university of groningen / faculty of science and engineering

Underwater object localization and identification using an extreme learning machine and artificial lateral line sensors

A comparative study on the performance of regression and classification

Arjen Brussen, s2695332, A.Brussen@student.rug.nl,
Supervisors: Dr S.M. van Netten, B.J. Wolf & Dr P. Pirih

Abstract: Fish possess a lateral line organ, consisting of flow detectors along the body of the fish. With this organ, they are able to detect water flow in surrounding waters. Using excitation patterns of these flow detectors, the location and direction of moving objects can be inferred. Using these findings as a basis, this paper shows if and how we can identify these objects in terms of shape, size and velocity using an artificial lateral line in combination with an extreme learning machine (ELM). Input normalization based on the maximum velocity outperforms other tested techniques. For shape identification, three object shapes were used (single object, school of fish, snake) and an identification accuracy of 66.3% is reported. For size identification (0.025m, 0.05m, 0.1m), the ELM achieves an overall accuracy of 68.3%. For velocity identification (0.065m/s, 0.13m/s,0.26m/s), an overall accuracy of 53.8% was measured.

1 Introduction

Some fish possess a lateral line organ. This organ gives fish the ability to detect object movement, vibration and pressure changes in surrounding waters [5]. The cells which make up the organ respond to the velocity of water flow hitting the organ. Fish can use this source of information to detect and hunt prey, escape predators, avoid objects and stay organized in a school of fish [11]. Having knowledge about the size, shape or potentially even the speed of nearby objects is vital for the fish in order to make effective decisions regarding their own velocity and direction.

The significance of this organ in biomimetics is explained by its highly effective short-range sonar-like ability to perform source localization. Progress has been made to implement an artificial lateral line (ALL) system using an array of underwater sensors. These capture incoming velocity patterns. Furthermore, the location [2] and direction of an object [14] can be determined by training a neural network on simulated data. There is, however, less research available about object identification.

Previous research has shown that, under the conditions of potential flow, dipole fluid flow models can be used to predict excitation patterns along an array of underwater sensors for objects vibrating in a direction with respect to the sensors [1, 3, 4, 8] or for objects moving in a specific direction [7].

The extreme learning machine (ELM) architecture from Huang et al. [12], and later its possible applications by Ding et al. [6], were used in the research done by Boulogne et al. [2]. They have shown that the ELM, in general, outperforms the Echo State Network (ESN) architecture and the Multilayer Perceptron (MLP) for underwater object localization, although the MLP performs best under high noise conditions. Objects can differ in shape, size or velocity and dipole fluid models have already shown that size and velocity significantly influence the captured velocity patterns as well (see also Formula 2.1). Using their implementation of an ELM as a base, we will find out whether the ELM is able to differentiate between different kinds of objects by means of classification.

To the best of our knowledge, there is very little research available about object identification using an ALL and an artificial neural network in 2D space. Herzog et al. [10], have done similar research with tap water flow, but not with still water,

which is used in this work. Therefore, this research is mostly exploratory.

The goal of this research is to provide more information about object identification. In Section 2, the simulation space, data generation, ELM, objects and possible normalization methods are discussed. The problems of object localization and object identification are also explained.

2 Methods

The simulation environment will be discussed first. After, the data generation methods and model parameters are explained. Then, the ELM is introduced along with the setup of the localization and identification tests. localization is used to validate the object types, to see if all object types can be localized in the same order of accuracy. identification is used to see whether the ELM can differentiate between object types. Finally, the normalization techniques are discussed.

2.1 Simulation environment

The simulation environment is a two-dimensional 2-by-1 area in a 3D volume, currently measured in meters. The sensor array consists of 16 sensors measuring x and y velocities (m/s), and is linearly spaced across $(-0.5, -0.1)$ and $(0.5, -0.1)$. See Figure 2.2 for an example of the simulation environment.

2.2 Data generation

The velocity profile data can be generated by two methods. Hermes [9] has implemented the 'grid' data generation method and Boulogne et al. [2] has implemented the 'path' data generation method. This research has added two generation methods which are based on the path generation method, but are more complex. These methods are described in Sections 2.2.1, 2.2.2, 2.5.2 and 2.5.3.

For each time-step in the simulation, the intensity vs. noise of the velocity patterns for each sensor is captured using Equation 2.1:

$$v_c = \frac{r^3}{d^3} * w \tag{2.1}$$

where r is the radius of the object, d is the distance between the object and the sensor and w is the velocity of the object.

2.2.1 Grid

In the grid generation method, the simulation environment is divided evenly into grid points with a set distance between each other. The object is then put on all these points one by one, vibrating in a certain direction. This prevents bias towards a certain area of the simulation, since the ELM is trained on each evenly-divided grid point. The method uses $k = 6$ orientations for each grid point as in Hermes [9], divided evenly between $[0, 2\pi]$. A training/test length equal to $k*$gridpoints is taken.

2.2.2 Path

The path generation method consists of a continuous environment where the object moves in a more natural fashion. Each time-step, it moves in its current direction and then chooses a bounded random new angle such that it stays within bounds. The object produces more possible bias, since the randomization allows the object to, in theory, stay in one specific side of the simulation environment. However, with a big enough training length and a statistical mean across the results of multiple runs, this issue is reduced. The path method uses a step size of 0.1m and a random direction change of maximally $\frac{\pi}{4}$. 10000 iterations for training is used and 2000 for testing/validating, separately.

2.3 Model parameters

The object in the simulation environment can differ in shape (see Section 2.5), radius and velocity. Unless explicitly mentioned otherwise, the object's shape is approximated by a single object, its radius is 0.05m and the instantaneous velocity of the object is 0.13m/s.

According to Wolf and van Netten [14], the lowest sampled velocities using these settings are in the order of 10^{-6} m/s, so this noise level will be used. Higher noise levels would make the ELM train on the noise level, instead of the velocity patterns, while lower noise levels would decrease the noise robustness of the ELM. Using different noise levels

would also not allow us to classify in the whole area of interest.

2.4 Extreme Learning Machine

An ELM is a single layer feedforward neural network. It consists of the input layer, one hidden layer and the output layer. The weights from the input layer to the hidden layer are fully-connected, initialized randomly and not altered further. The weights from the hidden layer to the output layer are fully-connected as well and trained with the activation function $f(x) = \tanh(x)$. Section 2.6.3 contains more information about this choice.

To allow the ELM to learn, every layer contains a bias node with a fixed value of 1.

To prevent overfitting and to monitor the performance of the ELM, we calculate the Mean Squared Error (MSE) over the entire dataset. The MSE is calculated using Formula 2.2. L is the length of the array, M is the number of samples in the dataset, D is the amount of output samples. Overfitting can then be detected by comparing the MSE of the training set to the MSE of the testing set. If the training MSE is significantly lower than the testing MSE, overfitting is present and the model or ELM parameters should be adjusted accordingly.

$$\text{MSE} = \frac{1}{L^2 M} \sum_{n=1}^{M} \frac{1}{D} \sum_{i=1}^{D} (t_i(n) - o_i(n))^2 \qquad (2.2)$$

Additionally, for localization, the Mean Euclidean Distance (MED) (Formula 2.3) is also reported to give a more intuitive error for the distance.

$$\text{MED} = \frac{1}{LM} \sum_{n=1}^{M} \sqrt{\sum_{i=1}^{D} (t_i(n) - o_i(n))^2} \qquad (2.3)$$

The two formulas above are after Boulogne et al. [2].

2.4.1 Hyper-parameter optimization

The only hyper-parameters to be optimized in this ELM is the hidden layer size. For each specific problem, a repeated 6-fold validation is performed N times to monitor the MSE. The hidden layer sizes are chosen in a logarithmic fashion, with smaller step sizes near the local minima found in the logarithmic sweep.

2.4.2 Localization

For localization, the output will be the location of the object, encoded as x and y coordinates, and the direction of the object, encoded as $\cos\theta$ and $\sin\theta$.

Since object shape belongs to the model parameters and not the hyperparameters, the ELM should be able to predict the location and direction of each new object shape about as well as the standard single object, as will be explained in Section 2.5.1. Therefore, the ELM is trained on each different shape and is tested on the 'path' generation method. For each localization test, the ELM hidden layer size is set to 240 units. The results can be seen in Section 3.1.2.

2.4.3 Identification

The ELM implementation focused on identification will use one-hot encoding. Every unique object is assigned their own output slot.

Three possible values will be taken for each specific identification problem. For shape identification (see Section 2.5), the ELM has to differentiate between a single, school and snake object. For size identification, the ELM has to differentiate between the radii 0.025m, 0.05m and 0.1m on a single object. For velocity identification, the ELM has to differentiate between the instantaneous velocities 0.065m/s, 0.13m/s and 0.26m/s on a single object.

Furthermore, an experiment has been designed where all possibilities are combined ($3^3 = 27$ objects). In this experiment, the ELM is tested on shape identification.

The results can be found in Section 3.2.

2.5 Objects

This research uses three different object types: a singular object, a school of smaller objects and a snake object. The last two are more complex than the singular object. For each object type, the setup and their expected output are explained. Finally a single captured velocity profile for each object type can be seen in Figure 2.1. Visualization of the location and angle of the objects which generated the velocity profile can be seen in Figures 2.2 and 2.3.

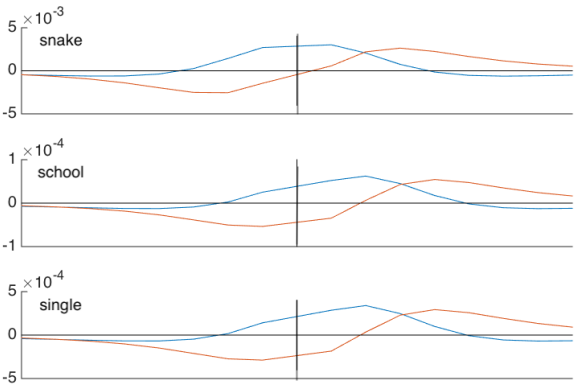

Figure 2.1: An example velocity profile for a snake, school and single object, respectively. Mean object location and angle are $x = 0.09$, $y = 0.19$ and $\theta = \pi$. Blue line indicates x-velocity, red line indicates y-velocity. Note the different order of magnitude.

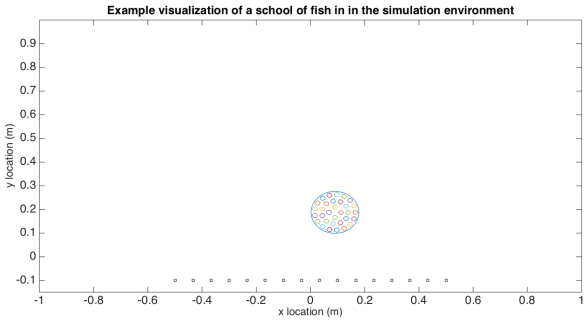

Figure 2.2: Example of a 'school' object in the middle of the simulation environment. The outer object is the reference object. The 'school' moves as a disk through the simulation

2.5.1 Single object

The default case is a singular object moving through the simulation environment with a certain radius. It works with either the grid or path generation method. The expected output is the true location and direction of the object.

2.5.2 School of objects

Working with the path generation method, there are multiple smaller objects moving together in the same direction packed in a larger reference circle.

These smaller objects are positioned using the optimization problem of circle packing*. circle packing finds the biggest amount and relative location of smaller circles in a larger circle such that the area is maximally filled, given the radius of the smaller and larger object. A maximally dense school of objects is not realistic, however, since a school of fish allows for inter-fish movement, which requires free space around each fish. By using a larger radius for the smaller objects when finding their positions, we can create a more natural density if we then place relatively smaller objects at the actually generated positions.

The result is a very simplistic version of a school of fish (see Figure 2.2). We can sum the produced input of all smaller objects together, due to the linearity of potential flow. Since circle packing with objects of the same radius prefers symmetry, the large reference object is taken as the true location and direction.

*https://en.wikipedia.org/wiki/Circle_packing_in_a_circle

2.5.3 Snake object

Also working with the path generation method, a line of objects is used, resembling an snake, as seen in Figure 2.3. The line moves in the same fashion as a single object, and thus produces a visualization resembling the game Snake. The segment in the front updates its location and angle, and each subsequent segment updates their own location and angle according to the first segment. The inter-segment distance is calculated such that the objects barely touch each other. The expected output values here are the means of the combined expected output values of all segments, producing the centroid.

2.6 Data Normalization

The dynamic range of the sampled velocity patterns are considerably large. This is explained by the cubed relationship of the distance between the source and the sensors as seen in Formula 2.1. In order to map this dynamic range and make the process of localization robust to variance in amplitude, data normalization techniques are explored in order to find out if certain techniques were better suited for localization, identification or the combination of them. All techniques ex-

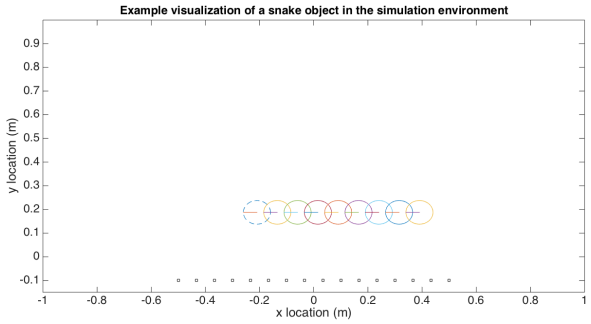

Figure 2.3: Example of a 'snake' object after several iterations. The lines in each object indicate the current direction.

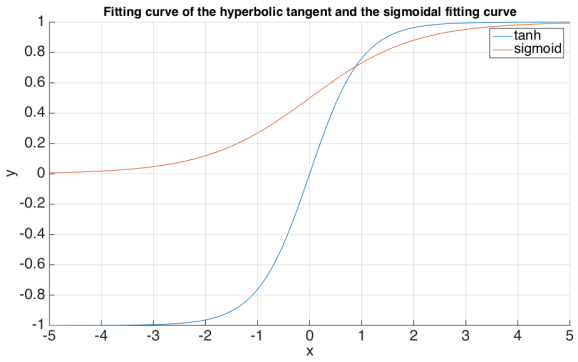

Figure 2.4: Hyperbolic tangent and sigmoidal fitting curve for $[-5, 5]$

cept for velocity normalization use input scaling, where the input is multiplied with the scaling sequence $[100, 200, 500, 1000, 2000, 5000, 10000]$ before data normalization. The scaled data is concatenated vertically such that every time-step now has $16 * 2 * 7 = 224$ input velocities. For all normalization techniques, lower scalings (i.e. lower values) gives more erratic normalized output, while higher scaling gives more information loss for the relatively lower scalings. To determine the normalization technique used in all other tests, 10 localization runs are performed for each normalization technique. Training is done with the grid method, testing on the path method. The mean over those 10 runs is taken and can be seen in Figure 3.1.

2.6.1 Velocity normalization

Velocity normalization scales the input in the range $[-1, 1]$ by dividing each x- and y-velocity by the maximum absolute velocity encountered of the measured velocity profile each time-step [2]:

$$\text{new } q[n] = \frac{q[n]}{\max(q)} \tag{2.4}$$

This method allows the information to keep its spatial scaling intact. It will be the performance baseline for the other normalization techniques, because it had a positive effect on the work done by Boulogne et al. [2], and Wolf and van Netten [14].

2.6.2 Clip normalization

Clip normalization uses a hard cut-off range $[-1, 1]$ where the values must be in. If there are values outside the given range, they are clipped back to the closest in-range value.

Clipping had a positive effect on the performance in the work done by van de Wolfshaar et al. [13].

2.6.3 Activation Functions

Two popular functions called hyperbolic tangent (tanh) and sigmoidal (sigmoid) were chosen as possible activation functions. Figure 2.4 show the fitting curve of these functions. The hyperbolic tangent shows a much steeper curve for lower velocities. Since the input is already normalized when it reaches the activation function, a steeper curve might be more favourable, because the data becomes more diverse.

3 Results

First, the results of the normalization techniques tested are shown. All other results and tests are performed with the best performing normalization technique. Then, the comparison between different object types for localization is shown. After, for every identification problem described in Section 2.4.3, the results are shown, represented in a confusion matrix, together with a parameter sweep for the hidden layer size.

3.1 Localization

3.1.1 Normalization

The results from the test described in Section 2.6. Velocity normalization outperforms every other method for predicting location and angle for localization problems.

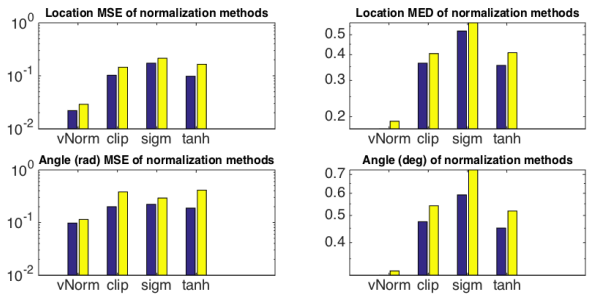

Figure 3.1: Localization errors of the ELM using different data normalization techniques. All but vNorm used input scaling. Logarithmic axes were used. The blue bar indicates the training MSE, while the yellow bar indicates the testing MSE. ($N = 10$)

3.1.2 Comparison of object shapes

The results from the test described in Section 2.4.2. No overfitting is present, as the training error is not significantly lower than the testing error in any of the errors. Furthermore, for every object type, the error is in the same order of magnitude.

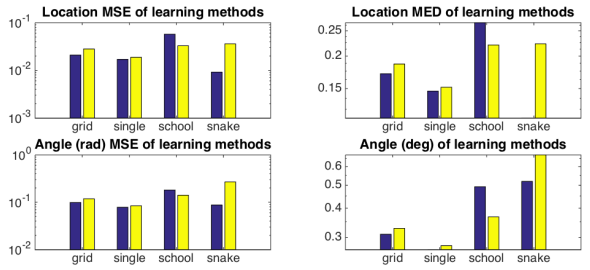

Figure 3.2: A comparison of localization errors of the ELM for different training object types. Logarithmic axes were used. The blue bar indicates the training MSE, while the yellow bar indicates the testing MSE. ($N = 10$)

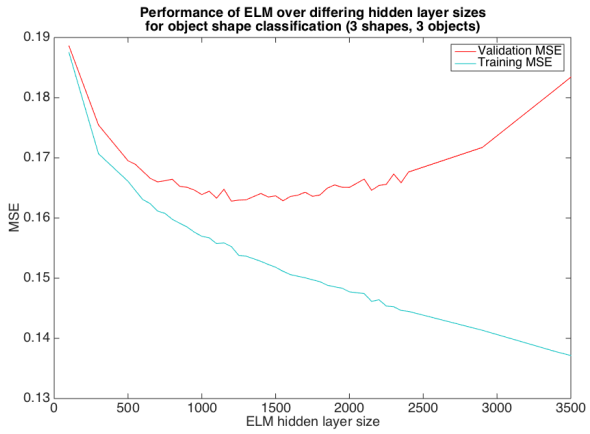

Figure 3.3: parameter sweep for the hidden layer size of the ELM using one-hot encoding for object shape with 3 objects. ($N = 6$)

3.2 Identification

3.2.1 Different shapes

A parameter sweep for the hidden layer size has been conducted, with the result shown in Figure 3.3. The lowest MSE corresponds with a hidden layer size of 1200.

Figure 3.4 shows the results of a identification run with the optimal hidden layer size in the form of a confusion matrix. Here, each green/red tile shows the percentage relative to the entire dataset. This means that perfect identification would result in a green tile percentage of 33.3% and a red tile percentage of 0%. Summaries for each identification possibility can be seen in the grey/blue tiles.

If we look at the grey tile of the 'single' *column*, we can see that 40.6% of the objects that should have been classified as 'single' are actually classified as 'single', while 59.4% of the objects are not classified as 'single' (false-negatives).

If we look at the grey tile of the 'single' *row*, we can see that 59.1% of the objects that are actually classified as 'single' are correct identifications, while 40.9% are incorrect identifications (false-positives).

A total accuracy of 66.3% has been achieved, with the weakest and strongest being 'path' and 'school', respectively. Most false-negatives are predicted as 'snake', while it should have been 'single'.

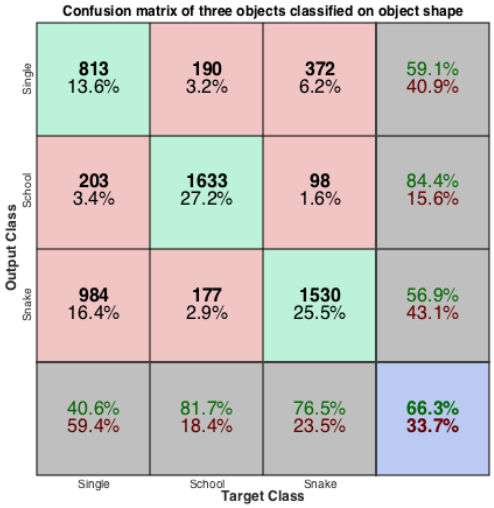

Figure 3.4: identification accuracy of the ELM using one-hot encoding for object shape with 3 objects.

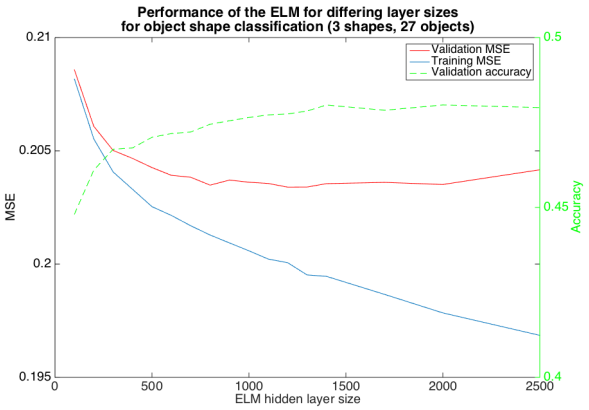

Figure 3.5: parameter sweep for the hidden layer size of the ELM using one-hot encoding for object shape with 27 objects. 5,000 train samples and 1,000 test samples per velocity pattern type. ($N = 8$)

3.2.2 Different shapes, many objects

For the 27 objects setup, the parameter sweep for the hidden layer size can be seen in Figure 3.5. A maximum hidden layer size of 2500 has been used in the parameter sweep, because of the relatively long computation time this setup comes with and due to the flat performance line in Figure 3.5. The optimal hidden layer size might be higher than this, but the computation time far outshines the relative increase in accuracy. The results of a single run can be seen in Figure 3.6 with a hidden layer size of 1500. With a training length of 10.000 time-steps and test length of 2000 time-steps per object the run-time exceeded four hours on a single core. An overall accuracy of 48.1% has been achieved, with the 'single' object being classified incorrectly 81.7% of the time.

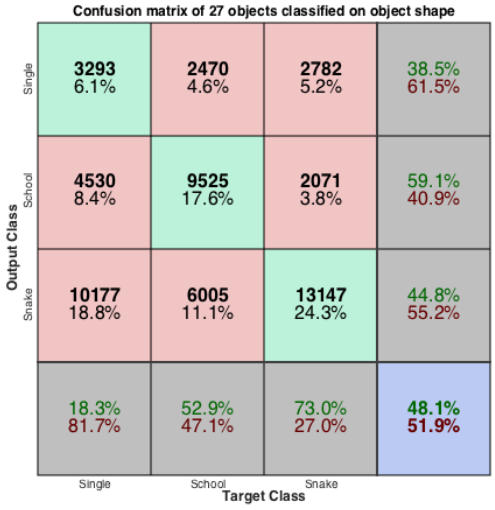

Figure 3.6: Identification accuracy of the ELM using one-hot encoding for object shape with 27 objects.

3.2.3 Different sizes

Figure 3.7 shows the parameter sweep for the hidden layer size for identification with 3 objects of different sizes. The validation accuracy is also shown, since it still increases when the MSE also increases. The optimal hidden layer size is at 1400 hidden layer units. An overall identification accuracy of

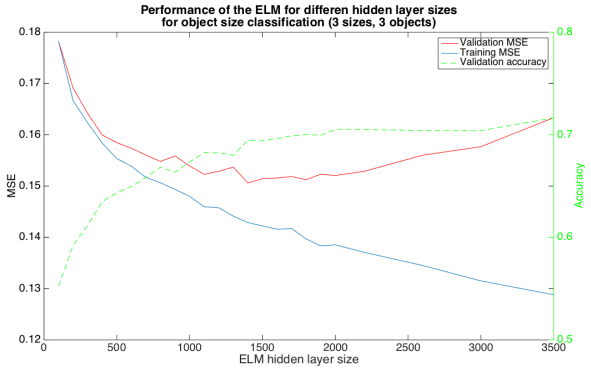

Figure 3.7: parameter sweep for the hidden layer size of the ELM using one-hot encoding for three objects with radius 0.025m 0.05m and 0.1m, respectively. ($N = 6$)

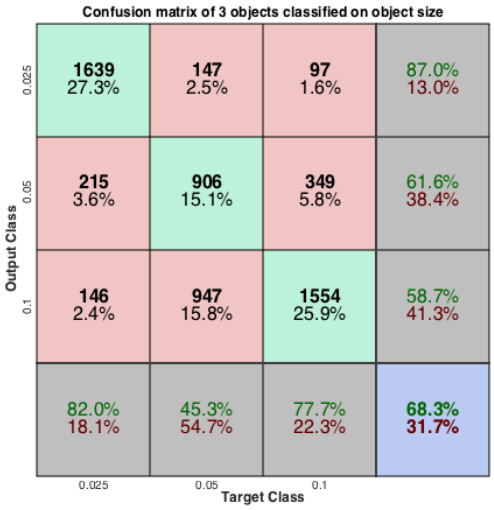

Figure 3.8: Identification accuracy of the ELM using one-hot encoding for object size with 3 objects.

68.3% is reached, as seen in Figure 3.8. The size of 0.05m is classified incorrectly the most. The ELM confuses the size with the third size (0.1m) in about half of the cases.

3.2.4 Different velocities

Figure 3.9 shows the parameter sweep for the hidden layer size for identification with 3 objects with different velocities. The minimum validation MSE

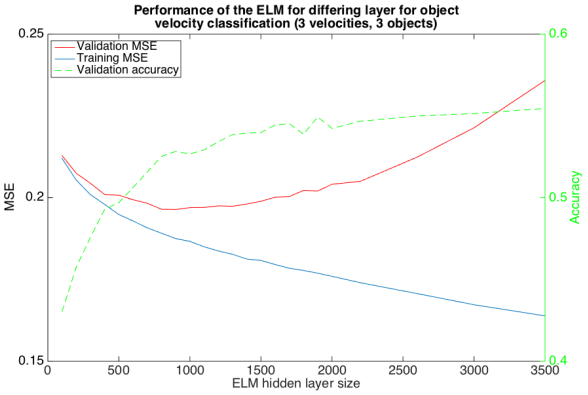

Figure 3.9: parameter sweep for the hidden layer size of the ELM using one-hot encoding for three objects with velocities 0.065m/s 0.13m/s and 0.26m/s, respectively. ($N = 6$)

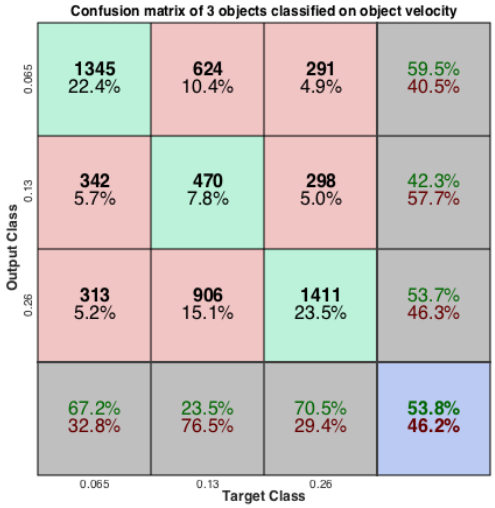

Figure 3.10: Identification accuracy of the ELM using one-hot encoding for three objects with velocities 0.065m/s 0.13m/s and 0.26m/s, respectively.

is found at a hidden layer size of 900 units. Figure 3.10 shows the identification accuracy per velocity, with an overall accuracy of 53.8%. The ELM has trouble classifying objects with an average velocity (0.13m/s), with an accuracy of only 23.5%.

4 Discussion

4.1 Velocity profiles

All velocity profiles, as seen in Figure 2.1, seem to have the same general shape. The snake object, however, is one order of magnitude higher for the intensity. This is explained by the size of the objects, which is further explained in 4.3.1.

4.2 Localization

4.2.1 Comparison of object shapes

The single object shows to be the easiest for determining the direction. For the location, however, all shapes seem to perform in the same order of magnitude. This allows us to conclude that the ELM is able to predict location and direction for the new objects shapes equally well compared to the single object.

4.3 Identification

We compare the results to the random chance as a baseline. This means for the confusion matrices, each identification possibility has a baseline of 33%.

4.3.1 Object shape

It is clear from the results that the single object has the most trouble being classified correctly. This might have several reasons. First of all, the combined size of the snake-like object is n times larger than the size of a single object, where n represents the amount of segments in the line (see Figure 2.3 for an example). This means that the combined radius is also larger and this influences the intensity of the velocity patterns generated by Equation 2.1. The school object does, however, have the same combined size as the single object. In the results, we can see that the school object is classified rather well (e.g. 81.7% in Figure 3.4). This suggests that the issue might be somewhere else.

A second possible explanation is the genericness of the 'single' object. The school and snake objects consist of multiple objects, which are all at slightly different locations in the simulation each time-step. The shape of these objects allow for unique velocity profiles, but the shape of a noisy 'single' object velocity profile might be generic enough that it sometimes passes as one of these objects in simulation in terms of generated velocities.

For the shape identification with 27 different objects, all differing in shape, size and velocity, it is noteworthy that the snake-like still performs well, even though the rest performs significantly worse. This might be explained by the first reason listed above about the inherent difference in size.

4.3.2 Object size

The second size used in the test (0.05m) performs worst. The ELM confuses the size with the third size (0.1m) in about half of the cases. This may be, because the normalization technique used (velocity normalization) scales the velocity profiles based on the maximum velocity encountered per profile. For the low size, the normalized velocity profiles may be unique, while the normalized velocity profiles for higher sizes may share too much resemblance.

4.3.3 Object speed

The linear relationship in Equation 2.1 seems to have a significant effect on the performance of the ELM for velocity identification. It is strange that the velocity of 0.13m/s is significantly worse at being classified correctly than the other velocities, although it might share the same reasons as the issue discussed in Section 4.3.2. The other velocities, however, are far above the baseline accuracy and therefore show a relationship between velocity and the generated velocity pattern.

4.4 Future work

The issue with the current abstract implementation of the simulation is that viscosity and wavelet refraction is not taken into account. This keeps things relatively simple to work with and gives better computation times, but also provides unrealistic readings (e.g. a long enough snake object could clip itself).

Changing the combined size of the snake-like object is something that could be done in order to give a more fair comparison between objects.

Furthermore, the school object could be changed to simulate a more realistic school of fish. Currently, every fish moves in exactly the same direction and speed, and has the same size. The only difference is

the offset in location from the center of the school. In a more realistic setting, there is inter-fish movement. A difference in velocity (i.e. acceleration) and location for each fish should give more dynamic and unique data.

To expand on the localization part of this research, a function was added such that the velocity could differ over time. This simulated acceleration is implemented, but there was no time to do further research on this. It would be interesting to see if the ELM could predict the velocity of an object with a certain acceleration. The velocity function could also be used to see if the ELM could classify accelerating, decelerating and non-accelerating objects correctly.

The path data generation method currently works with a movement type where the object can change its direction each time-step by some bounded limit. Simulating different movement type, such as a changing step size would allow for more unique data patterns to emerge from the simulation.

Finally, a more straightforward idea is to change the normalization technique used in identification. Although input scaling did not work well for localization, it might perform well on identification, as it shows difference in data when objects are scaled differently for some parameter.

4.5 Conclusion

The similar shape in the velocity profiles for each object is promising for future research, as it suggests that different object shapes still produce the same general velocity pattern.

There is, in general, a relationship between object shape, velocity and shape, and the generated velocity pattern. In any case, the ELM performs better than the baseline, so if an object would be classified over a time period, the ELM would almost always be correct.

References

[1] Abdulsadda, A. T. and Tan, X. (2013). Underwater tracking of a moving dipole source using an artificial lateral line: algorithm and experimental validation with ionic polymer–metal composite flow sensors. *Smart Materials and Structures*, 22(4):045010.

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

[3] Ćurčić-Blake, B. and van Netten, S. M. (2006). Source location encoding in the fish lateral line canal. *Journal of Experimental Biology*, 209(8):1548–1559.

[4] Dagamseh, A., Lammerink, T. S., Kolster, M., Bruinink, C., Wiegerink, R. J., and Krijnen, G. J. (2010). Dipole-source localization using biomimetic flow-sensor arrays positioned as lateral-line system. *Sensors and actuators A: Physical*, 162(2):355–360.

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

[6] Ding, S., Xu, X., and Nie, R. (2014). Extreme learning machine and its applications. *Neural Computing and Applications*, 25(3):549–556.

[7] Franosch, J.-M. P., Sichert, A. B., Suttner, M. D., and Van Hemmen, J. L. (2005). Estimating position and velocity of a submerged moving object by the clawed frog xenopus and by fisha cybernetic approach. *Biological cybernetics*, 93(4):231–238.

[8] Goulet, J., Engelmann, J., Chagnaud, B. P., Franosch, J.-M. P., Suttner, M. D., and Van Hemmen, J. L. (2008). Object localization through the lateral line system of fish: theory and experiment. *Journal of Comparative Physiology A*, 194(1):1–17.

[9] Hermes, R. (2017). Source localization using a artificial lateral line measuring 2d velocity profiles. *Bachelor Project Artificial Intelligence*.

[10] Herzog, H., Steltenkamp, S., Klein, A., Tätzner, S., Schulze, E., and Bleckmann, H. (2015). Micro-machined flow sensors mimicking lateral line canal neuromasts. *Micromachines*, 6(8):1189–1212.

[11] Hoekstra, D. and Janssen, J. (1985). Non-visual feeding behavior of the mottled sculpin, cottus bairdi, in lake michigan. *Environmental biology of fishes*, 12(2):111–117.

[12] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2004). 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.

[13] van de Wolfshaar, J., Wolf, B. J., and van Netten, S. M. (2018). Localizing multiple moving underwater objects using an artificial lateral line and deep convolutional neural networks. *Manuscript in preparation.*

[14] Wolf, B. J. and van Netten, S. M. (2018). Training submerged source detection for a 2d fluid flow sensor array with extreme learning machines. Manuscript submitted for publication.