Freie Universität Berlin

Master's Thesis at the Department for Informatics and Mathematics

Dahlem Center for Machine Learning and Robotics - Biorobotics Lab

Imitation learning of fish and swarm behavior with Recurrent Neural Networks

Helmut Moritz Maxeiner
Matriculation Number: 4473930
moritz.maxeiner@fu-berlin.de

Supervisor:	Prof. Dr. Tim Landgraf
First Examiner:	Prof. Dr. Tim Landgraf
Second Examiner:	Prof. Dr. Dr. (h.c.) habil. Raúl Rojas

Berlin, September 23, 2019

Abstract

In the field of collective behavior group-level phenomena emerge from interactions between individuals. To study inter-individual rules the Landgraf lab has built a robotic guppy that replaces live animals in the shoal, RoboFish. The primary purpose of this thesis is to examine if the pair interaction behavior of female guppies can be learned by recurrent neural networks via supervised learning and to develop the software components required to have the RoboFish system run the resulting models in the real world. Two distinct datasets are studied and RNN models trained to try to imitate the behavior seen in them: One dataset was synthetically generated from a simple deterministic model as a baseline and one was captured from live fish. Training different kinds of RNNs on the datasets revealed a capability of a simple stacked RNN to learn swarm behavior, further improved upon by putting a ConvLSTM input network in front of it. I was also successful in showing that a more complex network was able to learn some basic interaction behavior as seen in real fish.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

September 23, 2019

Helmut Moritz Maxeiner

Contents

1 Introduction **1**

2 Related Work **2**

3 Infrastructure and model **3**
- 3.1 Preexisting infrastructure . . . 3
- 3.2 Model . . . 3
  - 3.2.1 Locomotion . . . 5
  - 3.2.2 View of agents . . . 5
  - 3.2.3 View of walls . . . 6
- 3.3 Required modifications . . . 7

4 Data acquisition **9**
- 4.1 Synthetic data . . . 9
- 4.2 Live data . . . 9
  - 4.2.1 Organism and video capture . . . 9
  - 4.2.2 Pose extraction . . . 12
  - 4.2.3 Finalization . . . 13

5 Experiments **16**
- 5.1 Behavior quantification . . . 16
- 5.2 Network training . . . 16
- 5.3 Couzin torus behavior . . . 17
- 5.4 Live fish behavior . . . 25

6 Conclusions **37**
- 6.1 Gaussian mixtures . . . 37
- 6.2 No locomotion input . . . 37
- 6.3 Outline-based raycasting . . . 37
- 6.4 Attention-based models . . . 37

List of Abbreviations **39**

List of Figures **40**

Bibliography **41**

A Acknowledgements **43**

B Appendix **44**

1 Introduction

Collective behavior in animals appears to show common characteristics across many different species [20]. It is thus not unreasonable to assume that research into the behavior of one species showing collective behavior, such as guppy fish, can lead to a better understanding of collective behavior in general and potentially provide insights into the behavior of other collective species that we may not be able to study in a laboratory setting, e.g. due to ethical considerations.

Traditional investigations into collective animal behavior have been coined by minimalistic models with rules handcrafted by researchers in order to see what the least complex model would be that can capture the characteristics of what makes the observed behavior collective (see [9], [14], [20]).

These models have the benefit of being easy to understand and implement, but while they have been successful in approximating characteristic swarm behaviors seen in many different species (see [20]), they have so far not been able to account for individual-specific behavior. The principal reason for that is that there are a multitude of sources for variations in behavior: Personality differences between animals and physiological conditions such as hunger or fatigue, for example. How these affect the rules underlying animal behavior (both on an individual level as well as on the collective result) is not understood.

The still continuing advances in computational power over the last decade make another approach ever more feasible, however: Handcraft only a few peripheral components and have the central part(s) of your model be parameterized by a neural network. Whether you then choose to use traditional supervised learning methods or opt for more novel approaches such as reinforcement learning, it is still easier to try to transfer your results to other lines of inquiry (e.g. to different species).

In this thesis I aimed to apply these advances to the study of both swarm and fish behavior. In section 2, I first review similar work and clarify where and how this thesis differs from it. After that in section 3, I take stock of the state of the *RoboFish* system[1] as it pertains to this thesis, what kind of model I designed to fit with it, and what the most important modifications to it were that needed to be made. Section 4 then explains how data for the swarm model was generated and how data from live guppy fish (poecilia reticulata) was captured and processed. Following that section 5 details the supervised learning experiments performed and presents, as well as discusses, their results. Finally, section 6 provides a summary of the results and an outlook over possible improvements and further experiments.

[1]Fish tank with observation camera, two-wheel robot pulling in-tank fish dummy from below via magnetic coupling, and associated software

2 Related Work

There have been several different approaches for predicting fish locomotion.

In Eyjolfsdottir et al., 2016 the authors applied GRU RNN networks to fly locomotion and handwriting prediction. As this publication was the principal motivator behind my thesis it is unsurprising that I adopted many of their design decisions, such as the raycasting based view vectors, feeding the model its the locomotion from the previous time step as an explicit input, and the discriminator/generator network architecture. The key difference is that fish in general are inherently more complex organisms than flies.

In Khan and Zhang, 2017 the authors used an LSTM RNN to predict fish locomotion. In their model, however, the locomotion is based on an eight cell grid projected around the agent's current position; the agent can only move from its current position into one of those cells. The agent's orientation is not being modelled. Its network input also only models wall distance and current fish heading, which means it cannot be used to model fish interaction.

In Girdhar et al., 2015 the authors constructed a backbone/spline model for zebrafish and successfully trained a custom neural network model via a genetic algorithm to reproduce the same postural space as live zebrafish. Their model, however, does not attempt to learn interaction between fish, the training data is always of a fish swimming alone.

In Cazenille et al., 2017 the authors trained a custom stochastic model on zebrafish and used a design similar to *RoboFish* to show how a dummy fish controlled by that model was able to move together with groups of live zebrafish without it being apparently rejected by them.

In Iizuka et al., 2018 the authors trained a three layer feedforward neural network with ReLU cells on lampeye fish data, which was then able to reproduce the same kind of swarm behavior seen in live lampeye fish. Their sensory input, however, consists of the full, distance-sorted pose information for the three nearest neighbor fishes, which makes the task to learn significantly easier.

3 Infrastructure and model

This section describes the relevant parts of the preexisting RoboFish infrastructure, the model chosen for supervised learning, and any resulting required modifications to the infrastructure.

3.1 Preexisting infrastructure

The RoboFish system (see [19]) consists of an obstacle-free $100\ cm \times 100\ cm$ fish tank in which live fish and fish dummies can be placed for experiments. To mimic the locomotion of a live fish the dummy is pulled by a two-wheel robot from below the fish tank via magnetic coupling. The robot's wheel speeds are controlled via Wi-Fi using the RoboTracker software (see [3]), which receives positional and directional information about fish in the tank from the BioTracker software (see [1]). Different behavioral models for the RoboTracker are implemented as dynamically loadable plugins using the (abbreviated) C++ interface shown in Figure 1, where *IBody* specifies the position and orientation of a fish (live or dummy) in the tank, and *RobotAction* specifies a single action the robot is required to perform (see [2]).

IBehavior
nextSpeeds(robots: List<IBody>, fish: List<IBody>) : List<RobotAction>

Figure 1: IBehavior (old)

3.2 Model

For the purposes of this thesis it was decided to use an *agent-centric* model using recurrent neural networks (RNN) similar to the one used in [10] as I was interested to see whether their results with flies would translate to more complex organisms. As a result of this the model is inferred for one timestep at a time, predicting the next agent locomotion. It is furthermore important to note that the model is strictly two dimensional due to the design of the *RoboFish* system.

The model receives as input the handcrafted *feature vector* x_t consisting of three components: The locomotion $\vec{l}_{t-1}$ performed by the agent to reach time step t from time step $t-1$ (see subsubsection 3.2.1), the view of other agents $\vec{v}_{agents,t}$ the agent has at time step t (see subsubsection 3.2.2), and the view of (fish tank) walls $\vec{v}_{walls,t}$ the agent has at time step t (see subsubsection 3.2.3). The latter two are represented as separate channels of information to the model in order to enable having different parameters for their respective raycasting; also, we know that guppies can distinguish between other guppies and the walls of the fish tank. What this means in practice can be seen in Figure 2, which shows a screenshot taken from the *Trackset*[2] *Viewer* tool[3] in which which you can see a schematic of a track at some time step t together

[2]One track (time series of poses) per agent

[3]Developed by me for this thesis to facilitate exploration and qualitative evaluation of tracksets

with a visual representation of each agent's view vectors[4] shaded in the same way as detailed in subsubsection 3.2.2.

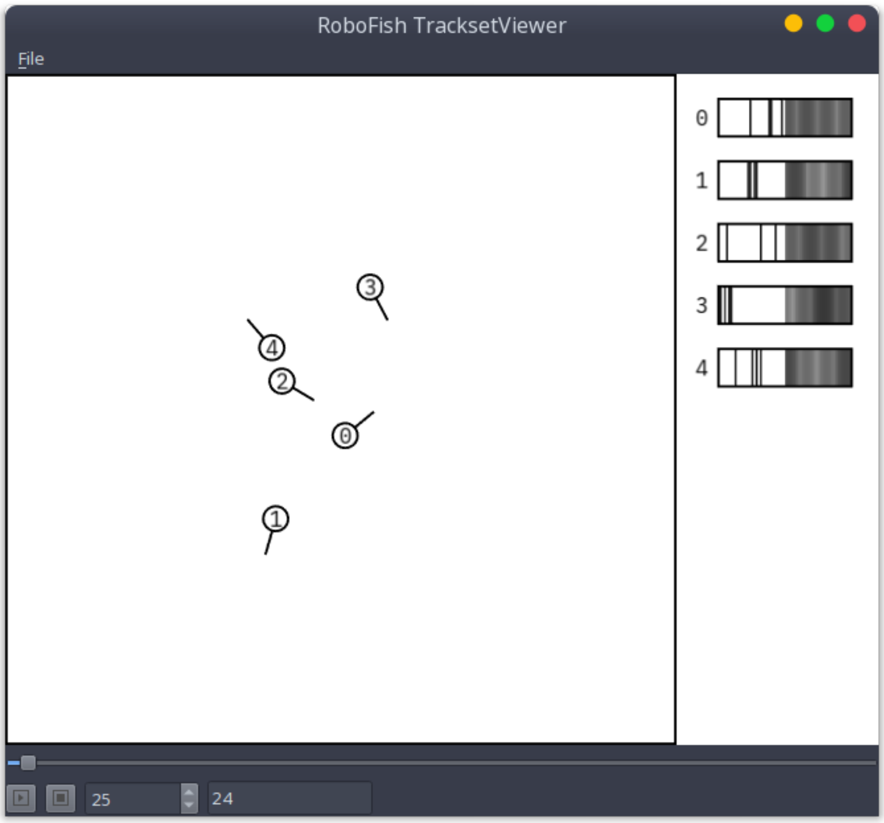


Figure 2: Trackset viewer example

There were two apparent methods to model the prediction of the locomotion vector components:

1. Predict a gaussian mixture for each component
2. Predict a multinomial distribution for each continuous component

The first of these options would require $2 \times k$ learnable parameters per locomotion component with k being the number of chosen normal distributions, while the second option would require n learnable parameters per locomotion component with n being the number of bins the component is discretized with. It should be noted, though, that the minimum k required for successful training is likely low enough for the first option to still be preferable in terms of parameter count. Another advantage of the first option would be the lack of fixed minimum and maximum values for each locomotion component, which would restrict the model less in what it can learn. The principal advantage of the second option, however, is that while k would need to be manually fine tuned, reasonable values for minimum and maximum values, as well as n can be extracted for each component from the input data. In accordance with the

[4]left half: view of agents, right half: view of walls

initial decision to see if the results of [10] could be transferred to guppies I decided to use the second option. That means the *output* of the model is one multinomial distribution per locomotion component.

3.2.1 Locomotion

As shown in Figure 3 I defined the locomotion an agent can perform between two time steps as the following pair based on observed real guppy locomotion:

- The angular turn v_a in radians for the change in orientation
- The linear speed v_l in cm for the forward/backward motion, which is the change in position projected onto the new orientation with the dot product.

As a consequence of this, the model cannot represent lateral motion, but since (1) the two-wheel robot in use by the RoboFish system would not be able to reproduce such motion, and (2) I did not observe any guppy actually performing such lateral movement (see captured videos of live fish) I do not consider this to be an issue.

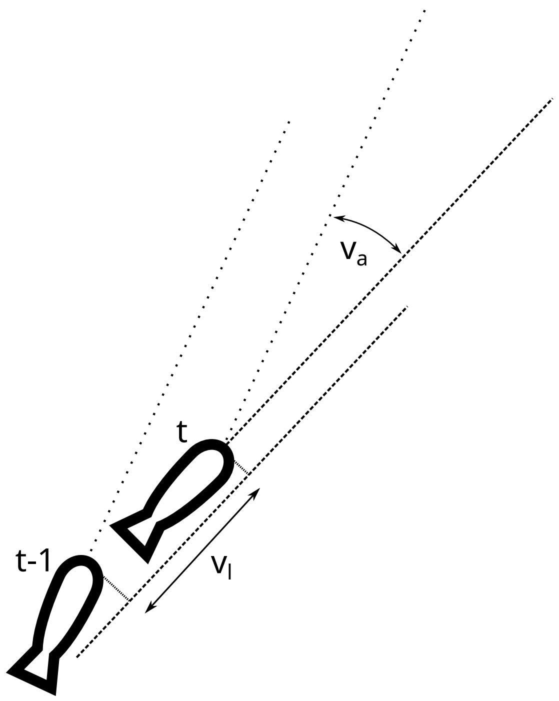


Figure 3: Agent locomotion between two time steps

3.2.2 View of agents

The sensory information an agent receives about other agents can be considered a minimalistic virtual eye: Given a *field of view* (*fov*), a number of discretization bins n, and a maximum viewing distance (*far_plane*), the *fov* is separated into n sectors, whereafter each sector is searched for the closest other agent within far_plane distance; the distance to each such found agent is then linearly scaled (normalized) from a $[0, far\_plane]$ distance value to a $[1, 0]$ intensity value and placed in the corresponding bin, which means the closer another agent is the higher the resulting value. Sectors with no other agent closed than far_plane in them have an intensity value of 0. Figure 4 shows a schematic of this process for an *fov* of 180°and 3 sectors with the sectors shaded from white (intensity = 0) to black (intensity = 1).

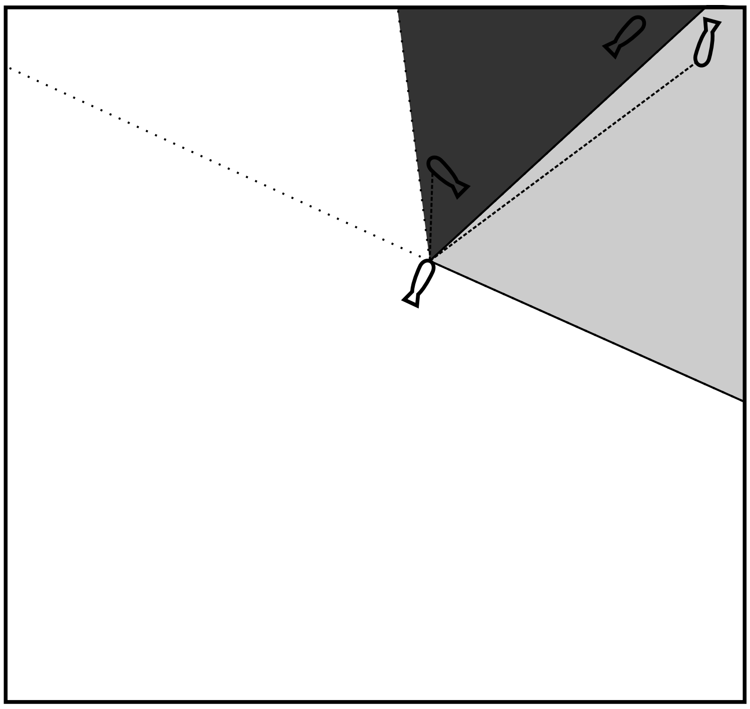


Figure 4: View of agents raycasting: 180°fov, 3 sectors

3.2.3 View of walls

The sensory information an agent receives about the walls of the fish tank are raycast in a similar fashion to the view of agents, with the key difference being that instead of having sectors in which the closest of multiple agents is searched for, rays are directly cast at the walls of the fish tank. The distances on each of those rays are then calculated and normalized into intensity values the same way as is done for the view of agents (using a shared value for far_plane). Figure 5 shows a schematic of this process for an *fov* of 120°and 3 rays, which also show cases that when both of the view vectors are of the same size[5] the wall rays split each agent sector in half.

[5]Number of agent sectors equals number of wall rays

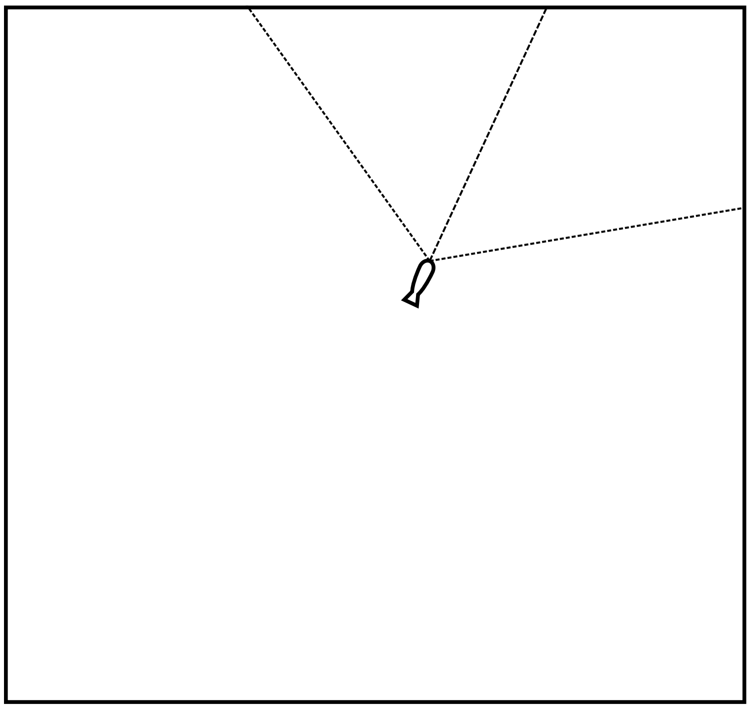


Figure 5: View of walls raycasting: 120°fov, 3 rays

3.3 Required modifications

The principal issue with the existing infrastructure was that it did not allow for a robot behavior to specify allowed time steps between successive *nextSpeeds* calls. This was not acceptable since my RNN model was going to be implicitly trained on a single time step: The one between two successive frames in the training examples shown to it. To address this, these two changes were made to the interface (see Figure 6):

1. The *supportedTimesteps* methods was added so a behavior can notify the *RoboTracker* which time steps (in milliseconds) it supports; an empty list signals that any time step is acceptable.

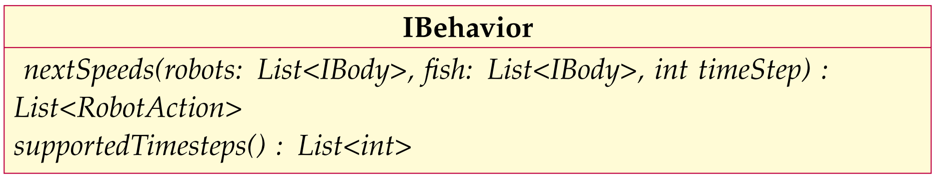
2. The *nextSpeeds* method received an additional *timeStep* (in milliseconds) parameter so the *RoboTracker* may decide which of the supported time steps it wishes to infer for.

It is thus the responsibility of the *RoboTracker* to

- Only call *nextSpeeds* with a supported time step[6]

- To schedule the next call to *nextSpeeds* after the amount of time specified in the previous call's *timeStep* parameter has passed

[6]Violations may result in undefined behavior



IBehavior

nextSpeeds(robots: List<IBody>, fish: List<IBody>, int timeStep) : List<RobotAction>
supportedTimesteps() : List<int>

Figure 6: IBehavior (new)

The next issue was that there was no suitable existing *RobotAction* capable of turning my model's locomotion predictions into motor speeds. To address this, I added a new action called *RobotActionTurningForward* that takes two parameters on construction:

- *turn*, which is the angular component of the locomotion (in radians)
- *forward*, which is the linear component of the locomotion (in centimeters)

Given the distance in centimeters between the center points of the two robot wheels (wd) and the time step in milliseconds the motor speeds are then calculated as follows:

$$\text{forward}' \leftarrow \text{forward} \times \frac{1000}{\text{time\_step}}$$

$$\text{turn}' \leftarrow \text{turn} \times \text{wd} \times \frac{1000}{\text{time\_step}}$$

$$\text{left speed} \leftarrow \text{forward}' + \frac{\text{turn}'}{2}$$

$$\text{right speed} \leftarrow \text{forward}' - \frac{\text{turn}'}{2}$$

This means that when executing an action from the model the robot will drive an arc of length *forward* with a curvature such that *turn* is the difference between the start and end orientations of the robot. While this implementation is slightly different from the previous definition of locomotion shown in Figure 3 it does not require splitting a single model prediction into multiple sequential robot actions and thus also avoids the potentially complex time scheduling that would require. Due to guppies most of the time either turning in place or moving forward (see videos) this should not pose an issue with a low enough time step value.

4 Data acquisition

This section describes how the synthetic couzin data set was generated and the live fish data set was captured and processed.

4.1 Synthetic data

The model used in this thesis to generate synthetic collective behavior ("couzin" model) is largely the same as the one described in [9]. The principal difference is that in addition to the three interaction zones present in the original I added fourth zone, the zone of (tank) wall repulsion (zowr), which has the highest priority: Each tank wall has a repulsion vector that is directed orthogonally away from it. Whenever the agent has at least one wall in this zone the target direction vector is the sum of all of those wall repulsion vectors whose respective wall is within the zone.

Of the four distinct behaviors described in [9] the two of most interest to me with respect of trying to have an RNN learn their rules were the torus behavior (see Figure 7) and the dynamic parallel group behavior (see Figure 8). Out of those two I decided to focus on the torus behavior due to time constraints, selected appropriate parameters for it (see Table 1), and generated synthetic tracksets for it with a simple simulation loop. It should be noted that agents are spawned uniformly accross the tank, except a 20 cm wide area from each tank wall[7], with uniformly distributed orientations. I chose to generate a dataset containing 120/number of agents 30 second tracksets each for 1, 2, 4, 6, and 8 simultaneous agents at 25 frames per second; this amounts to 450000 individual frames per dataset. This dataset was then split into training / validation / test data sets with a 0.8 / 0.1 / 0.1 distribution.

Parameter	Value
Zone of wall repulsion	20 cm
Zone of (agent) repulsion	1 cm
Zone of orientation	4 cm
Zone of attraction	40 cm
Field of perception	300°
Turn rate	45°
Speed	8 cm/s
Direction error (standard deviation)	0°

Table 1: Couzin model parameters for torus behavior

4.2 Live data

4.2.1 Organism and video capture

The fish used were from the same stock of originally wild-type guppies as detailed in [6] and were maintained in the same kinds of conditions, though it should be noted

[7]So agents are not spawned such that they cannot avoid crashing into the walls

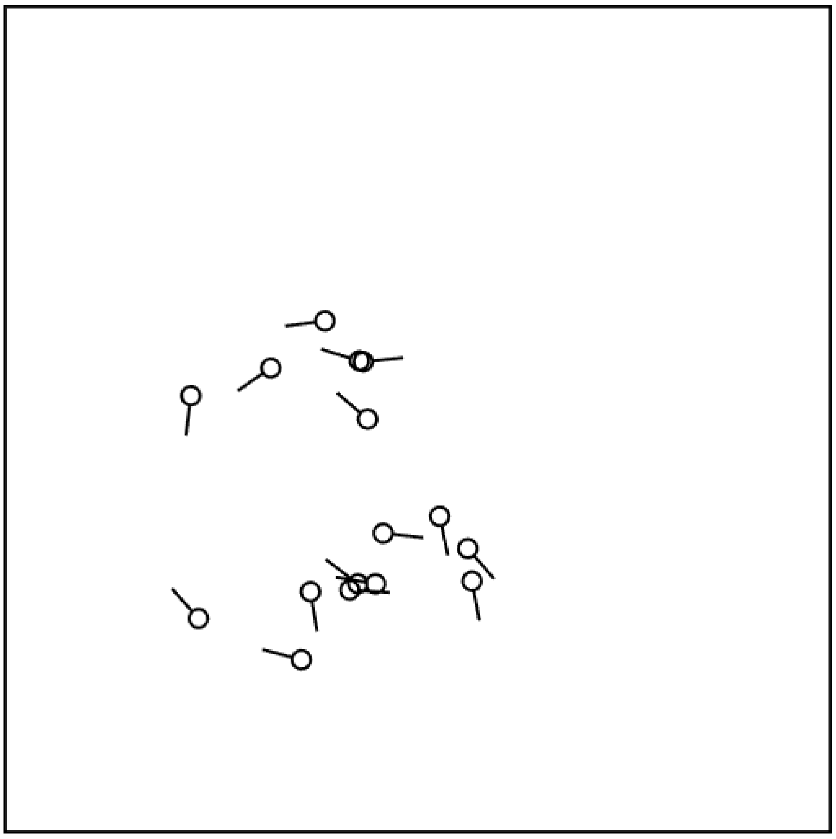


Figure 7: Example frame: Couzin torus behavior

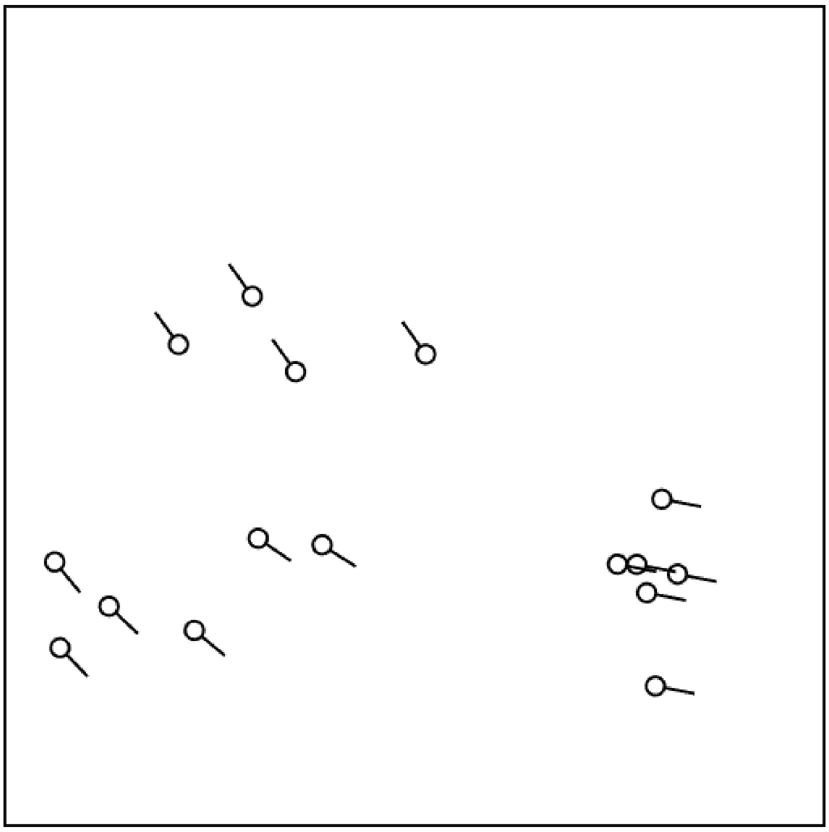


Figure 8: Example frame: Couzin dynamic parallel group behavior

that by now they have been bred in laboratory conditions for several more generations.

For each captured video the following process was followed:

1. Scoop two female guppies from the aquarium of fish who have never before been in the fish tank with a hand net and place them in a small container
2. Transfer that container to the experiment fish tank
3. Start the video capture
4. Move the fish from the container into the experiment fish tank with the hand net
5. Close the fish tank roller blinds so the fish cannot see outside the tank
6. Capture video for at least 10 minutes
7. After capturing, with the hand net move the fish into the container again
8. Take a picture of the fish in the container alongside a ruler to establish their size
9. Move the fish into the second aquarium of fish who have already been into the experiment fish tank once

The videos were captured by a Basler acA2040-90um camera with a resolution of 2048×2048 pixels at 25 frames per second[8], and example frame of one such video is shown in Figure 36.

4.2.2 Pose extraction

Given that my model requires pose information (position and orientation) and does not receive raw visual data as input the videos need to be processed to be of use. While it would have been preferable to use the existing *BioTracker* [1] platform for this there was no available tracking plugin for it that did not yield unacceptable levels of identity switches[9]. Since I did not believe it to be feasible for me to hand correct all the tracksets after such an erroneous extraction I searched for automated open source solutions.

What worked out well for me was the idtracker.ai software [24], which I used as a black box system that I gave each video as input and received detected fish as accurate contours without any observed id switches for each frame in each video as output. After that I extracted the pose information from each such contour as follows:

1. Pass the contour through a one dimensional gaussian filter ($\sigma = 10$)
2. Calculate the curvature of the resulting smoothed contour
3. Find the two biggest local maxima of the curvature; due to the form of guppies as seen from above these match extremely well with the tail end and the tip of the snout of the corresponding fish

[8]Equal to a time step of 40 milliseconds

[9]Fish A suddenly being reported as fish B and vice versa

4. Calculate the fish position as the centroid of the subset of the contour within 20 pixels of the tip of the snout
5. Remove camera distortion(s) from fish position and tip
6. Transform from the camera coordinate system in pixels to the *RoboFish* world coordinate system in centimeters (origin being at the top left)
7. Calculate the fish orientation as the normalized vector from fish position towards fish tip

For any instance where the contour for a fish was missing or the above process failed (e.g. due to not enough local maxima being detected) the fish pose was linearly interpolated from previous and subsequent frames. If there are no previous frames to interpolate from the frame is dropped. Both the amount of frames that needed to be interpolated like this (< 1 percent per video) and the mean length of contiguous series of such frames (< 100 milliseconds in all videos) were considered to have negligible impact on the viability of the data for supervised training.

4.2.3 Finalization

The standard supervised learning of RNNs I employ requires unrolling them in the temporal axis to some fixed sequence length. While this still allows for having differently sized training examples (by slicing them into different amounts of such fixed sequences) I also wanted to employ the common optimization technique of minibatching [21]. This requires that all those examples that may be batched together be of the same overall temporal length (before sequence slicing), as the state of the RNN needs to be retained between sequences of the same example.

This requirement, however, is not fulfilled in the live dataset after pose extraction:

- The captured videos do not all have the same length
- Using the idtracker.ai software often requires removing an untrackable prefix of the video
- Pose extraction may cut off another prefix as previously detailed

In addition to these factors the fish sometimes do not start moving at all for an extended period of time after being placed into the tank. As this initial behavior is not of interest to me w.r.t. swimming of and interaction between fish I also cut off the prefixes of the tracksets after pose extractions such that at least one fish was exhibiting absolute linear speed above 2 cm / s after a gaussian filter ($\sigma = 5$). At this point the live dataset consisted of 579212 individual frames with a distribution of track lengths shown in Figure 9.

In order to best fulfill the above requirement, the tracks need to be sliced into subtracks of equal total temporal length (so any example may be batched with any other example). This has the disadvantage of losing any temporal dependency between subtrack boundaries, which means there are two parameters to optimize: The amount of total frames lost should be minimized (more training data) and the subtrack length should be maximized (more temporal dependencies that can be learned). The relationship between the two can be seen in Figure 10. After balancing the two against each other I decided to settle for a subtrack length of 8990 frames, which results in a frame loss of 9.98%. This means at best roughly 6 minutes worth of temporal dependencies can be learned from the dataset and the finalized total amount of individual frames in the dataset is 521420. This dataset is then split into training / validation / test data sets with a 0.8 / 0.1 / 0.1 distribution.

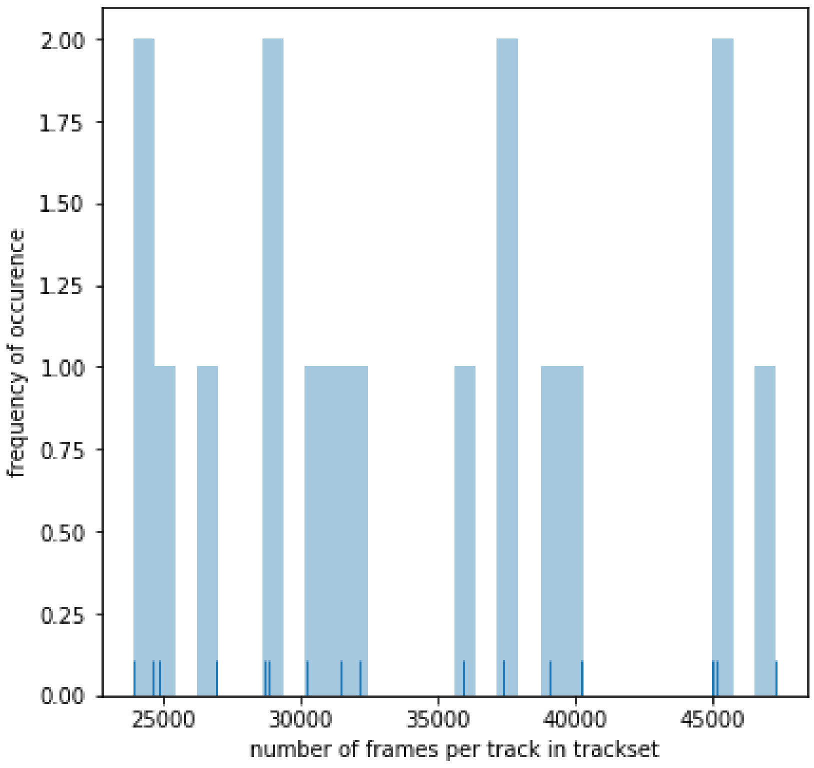


Figure 9: Distribution of live trackset lengths

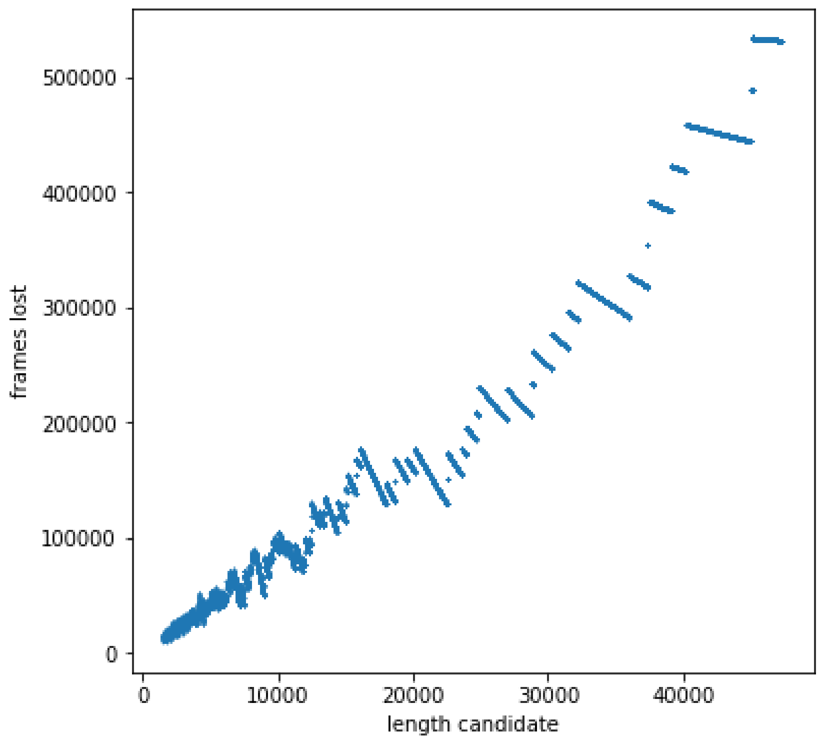


Figure 10: Number of frames lost by subtrack slicing

5 Experiments

This section describes, present the results of, and discusses the experiments performed.

5.1 Behavior quantification

In order to quantify the observed behavior I decided on using the following non-standard measures in addition to the basic measures of interindividual distance (iid), linear speed, and angular speed:

Definition. *For two agents a and b, let the follow measure $follow(a, b)$ be the dot product of the a's velocity and the normalized direction from a's position to b's position.*

The follow measure was initially designed for the not yet published Mönck et al. [23] and is only being reused here.

It is maximal if a currently moves towards b, and minimal if a moves away from b.

Definition. *For two agents a and b, let the time lagged velocity correlation $tlvc(a, b, \tau_{min}, \tau_{max})$ be the mean of the dot products of a's velocity and b's future velocities, starting at τ_{min} seconds in the future and stopping at τ_{max} seconds in the future.*

The time lagged velocity correlation was designed by me with inspiration from Bierbach et al, 2018 [6].

It is maximal if b is fully aligned with a over the future time window given by τ_{min} and τ_{max} and minimal if b is fully aligned in the opposite direction as a over that time window.

5.2 Network training

All networks presented were implemented using the MXNet machine learning library [8] and trained with the Adam optimizer [18] using mini-batches randomized at every training epoch. Recurrent states were zero initialized. To mitigate the effects of vanishing/exploding gradients [15] I chose to employ layer normalization [4] on all LSTM cells. To achieve better generalization[10] I also chose to employ variational dropout [11] on all LSTM cells.

The following measures are used when discussing the training process:

Definition. Loss *refers to the mean of the cross entropies [13] between the network's locomotion predictions and the actually correct locomotions in the training/validation data.*

Definition. Accuracy *refers to the mean of individual locomotion 0/1 accuracies, which reflect whether the network correctly predicted a locomotion bin as given by the training/validation data.*

Definition. Confidence *refers to the mean of individual locomotion confidences, each of which reflect the highest prediction probability per locomotion component.*

[10]Mitigate overfitting

5.3 Couzin torus behavior

As the couzin model is low in complexity I chose to start training on the couzin torus dataset with a stacked LSTM network followed by a fully connected output layer. The training parameters are listen in Table 2. The view vector values were arbitrarily chosen, whereas the far plane matches the diagonal of the tank[11]. The number of linear speed bins was chosen such that their width (and thus the maximum continuous error on correct bin prediction) is less than 5% of the linear speed (see Table 1). The number of angular speed bins was chosen such that there are at least 10 bins within the high density area of their distribution (see Figure 11). The number of hidden units and recurrent layers were chosen arbitrarily.

Parameter	Value
View of agents	300°, 21 sectors
View of walls	180°, 15 rays
Far plane	142 cm
Angular speed	-0.75 - 0.75 radians/frame, 217 bins
Linear speed	-0.32 - 0.32 cm/frame, 41 bins
Hidden units	10
LSTM layers	1

Table 2: LSTM torus: Model parameters

Parameter	Value
Target accuracy	99%
Sequence length	75 frames
Batch size	10
Dropped cell input	0%
Dropped recurrent state	10%

Table 3: LSTM torus: Training parameters

As can be seen in Figure 12, the loss converges extremely fast after only a few epochs and the training terminates early after reaching the selected target accuracy (see Figure 13) with a high confidence (see Figure 14). Thus, for simulation purposes the last epoch was always chosen. These figures show that the network performs better on the validation data set than on the training data set. This is highly unusual and could result from the network learning (or overfitting on) only a subset of the training data that happens to line up well with the validation data set. To investigate this cross-validation should be employed, which I refrained from doing due to time constraints.

In order to determine whether the network was actually able to learn the torus behavior instead of overfitting I first simulated a single agent (see Figure 15) for 3 minutes in the same manner as described in section 4. It shows the network generally favoring curved trajectories as opposed to the straight trajectories the actual couzin

[11]So no agent is too far from another to see it

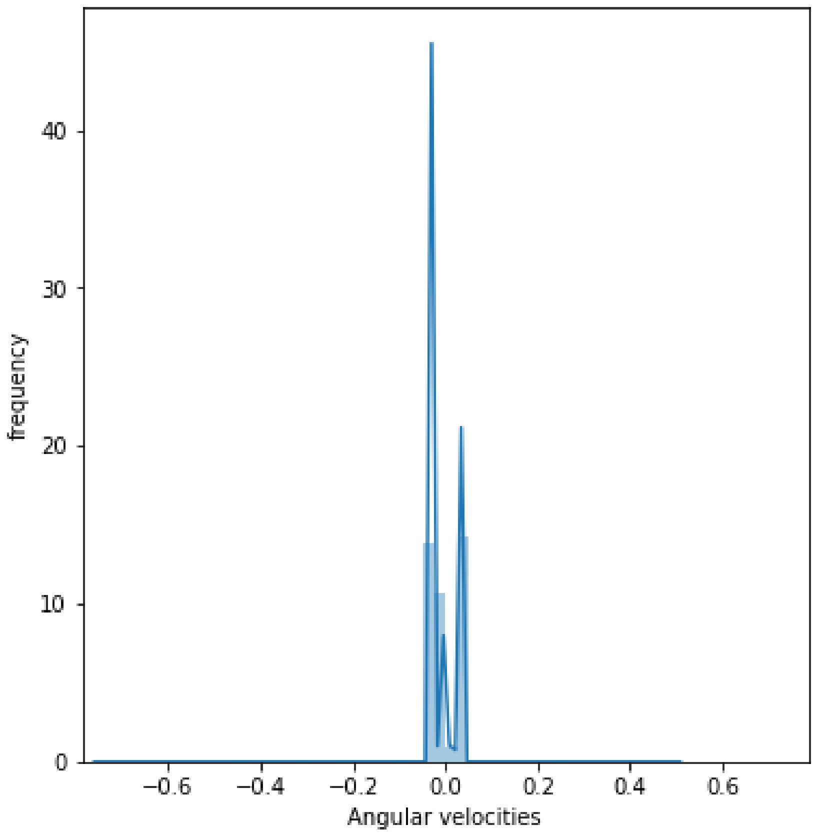


Figure 11: Couzin torus: Angular speed distribution

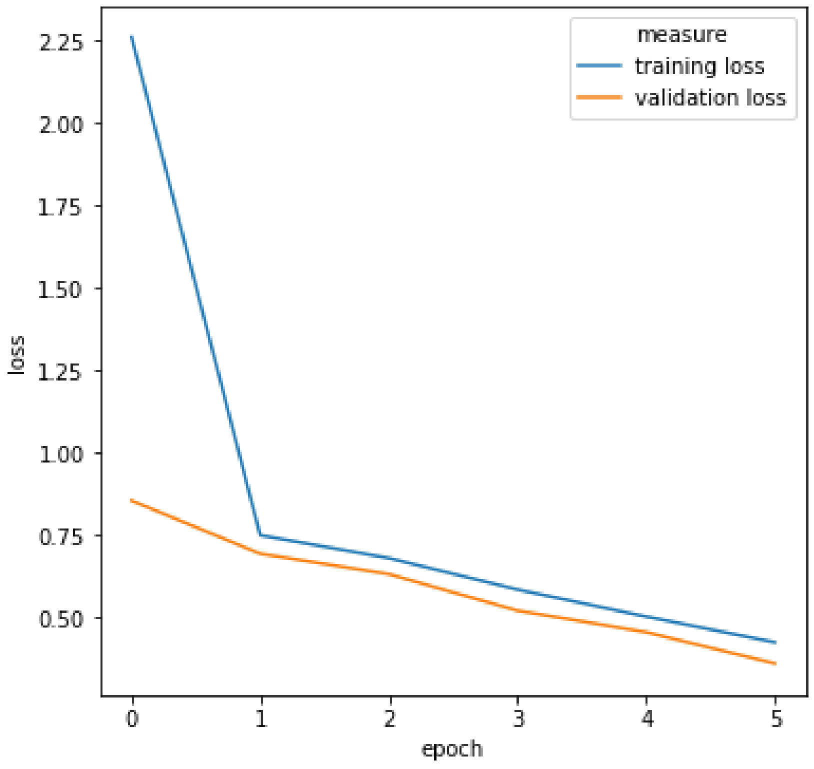


Figure 12: LSTM torus: Loss progression

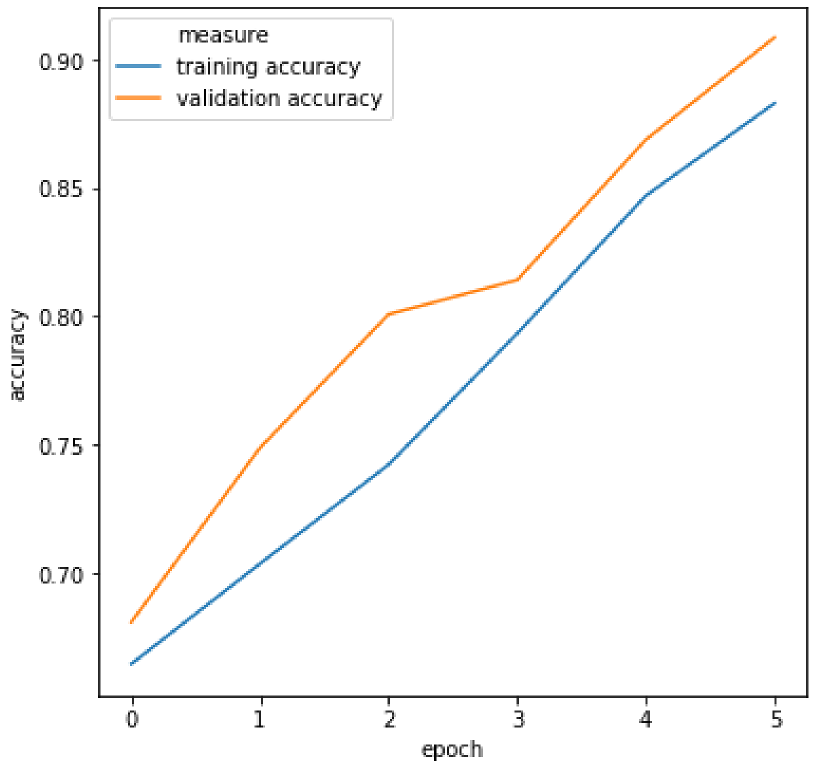


Figure 13: LSTM torus: Accuracy progression

model would yield. It does, however, generally try to avoid the tank walls and does not simply keep rotating around some center point, both of which I consider to be positive indicators on the network having learned the torus behavior.

After that I simulated multiple agents with the simulation parameters as detailed in section 4 for the input data.
Comparing Figure 16 with Figure 17 shows the network concentrating too much on the tank center, which to me indicates that regardless of whether the interaction component of the behavior was learned, the network also learned the *much easier* task of being near the center of the tank. Further investigations into this might try to replace the linear scaling of the wall view vector with an exponential scaling as was done in Eyjolfsdottir et al., 2016 [10]. Another approach would consist of placing convolutional layers, standard or LSTM [25], as a *view* network in front of the current *action* network.

Comparing Figure 18 with Figure 19 shows a similar correlation of proximity to both following and avoiding another agent; further comparing Figure 20 with Figure 21 shows similar correlations between agent proximity and velocity con- and divergence over the next second. I consider both of these positive indicators that basic interaction rules of the couzin model were successfully learned by the network. One major difference is that while the couzin model has a peak IID of about 2 cm, the network peaks at about 20 cm. One possible explanation for this would be that the network has only partially learned the rules behind the torus behavior: Enough to

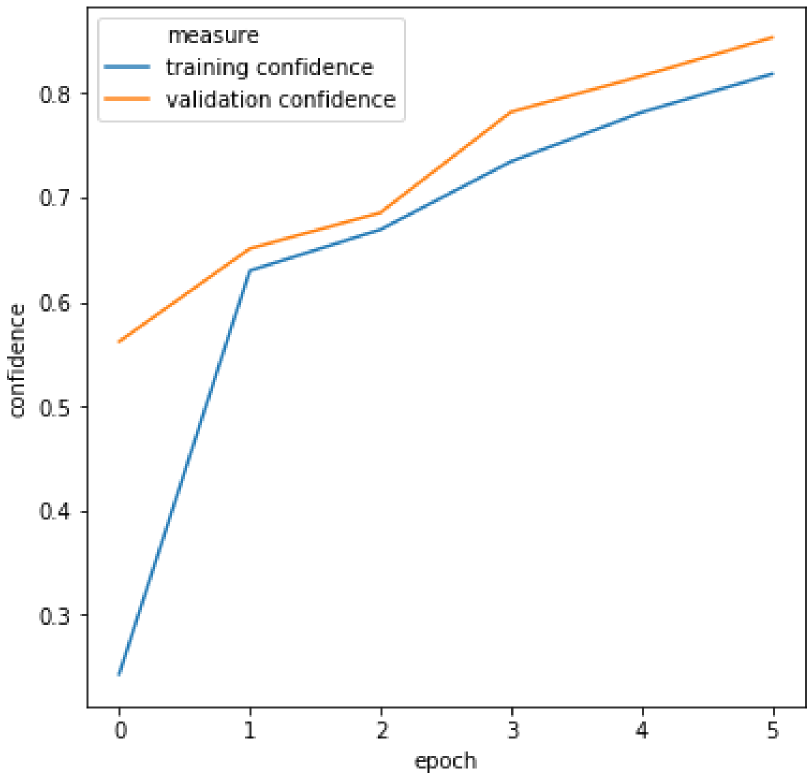


Figure 14: LSTM torus: Confidence progression

attract agents towards each other and react to each other, but only within the bounds of being a certain distance near the center of the tank. Another possibility is that the network has learned the 20 cm zone of border repulsion from the training data as applying to both the walls and other agents. Further investigations into this peak were not done due to time constraints.

In an attempt to improve upon the above results I decided to implement the *view / action* network split mentioned in the above. In this setup the two view vectors are stacked, not concatenated, and then given as the input to a convolutional LSTM (ConvLSTM) layer with each view vector being an input channel. As a consequence of this, both view vectors must be of equal size. After going through the ConvLSTM layer the output is then flattened and concatenated with the locomotion input. That vector then is the input to the action network, which is again a single layer LSTM. The parameters used match those already detailed in Table 2 and Table 3, except the view vectors. View of agents now has a granularity of 150 sectors (same field of view), and view of walls has rays going through the center of view of agent's sectors (298°field of view, 150 rays); also, the ConvLSTM layer has 8 output channels and kernel size (input to input and input to hidden) 9. The training progression can be seen in Figure 22, Figure 23, and Figure 24. It is not surprising that the addition of a ConvLSTM layer and a steep increase in raycast input significantly increases the required number of epochs. The training was stopped early after 25 epochs due to time constraints and the last epoch was chosen for simulation purposes.

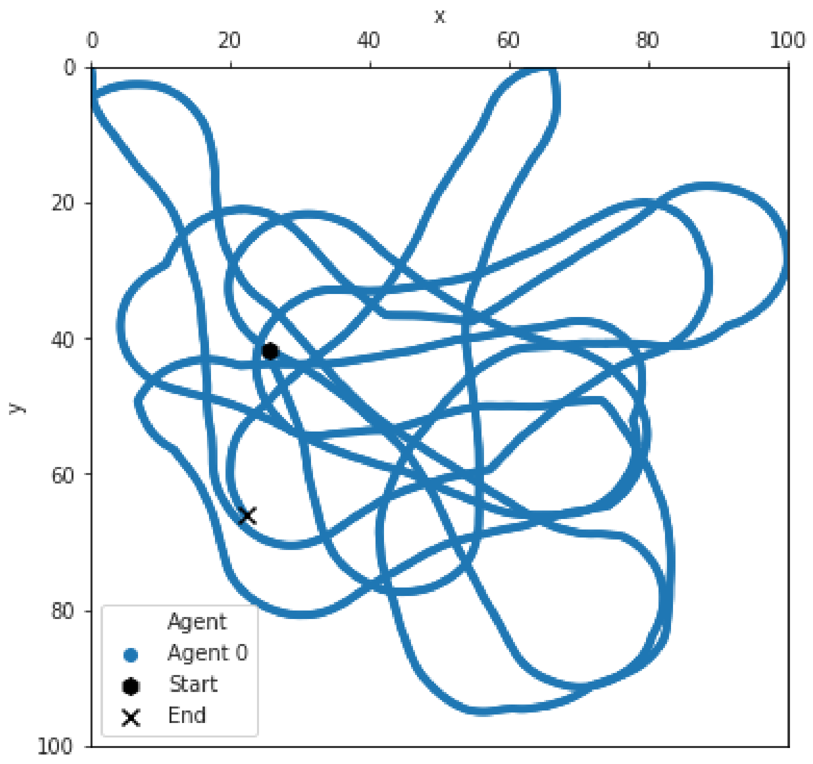


Figure 15: LSTM torus: Single agent tank trajectory

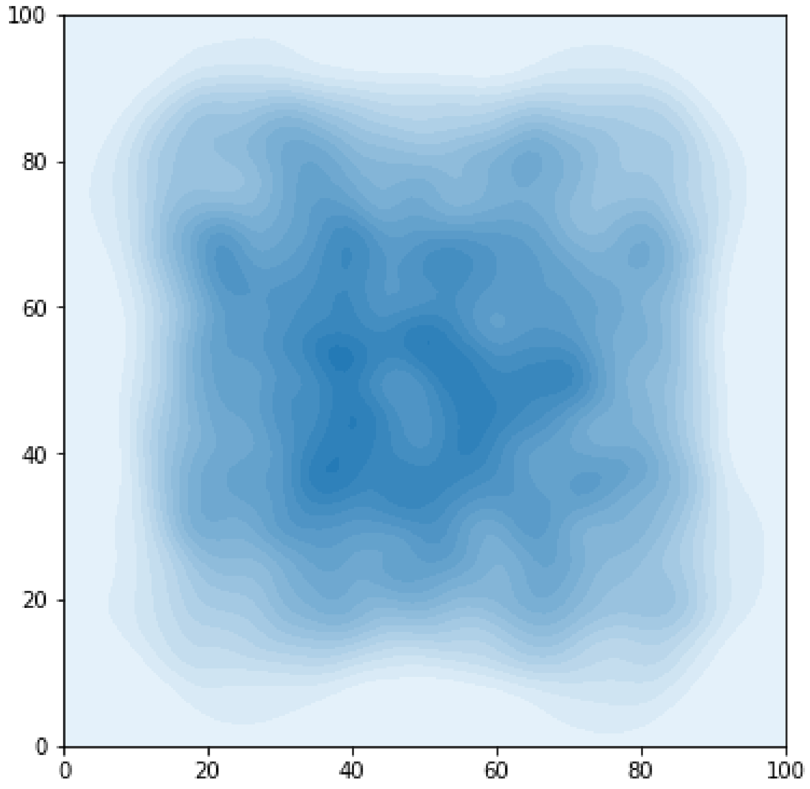


Figure 16: Couzin torus: Average tank positions

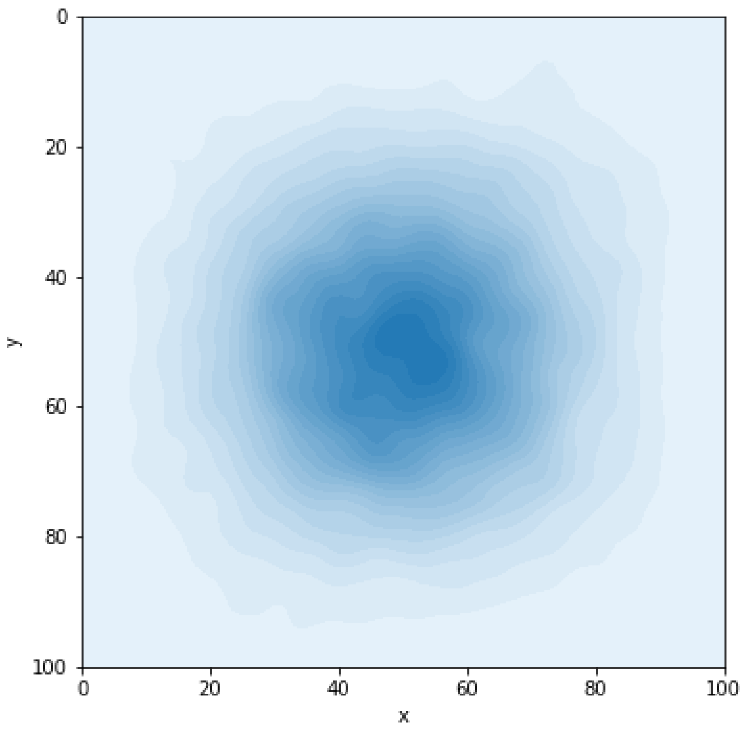


Figure 17: LSTM torus: Average tank positions

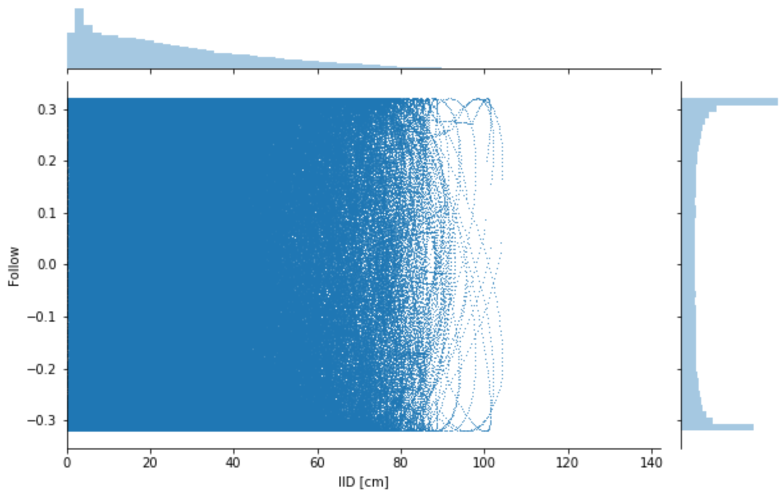


Figure 18: Couzin torus: Follow / IID

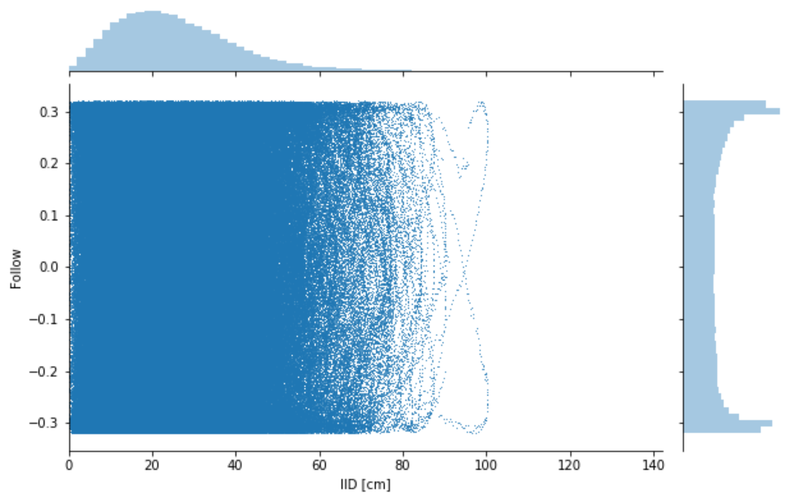


Figure 19: LSTM torus: Follow / IID

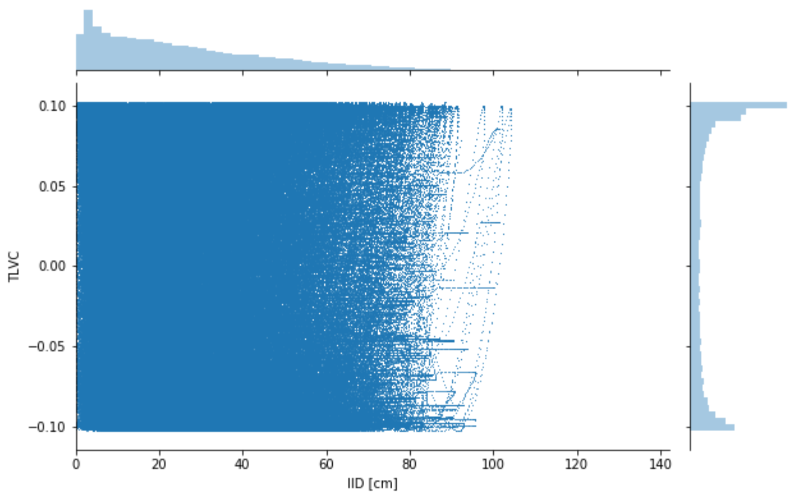


Figure 20: Couzin torus: TLVC / IID ($0s \leq \tau \leq 1s$)

A 3 minute single agent simulation of the new network shows a remarkably stabilized trajectory (see Figure 25) and simulations with multiple agents[12] show a slighly decreased concentration on the tank center, both of which I consider positive indicators for the viability of the view / action network split.

Further investigations into this were not feasible due to time constraints.

5.4 Live fish behavior

Due to the unknown internal complexity of the guppies' internal behavior I trained networks with many different combinations of model parameters, as shown in Table 4.

Parameter	Value range
Action network	Stacked LSTM, Eyjolfsdottir[10]
Hidden units	10 - 100
LSTM layers	1 - 3
ConvLSTM layers	0 - 2
ConvLSTM output channels	8, 16, 32
ConvLSTM kernel sizes	5, 9, 21

Table 4: Explored model parameters

Simulations on such trained models were then performed as follows: One trackset was uniformly chosen from the validation set. For each agent in that validation trackset[13] a network agent was created and spawned with the same initial position and orientation as in the validation trackset, after which the simulation loop previously mentioned was run.

[12]Same simulation parameters as in section 4

[13]Always 2

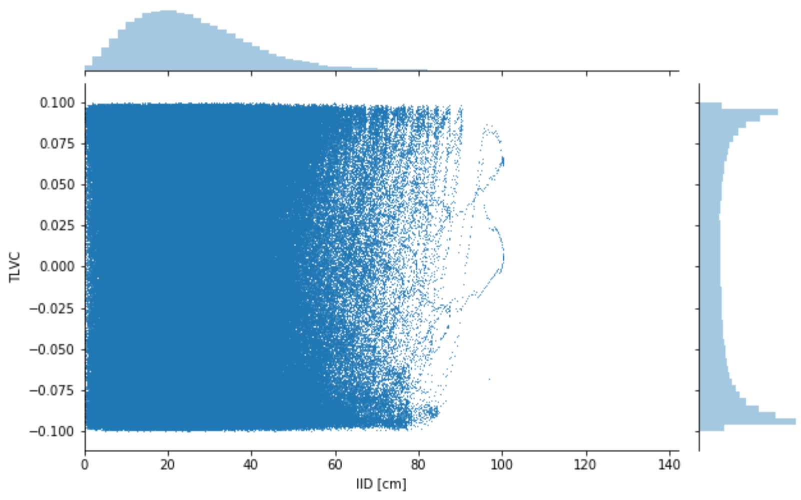


Figure 21: LSTM torus: TLVC / IID ($0s \leq \tau \leq 1s$)

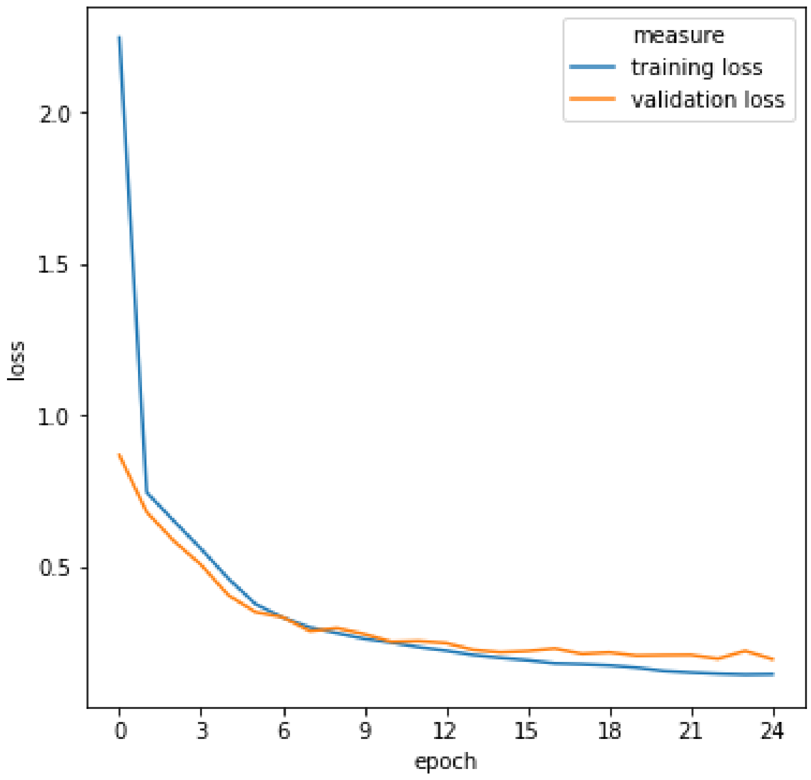


Figure 22: ConvLSTM+LSTM torus: Loss progression

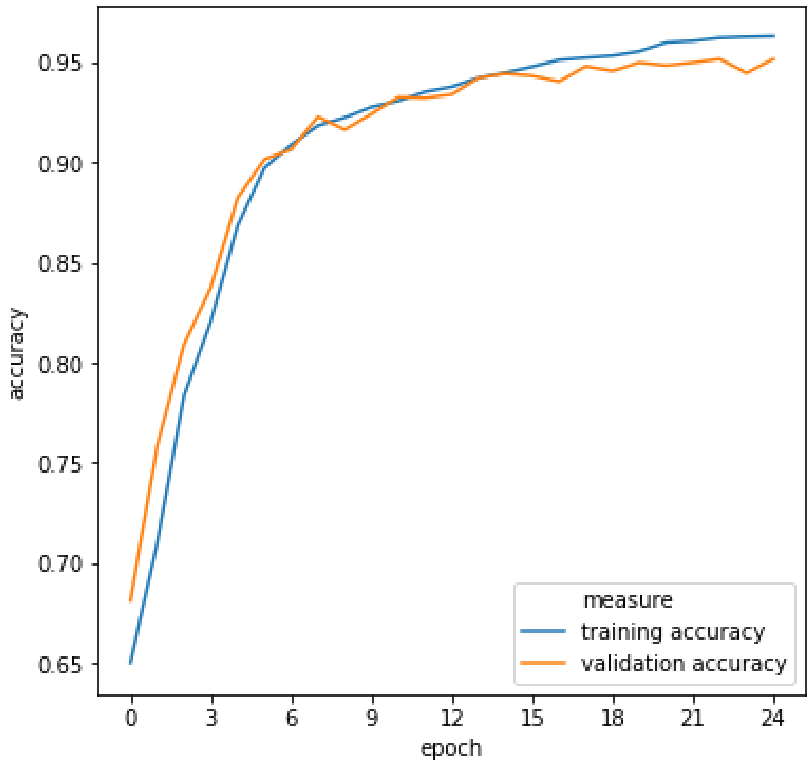


Figure 23: ConvLSTM+LSTM torus: Accuracy progression

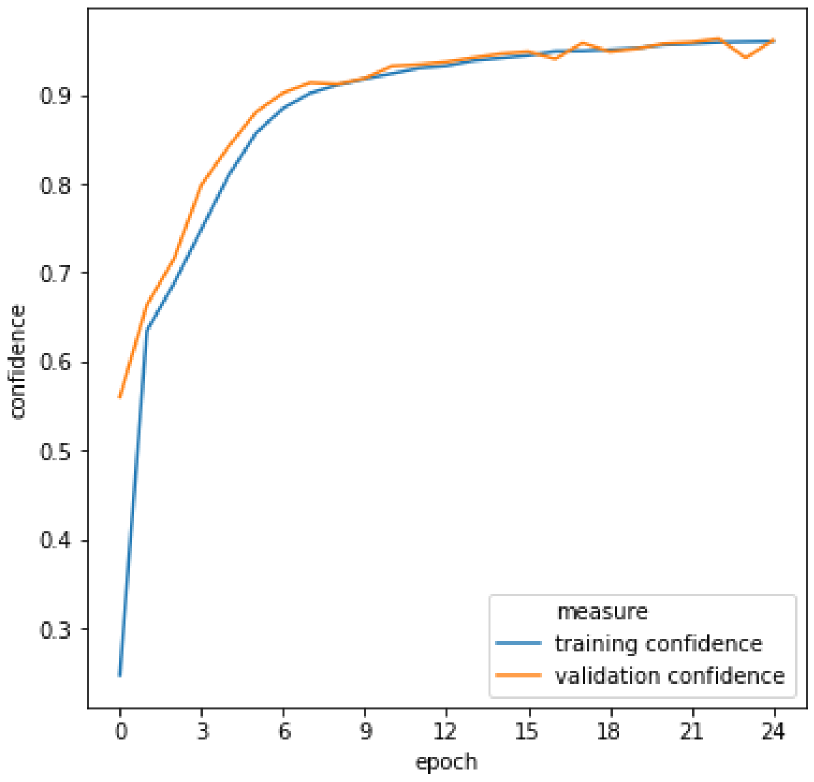


Figure 24: ConvLSTM+LSTM torus: Confidence progression

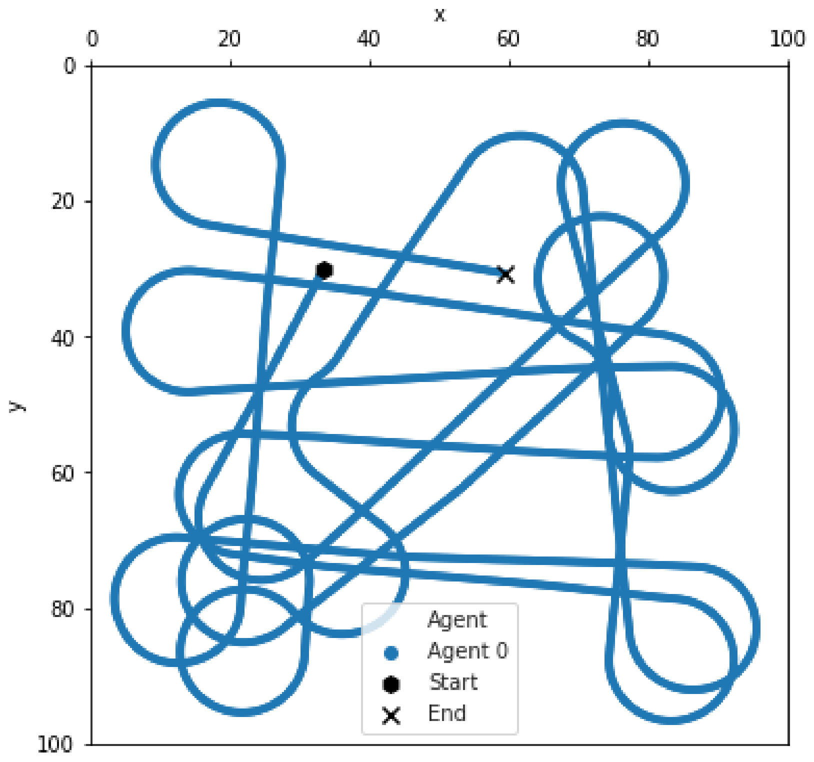


Figure 25: ConvLSTM+LSTM torus: Single agent tank trajectory

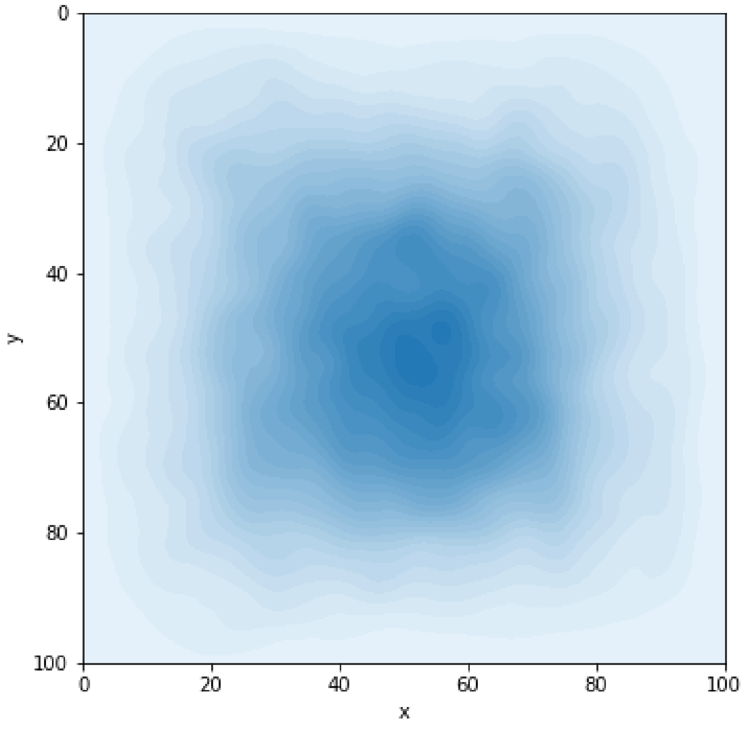


Figure 26: ConvLSTM+LSTM torus: Average tank positions

The networks were then qualitively compared against each other based on how realistic the behavior observed in those simulations appeared to me and a representative network was chosen. This network's model parameters are shown in Table 5. The training parameters are the same as in Table 3 and the training progression can be seen in Figure 27, Figure 28, and Figure 29. The loss progression shows three similar local optima on the validation set: At epochs 13, 16, and 20. Furthermore, training and validation performance start to diverge after epoch 16. These two together suggest to me that the network is overfitting on the training set after epoch 20. Thus, epochs 13, 16, and 20 are the most interesting ones in terms of learning, which lead me to examine those three with the same measures used for the networks trained on the couzin torus behavior. For each examined epoch 20 10 minute simulations were done with initial poses from the test set[14].

Parameter	Value
View of agents	300°, 150 sectors
View of walls	298°, 150 rays
Far plane	142 cm
Angular speed	-3.14 - 3.14 radians/frame, 43 bins
Linear speed	-3.38 - 3.38 cm/frame, 91 bins
Action network	Eyjolfsdottir
Hidden units	100
LSTM layers	2
ConvLSTM layers	1
ConvLSTM output channels	8
ConvLSTM output kernel size	9

Table 5: Live fish network: Model parameters

When comparing the different epochs shown in Figure 30, Figure 31, and Figure 32 with the real fish behavior shown in Figure 33, Figure 34, and Figure 35 it becomes apparent that the network first learned some fish interaction and to avoid the tank walls, but not where in the tank the fish generally prefer to be in. When it then starts learning that the fish generally prefer to be near the tank walls it loses a significant part of the learned fish interactions. Why the follow and tlvc differences between simulated and real fish are so noticeable before that switch in learning direction remains unknown. One possibility could be that the network did not receive enough training data. Another that the instances of important changes in interaction between fish are far outweighed by the general "swimming about" seen in the training data. I believe further investigations into this would thus benefit greatly from capturing much more live fish behavior and training on only "interesting" subtracks; this was not done due to time constraints.

[14]Same as above with the validation set

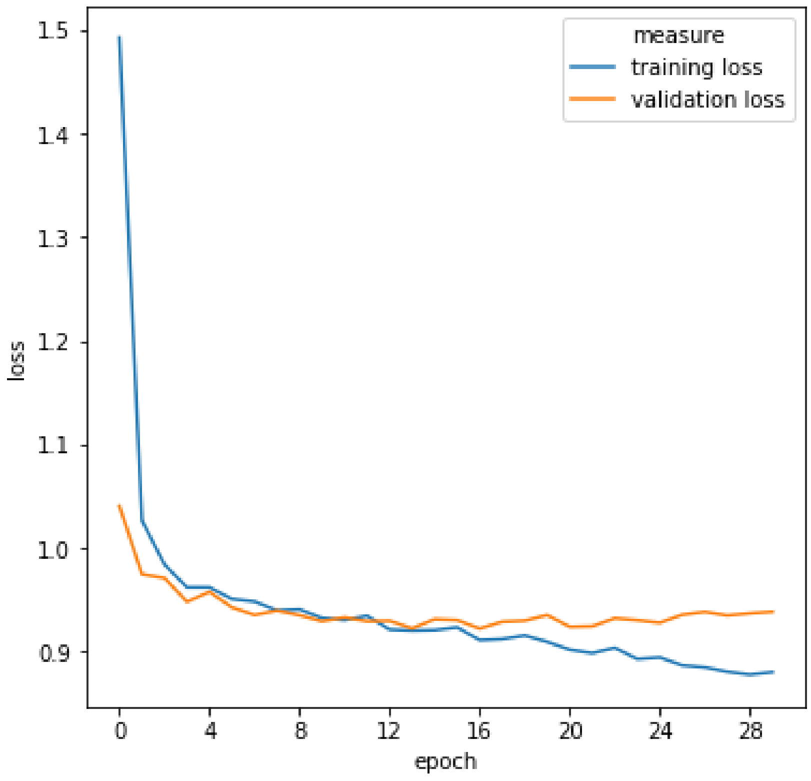


Figure 27: Live fish network: Loss progression

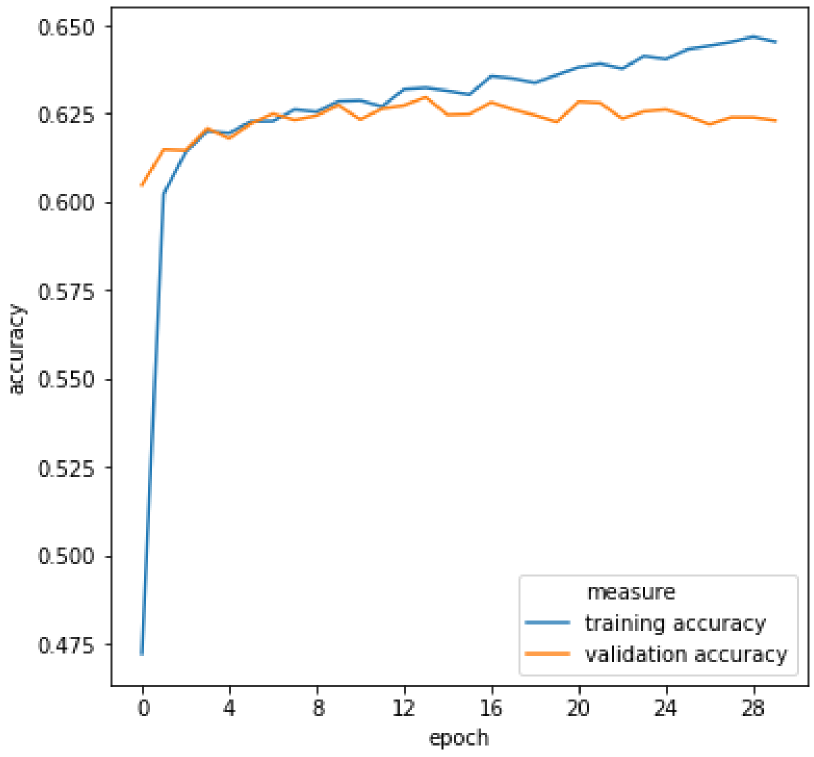


Figure 28: Live fish network: Accuracy progression

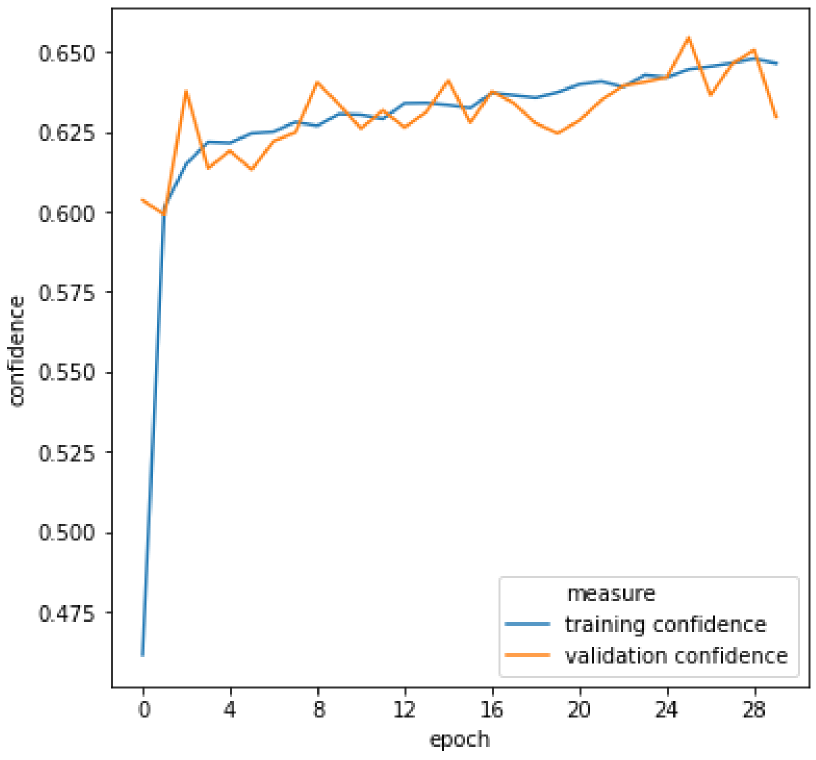


Figure 29: Live fish network: Confidence progression

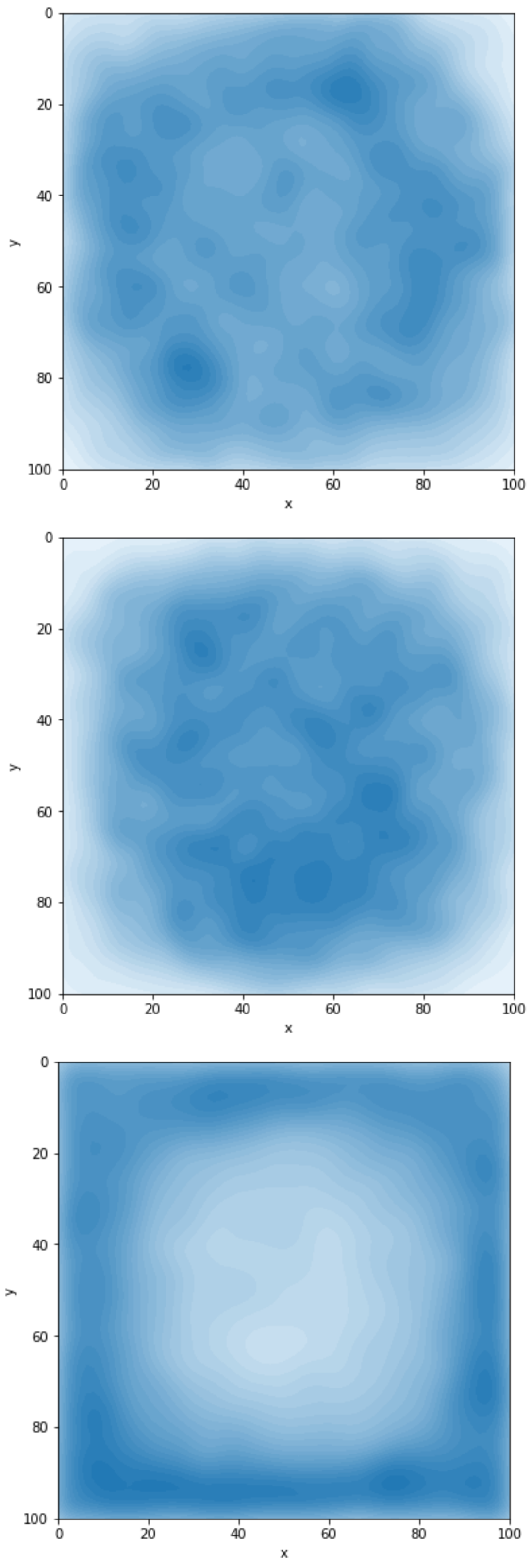


Figure 30: Live fish network: Average tank positions at epochs 13, 16, and 20

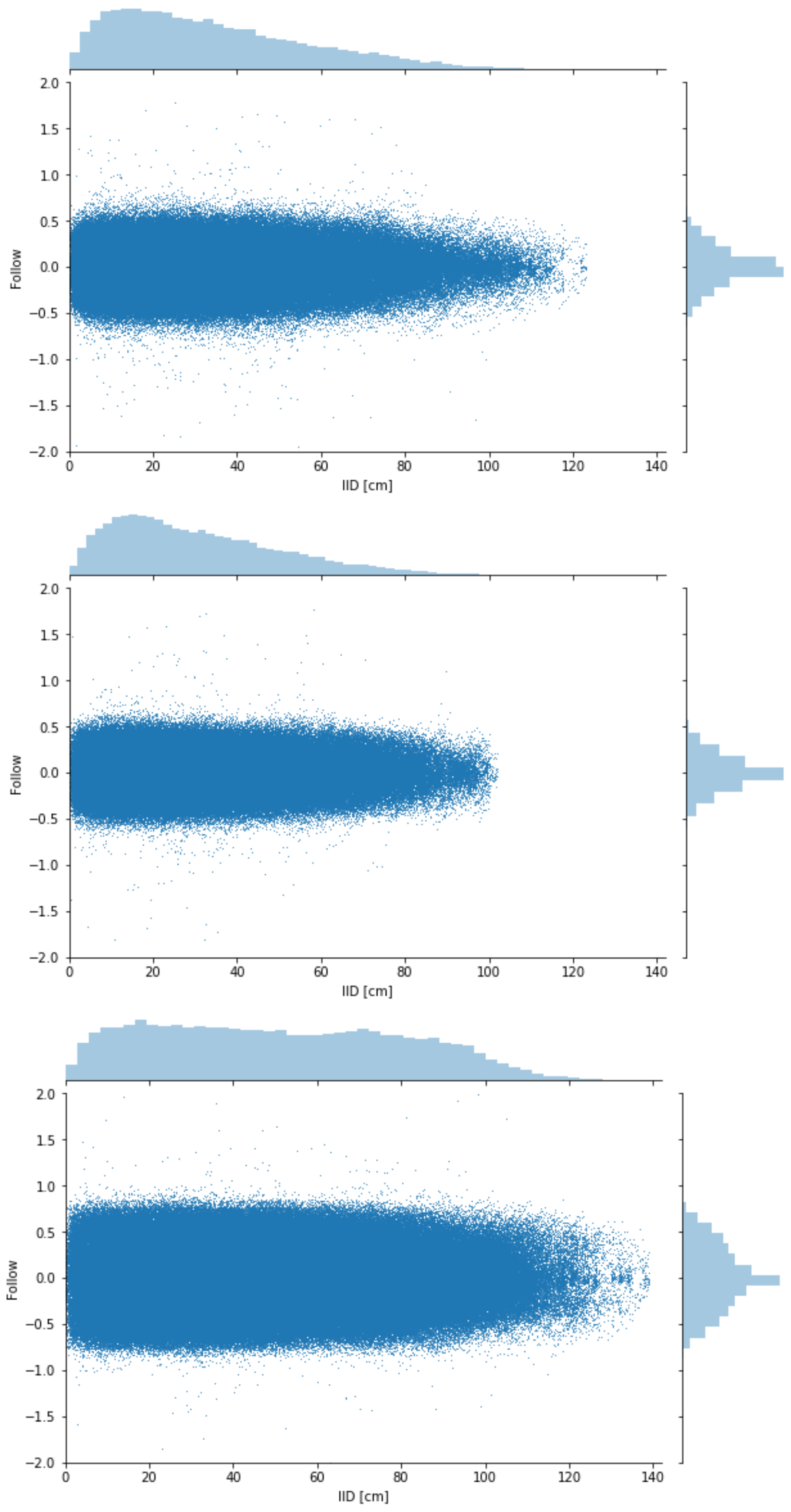


Figure 31: Live fish network: Follow / IID at epochs 13, 16, and 20

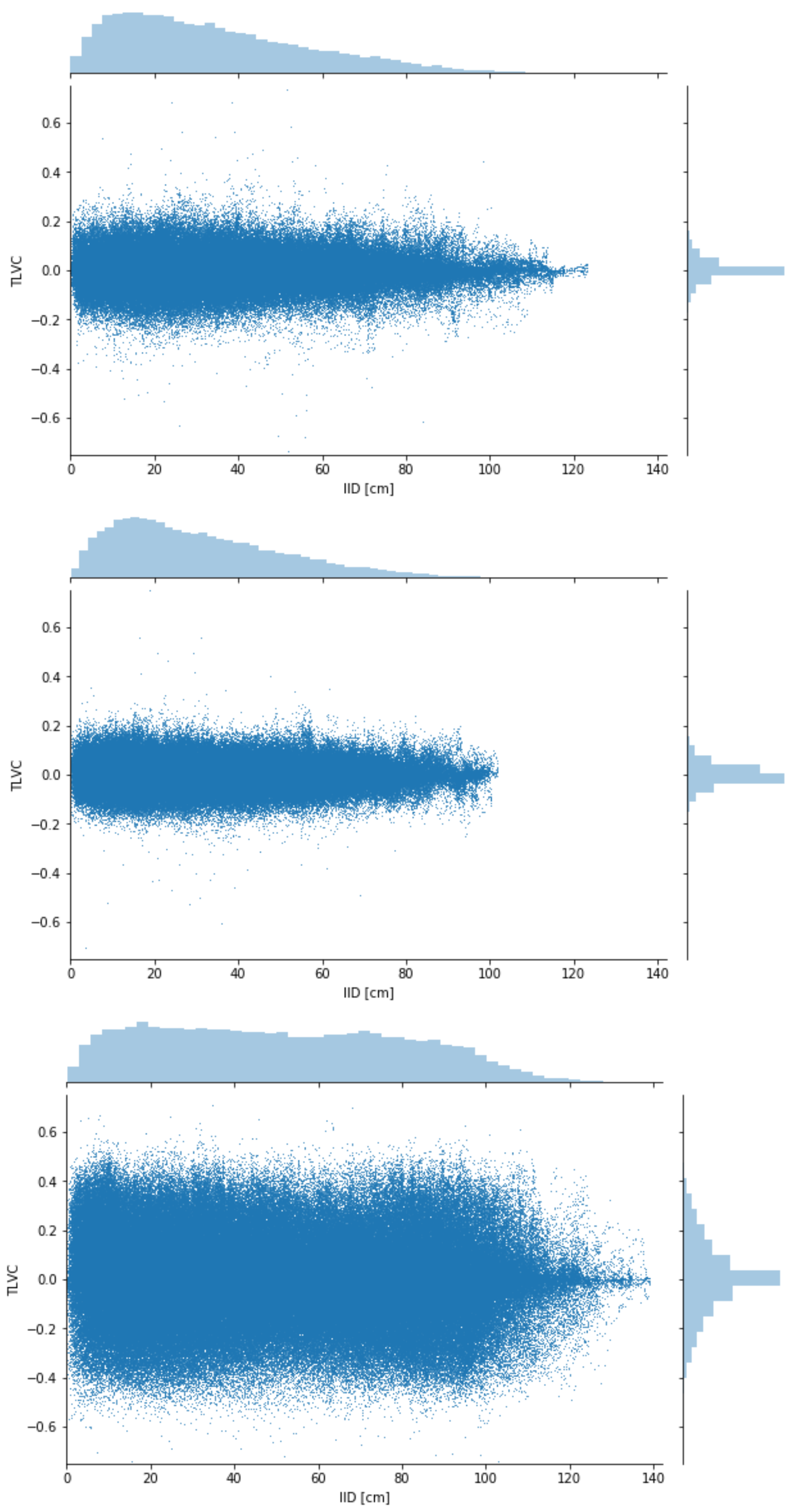


Figure 32: Live fish network: TLVC / IID ($0.3s \leq \tau \leq 1.3s$) at epochs 13, 16, and 20

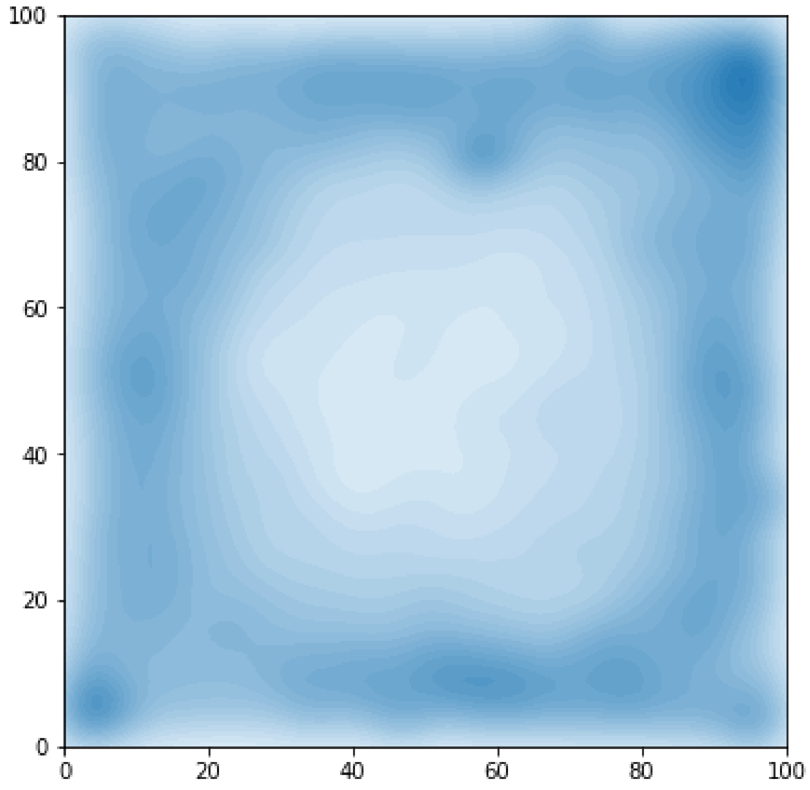


Figure 33: Live fish: Average tank positions

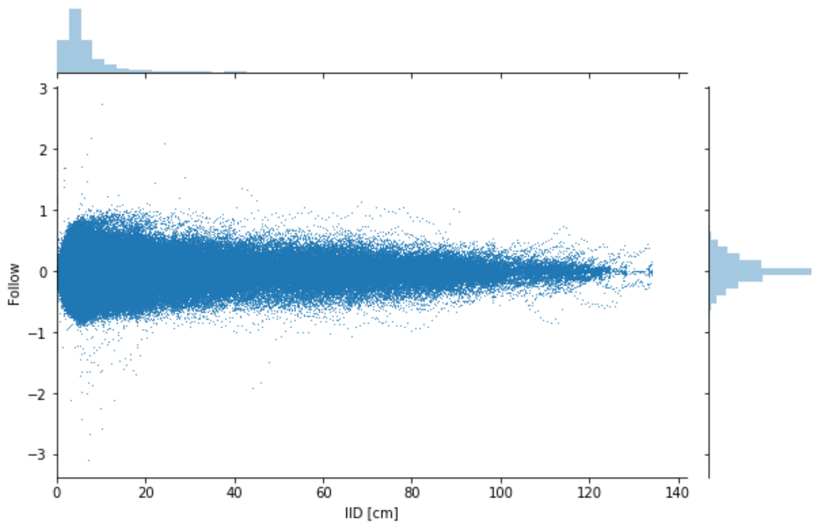


Figure 34: Live fish: Follow / IID

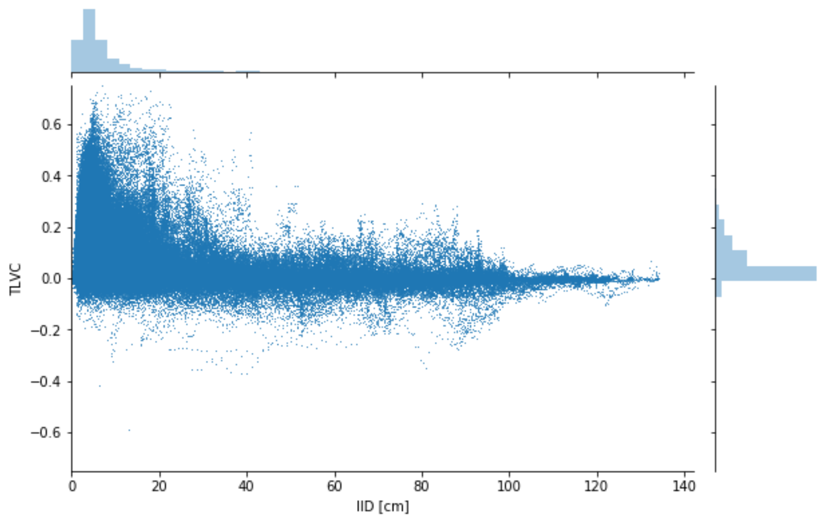


Figure 35: Live fish: TLVC / IID ($0.3s \leq \tau \leq 1.3s$)

6 Conclusions

In this thesis I was able to train an RNN to successfully imitate the behavior of the deterministic couzin model. Furthermore, I was able to train another RNN to partially imitate fish behavior. To the best of my knowledge this is the first work that has done either. I was not, however, able to fully imitate fish behavior, which is why further study of this subject is necessary.

Unfortunately, I was - due to time constraints - not able to run the trained RNNs in the real world with the RoboFish system and capture data on how they interact with live guppy fish. Had I been able to do so I would have liked to have humans try to distinguish the real from the robot fish based on their trajectories. Future work should strive to employ such tests.

Given how many different hyper parameters need to be carefully chosen when dealing with neural networks it is also always possible to try out a different design, or to further refine the current one.

6.1 Gaussian mixtures

While multinomial prediction of locomotion has been shown to be a viable path, the number of bins required were significantly larger (over 300 bins for live fish locomotion) than initially expected and the model inference speed could thus profit from using gaussian mixtures as the locomotion prediction instead. This would be of even greater impact should one wish to infer the network on embedded hardware.

6.2 No locomotion input

In the current design the network receives the old locomotion explicitly at each inference step. It would be interesting to investigate the consequences of removing those inputs and instead only providing the network with the current sensory input (view vectors). Intuitively, this should make the learning task harder, since there's more information the network has to encode the recurrent states.

6.3 Outline-based raycasting

The raycasting currently only uses agent positions without any information about agent orientation, shape, or size (limitation of the RoboTracker software). This necessarily puts a severe limit on what the network can actually learn, since e.g. if the difference in interaction behavior between two sets of fish pairs is due to their individual shape or size differences there is no way for the network to know this. It should prove beneficial to at least raycast against agent outlines instead of their raw positions.

6.4 Attention-based models

While standard RNNs have been the state of the art when it comes to temporal series prediction for a while now, recent advances in attenion-based models have shown both the viability of attenion-based RNNs [5] and attention networks [26] for machine translation tasks. It would be interesting to examine if these advances translate to fish locomotion prediction.

List of Abbreviations

RNN	Recurrent neural network
LSTM	Long short-term memory
ConvLSTM	Convolutional LSTM
TLVC	Time lagged velocity correlation
IID	Interindividual distance

List of Figures

1 IBehavior (old) . . . 3
2 Trackset viewer example . . . 4
3 Agent locomotion between two time steps . . . 5
4 View of agents raycasting: 180°fov, 3 sectors . . . 6
5 View of walls raycasting: 120°fov, 3 rays . . . 7
6 IBehavior (new) . . . 8
7 Example frame: Couzin torus behavior . . . 10
8 Example frame: Couzin dynamic parallel group behavior . . . 11
9 Distribution of live trackset lengths . . . 14
10 Number of frames lost by subtrack slicing . . . 15
11 Couzin torus: Angular speed distribution . . . 18
12 LSTM torus: Loss progression . . . 19
13 LSTM torus: Accuracy progression . . . 19
14 LSTM torus: Confidence progression . . . 20
15 LSTM torus: Single agent tank trajectory . . . 21
16 Couzin torus: Average tank positions . . . 22
17 LSTM torus: Average tank positions . . . 23
18 Couzin torus: Follow / IID . . . 23
19 LSTM torus: Follow / IID . . . 24
20 Couzin torus: TLVC / IID ($0s \leq \tau \leq 1s$) . . . 24
21 LSTM torus: TLVC / IID ($0s \leq \tau \leq 1s$) . . . 25
22 ConvLSTM+LSTM torus: Loss progression . . . 26
23 ConvLSTM+LSTM torus: Accuracy progression . . . 26
24 ConvLSTM+LSTM torus: Confidence progression . . . 27
25 ConvLSTM+LSTM torus: Single agent tank trajectory . . . 27
26 ConvLSTM+LSTM torus: Average tank positions . . . 28
27 Live fish network: Loss progression . . . 30
28 Live fish network: Accuracy progression . . . 30
29 Live fish network: Confidence progression . . . 31
30 Live fish network: Average tank positions at epochs 13, 16, and 20 . . . 32
31 Live fish network: Follow / IID at epochs 13, 16, and 20 . . . 33
32 Live fish network: TLVC / IID ($0.3s \leq \tau \leq 1.3s$) at epochs 13, 16, and 20 34
33 Live fish: Average tank positions . . . 35
34 Live fish: Follow / IID . . . 35
35 Live fish: TLVC / IID ($0.3s \leq \tau \leq 1.3s$) . . . 36
36 Example frame: Pair of live fish . . . 44

Bibliography

[1] Biotracker. https://git.imp.fu-berlin.de/biorobotics lab/biotracker/biotracker.

[2] Robot action control and virtual hardware. https://git.imp.fu-berlin.de/bioroboticslab/robofish/robocontrol/,.

[3] Robotracker. https://git.imp.fu-berlin.de/bioroboticslab/robofish/robotracker,.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv:1607.06450 [cs, stat]*, July 2016. URL http://arxiv.org/abs/1607.06450. arXiv: 1607.06450.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] David Bierbach, Tim Landgraf, Pawel Romanczuk, Juliane Lukas, Hai Nguyen, Max Wolf, and Jens Krause. Using a robotic fish to investigate individual differences in social responsiveness in the guppy. *bioRxiv*, page 304501, April 2018. doi: 10.1101/304501. URL https://www.biorxiv.org/content/early/2018/04/19/304501.

[7] Leo Cazenille, Bertrand Collignon, Yohann Chemtob, Frank Bonnet, Alexey Gribovskiy, Francesco Mondada, Nicolas Bredeche, and José Halloy. How mimetic should a robotic fish be to socially integrate into zebrafish groups ? *Bioinspiration & Biomimetics*, 2017. ISSN 1748-3190. doi: 10.1088/1748-3190/aa8f6a. URL http://iopscience.iop.org/10.1088/1748-3190/aa8f6a.

[8] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[9] Iain D. Couzin, Jens Krause, Richard James, Graeme D. Ruxton, and Nigel R. Franks. Collective Memory and Spatial Sorting in Animal Groups. *Journal of Theoretical Biology*, 218(1):1–11, September 2002. ISSN 00225193. doi: 10.1006/jtbi.2002.3065. URL https://linkinghub.elsevier.com/retrieve/pii/S0022519302930651.

[10] Eyrun Eyjolfsdottir, Kristin Branson, Yisong Yue, and Pietro Perona. Learning recurrent representations for hierarchical behavior modeling. *arXiv:1611.00094 [cs]*, October 2016. URL http://arxiv.org/abs/1611.00094. 00001 arXiv: 1611.00094.

[11] Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *arXiv:1512.05287 [stat]*, December 2015. URL http://arxiv.org/abs/1512.05287. arXiv: 1512.05287.

[12] Kiran Girdhar, Martin Gruebele, and Yann R. Chemla. The Behavioral Space of Zebrafish Locomotion and Its Neural Network Analog. *PLOS ONE*, 10(7):e0128668, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0128668. URL `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0128668`.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[14] J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tâche, I. Saïd, V. Durier, S. Canonge, J. M. Amé, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart, and J. L. Deneubourg. Social Integration of Robots into Groups of Cockroaches to Control Self-Organized Choices. *Science*, 318(5853):1155–1158, November 2007. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1144259. URL `http://science.sciencemag.org/content/318/5853/1155`.

[15] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[16] Hiroyuki Iizuka, Yosuke Nakamoto, and Masahito Yamamoto. Learning of Individual Sensorimotor Mapping to Form Swarm Behavior from Real Fish Data. In *The 2018 Conference on Artificial Life*, pages 179–185, Tokyo, Japan, 2018. MIT Press. doi: 10.1162/isal_a_00039. URL `https://www.mitpressjournals.org/doi/abs/10.1162/isal_a_00039`.

[17] A. Khan and F. Zhang. Using recurrent neural networks (RNNs) as planners for bio-inspired robotic motion. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1025–1030, August 2017. doi: 10.1109/CCTA.2017.8062594.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Tim Landgraf, David Bierbach, Hai Nguyen, Nadine Muggelberg, Pawel Romanczuk, and Jens Krause. *RoboFish: Increased acceptance of interactive robotic fish with realistic eyes and natural motion patterns by live Trinidadian guppies*, volume 11. January 2016. doi: 10.1088/1748-3190/11/1/015001.

[20] Ugo Lopez, Jacques Gautrais, Iain D. Couzin, and Guy Theraulaz. From behavioural analyses to models of collective motion in fish schools. *Interface Focus*, 2(6):693–707, December 2012. ISSN 2042-8898, 2042-8901. doi: 10.1098/rsfs.2012.0033. URL `http://rsfs.royalsocietypublishing.org/content/2/6/693`.

[21] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks. *arXiv:1804.07612 [cs, stat]*, April 2018. URL `http://arxiv.org/abs/1804.07612`. arXiv: 1804.07612.

[22] H. Moritz Maxeiner. Data: Imitation learning of fish and swarm behavior with Recurrent Neural Networks, September 2019. URL `https://doi.org/10.5281/zenodo.3457834`.

[23] H.J. Mönck, D. Bierbach, G.H.W. Gebhardt, N. Weimar, Jens Krause, and Tim Landgraf. Social competent robotic leaders enhance interaction performance with live fish.

[24] Francisco Romero-Ferrero, Mattia G Bergomi, Robert C Hinz, Francisco JH Heras, and Gonzalo G de Polavieja. idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. *Nature methods*, 16(2):179, 2019.

[25] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv:1506.04214 [cs]*, June 2015. URL `http://arxiv.org/abs/1506.04214`. arXiv: 1506.04214.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

A Acknowledgements

Besides the author, there are several people involved in the successful elaboration of a thesis. Hereby I want to thank some of them.

I am grateful to Prof. Dr. Tim Landgraf of the Biorobotics Lab at Freie Universität Berlin for giving me the opportunity to work on this project in his lab. I always received every support I needed but I also had a lot of freedom.

I want to thank Dr. David Bierbach for helping with his unique knowledge and experience with guppies.

Leon Sixt was always willing to discuss statistics and how to best evaluate results.

Benjamin Wild and Julian Stastny could always be counted on to provide invaluable insight into machine learning details.

Whenever I needed someone as a rubber duck or test group for design decisions Julian Petrasch could always be depended on to make time.

Für die zur Verfügung gestellte Rechenzeit möchte ich mich beim HPC-Dienst der ZEDAT, Freie Universität Berlin bedanken.

B Appendix

The data is available in [22].

The source code is available at
`https://git.imp.fu-berlin.de/biorobotics lab/robofish/mm_master`

The official thesis source code git commit is
`c2e0375d2e5f7722eb78a00b38c58185b261a824`

The QR code combining the above is as follows:

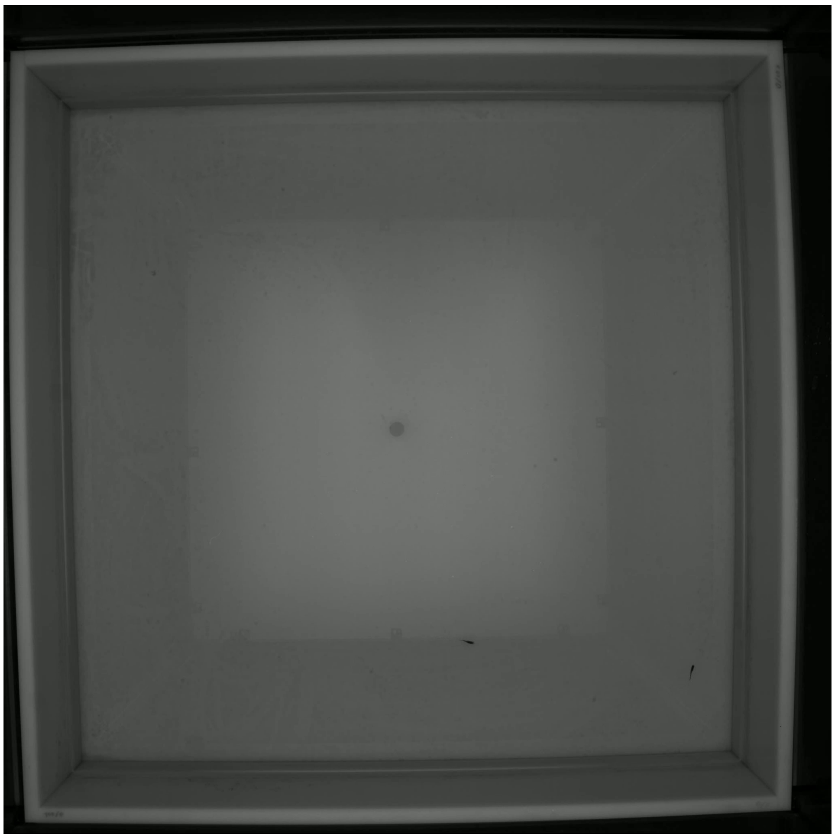


Figure 36: Example frame: Pair of live fish

Video	Fish A	Fish B
CameraCapture2019-05-03T11_22_33_8108	22.54	28.45
CameraCapture2019-05-03T14_58_30_8108	23.93	23.26
CameraCapture2019-05-03T15_46_57_8108	22.58	22.82
CameraCapture2019-05-06T11_20_16_8108	25.54	23.61
CameraCapture2019-05-06T11_44_55_8108	23.89	22.74
CameraCapture2019-05-06T14_05_36_8108	24.56	21.08
CameraCapture2019-05-06T15_30_28_8108	26.97	22.16
CameraCapture2019-05-06T16_21_16_8108	26.95	25.40
CameraCapture2019-06-20T15_35_23_672	24.51	21.79
CameraCapture2019-06-20T16_13_11_672	24.74	26.43
CameraCapture2019-06-27T15_23_01_9052	20.23	19.87
CameraCapture2019-06-28T11_24_26_9052	18.20	19.34
CameraCapture2019-06-28T11_57_23_9052	17.52	17.37
CameraCapture2019-06-28T13_14_08_9052	20.49	19.88
CameraCapture2019-06-28T13_58_01_9052	17.97	18.82
CameraCapture2019-06-28T15_02_41_9052	15.15	17.87
CameraCapture2019-06-28T15_40_01_9052	33.10	23.04

Table 6: Observed fish standard lengths in millimeters