| | | | |
|---|---|---|---|
| **Exercise Sheet** | 04 | **1. Team Partner:** | Lennart Slusny |
| **Task** | 01 | **2. Team Partner:** | Joschka Strüber |

Based on the Differential Evolution Example, we implemented an echo state network with differential evolution. The model is implemented using a three-layer recurrent neural network for three inputs, a small reservoir and three outputs for the three-dimensional coordinates of the target.

For training, we collected a data set of about 2400 consecutive data points. Surprisingly, the model is fairly stable for a wide variety of hyper parameters. Using a smaller reservoir of 30 or less neurons also works, but it requires a larger amount of training steps. The same appears to be even more true, if using a smaller training set. While the cost of each iteration is lower, the amount of steps to reach good results, is much higher.

If the training set is too short, the RMSE also increases significantly and we recommend to use at least 1000 training points. For example, using 500 instead of 1600 training steps for our final parameters, increases the RMSE by one order of magnitude. In practice, this model still performs good and there is no visual difference compared to the one performing better on paper.

One of the most interesting parameters is the feedback scaling that greatly decreases the learned output weights. We were able to get good results for any values between $10^{-8}$ and $10^{-12}$. If we increased it too much, e.g. $10^{-7}$, we were still able to get good RMSE values on the test set, but for some reason the predicted trajectories were both wrong and unstable.

When the hyper parameters are chosen sensibly, the neural evolution algorithm finds a working set of weights very quickly, often getting solid results in ten iterations or less. Usually, a few hundred training iterations are more than enough to reach very good results. Training for much longer (500, 1000 or 10,000 iterations) has diminishing returns and the chance to find a very slightly better model decreases quickly.

Our final model was trained using 300 washout steps, 1600 for training and 400 for testing. The reservoir size is 50 and the feedback scaling $10^{-9}$. 1000 optimization steps were used, but 100 probably would have been sufficient as well. The model reaches an RMSE of about $3 \cdot 10^{-4}$ on the test set.

The trajectory projection simply uses a copy of the pretrained model and computes the next 100 predicted time steps as absolute coordinates. Taking a copy instead of the original network is important, because by computing

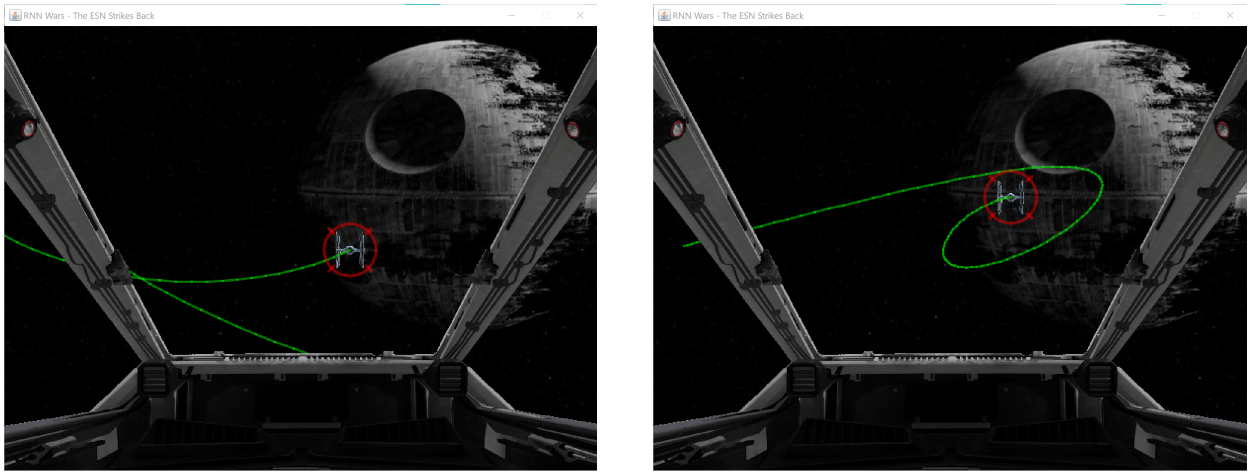


Figure 1: The expected trajectories during and after washout.

100 forward passes, we change the model's activations. Continuing with those activations at the next simulation step, would return the wrong results, because the model expects to be 100 time steps in advance. For this reason, we copy the activations of the model to the copy, which is used for the projection.

After locking on the target, the trajectory starts to be unstable in the beginning. This is to be expected, because we are essentially in the washout stage of the ESN. This can be seen in Figure 1 on the left. After a few hundred time steps, the predictions stabilize and remain basically unchanged after each simulation step, as seen on the right.

Based on visual confirmation, the predicted trajectories are very good and the TIE fighter is at the expected position at each time step. The provided RNN to steer the missiles is not perfect, but we are able to hit the space ship in about 90 to 95% of cases.