university of groningen / faculty of science and engineering

Extreme learning machine using filters for artificial lateral line source localisation

Exploring the use of filters and simultaneous use of two activation functions in an extreme learning machine for underwater source localisation using an artificial lateral line.

Jelle Egbers, s2701227, J.T.Egbers@student.rug.nl,
Supervisors: Dr. S.M. van Netten, B.J. Wolf & Dr.P. Pirih

Abstract: Data of an artificial lateral line, an array of sensors which are able to sense the differences in water flow, can be used for source localisation and angle prediction. In this research, possibilities to improve an extreme learning machine used for this purpose are explored. This is tried with filters, by changing the input representation, and by changing the activation function. Changing the input representation to a square matrix improves the accuracy of the algorithm, with the mean square error reaching an asymptotic line as increasingly more filters are added. An algorithm with both the ReLu and tanh activation functions also turned out to work well, because of the fact that one function was good at predicting the location and the other function was good at predicting the angle.

1 Introduction

1.1 Artifical Lateral Line sensors

Fish that live so deep down in the sea that they can't see anything due to the darkness, have to trust on other senses. Most of these fish have a lateral line organ, allowing them to detect objects and obstacles without actually seeing them [3]. This organ detects the disturbances in water, caused by nearby sources, relative to the body of the fish. Due to these disturbances within water, the fish can determine the location and the angle of motion of the source [2].

With artificial lateral line (ALL) sensors, the disturbances in water can also be measured. Using the measured profile, the location and angle of a moving object can be inferred [1]. This has been tried with a multilayer perceptron (MLP), an extreme learning machine (ELM), and an echo state network [1]. They showed that the ELM achieved the best results overall, while the MLP showed better results when increasingly more noise was added. Another advantage of an ELM over a MLP is the training time, an ELM may take in the order of seconds to train, where the MLP needs in the order of hours to train [6]. Both take in the order of seconds to test [1]. In this research, the possibility to achieve even better results with an ELM is explored.

1.2 Extreme learning machine

Since its discovery by Huang [6], the ELM has been researched a lot, especially by its inventor. One of the variants was the local receptive field ELM (ELM-LRF) [5]. In this ELM, the random layer in a plain ELM is replaced with a layer consisting of what are called orthogonal filters. This layer is followed up by a pooling layer. The idea behind this ELM is to have the feature extracting power of filters while having the speed of a plain ELM.

Orthogonal filters are filters where the matrix which represents the filter shows orthogonal characteristics, i.e. the inverse of the matrix is the same as the transpose of the matrix. Orthogonalisation allows to extract a more complete set of features than non-orthogonal ones, improving the generalisation of the network [5], [7].

In [5], square-root pooling was used to scale down the amount of information. Specifically square-root pooling was used because it has been proven to be frequency selective and translational invariant, meaning that the system produces exactly the same response, regardless of how its input is shifted [5], [9]. Because the spatial features contain most infor-

mation, this is more suitable for image processing, and making it less interesting for source localisation.

In [5], Huang states that for the problem of 3D image classification the ELM-LRF algorithm outperforms the other state of the art (neural network) algorithms with an error of 2.74% against errors of 2.8% to 6.6%. Despite the fact that the task of 3D image classification is not comparable to source localisation, the improvement in performance of the algorithm was a reason to experiment with it. The main difference between the ELM-LRF and the algorithms used in this research is that the pooling layer will be omitted completely.

In this research, the goal is to find out if (orthogonal) filters can increase the accuracy of an ELM used for source localisation using ALL data. A second requirement for the algorithm is not to take 'too' long, i.e. it is still possible to train and test the algorithm in the order of seconds.

2 Method

2.1 Extreme learning machine

An ELM is a neural network with one hidden layer, first described in [6]. The weights from the input to the hidden layer are fully connected and randomly initialised. The values this generates are then evaluated with a predetermined activation function, deciding for each node in the random layer individually which value is used for the continuation of the calculation. This is layer H, the output of the hidden layer. This hidden layer is fully connected to the output layer with weight matrix W. When W would be perfectly chosen, the multiplication of W and H would give the training output T:

$$WH = T. \tag{2.1}$$

Because W is not known beforehand, the algorithm needs to be trained to find it. This is done using batch learning with the formula:

$$W = TH^{\dagger}, \tag{2.2}$$

where $H^{\dagger}$ is the MoorePenrose generalised inverse of matrix H.

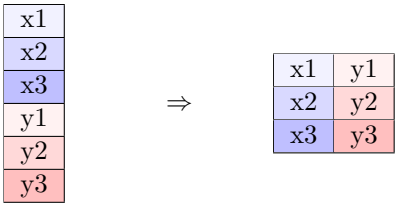


Figure 2.1: Example of how a vector [x1, x2, x3, y1, y2, y3] is put into two columns. This to be able to apply a 2 x 2 orthogonal filter. In the research there are 16 x-values and 16 y-values for the sensed velocities.

2.1.1 Input

In this research, different kind of filters are applied to the input. Because the filters have different shapes, it was necessary to change the dimensions of the input to be able to apply all the filters.

The inputs of the algorithm are the x- and y-velocities of 16 ALL sensors, creating a 32 x 1 vector. In the simulation, noise of $10^{-6}m/s$ flow is added and the values are normalised before being fed to the algorithm [1]. In the input vector, the 16 x-values are stored first and then the 16 y-values.

Because the orthogonal filters are square, and so have at least a width of two, the input also needs to have at least a width of two. This was achieved by storing the x- and y-values in separate columns, instead of storing the x- and y-values all in one column, creating an input matrix of 16 x 2. This can be seen in table 2.1. This input format is only used for orthogonal filters.

Filters are used for feature detection, because observed values might not match the filter features exactly or they might be alike but shifted, also a square instance of the input vector was created. This was achieved by creating a 32 x 32 matrix, where each column is shifted 1 place with respect to the previous column. This is from now on referred to as squared input. Table 2.1 shows an example of a vector which is squared. This input format is used for both orthogonal as signal shaped filters.

2.1.2 Filters

In this research, three kinds of filters are investigated: orthogonal filters, signal shaped filters, and random filters.

Orthogonal filters are filters where the inverse of

x1	⇒	x1	y3	y2	y1	x3	x2
x2		x2	x1	y3	y2	y1	x3
x3		x3	x2	x1	y3	y2	y1
y1		y1	x3	x2	x1	y3	y2
y2		y2	y1	x3	x2	x1	y3
y3		y3	y2	y1	x3	x2	x1

Table 2.1: Example of how a vector [x1, x2, x3, y1, y2, y3] is squared, both for signal shaped as orthogonal filters to be applied. In the research there are 16 x-values and 16 y-values and so this results in a 32 x 32 matrix.

the matrix is the same as the transpose, only square filters are able to meet these requirements. To create an orthogonal filter, first a randomly initialised matrix (of the wanted size) is constructed, M. Then with Singular Value Decomposition (SVD) U, S and V are calculated [4]:

$$U, S, V = \text{SVD}(M). \tag{2.3}$$

For matrix M with size m x n and rank r, U is a m x m orthogonal matrix, V is a n x n orthogonal matrix and S is a m x n pseudo-diagonal matrix where the first r elements are the singular values of of M and all other elements are zero. The columns of U are the left singular vectors, the columns of V are the right singular vectors. Here U and V are not unique while S is. When multiplying U and V the closest orthogonal filter to M is found [10].

Signal shaped filters are simulated and normalised signals without noise. An example of a signal shaped filter, and thus a velocity profile without added noise, is showed in figure 2.2. The two lines repsresent the observed x- and y-velocities.

Random filters are randomly initialised. These are, in contrast to the orthogonal filters, not square. They are, like the signal shaped filter, as long as one input velocity profile.

The amount of filters used is experimented with throughout the research and varies from 2 to 350. The size of the orthogonal filters is for the 16 x 2 input always a 2 x 2 orthogonal matrix. The size of the filters for the 32 x 32 squared input, varies from a 2 x 2 to 32 x 32 orthogonal matrices.

2.1.3 Activation function

After applying weights or filters to a layer, the nodes in the following layer have a certain activation value. An activation function can be applied to these activation values, this can serve to scale the activation values to predetermined ranges, but they also serve to give the network non-linear characteristics.

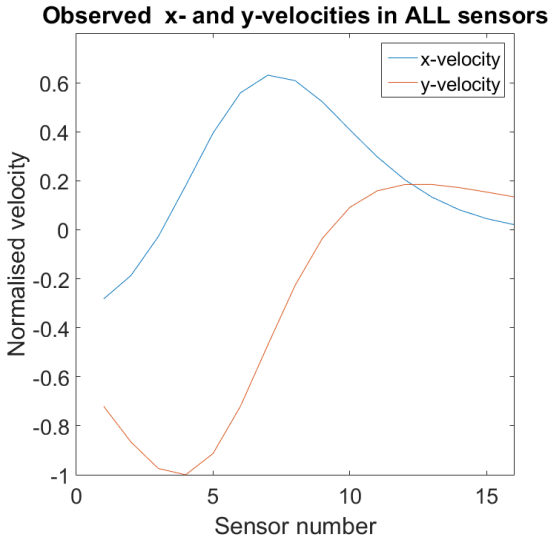


Figure 2.2: Example of an observed velocity profile. When used as filter the x- and y-values are placed in one column.

In this research, the possibilities of two activation functions are explored, the tanh and ReLu activation functions. Both are shown in figure 2.3. The $\tanh(x)$ function scales the value between -1 and 1 in a non-linear manner. The ReLu function is a linear function when the input value is 0 or higher and otherwise 0 ($\max(0, x)$). This is particularly interesting when the value after applying a filter is high. This means there is a high correlation between the filter and the input which most likely means that the feature of that particular filter is matched. When the filter does not match at all, the value is negative, resulting in an output of 0. This means that that node does not propagate any information further.

2.1.4 Second hidden layer

There are two possible continuations after the filters and activation function are applied. The first possibility is to apply the MoorePenrose Inverse (equation 2.2) to get the output weights and have

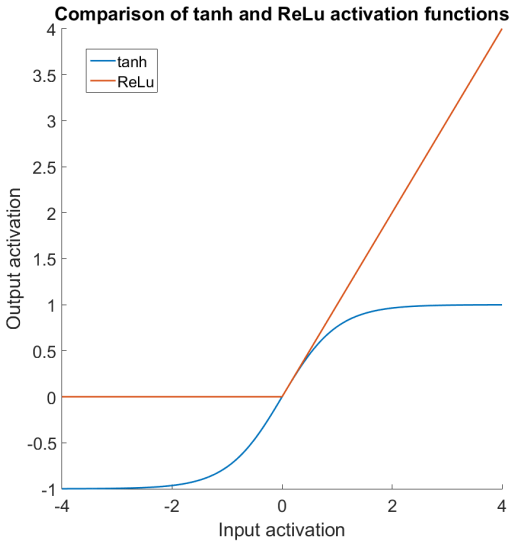


Figure 2.3: Plot of the ReLu and tanh activation functions.

a finished ELM.

The second possibility is to add a random layer, one that in a plain ELM is directly placed after the input, after the filters. The idea is that in this way the algorithm can take advantage of the power of filters, as well as random transformations.

2.2 Simulation environment

All the training and testing samples are obtained from a simulated environment stretching from $(-1, 0)$ to $(1, 1)$, comparable to the simulation environment as in [1]. The difference is that the sensors are placed between $(-0.5, -0.1)$ and $(0.5, -0.1)$. The amount of sensors is set to 16, creating a 32 input vector. 10^{-6} m/s flow of noise is added to the sampled signals. The source which creates the disturbances in the simulated water has a radius of 0.06 meter and moves with a speed of 0.16 m/s. The environment can be thought of as a small swimming pool of 1 by 2 meter, this means this is a closed environment where the borders can't be passed.

As in [1], potential flow is assumed. This is justified by previous studies on fish. The shape of the wavelets and the distance coding predicted by potential flow are also observed in a biological lateral line response of fish [2]. The movement made by the simulated source generates a potential flow which can be measured by the sensors in the ALL and from this the angle and location predictions are made.

For training, 1386 samples are taken in a grid-like manner with random angles. For testing, multiple motion paths are simulated where each new state has a change of direction within the magnitude of $\pi/4$ with respect to the previous state.

2.3 Bias

By adding either a single or multiple ones to every layer except the output layer, a bias was added to the algorithm. In the case the input is squared, a whole row of ones is added after the squared input. Bias has the effect of shifting the result. By adding bias to the network, the network is capable of learning the necessary shift, which is harder/not possible for the network to learn without bias.

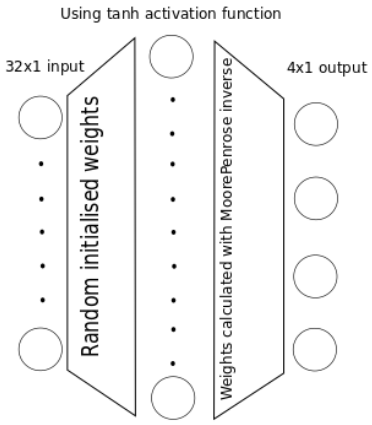
2.4 Used performance measure

The results will be evaluated based on the achieved average mean squared error (MSE) on the test set. The average MSE is the average value of two other MSE's. The location MSE is the error in the prediction of the location of the source and the angle MSE is the error in the prediction of the angle of the source. From here on, when referring to MSE, the average MSE is meant except when explicitly mentioned otherwise.

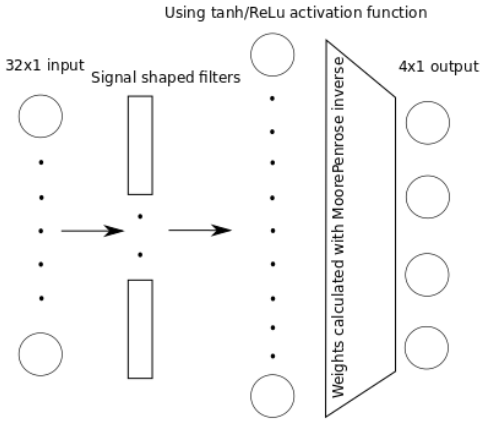
3 Results

3.1 Baseline: A plain ELM

A plain ELM, as seen in figure 2.4a, with normal input and 400 hidden nodes was used as baseline. This algorithm with a tanh activation function achieves a MSE of 4.96%. Using the ReLu activation function a MSE of 5.26% was achieved.

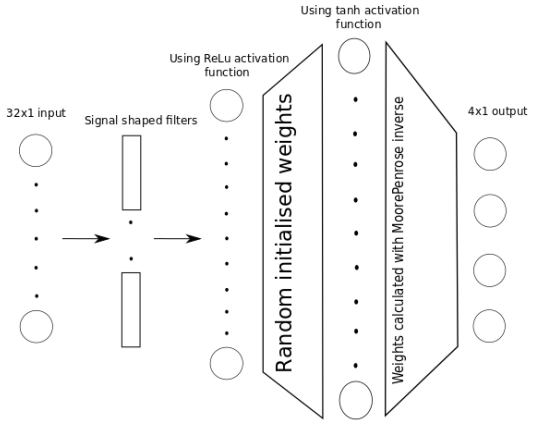


(a) Visualisation of a plain ELM.

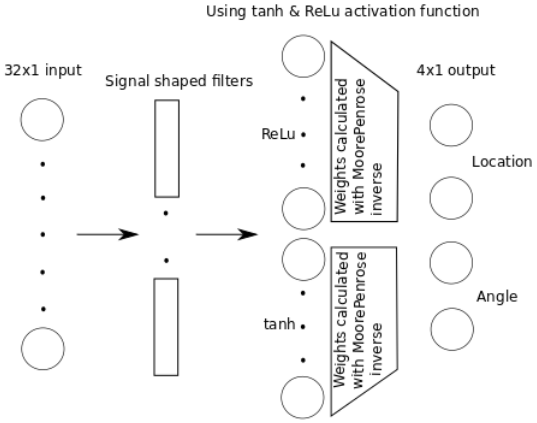


(b) Visualisation of an ELM with added signal shaped filters

Figure 2.4



(a) Visualisation of an ELM with added signal shaped filters and a random layer.



(b) Visualisation of an ELM with added signal shaped filters and a split connected hidden layer.

Figure 2.5

	Input format	Filters	# of filters	Activation function	Random layer	# of nodes	Activation function	Bias	MSE
1	Normal	None	None	None	Yes	400	tanh	Yes	4.96%
2	Normal	None	None	None	Yes	400	ReLu	Yes	5.26%
3	Normal	Signal shaped	82	Relu	None	None	None	No	12.47%
4	Normal	Signal shaped	162	tanh	None	None	None	No	18.41%
5	Normal	Signal shaped	102	Relu	None	None	None	Yes	7.82%
6	Normal	Signal shaped	142	tanh	None	None	None	Yes	4.79%
7	Normal	Signal shaped	50	ReLu	Yes	500	tanh	No	3.92%
8	Normal	Signal shaped	50	ReLu	Yes	500	tanh	Yes	3.84%
9	Normal	Signal shaped	390	tanh & ReLu*	None	None	None	No	4.13%
10	Normal	Signal shaped	330	tanh & ReLu*	None	None	None	Yes	3.54%
11	Normal	Signal shaped	570	tanh & ReLu**	None	None	None	No	**3.18%**
12	Normal	Signal shaped	330	tanh & ReLu**	None	None	None	Yes	3.37%
13	Normal	Random	90	tanh & ReLu**	None	None	None	No	19.27%
14	Normal	None	None	None	Yes	320	tanh & ReLu**	No	5.08%
15	2 columns	2x2 orthogonal	8	Relu	None	None	None	No	12.62%
16	2 columns	2x2 orthogonal	8	tanh	None	None	None	No	20.08%
17	2 columns	2x2 orthogonal	8	ReLu	None	None	None	Yes	11.14%
18	2 columns	2x2 orthogonal	10	tanh	None	None	None	Yes	**7.33%**
19	Square	Signal shaped	4	ReLu	None	None	None	No	14.12%
20	Square	Signal shaped	4	tanh	None	None	None	No	21.52%
21	Square	Signal shaped	274	ReLu	None	None	None	Yes	2.89%
22	Square	Signal shaped	290	tanh	None	None	None	Yes	3.62%
23	Square	32x32 orthogonal	86	ReLu	None	None	None	No	11.91%
24	Square	32x32 orthogonal	200	tanh	None	None	None	No	17.67%
25	Square	20x20 orthogonal	248	ReLu	None	None	None	Yes	**2.80%**
26	Square	20x20 orthogonal	248	tanh	None	None	None	Yes	3.56%

Table 3.1: Results of the different variations on the plain ELM. For each variation, the results of the algorithm with the optimal hyper parameters are shown.
*** is fully connected, ** is split connected.**

3.2 Input in 2 columns with orthogonal filters

Table 3.1, rows 15-18, show the result of the algorithm using 2 x 2 orthogonal filters on input which was split up in 2 columns. The algorithm using the tanh function and added bias achieved the lowest MSE. The highest MSE was also using the tanh function but without bias.

3.3 Signal shaped filters with different activation functions

Table 3.1, rows 3-6, show the results of the algorithm using signal shaped filters and the different kind of activation functions. A visual representation of this algorithm can be seen in figure 2.4b. Using the tanh function with no bias gives the highest MSE. The lowest MSE was achieve by the tanh function with bias.

3.4 Signal shaped filters followed by a random layer with and without bias

Table 3.1, rows 7-8, show the results of an algorithm with filters and a random layer without and with bias. A visual representation can be seen in figure 2.5a. The first layer which consists of the filters, uses the ReLu activation function, the second, random, layer uses the tanh activation function. In both cases the algorithm achieves the best result with 500 random nodes and 50 filters. Figure A.2a shows a peak in the MSE for the algorithm with bias at 250 filters for 500 and 700 random nodes.

3.5 Two different activation functions in hidden layer

Table 3.1, rows 9-14, show the results of the algorithm with a hidden layer with the two activation functions and signal shaped filters. In the fully connected hidden layer, the hidden nodes are fully connected to the four output nodes. In the split connected hidden layer, the nodes using the tanh activation function predict the angle while the nodes using the ReLu activation function predict the location. A visual representation of this is shown in figure 2.5b.

The algorithm with filters that are randomly initialised is also shown in the table. This performance is worse than when signal shaped filters are used.

The result of a plain ELM with the tanh and ReLu activation function split connected in the hidden layer is also shown. This variant achieves a higher MSE than when filters are used.

3.6 Squared input with signal shaped filters

Table 3.1, rows 19-22, show the results of the algorithm with a squared input and signal shaped filters. Both the algorithms with no bias achieve a high MSE. When a bias is added the MSE is lower, and when looking at figure A.3a, they seem to approach an asymptotic line.

3.7 Squared input with orthogonal filters

Table 3.1, rows 23-26, show the results of an algorithm with squared input and orthogonal filters with and without bias. The results without bias are again worse than with bias. With bias and ReLu function the best MSE is achieved and it again seems that for both activation functions an asymptotic line is approached, which is shown in figure A.4a and A.4b.

3.8 Location and Angle prediction

When using the algorithm with 150 signal shaped filters, the different activation functions and whether a bias is added, have large effects on the results. Table 3.2 shows that when no bias is used, the algorithm using the ReLu activation function is better at predicting the location while the algorithm using the tanh activation function is better at prediction the angle. When a bias is added, these differences get smaller for the ReLu function while disappearing almost completely for the tanh function.

3.9 Training and testing time

An argument to use an ELM instead of an back propagating algorithm is its speed. The plain ELM trains on 1386 training examples in 0.2 seconds. The algorithm with a squared input and 250 2 x

	ReLu-	tanh-	ReLu+	tanh+
average MSE	13%	19%	7.5%	**4.8**%
location MSE	**1.2**%	31%	**1.2**%	4.2%
angle (rad) MSE	25%	5.7%	14%	**5.4**%
location MED	**13**%	73%	**13**%	23%
angle (deg)	63%	20%	35%	**18**%

Table 3.2: Comparison of location and angle MSE. - means no bias, + means with bias. Results were measured with an algorithm using 150 signal shaped filters on a normal input.

2 orthogonal filters trains on that same amount of samples in 94.9 seconds, which is 474 times longer. Testing takes for the ELM also 0.2 seconds, for the algorithm with squared input and 250 2 x 2 orthogonal filters this takes 7.15 seconds, which is 35 times longer.*

li

3.10 Best performing algorithm variation

Figure 3.1 shows the different variations of the algorithm that performed best, i.e. better than the plain ELM. The algorithm with a squared input and 20 x 20 orthogonal filters using the tanh activation function and with added bias achieved the best results with a MSE of 2.8%.

4 Discussion

4.1 Input

When comparing figure A.1b and figure A.3a, the difference between signal shaped filters applied to a normal and a squared input can be seen. In both cases, the performance of the algorithm is much better when a bias is added. The MSE of the algorithm with the squared input keeps getting smaller. The MSE of the algorithm with a normal input and signal shaped filters reaches a minimum and then starts getting higher again. The reason why the MSE of the squared input keeps dropping may be because using signal shaped filters boils down to pattern matching. The chance that a filter is similar to the input increases when increasingly more filters are added. In this way, the algorithm has a 'memory' of every possible location for the source and because the input is squared, the algorithm is also able to recognise it when it is shifted slightly. This might not be maintainable when the problem space gets larger. The amount of filters necessary to have one for almost every location will not be realistic because every filter first needs to be sampled.

4.2 Filters

When looking at figure A.1a and figure A.1b, the results of the two different filters can be analysed. In both cases, choosing the tanh activation function is worse when using no bias, but better when using a bias. Also, a lot less filters are necessary when using an orthogonal filter to reach the minimum MSE. This is due to the fact that an orthogonal filter does not have the same shape as the input, meaning that when a filter is applied, multiple hidden layer nodes are created.

However, the difference is that when signal shaped filters are used, the MSE goes as low as 4.83% while the lowest achieved MSE using the 2 x 2 orthogonal filters was 7.33%.

This changes when orthogonal and signal shaped filters are applied on a square input, these situations can be seen in figures A.4a and A.4b for orthogonal filters and in figure A.3a for signal shaped filters. In both cases, the MSE seems to go to an asymptotic line which is around 2.8% for the signal shaped filters and lower, but not yet good definable for the orthogonal filters. The problem is that sources from different locations show different characteristics which are 'recognised' by the filters, and when the problem space gets bigger, increasingly more filters will be necessary to recognise every characteristic from the different locations.

For the optimisation of the number of filters, the amount of steps for each filter increment taken goes up to 20, this gives a good estimation of where the (local) minimum for the MSE lies, but this might miss the optimum amount of filters. Smaller increments could solve this problem but increases the time necessary for testing.

*On an Intel Core i7 3770 @ 3.40GHz.

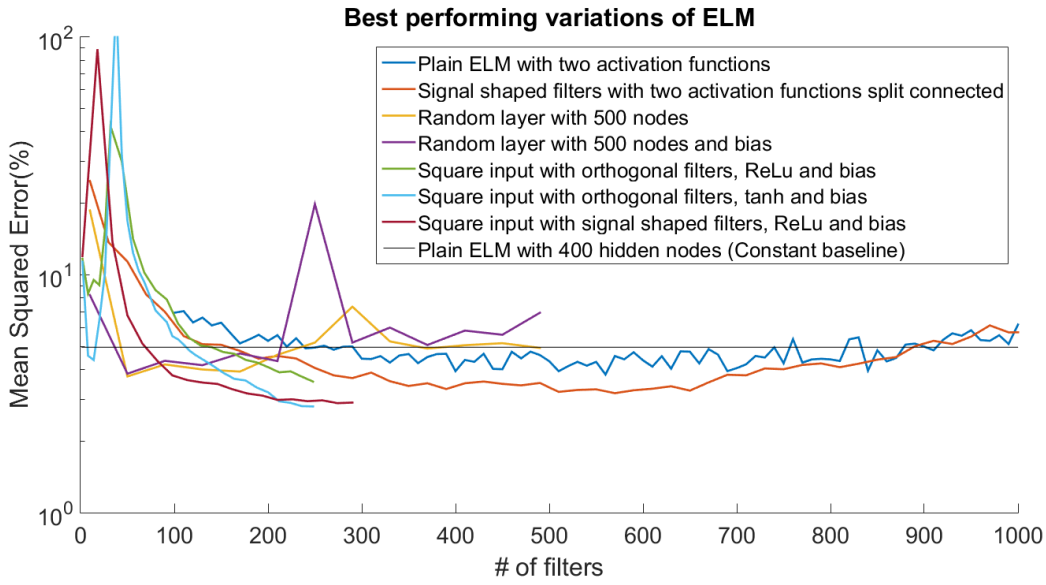


Figure 3.1: Best performing variations of the plain ELM

4.3 Random layer

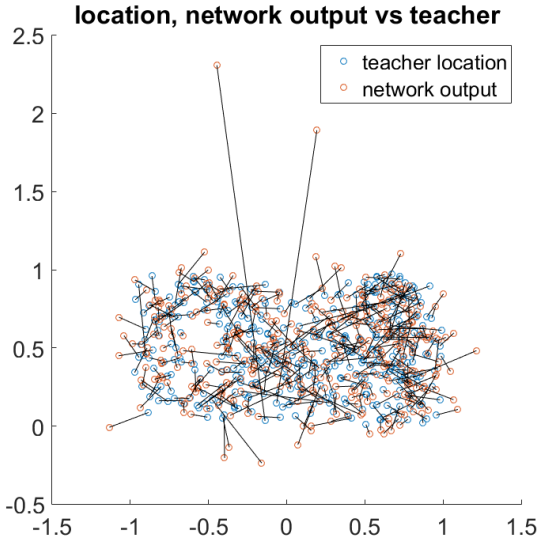
When looking at figure A.1c and figure A.2a, showing the performance of an algorithm with filters and a random layer, and figure A.1b for the same filters but without random layer, it is visible that adding a random layer after the filters can further increase the performance of the algorithm. Where only signal shaped filters with the tanh activation and bias achieved a MSE of 4.84%, the algorithm with 50 filters, 500 random and no bias achieved a MSE of 3.74%. One possible hypothesis why this happens, is that the extra random layer can derive more information based on which nodes in the previous layer are activated than the MoorePenrose function can, giving the algorithm with the extra random layer an advantage over the algorithm without random layer.

The high peak in figure A.2a for 250 filters and 500 and 700 random nodes is noteworthy, especially because other configurations do not show this peak. When running the algorithm with 10 filters less, the peak does not show up. There is likely a bad combination of filters with an unlucky initialisation of random weights present causing one or two predictions to be completely off. Figure 4.1 shows the predicted locations. It is clearly visible that the high peak in the MSE is caused by 2 network outputs which are way off, while all the other outputs are in an acceptable range of the actual teacher location.

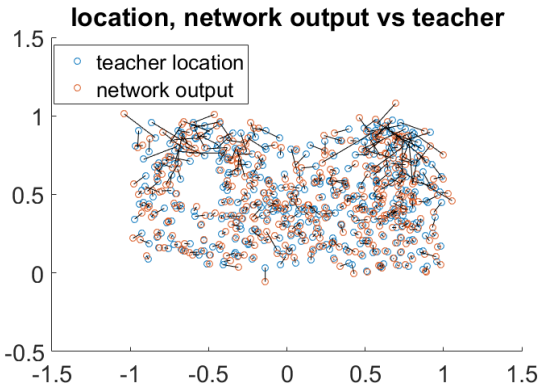
4.4 Activation function

The different activation functions show different results, which are shown in table 3.2. It is clear that the algorithm with the tanh activation function was not capable of predicting the location very well, while with an added bias this is less of a problem. This is presumably the case because the algorithm predicted the source locations around the coordinates of $(0, 0)$, while the problem space was between$(-1, 0)$ and $(1, 1)$ causing the actual centre of locations to be around $(0, 0.5)$, adding a bias allowed shifting the results enough to achieve a lower MSE. The algorithm with the tanh function did predict the angle right all along. This is probably due to the fact that the angle is location independent and because there are less different angles to predict than locations. Due to the limiting nature of the tanh function, scaling all activation between -1 and 1, it is better in 'storing' information for answers which have a smaller, rather than a larger problem space.

The algorithm using the ReLu activation function is a lot better at predicting the location than the angle. This may be due to the fact that the



(a)



(b)

Figure 4.1: Plots of the predicted location vs the actual location. Figure 4.1a shows an unfortunate filter/random weight initialisation in the algorithm with a second hidden layer with 500 nodes, 250 filters and added bias. Two locations are predicted far off, causing the MSE to increase. Figure 4.1b shows how it looks like in the case of a MSE of 3.5%.

ReLu function has in theory no maximum, giving it the ability to store a lot of information about the different locations. The prediction of the angle is a lot worse, most likely due to what makes the ReLu function good for location prediction, i.e. the ability to get a large activation level. This ability is less suitable for angle prediction with a smaller problem space, where fewer values are a possible right answer.

Bias increases the angle prediction performance by a factor of two for the ReLu function. This may be caused by the values in the hidden layer. Because the filters do not fit as perfectly on the input as without bias, the activation in the hidden layer tends to be smaller. In line with the observation made before, smaller activations in the hidden layer are better at predicting the smaller problem space of angle prediction.

This results in the algorithm with two activation functions to work quite well, because each activation function is used for the task it showed to perform best in. The use of filters does increase the need to think well about the activation functions. This is because the results of the plain ELM were 4.96% for the tanh, 5.26% for the ReLu and 4.84% for two split connected activation functions. These are less far apart than when filters were used.

4.5 Bias

When looking at table 3.1, it is clear that a bias is necessary to improve the results. The only instance adding a bias does not improve the performance of the algorithm is with the split connected hidden layers. In this variant of the algorithm, the hidden nodes using a ReLu activation function are connected to the output nodes which predict the location. The hidden nodes which use the tanh activation function are connected to the output nodes which predict the angle. The fact that in this case bias does not improve the performance, may be so because every node is predicting what it showed to perform best in.

4.6 Conclusion

It is not easy to conclude that one algorithm is better than another. When looking just at the performance, the algorithm with squared input, 20 x 20 orthogonal filters, tanh activation function and

added bias is better than the others. But because the algorithm can use increasingly more filters and thus achieve a better result, it is not clear when to stop adding filters. By adding increasingly more filters the training and testing time also goes up, while the short training and testing times were one of the arguments for an ELM in the first place. The testing time increases about 10-fold, but the algorithm still is able to give back the result in the order of seconds, which meets the requirement set in the introduction.

The algorithm with two activation functions in the hidden layer which are split connected to the output shows a clear minimum MSE for a specific amount of filters. This gives the algorithm a more predictable training time. However, the MSE of this algorithm lies around 1% higher than the asymptotic line of the algorithm with orthogonal filters seems to approach.

The use of filters as single alteration to the algorithm does not improve the algorithm drastically. Meaning that the use of filters is almost comparable to the standard random layer. But when the input and/or the hidden layer is altered in a way to have maximal profit from filters, the accuracy can increase with respect to the plain ELM.

Adding a bias to the algorithm is generally a good idea. However, this is not the case when the algorithm is specifically designed to not need one. The algorithm with two activation functions shows that it is possible to design such an algorithm and still be able to compete with the algorithms that use a bias.

4.7 Further research

All the variations on the algorithm are tested in a simulated environment. This means, the results may not be representative for the performance of the algorithm in a real-world environment. Variations that perform better than the original algorithm might not work that well with the extra amount of noise observed in water. As observed in [8], in a more realistic simulation environment the location estimation accuracy got worse by a factor two and the angle estimation accuracy by a factor five. Experiments should be conducted to test the algorithms with data from a more advanced simulation environment or from real sensors.

The enhanced performance from the filters might not transfer to other problems. It is obvious that the signal shaped filters are suitable for this problem, but for problems where the input is for example an image, it makes less sense to have filters shaped like previously seen pictures.

In section 2, a lot of different possibilities to alter the algorithm are mentioned. In this research, different kinds of alteration are combined with each other and for each alteration the optima for the different kind of parameters was looked for. However, not every possible combination was tried and to know for sure the results can't be optimised further, it might be worth to implement and test the untried alterations in the algorithm. This could also mean to apply a filter which is made from only a part of a signal, for example the x- and y-values from only the first 8 sensors.

The choice to not use (square-root) pooling was partly made because the expectations were that the hidden layer would not become large enough to profit from the down scaling that pooling causes. Using a 2 x 2 orthogonal filter on a squared input (32 x 32 matrix), results in a 31 x 31 matrix in the hidden layer. Figures A.4a shows the results for this scenario upto 250 filters. This means there are 250 31 x 31 matrices in the hidden layer. This amount would make it interesting to test if adding a (square-root) pooling layer improves the result of a MSE of 32%.

For the signal shaped filters, the normalised observations were taken and used as filters. It might be that non-normalised filters cause a different behaviour of the algorithm, especially when the ReLu activation function is used and the activation in the hidden layer is not normalised.

In conclusion, it was shown that using signal shaped and orthogonal filters can improve the performance of an ELM. Using two different activation functions after using filters gave interesting results because each activation function had specific outputs it was better at to predict.

References

[1] Luuk H. Boulogne, Ben J. Wolf, Marco A. Wiering, and Sietse M. van Netten. Performance of neural networks for localizing moving objects with an artificial lateral line. *Bioinspiration & Biomimetics*, 12(5):056009, 2017.

[2] Branislava Ćurčić-Blake and Sietse M. van Netten. Source location encoding in the fish lateral line canal. *Journal of Experimental Biology*, 209(8):1548–1559, 2006. ISSN 0022-0949. doi: 10.1242/jeb.02140.

[3] S. Dijkgraaf. The functioning and significance of the lateral-line organs. *Biological Reviews 38*, 1:51–105, 1963.

[4] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, Apr 1970. ISSN 0945-3245. doi: 10.1007/BF02163027.

[5] G. B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, 10(2):18–29, May 2015. ISSN 1556-603X. doi: 10.1109/MCI.2015.2405316.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, volume 2, pages 985–990 vol.2, July 2004. doi: 10.1109/IJCNN.2004.1380068.

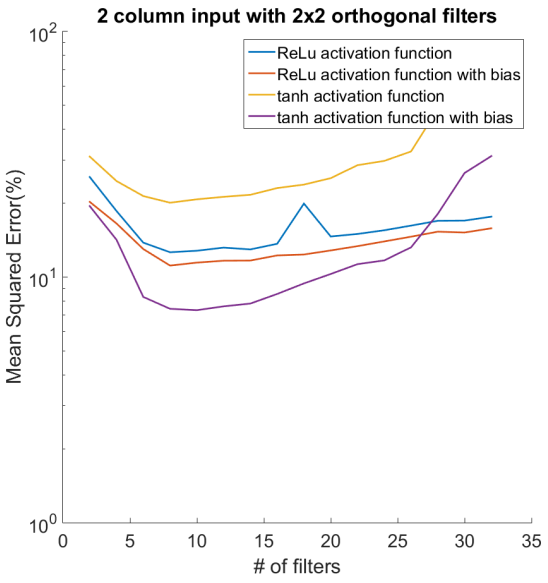
[7] Liyanaarachchi Kasun, Hongming Zhou, G.-B Huang, and Chi-Man Vong. Representational learning with elms for big data. 28:31–34, 11 2013.

[8] Jonathan Reid. Source detection performance comparison between potential and turbulent flow. *Bachelor's Thesis, Artificial Intelligence*, 2018.

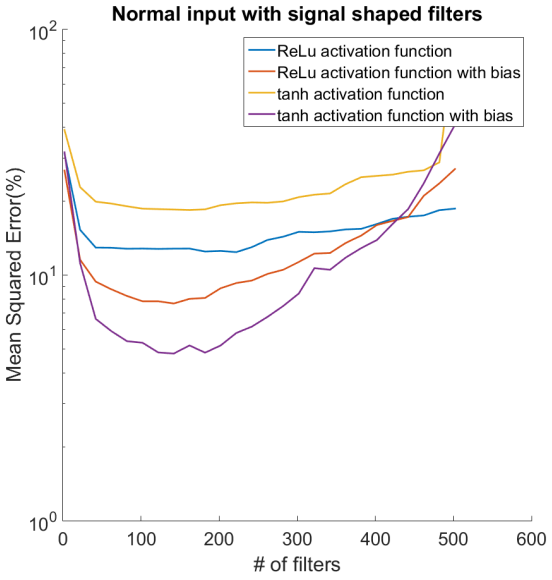
[9] Andrew M. Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, and Bipin Suresh. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1089–1096, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5.

[10] Jian Zhang, Jian Yang, Jianjun Qian, and Jiawei Xu. Nearest orthogonal matrix representation for face recognition. *Neurocomputing*, 151:471 – 480, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2014.09.019.

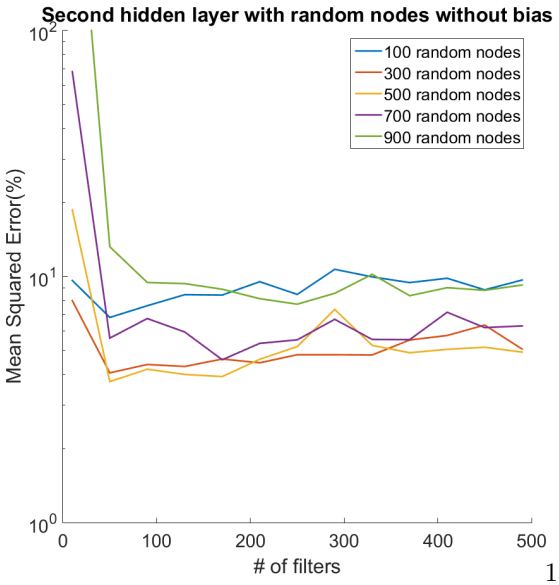
A Appendix



(a)

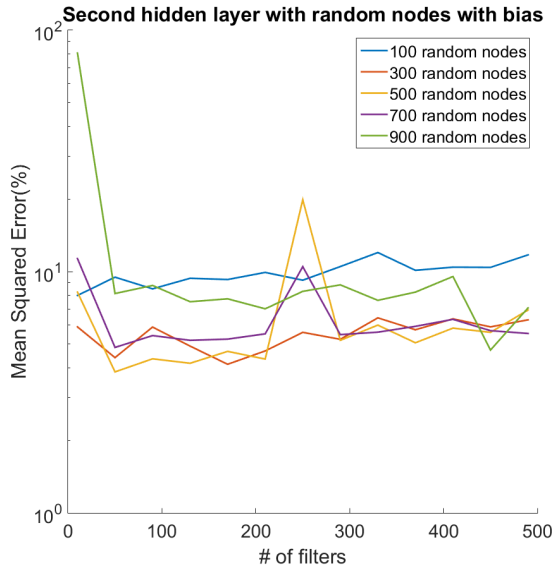


(b)

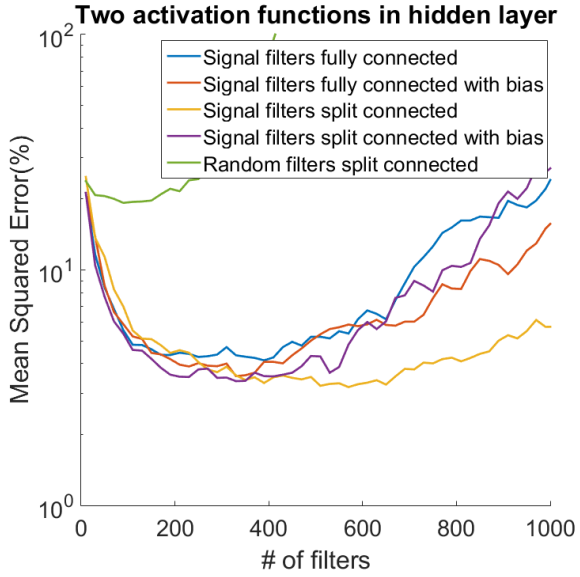


(c)

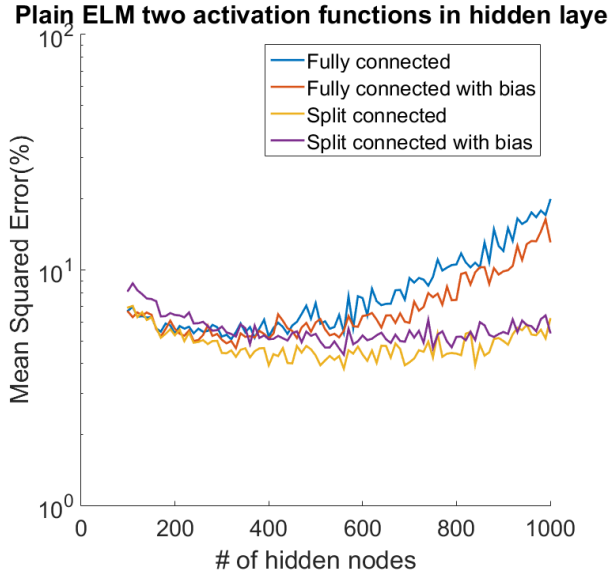
Figure A.1



(a)

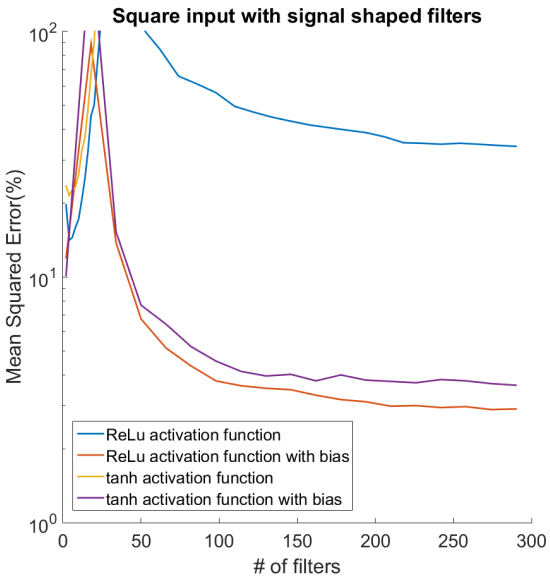


(b)

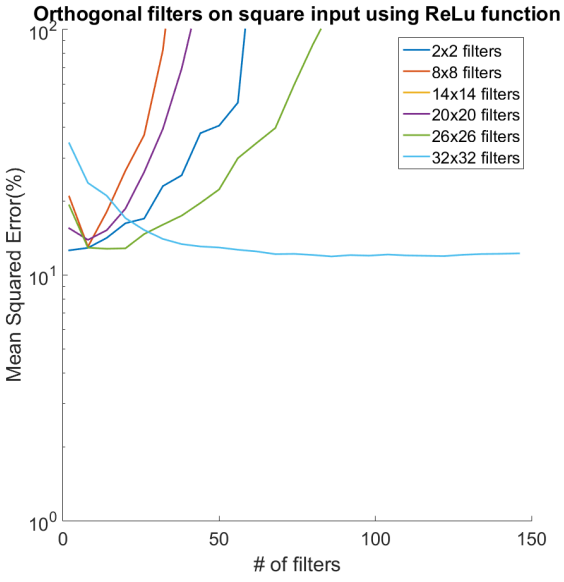


(c)

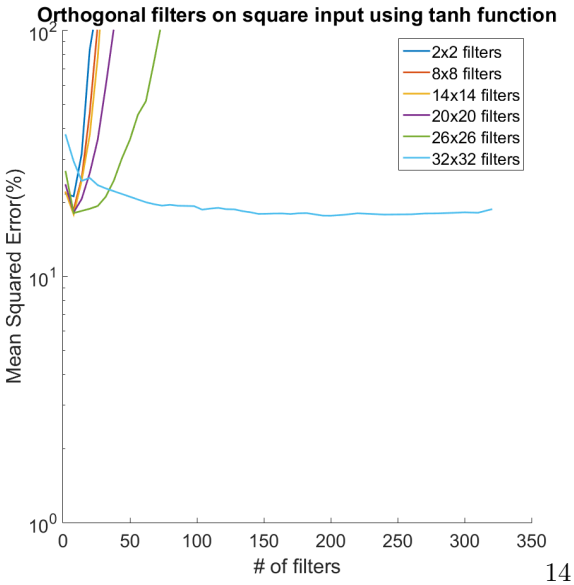
Figure A.2



(a)

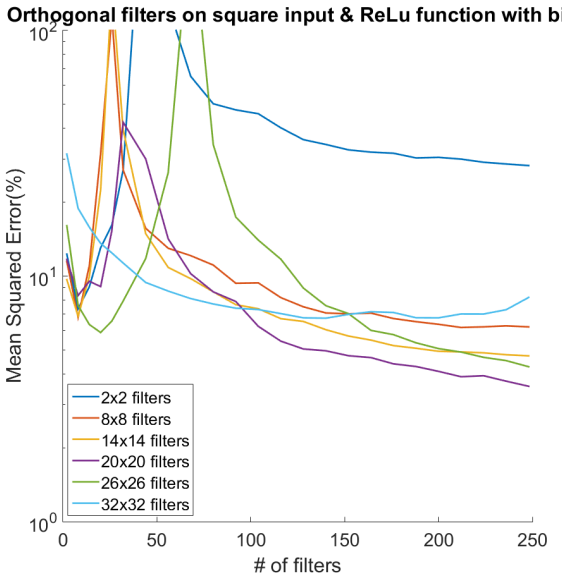


(b)

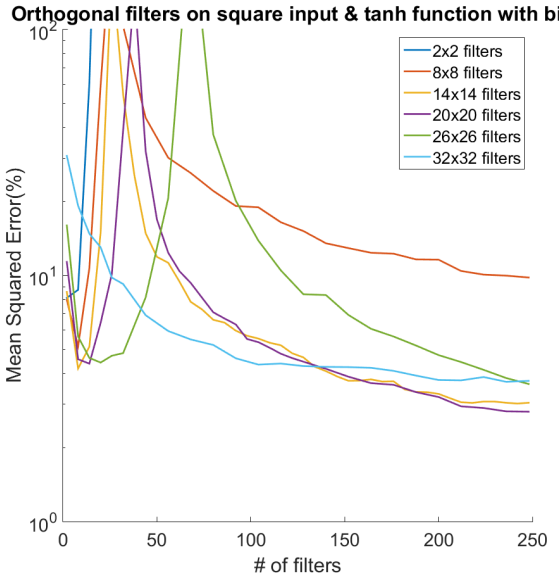


(c)

Figure A.3


Orthogonal filters on square input & ReLu function with bias
Mean Squared Error(%)
# of filters
2x2 filters
8x8 filters
14x14 filters
20x20 filters
26x26 filters
32x32 filters

(a)


Orthogonal filters on square input & tanh function with bias
Mean Squared Error(%)
# of filters
2x2 filters
8x8 filters
14x14 filters
20x20 filters
26x26 filters
32x32 filters

(b)

Figure A.4