neuron controlled by the Knight DOE.

By requesting all 8x8 neurons to be on the neuron network will optimize the request and produce a result with as many neurons as possible on. The result is very dependent on the values of A and b which have to be determined by the user. They will not have the same values for all the different DOEs and must be determined for each DOE depending on how the neuron network should perform. For example we want the result to be valid and have as many neurons as possible to be on, but we find that the values of A and b

for which these conditions are fulfilled, depend on the neuron network load. This presents the operator with a choice, should the neuron network produce valid results with as many neurons as possible on for high loads and not care if validity drops for lower loads, or should the network sacrifice the number of neurons on for an increased validity at lower loads. Due to the nature of this application, namely to get as many Kings, Queens or Knights in a 8x8 grid as possible, we choose to sacrifice validity at lower loads in order to get optimum result and better validity at full load.

2.1 Checking validity of result

In this section we go through the basis of the code for checking the validity of the different DOEs.

The code used to check validity is based on the following principle.

1. Search result array for first/next neuron that is on.
2. Check the neuron's range and if any neurons within the range is on, set result as invalid and go to step one. If no neurons are on go to step one.
3. Loop step 1 and 2 until all 8x8 neurons have been checked.

2.1.1 Step one

Step one is the same for the King, Queen and Knight. It finds a neuron that is on and gives it a coordinate, (*ii*, *jj*), where (0, 0) is the bottom left hand corner as in a right

handed 2D coordinate system.

2.1.2 Step two: King

For this we divide the range of the King in two, a and b, so the range of the neuron is a or b. This is illustrated in figure 13.

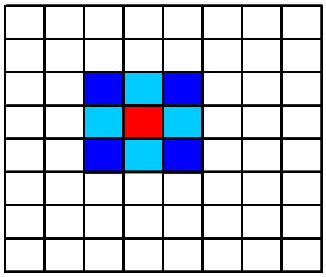

Figure 13: *A neuron is found on, marked red here, and its range is marked in two shades of blue. Dark blue is a and light blue is b.*

Remember, red represents a neuron that is on and is given the coordinate (*ii*, *jj*).

Then a is given as $i = ii +/- 1$ and $j = jj +/- 1$.

b is given as $i = ii$ and $j = jj +/- 1$ or $i = ii +/- 1$ and $j = jj$.

When a neuron is found that is on, the code goes through the 8x8 array again, every neuron that is found to be on is given the coordinate (*i*, *j*) and a and b are evaluated. If a or b is true for any neuron, then the result is invalid.

2.1.3 Step two: Queen

As with the King DOE, we divide the range of the Queen in two sections, a and b. This is illustrated in figure 14.

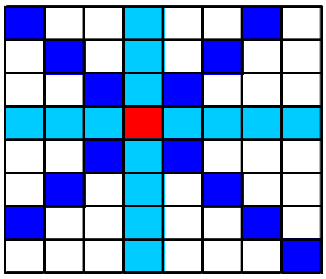

Figure 14: *The range of a Queen neuron divided in two. Dark blue is a and light blue is b.*

a is given as $j = i + jj - ii$ or $j = -i + jj + ii$.

b is given as $i = ii$ or $j = jj$ and $(i, j) != (ii, jj)$.

When a neuron is found that is on, the code goes through the 8x8 array again and every neuron that is found to be on is given the coordinate (i, j). If a or b is true for any neuron, then the result is invalid.

2.1.4 Step two: Knight

As before the range is divided in two regions. Illustrated in figure 15.

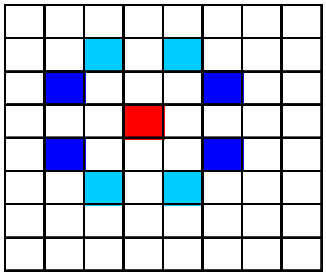

Figure 15: *The range of a Knight neuron divided in two. Dark blue is a and light blue is b.*

a is given as $i = ii +/- 2$ and $j = jj +/- 1$.

b is given as $i = ii +/- 1$ and $j = jj +/- 2$.

When a neuron is found that is on, the code goes through the 8x8 array again and every neuron that is found to be on is given the coordinate (*i, j*). If a or b is true for any neuron, then the result is invalid.

3 DOE fabrication

The DOEs used in this project were not available when we started, so we had them manufactured in the facilities available at Heriot-Watt. The DOEs are binary DOEs and are relatively easy to make.

First, photo resist is spun on the glass substrate to form a thin, even layer. Then a mask is applied and the photo resist is exposed to UV radiation through the mask. The unexposed photo resist is wash off in a developer and the DOEs are at this stage placed in an oven at 198 degrees for 30 minutes until the resist hardens enough to withstand the next process which is ion etching. When the photo resist has hardened, the DOEs are placed in an ion etcher where they are exposed for 2 hours, which is the time necessary to get the etch deep enough for the DOEs to work at 960 nm, which is the wavelength the VCSEL array operates at.

4 Experiment

In this section we describe the main experimental effort of the project. Namely using the Neural Network Demonstrator illustrated in figure 4.2 to optimize the positioning of Kings Queens or Knights in a 8x8 grid using the corresponding DOEs. First we determine the best values of the neural network parameters for each DOE before we go on to investigate how the network performs.

4.1 Determine A and b

We have already mentioned that we want to run the network at full load and get valid results with as many neurons as possible to be on. In this section we will discuss and present the results from when we determined the *A* and *b* values for the different DOEs.

Determining the optimum value of *A* and *b* takes quite a while since we have to run an optimization with 100 requests for every value of *A* and *b* that we wish to investigate.

As a reference the values of *A* and *b* for the Cross bar and Banyan DOE are given in table 1.

	A	b
Crossbar	1.05	16
Banyan	1.05	9

Table 1: *Values of A and b for Crossbar and Banyan DOE*

4.1.1 King DOE

We know from the introduction that we can have a maximum of 16 Kings in a 8x8 grid, so we are looking for values of *A* and *b* that produces valid results with 16 neurons on. The best way of doing this is doing what we call an A-b test where we look at validity and number of neurons on as a function of *A* and *b*.

A \ b	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
1	100	100	100	100	100	100	100	100
2	97	100	100	100	100	100	100	100
3	95	100	100	100	100	100	100	100
4	1	96	100	100	100	100	100	100
5	0	5	97	100	100	100	100	100
6	0	0	21	96	100	100	100	100
7	0	0	0	52	96	100	100	100
8	0	0	0	0	39	99	100	100

***Table 2**: Validity as function of A and b.*

Table 2 list the validity as a function of *A* and *b*. The results are plotted in figure 16.

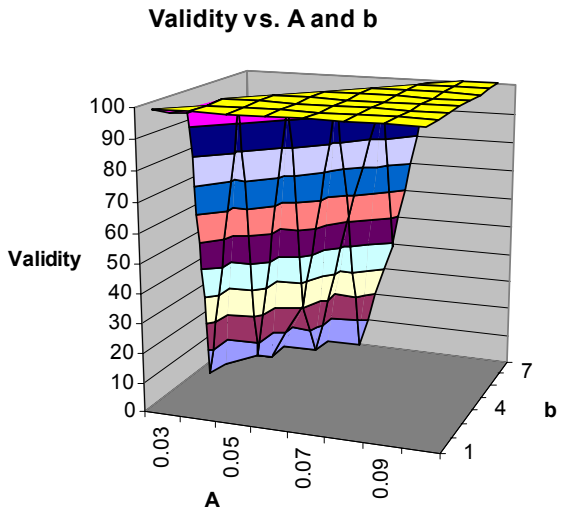

Figure 16: *Validity of King DOE as a function of A and b.*

Choosing the right range of *A* and *b* is just an exercise in trial and error.

A \ b	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
1	5.02	5	4.86	4.3	4.37	4.16	4.36	4.19
2	12.98	11.32	9.67	9.19	8.56	8.15	7.77	7.42
3	15.66	14.12	12.65	11.86	10.58	10.26	9.22	9.16
4	16	15.91	14.86	13.87	12.85	12.13	11.35	10.65
5	0	16	15.94	15.19	14.34	13.79	13.14	12.31
6	0	0	16	15.91	15.52	14.52	14.02	13.74
7	0	0	0	15.98	15.99	15.89	15.07	14.37
8	0	0	0	0	16	15.99	15.75	15.13

Table 3: *Number of neurons on as a function of A and b for King DOE.*

Table 3 shows the average number of neurons on as a function of *A* and *b*. The results are plotted in figure 17.

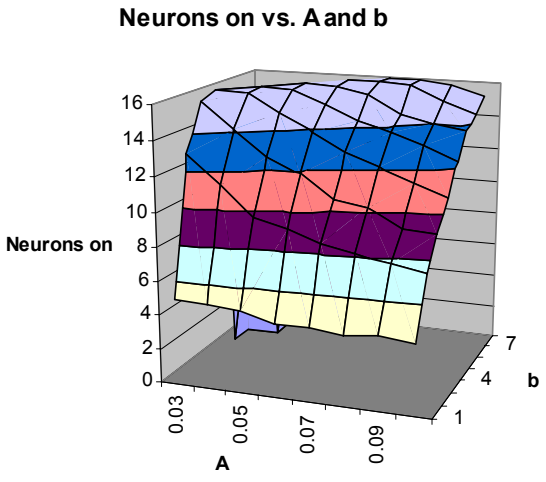

Figure 17: *Number of neurons on as a function of A and b for King DOE.*

The values of *A* and *b* that gives 100% validity and the most neurons on is *A*=0.08 and *b*=7 which results in and average of 15.89 neurons on. This is very good considering that 16 is the optimum result.

4.1.2 Queen DOE

The process of finding the optimum value of *A* and *b* is the same for all DOEs.

A \ b	0.6	0.7	0.8	0.9	1	1.1	1.2	1.3	1.4
1	100	100	100	100	100	100	100	99	100
2	100	100	100	100	100	100	100	100	100
3	100	97	98	100	100	100	100	100	100
4	94	99	99	100	100	99	100	98	100
5	84	96	96	99	100	99	99	100	100
6	63	82	95	91	99	100	100	100	98
7	58	72	75	93	96	99	99	98	100
8	40	38	70	76	77	94	96	94	96
9	21	36	46	67	76	82	93	97	96
10	2	15	43	50	71	89	87	91	97

Table 4: *Validity as a function of A and b for Queen DOE.*

Queen validity as a function of *A* and *b* is listed in table 4.

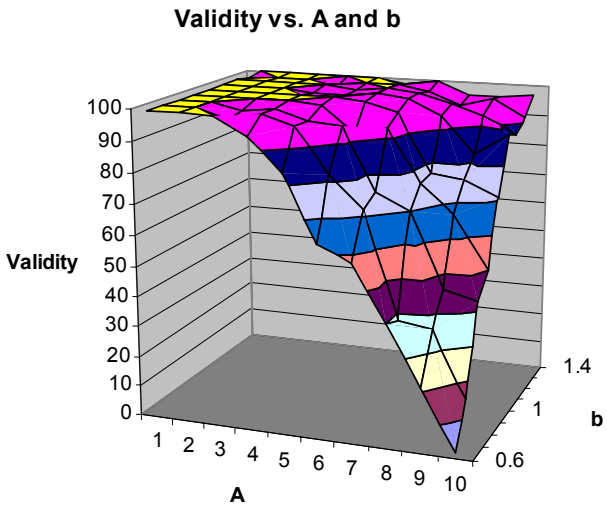

Figure 18: *Validity of Queen DOE as function of A and b.*

Figure 18 is a plot of the Queen DOE validity as a function of *A* and *b*.

A \ b	0.6	0.7	0.8	0.9	1	1.1	1.2	1.3	1.4
1	1.95	1.73	1.67	1.52	1.62	1.6	1.54	1.566	1.53
2	3.11	2.75	2.72	2.59	2.56	2.17	2.2	2.14	2.08
3	3.85	3.557	3.286	3.28	3.43	2.75	2.76	2.68	2.59
4	4.564	4.081	3.929	3.87	3.85	3.263	3.34	3.133	2.98
5	5.19	4.833	4.396	4.182	4.11	3.818	3.838	3.69	3.57
6	5.651	5.366	4.926	4.44	4.626	4.1	4.06	3.92	3.806
7	6.31	5.847	5.373	4.978	5.01	4.586	4.384	4.194	4.01
8	6.5	5.974	5.814	5.421	5.506	4.936	4.667	4.447	4.406
9	7.143	6.333	6.152	5.97	5.75	5.366	5.065	4.907	4.729
10	7	6.933	6.721	6.22	6.014	5.854	5.425	5.165	4.938

Table 5: *Number of neurons on as a function of A and B for Queen DOE.*

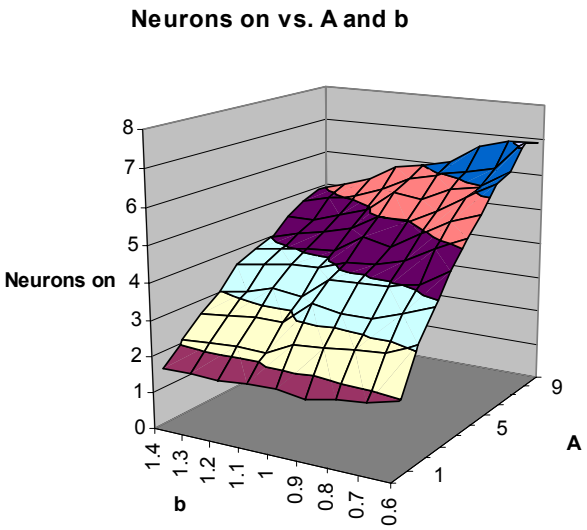

Figure 19: *Number of neurons on as a function of A and b for a Queen DOE.*

Table 5 and figure 19 show the result of determining the optimum *A* and *b* values for the Queen DOE. From this we get that for the most number of neurons on, *A*=1 and *b*=5, yielding an average of 4.11 neurons on.

We determined earlier in the introduction that the optimum number of queens in an 8x8 grid is 8 but we are getting 4.11 neurons an at best. The Queen DOE has a much larger fan-out than the other DOEs, so this might indicate that there is a limit to the DOE fan-out, and beyond this limit the resulting optimizations will not be optimum.

4.1.3 Knight DOE

The best value of *A* and *b* is *A*=0.14 *b*=16 and this produces results with an average of 23.99 neurons on.

	b										
A	0.1	0.14	0.18	0.22	0.26	0.3	0.34	0.38	0.42	0.46	0.5

12	97	99	100	96	99	99	100	100	100	100	100
13	99	100	100	98	100	99	100	100	100	100	100
14	0	100	100	100	100	99	100	100	100	100	100
15	0	99	100	100	100	100	100	100	100	100	100
16	0	100	100	99	99	100	100	100	100	100	100
17	0	97	100	100	100	100	99	100	100	100	100
18	0	0	100	100	100	100	100	100	100	100	100

Table 6: *Validity for Knight DOE as a function of A and b.*

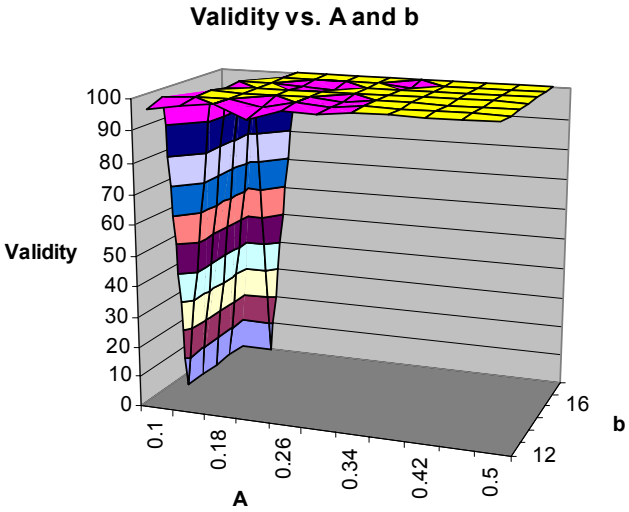

Figure 20: *Plot of validity vs. A and b for Knight DOE.*

A \ b	0.1	0.14	0.18	0.22	0.26	0.3	0.34	0.38	0.42	0.46	0.5
12	23.98	19.18	19.96	15.99	14.09	12.92	11.96	11.23	10.46	9.9	9.4
13	24	21.99	20	17.71	15.94	14.33	12.03	11.93	11.05	10.72	9.85
14	0	22.94	20.94	19.16	16.31	14.58	12.03	11.96	11.77	11	10.9
15	0	22.99	21	19.96	17.73	15.98	13.67	11.99	11.84	11.64	11.1
16	0	23.99	21.75	20.34	18.07	16.28	14.41	12.19	12.05	11.99	11.8
17	0	23.99	22	21	19.95	17.11	15.62	13.57	12.01	12.05	11.92
18	0	0	22.97	21.95	19.99	18.25	16	14.28	12.44	12.01	12.01

Table 7: *Number of neurons on as a function of A and B for Knight DOE.*

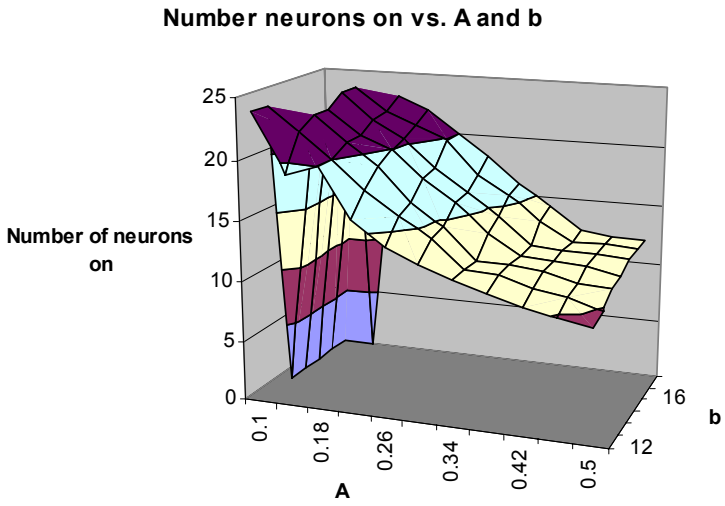

Figure 21: *Plot of number of neurons on as a function of A and b for Knight DOE.*

We only get an average of 24 neurons on for the Knight DOE when we would expect to get 32.

4.1.4 Summary

The result for the King DOE is optimum with an average of 15.89 out of 16 neurons on and A and b is 0.08 and 7 respectively. The result for the Queen is not optimum with 4.11 out of 8 neurons on and A and b is 1 and 5 respectively. We can achieve a larger number of neurons on by choosing different values of A and b but that would be at the cost of a reduction in validity. This is a combined consequence of the Queen DOEs large fan-out and the noise in the neural network. The Knight DOE has a small fan-out, so one would expect it to give optimum results, but it doesn't with only 24 out of 32 possible neurons on with A and b equal to 0.14 and 16 respectively. It is possible that this is a side effect

of the observed stable states. The Knight DOE has a very strong stable state, much stronger than observed in other DOEs, that repeats itself 90-95% of the results. It could be that the neural network reaches this stable state and refuses to optimize further to achieve a larger number of neurons on.

Listed below is a table of the different values of *A* and *b* that gave the best results.

	A	b	fan-out	A/b
King	0.08	7	8	0.011429
queen	1	5	56	0.2
knight	0.14	16	6	0.008125
crossbar	1.05	16	28	0.065625
banyan	1.05	9	43	0.116667

Table 8: *Best values of A and b for different DOEs and fan-out for the DOEs.*

It looks like there might be some kind of relation ship between *A/b* and the DOE fan-out. When the DOE fan-out increases, the ratio *A/b* increases.

DOE fan-out is defined as the number of allowed orders light is diffracted into. E.g. figure 22 illustrates the fan-out for the King DOE.

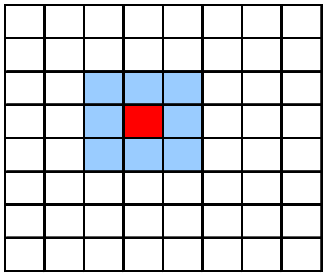

Figure 22: *The range of a King DOE, marked in blue.*

We see from figure 22 that the fan-out of the King DOE is 8.

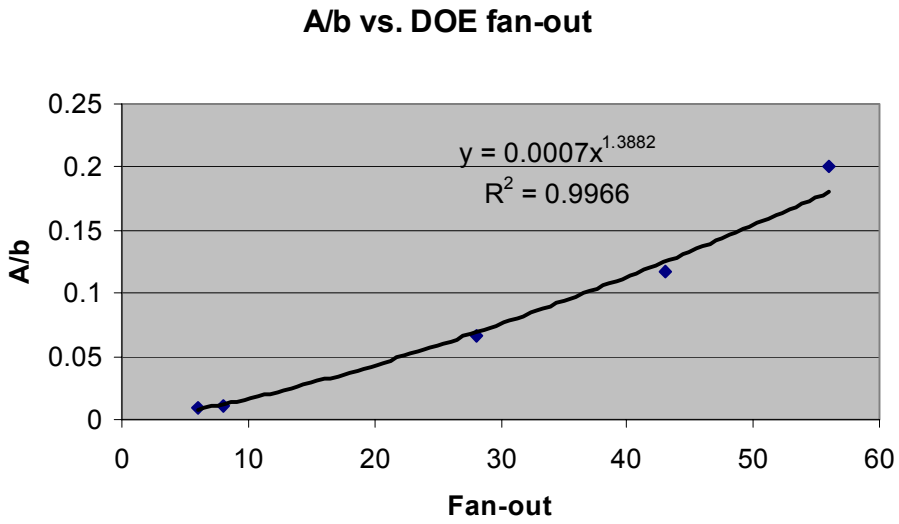

Figure 23: *Plot of A/b as a function of DOE fan-out.*

Figure 23 indicates that there is indeed a relationship between *A/b* and the DOE fan-out, and that it can be described with a power function shown in equation 4.1. This would indicate that *A/b* is constant, which would be nice, since that means it is easier to set up the neural network with new DOEs without having to go through the lengthy process of determining *A* and *b* for every DOE manually.

$$A/b=0.0007*x\text{^}1.3882 \tag{4.1}$$

This relationship is linked to the noise in the network and equation 4.1 is determined by the level of noise that is present in the network

4.2 Verify A/b relationship

King	A	0.034	0.057	0.069	0.103	0.126	0.171	0.206	0.229
	b	3	5	6	9	11	15	18	20
	Valid	100	100	100	100	100	100	100	100
	Neurons	14.95	15.84	15.85	15.88	15.97	15.88	15.88	15.93
Queen	A	0.600	1.000	1.200	1.800	2.200	3.000	3.600	4.000
	b	3	5	6	9	11	15	18	20
	Valid	100	97	94	94	95	98	96	100
	Neurons	3.93	4.05	4.09	4.02	3.77	4.08	4.15	4.11
Knight	A	0.026	0.044	0.053	0.079	0.096	0.131	0.158	0.175
	b	3	5	6	9	11	15	18	20
	Valid	100	100	99	99	100	99	100	100
	Neurons	21.59	23.04	22.92	23.94	23.99	23.99	23.99	23.98

Table 9: *Validity and number of neurons on for different values of A and B.*

In table 9 we have chosen a set of values for *b,* calculated the corresponding value of *A* determined by,

$$A=b*Ce \tag{4.2}$$

where *Ce* is the experimentally determined value of *A/b*, and measured the validity and number of neurons on as a function of these values.

We have done the same in table 10, only the value of *A* is determined using

$$A=b*Ct \tag{4.3}$$

where *Ct* is determined by the exponential curve-fit in figure 23 and shown in 4.1.

King	A	0.031	0.052	0.063	0.094	0.115	0.157	0.189	0.210
	b	3	5	6	9	11	15	18	20
	Valid	95	100	96	99	100	99	99	100
	Neurons	15.51	15.98	15.76	15.97	15.98	15.97	15.95	16.00
	A	0.557	0.928	1.113	1.670	2.041	2.783	3.340	3.711
	b	3	5	6	9	11	15	18	20
	Valid	98	97	90	98	84	95	100	100
	Neurons	3.81	4.23	4.18	4.13	3.95	4.14	4.08	4.16
	A	0.027	0.045	0.054	0.080	0.098	0.134	0.161	0.179

b	3	5	6	9	11	15	18	20
Valid	98	99	99	100	100	100	100	100
Neurons	21.83	22.81	23.63	23.32	23.00	23.26	23.01	24.00

Table 10: *Validity and number of neurons on for different values of A and b.*

We see from tables 9 and 10 that *A/b* is constant for the values of *A* and *b* that produces optimum results and the constant is determined by the DOE fan-out according to formula 4.1. It is worth pointing out that results seem to become better for large values of *A* and *b.* Also note that the results for the Queen DOE are not all 100% valid. E.g. validity for *A*=1 and *b*=5 is only 97% whereas before it was 99.3%. This might be caused by the DOE being misaligned, but we tried to realign and still couldn't achieve results as good as we had before. The power out-put of the VCSEL array is very temperature dependant [2] and might be the cause of this.

This provides us with a way of getting a good estimate for what values of *A* and *b* to use, depending on DOE fan-out. Use 4.1 to determine *A/b* depending on the DOE fan-out, choose a relatively high value of *b* and calculate *A*. It would seem that *b*=20 would be a good choice, larger values of *b* causes a drop in performance

4.3 Performance

In this section we look at the validity and optimality of the results as a function of neural network load and number of neuron network iterations.

4.3.1 King

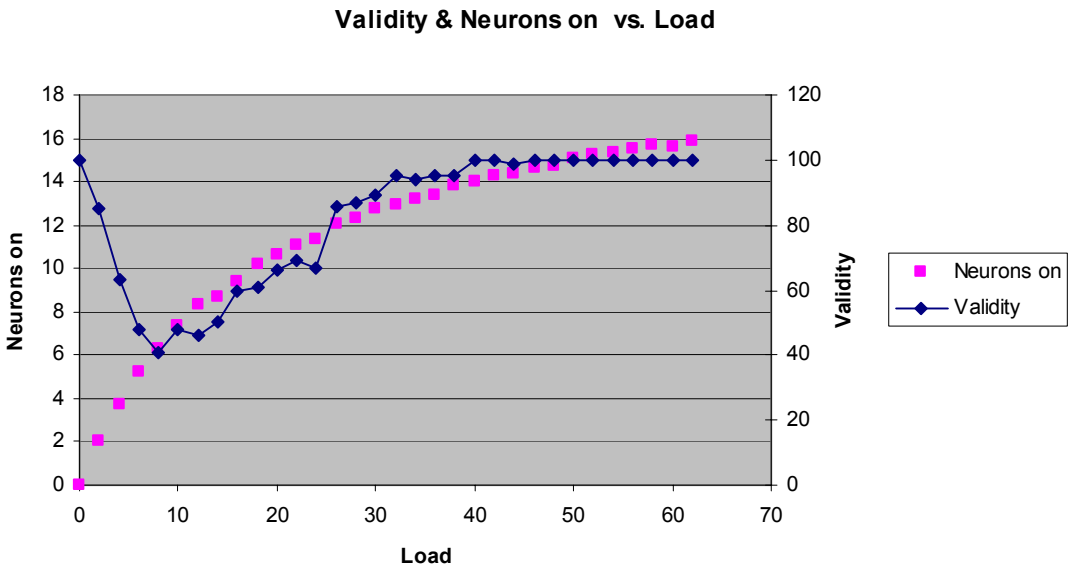

Figure 24: *Plot of Validity and no. of Neurons on against neuron network load for King DOE.*

In figures 24 and 25 we have plotted the no. of neurons on and the validity as a function of Load and no. of iterations, respectively, for the King DOE.

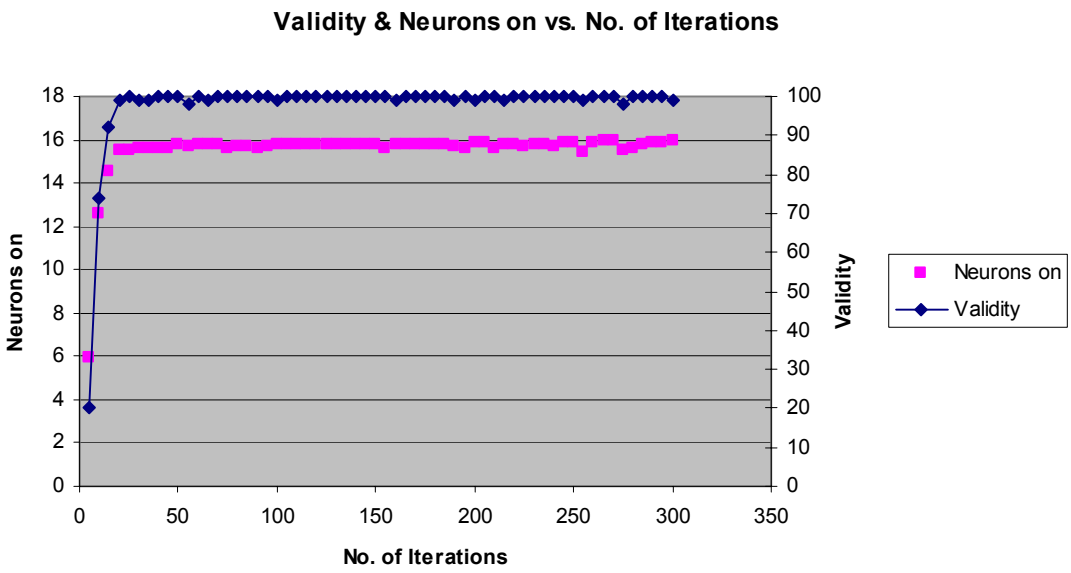

Figure 25: *Plot of Validity and no. of Neurons on against no. of Iterations for King DOE.*

4.3.2 Queen

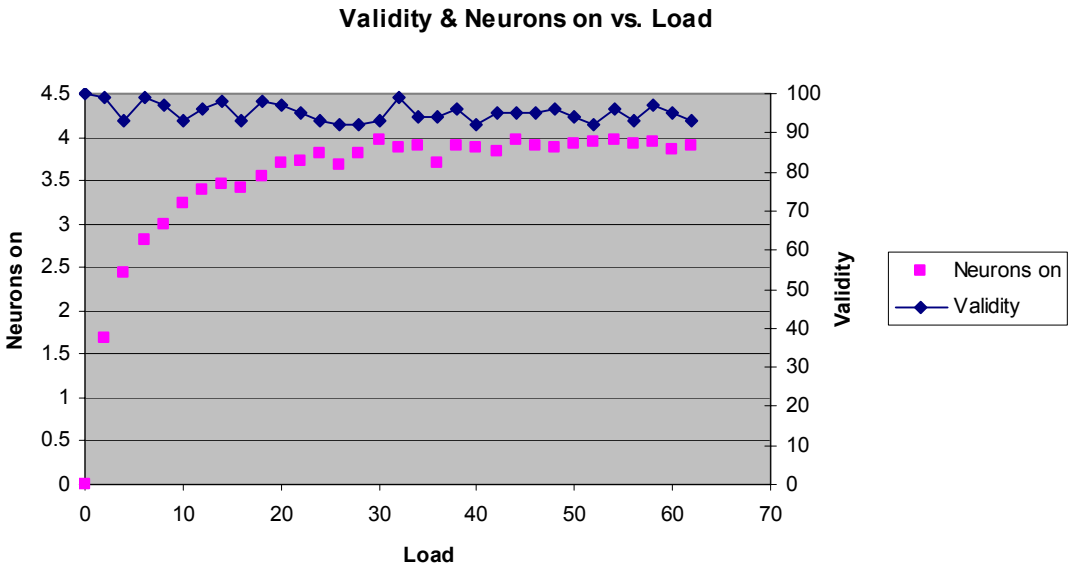

Figure 26: *Plot of Validity and no. of Neurons on against neuron network load for Queen DOE.*

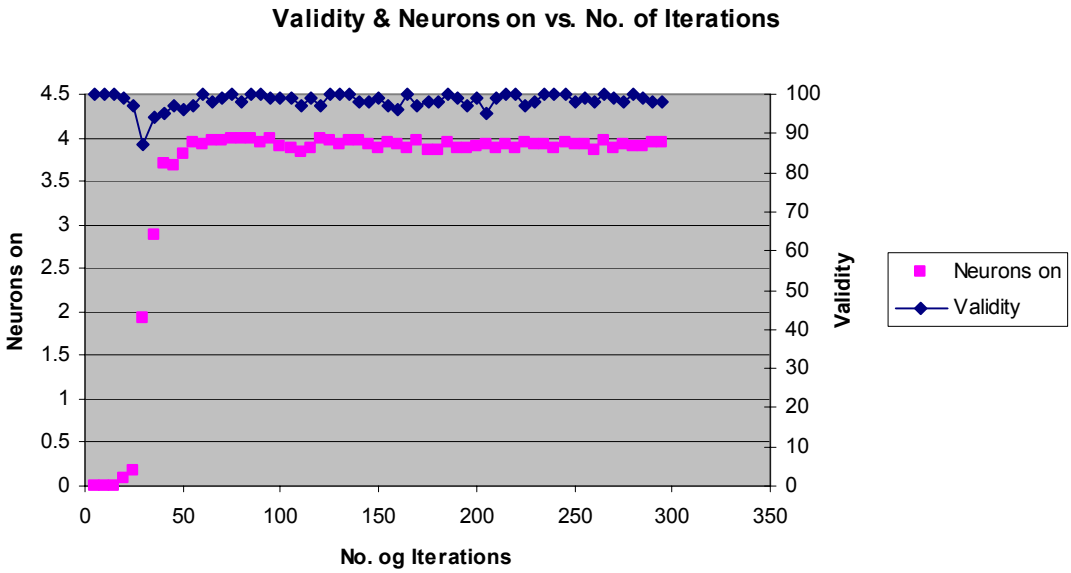

Figure 27: *Plot of Validity and no. of Neurons on against no. of Iterations for Queen DOE.*

In figures 26 and 27 we have plotted the no. of neurons on and the validity as a function of Load and no. of iterations, respectively, for the Queen DOE.

4.3.3 Knight

In figures 28 and 29 we have plotted the no. of neurons on and the validity as a function of Load and no. of iterations, respectively, for the Knight DOE.

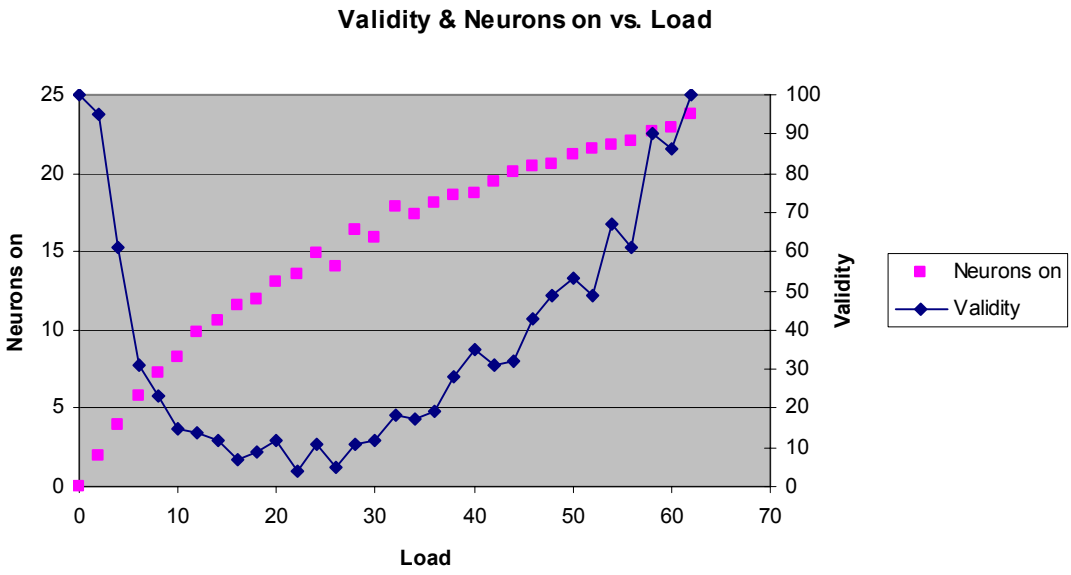

Figure 28: *Plot of Validity and no. of Neurons on against neuron network load for Knight DOE.*

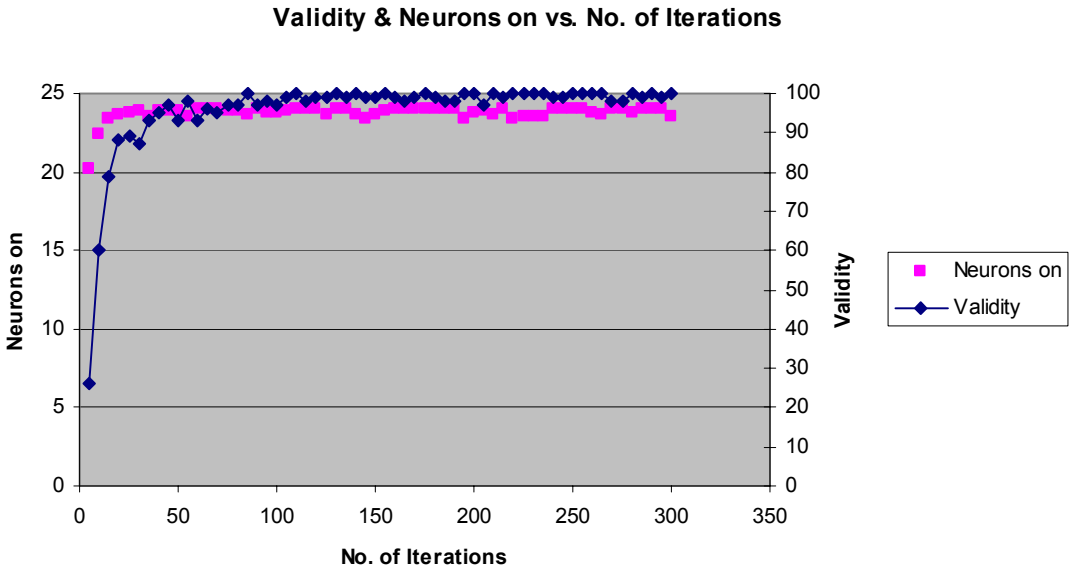

Figure 29: *Plot of Validity and no. of Neurons on against no. of Iterations for Knight DOE.*

4.3.4 Optimality vs. fan-out

We mentioned briefly in the start that we could not get optimal results for the Banyan DOE as for the Queen DOE. These are both DOEs with a large fan-out causing a low signal to noise ratio and the theory is that a low signal to noise ratio is the basis for sub-optimum results. In fact, data gathered during the project supports this theory.

	fan-out	Optimality
King	8	0.99
queen	56	0.51
knight	6	0.75
crossbar	14	1.0
banyan	19	0.64

Table 11: *Optimality vs. DOE fan-out.*

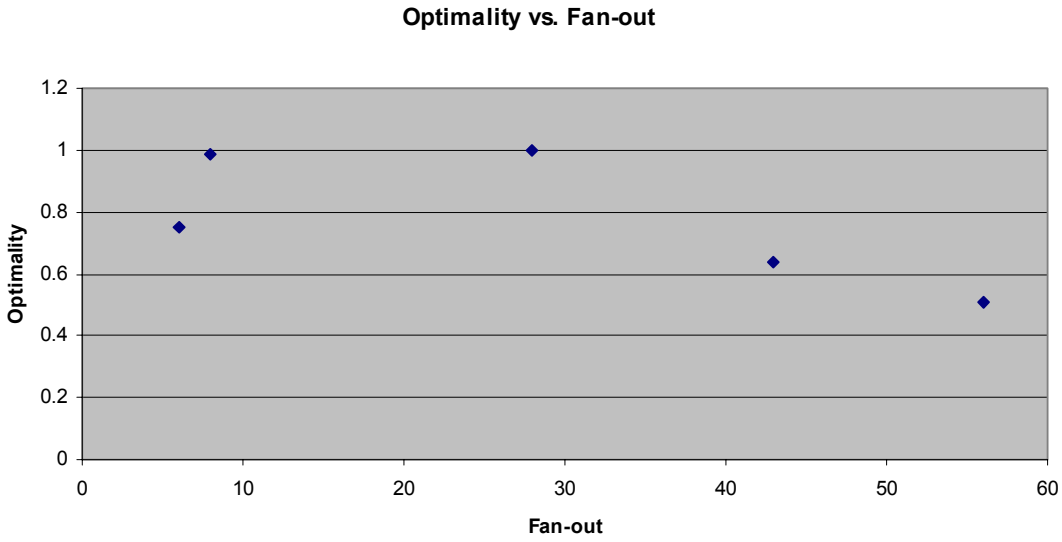

Figure 30: *Plot of optimality vs. DOE fan-out.*

Table 11 and figure 30 confirm our suspicion about the connection between optimality and DOE fan-out. This shows that there is a limit to the DOE fan-out that can be used with the neural network demonstrator. One would expect if the VCSEL power is increased, then the fan-out threshold will increase as well.

The low optimality of the Knight DOE is however something that needs further investigation.

4.3.5 Summary

Notice how each DEO behaves differently at the different levels of load and when using different numbers of iterations. Both King and Knight have a high validity for full load and as load decreases, so does the validity, but at different rates. Also the validity goes back up when the load is reduce further.

The number of neurons on behaves as one would expect as a function of load, it decreases when load decreases, but the rate at what it decreases is different for all DOEs.

Validity and number of neurons on as a function of the number of iterations is very similar for all DOEs, except from the Queen DOE where the validity hardly changes and the number of neurons on drops a lot earlier than for the King and Knight.

5 Conclusion

This project demonstrates one of the strengths of an optical neural network. It is very easy to change the interconnects in the neural network and prepare it for different tasks. The DEO is very easily to align since the performance of the network is not very sensitive to the position of the DOE. We also discovered that the network is very insensitive to the quality of a DOE. The network would perform well even with a DOE made for a completely different wavelength than that of the VCSEL.

The neural network did a good job of optimizing the positioning of Kings in a 8x8 grid, giving an optimality of 99% or better and 100% validity, but performance of the Queen and Knight DOE positioning could be better. We have seen that the reason for the Queen performing badly is related to a fan-out threshold that the Queen DOE is greater than and therefore produces less than optimal results. The reason for the Knight performing badly is not fully understood it is possible it is related to the system producing stable states, this should be investigated closer.

We have also shown that there is a relationship between the ratio A/b and the DOE fan-out. This will make it easier to determine the best values of the neuron network parameters for other DOEs.

Bibliography

[1] Optoelectronic Neural Network Demonstrator: Program and Results

Yves Randle, 2002.

[2] Optically Interconnected Computing Systems

PhD Thesis, Keith J Symington, 2001

[3] Neural Computing an Introduction

R Beale & T Jackson

IoP Publishing, ISBN 0852742622, 1990

[4] Optoelectronic neural-network Scheduler for Packet Switches

Roderick P Webb, Andrew J Waddie, Keith J Symington, Mohammed R Taghizadeh, and John F Snowdon

Applied Optics, Vol. 39, No. 5, 10 February 2000

[5] Artificial neural networks for parameter estimation in geophysics

Carlos Calderon-Macias, Mrinal K Sen and Paul L Stoffa

Geophysical Prospecting, 2000, 48, 21-47

[6] Neural network Design of a Banyan Network Controller

Timothy X Brown and Kuo-Hui Liu

IEEE Journal on Selected Areas in Communications, Vol.8, No. 8, October 1990