Deep learning tools to study collective behaviour

Francisco Romero Ferrero

Dissertation presented to obtain the
Ph.D. degree in Neuroscience

Instituto de Tecnologia Química e Biológica António Xavier | Universidade Nova de Lisboa

Oeiras,
April, 2021

UNIVERSIDADE NOVA DE LISBOA

Deep learning tools to study collective behaviuor

Francisco Romero Ferrero

Dissertation presented to obtain the
Ph.D. degree in Neuroscience

Instituto de Tecnologia Química e Biológica António Xavier | Universidade Nova de Lisboa

Research work coordinated by:

Champalimaud Foundation

Oeiras, April, 2021

Deep learning tools to study collective behaviour

Francisco Romero Ferrero

A Dissertation
Presented to the Faculty
of Universidade Nova de Lisboa
in Candidacy for the Degree
of Doctor of Philosophy

Supervisor: Gonzalo G. de Polavieja

April 2021

To my parents: Loli and Juan.

Acknowledgements

When as a kid my parents asked me what I wanted to do for a living when I grew up, I naively replied that I wanted to be an "observer". All these PhD years has brought me quite close to what I really meant. This PhD journey has not only taught me what I know about science, but it has also allowed me to grow personally in many directions. This would have not been possible without the people that supported and accompanied me throughout this journey.

First of all, Gonzalo, to whom I am deeply grateful for opening me the doors of his lab 10 years ago when I was a naive undergraduate student. I can truly say that I have become the scientist I am now thanks to him. I feel extremely fortunate for all the knowledge and ideas that he has shared with me during the innumerable enlightening discussions about science and life. The work on this thesis is as much his accomplishment as it is mine. Thank you.

Fran, my close collaborator and "scientific sparring" fellow during these last years, for sharing with me his admirable analytical thinking and his passion for computers and best coding practices.

Mattia, my collaborator during the first years of the PhD, for introducing me to the world of deep learning and pair programming.

My thesis committee members, Alfonso Renart and Michael Orger, for being so supportive and enthusiastic about my project.

All the current and former members of the Polavieja lab for all the scientific discussions and fun outside the lab. Alfonso, for being a source of inspiration since my early days in the lab. Robert for teaching me everything he knew about building experimental setups. Julian and Gabriel, for all the joy and positive spirit. Antonia, Tiago and especially Vicky, for being my constant companions during this journey. Dean for proof reading this thesis.

The members of the Champalimaud Research (CR) platforms, especially Catarina, Sandra, Joana, João, Felipe, Artur, Ricardo, Paulo for all their great work.

The CR members, for all the science and fun during these years. The basketball aficionados for all those Monday games. The members of the science communication office and the Ar initiative for trusting my photography skills. All the music lovers with whom I shared rehearsals and stages. The Carey Lab

members, Megan, Catarina, Jovin, Rita, Hugo and Ana for considering me a friend of the lab, you all brought a lot of fun into this PhD. Pietro for all the chats about the difficulties of the academic system. Tomas and Vicky for always being such good friends and hosts, especially Vicky for all those delicious dishes. Bruno for being such an easy going flat mate. Special thanks to Tatiana and Ines, you two have always been there for the good and for the bad, I feel extremely fortunate to have you as friends.

I must mention other people that have helped me to remember that there is life outside of CR. Ledi, for all the conversations about saxophones, music and life. My university friends, for being there for nice and warm gatherings in my visits to Madrid. My friends from Beneixama, especially to Xavi, Cris and Cristobal, for all the musical adventures. All my aunts, uncles and cousins of the Romero, Ferrero and Bellot families for all the memorable gatherings in Beneixama.

Music, photography, my cameras and my instruments, especially my sax, have been my faithful companions during all these years of PhD. I feel fortunate to have found these two vocations which have helped me to keep my mind away from science when necessary. I am grateful to all musicians and photographers that have inspired my artistic side.

Cris and Fran, you have truly been my family in Algés. I cannot put into words all the moments that I have shared with you and that have made my life during these last years really joyful, especially during the last year of COVID-19 pandemic. I will never ever forget it. Your warmness, caring personalities and sense of humor, really defines you as extraordinarily good people.

Silvia, for all your love, companionship and support, I feel extremely lucky to have found you in my life because you always bring so much *vidilla* to it. My sister Aurora, Tista and Joan, for always being there when I need it.

Finally, *mama* (mom) and *tata* (dad), to whom I dedicate this thesis. Everything I have accomplished so far is thanks to your more than admirable labor as parents. You have done all in your hands to educate me with your best values and this has marked me with the principles of hard work and perseverance, so important to do good science. Thank you for always believing in me and for your unconditional support.

Título

Ferramentas de aprendizagem profuda para o estudo de colectivos animais

Resumo

Nesta tese, investigamos como o uso de deep learning (aprendizagem profunda) pode ajudar no estudo de colectivos animais. A principal vantagem desta abordagem é a possibilidade de produzir modelos extremamente exatos. Com esta ideia em mente, desenvolvemos idtracker.ai, um software que, a partir de um vídeo, extrai as trajectories de cada animal presente no colectivo, sem necessidade de marcadores físicos. Ao contrário da tecnologia anteriormente desenvolvida no nosso grupo (idTracker), este algoritmo faz uso de redes neuronais convolucional a fim de identificar, de maneira auto-supervisionada, a identidade de cada um dos animais. Esta estratégia possibilita a monitorização, de alta fidelidade, de 100 indivíduos, de várias espécies, mesmo na presença de colisões e oclusões. Em adição ao idtracker.ai, desenvolvemos também: idmatcher.ai. Esta plataforma interage facilmente com idtracker.ai e é capaz de detetar os mesmos animais mesmo que presentes que múltiplos vídeos diferentes. Adicionalmente, testámos como estes métodos podem ser explorados a fim de derivar regras de interação entre indivíduos do colectivo animal. A dificuldade, neste caso, é garantir que o resultado do modelo é interpretável, já que a dimensionalidade e, inerentemente, a complexidade, de métodos envolvendo redes neuronais resulta no que é habitualmente chamado de "caixas negras". Trabalho anteriormente desenvolvido no laboratório sugere o uso de deep attention networks (redes neuronais com capacidade de atenção) como possível solução para este problema. Dada a sua estrutura modular e de baixa dimensão, este modelo, conseguiu, não só prever o comportamento dos animais, mas também revelou regras que governam as interações entre os animais: os animais pesam o comportamento dos outros animais, dependendo do seu comportamento. Esta flexibilidade pode ser explorada, por exemplo, a fim de dar mais peso a animais que trazem nova informação para o grupo. Para testar esta hipótese, desenvolvemos novas experiências, usando peixe-zebra, onde alguns elementos do grupo foram condicionados a procurar comida, numa zona específica da arena, na presença de uma indicação luminosa. Usamos as ferramentas

previamente mencionadas para extrair e identificar os animais do colectivo a fim de classificar cada indivíduo como "informado" ou "não-informado". Descobrimos que, em adição ao rápido movimento de peixes informados em direção à luz, após um pequeno intervalo de tempo, animais não-informados, também se movem na mesma direção. Usando redes neuronais, estudamos como estes últimos peixes agregam, e usam, a informação dos indivíduos informados, para guiar o seu processo de decisão. Concluímos que estes pesam mais a informação de animais que se movem a velocidades mais elevadas, fazendo que que animais não-informados sigam os animais informados.

Abstract

In this thesis, we investigate how deep learning can help to study animal collectives. The main strength of this approach is the ability to produce very accurate models. We developed idtracker.ai to extract from a video the trajectory of each animal in the collective. idtracker.ai is a marker-less multi-animal tracking system that works by identifying each animal, like its predecessor idTracker. The difference is that it trains a convolutional neural network in a self-supervised manner for animal identification. With this strategy, idtracker.ai can track with high identification accuracy sparse collectives of any species of up to 100 individuals even if animals touch or occlude each other frequently along the video. A new tool, idmatcher.ai, works seamlessly with idtracker.ai to identify animals across different videos. We also tested how deep nets can help to find interaction rules among animals in collectives. The difficulty, in this case, is how to make the model interpretable about the animals' interactions since the high complexity of the deep nets makes them basically "black boxes". Previous work in the lab proposed using a deep attention network to model an animal in the collective. This model could not only predict the behaviour of the animals but also was insightful thanks to an organization into modules of low input/output dimensionality. These results revealed a flexible information aggregation rule by which an animal could weigh other animals in the collective differently depending on their behaviour. We reasoned that this rule could in principle allow animals to weigh more other animals that bring new information to the group. To test this hypothesis, we performed new experiments with zebrafish collectives in which some fish in the group were trained to look for food in a given location after we turned a light on, and the rest of the group did not have such information. We used idtracker.ai to extract the trajectories and idmatcher.ai to identify which animals in the collective were informed and uninformed. We found that the informed animals moved fast towards the light and, with a delay, the uninformed fish also moved in the same direction. We used the deep attention network to study how the uninformed animals aggregated the information from their neighbours. We found that they weighted higher the animals moving at higher speeds, effectively following the informed more than the uninformed animals.

Author Contributions

This thesis comprises two main studies which appear in Chapters 2 and 3. Chapters 2 is an adapted version of the work already published by Romero-Ferrero et al. (2019). Author contributions appear at the end of each chapter. Francisco Romero Ferrero wrote this thesis with input from Gonzalo G. de Polavieja and Francisco J.H. Heras.

Financial Support

This work was funded by Fundação para a Ciência e a Tecnologia (FCT) and Fundação Champalimaud. Francisco Romero Ferrero was a student of the International Neuroscience Doctoral Program (INDP) 2014 and he was supported by the Ph.D. fellowship PD/BD/105946/2014 from FCT.

Overview

This thesis is structured into 4 Chapters. In Chapter 1, I briefly introduce the field of collective behaviour and present the zebrafish as an animal model for collective animal behaviour. Then, I review previous solutions on multi-animal tracking and introduce convolutional neural networks (CNNs). Finally, I summarize the relevance of deep attention networks models for study animal collectives, in particular for the problem of information aggregation. Chapters 2 and 3 include the original body of work produced during my doctoral studies. In Chapter 2, I present the already published multi-animal tracking software idtracker.ai. First, I explain idtracker.ai's algorithm. Next, I introduce the current interface of the software emphasizing the new features that can help users to obtain a better tracking performance. I continue presenting the already published results about the validation of the algorithm. Finally, I discuss the main limitations of the algorithm, and propose future directions of improvement. Chapter 3 includes original unpublished results of a work in progress. I present the application of deep attention networks to the study of information aggregation in animal collectives. In particular, I present new experiments of zebrafish collectives in which a fraction of the group has privileged information about a visual cue in the environment (informed fish) and the rest has no such information (uninformed fish). I summarize the behaviour during the training sessions preceding the experiments, and analyse the behaviour of the animals during the experiments, when the informed and uninformed fish swam together. I show that the uninformed fish are influenced by the informed ones after the presentation of the visual cue. I continue showing that a deep attention network model can predict well the turning side of the uninformed fish. An analysis of this model reveals a flexible aggregation rule that depends on the behaviour of the neighbours. I show that uninformed fish that followed the behaviour of the informed fish paid more attention to them. Lastly, I discuss some open-ended questions and propose further analysis. In Chapter 4, I briefly review the main finding of the thesis and discuss perspectives on the future application of deep learning techniques to multi-animal tracking, and about the project presented in Chapter 3.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Collective animal behaviour

Animal behaviour can be defined as the set of actions that animals do to adapt to and survive in a given environment (Darwin & Prodger, 1998; Gomez-Marin et al., 2014). The interaction among different individual units in the animal body (e.g. genes, neurons, muscles) converges to produce movements as a response to external stimuli. At the same time, animals living in groups interact among each other producing patterns on scales larger than themselves: flocks of birds, ant trials, termite nests, fish schools, human societies, etc. Understanding the emergence of these collective phenomena at all scales is a challenging endeavour that occupies many disciplines, from genetics and neuroscience to physics and mathematics (Sumpter, 2010; Easley et al., 2010).

Living in groups allows individuals to benefit from the information of others (Danchin et al., 2004). One of the earliest efforts to understand how the knowledge of each member in a group can be used to make better decisions was the theorem first expressed by the Marquis de Condorcet in his 1785 work *Essay on the Application of Analysis to the Probability of Majority Decisions* (De Condorcet et al., 2014). Since then, the study of the behaviour of the individuals units of a group and how the information in the collective can be used to make better decisions has attracted the attention of many disciplines like economics (Easley et al., 2010), psychology (Haslam et al., 2020), online social networks analysis (Guille et al., 2013), multi-agents systems in artificial intelligence (Dorri et al., 2018), and ecology (Westley et al., 2018), among others.

The study of the collective phenomena from real-world data of human social networks or animal collectives in the wild has the added challenge of controlling the multiple factors that can affect the behaviour of the individuals in the group (Hughey et al., 2018). However, studying collective behaviour using groups of animals under laboratory conditions allows a more precise control and manipulation of the environment and a detailed measurement of their behaviour.

Just as physicists measured the movement of the planets to study and model the forces that keep them together (Newton & Chittenden, 1850), measuring, analysing, and modelling the movements of the animals in a controlled environment can reveal important aspects of their behaviour (Anderson & Perona, 2014;

Egnor & Branson, 2016; Brown & de Bivort, 2018; Berman, 2018; Pereira, Shaevitz, & Murthy, 2020). More interestingly, animals are highly complex systems that live in rapidly changing environments, which makes their behaviour very difficult to predict (Honegger & de Bivort, 2018). In addition, the number of variables that one can measure from an animal and its surroundings is enormous, making the study of animal behaviour a very high-dimensional problem (Gomez-Marin et al., 2014). While this might seem hopeless, the study of collective animal behaviour has shown multiple examples where mathematical modelling has helped to understand animal behaviour (Sumpter, 2010).

1.1.1 Fish collective behaviour

Living in groups allows animals to benefit from the information of others, for example during foraging (Giraldeau & Caraco, 2018) or when avoiding risks (Pérez-Escudero & de Polavieja, 2017). Many animal species that live in groups show different levels of self-organization (Nicolis, 1977) that emerge from simple local interactions between individuals of the collective without a centralized control (Sumpter, 2006). A particular example of these phenomena that has brought the attention of scientists for years are fish schools (Partridge, 1982; Partridge et al., 1983; Parrish et al., 2002; Viscido et al., 2004).

Measuring and controlling the environment of fish collectives in the wild poses many challenges (Lennox et al., 2017). Luckily, many species of fish like the mosquitofish (*Gambusia holbrooki*), golden shiner (*Notemigonus crysoleucas*), threespine stickleback (*Gasterosteus aculeatus*), guppies (*Poecilia reticulata*), rummy-nose tetra (*Hemigrammus rhodostomus*), medaka (*Oryzias latipes*), zebrafish (*Danio rerio*), giant danio (*Devario aequipinnatus*), and danionella (*Danionella translucida*), can be raised and/or maintained in laboratory conditions where they also show schooling and shoaling like behaviours (Herbert-Read et al., 2011; Katz et al., 2011; Faria et al., 2010; Davis et al., 2017; Crosato et al., 2018; Jeon et al., 2013; Miller & Gerlai, 2012a; Viscido et al., 2004; Schulze et al., 2018). Among them, the zebrafish stands out for being a well established model in behavioural neuroscience (Levin & Cerutti, 2009; Kalueff et al., 2013; Oliveira, 2013; Gerlai, 2019; Orger & de Polavieja, 2017).

1.1.2 Zebrafish as an animal model for collective behaviour

The zebrafish is a small teleost fish that can be found shoaling in groups of variable sizes in natural shallow waters of India and Myanmar (Engeszer et al., 2007; Spence et al., 2008; Suriyampola et al., 2016).

Under laboratory conditions, Hinz & de Polavieja (2017) did a meticulous characterization of the ontogeny of the zebrafish collective behaviour and showed that after 2 weeks they swim in groups. After the first month of development, the zebrafish robustly shows shoaling and schooling behaviours (Buske & Gerlai, 2011, 2012; Miller & Gerlai, 2012b). The juvenile zebrafish at the age of 1 month are much smaller than the adult zebrafish. This allows studying their collective behaviour in relatively small experimental setups.

Studying free swimming shoals and schools of fish can provide insight about the nature of the underlying local interactions (Herbert-Read et al., 2011; Katz et al., 2011; Hinz & de Polavieja, 2017; Heras et al., 2019). However, it is sometimes necessary to disturb the collective to learn causal effects about the interactions between animals, such as how animals aggregate information from neighbours that have more knowledge about the environment. This can be done by training some animals in the group to move in a preferred direction after the presentation of a visual cue, like Strandburg-Peshkin et al. (2013) did with golden shiner fish.

Valente et al. (2012) characterized the ontogeny of classical and operant learning in groups of zebrafish showing that the ability to learn appears after 3 weeks post fertilization. While they performed the experiments using noxious electroshock stimuli, zebrafish can also learn to associate visual cues with food (Williams et al., 2002; Colwill et al., 2005). This suggests that, in principle, the juvenile zebrafish can be used to perform experiments, others than free swimming, in which different members of the group have different information about the environment.

1.2 Multi-animal tracking

The quantitative study of animal behaviour requires obtaining accurate and highly resolved data sets. From the multiple variables that can be measured

from an animal, its location throughout time in a given environment, i.e. its trajectory, has been proven to contain valuable information for the study of animal behaviour.

Scientists have used many techniques to obtain animal trajectories. Direct annotation of the location of the animals (Kareiva & Shigesada, 1983; Angilletta Jr et al., 2008) is prone to subjective biases of the observer and in addition it can influence the actual behaviour of the animals (Altmann, 1974). Footprint trails are a valuable source of information to obtain trajectories of extinct animals (Alexander, 1976) or to non-invasively track the path of animals in the snow (Fortin et al., 2005), however they lack the temporal component of the trajectory.

In the last decades, technological advances like acoustic telemetry, radio telemetry, harmonic radar, RFID, and GPS have allowed tracking terrestrial, aquatic or aerial animals in their natural habitat using animal-borne tags or biologgers (Daniel Kissling et al., 2014; Kays et al., 2015; Lennox et al., 2017; Toledo et al., 2020). Advances in computer vision have made video-based tracking methods to become a standard less invasive alternative that can provide high spatial and temporal resolutions with the possibility of tracking animals in the wild (Dell et al., 2014; Miao et al., 2019; Haalck et al., 2020; Francisco et al., 2020) and under laboratory conditions (e.g. Branson et al. (2009); Pérez-Escudero et al. (2014); Geissmann et al. (2017); Rodriguez et al. (2018); Werkhoven et al. (2019); Romero-Ferrero et al. (2019); Tadres & Louis (2020); Gal et al. (2020); Walter & Couzin (2021)).

1.2.1 Automated video-based animal tracking

Since the pioneering work of Eadward Muybridge (Muybridge, 1887), scientists have used sequences of images to study animal behaviour (Elliott et al., 1977; Bayne & Scullard, 1978). The development of modern digital videography has allowed scientists to automatize the process of capturing, tracking, and analyzing animal movements. Dell et al. (2014) did an extensive review on automated video-based tracking methods for animal behaviour. More recently, Werkhoven et al. (2019) and Sridhar et al. (2019) also reviewed and compared the features of multiple state of the art open-source video-based tracking software to obtain trajectories from videos in laboratory conditions.

Depending on the scientific question, researchers require trajectories that resolve the movement of the animals with different levels of spatio-temporal resolutions and information about the animals. For example, some scientific questions require fine details about the movement of the body parts of the animals (e.g. Machado et al. (2015); Marques et al. (2018); Mathis et al. (2018); Pereira et al. (2019); Graving et al. (2019)). Others require obtaining real-time information about the location of the animals to manipulate their environment in a closed-loop setup (e.g Lopes et al. (2015); Geissmann et al. (2017); Rasch et al. (2016); Marques et al. (2018); Werkhoven et al. (2019); Tadres & Louis (2020); Kane et al. (2020)). While sometimes, 3D trajectories are needed, but this typically requires extra hardware for the calibration and synchronization of different camera views (Cavagna et al., 2010; Straw et al., 2011; Butail & Paley, 2012; Stowers et al., 2017; Qian & Chen, 2017; Nath et al., 2019; Cheng et al., 2018; Pedersen et al., 2020).

Many questions in collective animal behaviour can be answered in experiments where animals move in quasi-planar environments and without manipulating the environment in a closed-loop. In these cases, 2D trajectories can be obtained off-line after the acquisition of the video. An important consideration when studying interactions in animal collectives is to obtain sets of trajectories where each trajectory corresponds to a unique animal in the group. We will call these types of trajectories identity-consistent trajectories. Identity-consistent trajectories are necessary to study behaviours like aggression (Laan et al., 2018), territoriality (Pérez-Escudero et al., 2014; Rasch et al., 2016), leadership and social influence (Strandburg-Peshkin et al., 2018), and other social interactions (e.g. Lorbach et al. (2015)).

1.2.2 Multi-animal video-based tracking with identification under laboratory conditions

1.2.2.1 The identities swap problem

Robie et al. (2017) reviewed the different steps and considerations to obtain trajectories of multiple animals in quasi-planar environments with a single camera view and under laboratory conditions. To obtain identity-consistent trajectories,

many tracking algorithms initialize the identities of the animals in the first frame of the video and use assumptions about the motion of the animals to match the location of the animals from frame to frame in the video. However, when animals get very close, or when the dynamics of the animal movement change abruptly these algorithms can fail to match the real location of the animals in the next frame, which generates identity swaps. Even if this occurs in very few frames, the errors will propagate generating sets of trajectories that are not identity-consistent (Pérez-Escudero et al., 2014; Kim et al., 2015).

When using these algorithms to obtain identity-consistent trajectories, users must check the frames in the video where potential identity swaps might have occurred. This is a very arduous and time-consuming task. Moreover, when animals are very similar to each other it might be difficult for the human eye to tell them apart from each other. A possible solution to the identity swap problem is to artificially mark the animals so that they can be easily identified (see Robie et al. (2017) for a review of different marking methods). Then the identity of the animals can be automatically matched after each potential identity swap using computer vision methods that can be incorporated to the tracking algorithm (e.g. Boenisch et al. (2018); Gal et al. (2020)). However, these techniques might require anesthetizing the animals and modifying their natural appearances which can potentially change their behaviour (Dennis et al., 2008; Dahlbom et al., 2011).

To answer some questions about animal collectives it is not necessary to maintain the identity of the animals along the video. In these cases, one can still compute population level measures using anonymous individual trajectory fragments that the tracking system detects as belonging to single animals with a high certainty. These individual anonymous trajectory fragments are typically known as "tracklets" (Robie et al., 2017) or simply as "fragments" (Pérez-Escudero et al., 2014; Rodriguez et al., 2017).

1.2.2.2 Marker-less tracking by identification

Pérez-Escudero et al. (2014) proposed for the first time a multi-animal marker-less tracking algorithm based on identifying each animal by its appearance, idTracker. Briefly, instead of matching the location of the animals from frame to frame of the video, idTracker first computes all fragments in the video and then it assigns

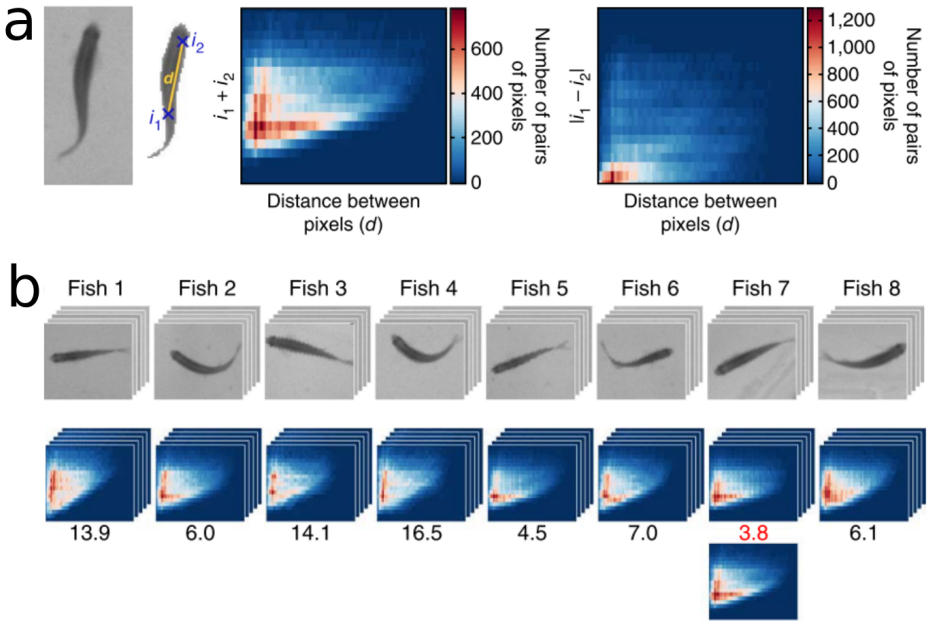

Figure 1.1. **idTracker animal identification paradigm.** Figure adapted from Pérez-Escudero et al. (2014). Summary of the identification paradigm that idTracker uses to assign an identity to each animal in a frame of a given fragment. **a.** Feature maps constructed from the blob of pixels of a given fish in a video. **b.** Collection of reference images and maps for each individual in a video of 8 zebrafish. The identity of a given fish image is determined to be identity of the reference collection with minimum distance to the feature map of the given fish.

a unique identity to each of them based on the visual features of the animals in the video.

For simplicity, we will ignore how idTracker constructs the fragments and we will focus on explaining the identification paradigm. idTracker extracts *blobs* of pixels (Moeslund, 2012) for each animal in the video. For each blob it constructs two feature maps that summarize the animal appearance at a given frame of the video (see Figure 1.1a). These maps are 2D-histograms that consider pairwise distances between pixels and differences and sum of pixel intensities. This makes the feature maps invariant under rotations and species-agnostic. Thus, each fragment in the video can be seen as a collection of such maps.

Using parts of the video where all animals are separated from each other, it automatically constructs a reference collection of 3000 maps for each animal (see Figure 1.1b). Typically, fragments are shorter than 3000 frames. Therefore, idTracker computes a similarity measure between maps in different fragments to gather samples to generate the reference collection. This reference collection acts as a fingerprint of the appearance of the animal throughout the whole video, i.e. it represents the animal identity.

To assign the identity of each fragment in the video, idTracker first computes pairwise similarities between the maps of each blob in the fragment and the maps in the reference collection of each animal identity. For each blob idTracker assigns the identity of the reference collection with the smallest distance (see Figure 1.1b). Then, the identities of all blobs in a fragment are combined with a Bayesian framework that assigns the identities of all fragments in the video preventing identity duplications in the same frame.

With this identification method, idTracker could extract highly accurate identity-consistent trajectories for small groups of animals of any species (≤ 20 individuals).

While idTracker feature maps can be computed without requiring any prior knowledge about the animal appearance, they are computationally expensive to generate (Rasch et al., 2016; Rodriguez et al., 2017). But more importantly, the identification paradigm of idTracker requires an enormous number of comparisons between the reference collection of maps and each blob in the video, which does not scale well for large groups.

After idTracker, other multi-animal tracking algorithms have integrated the idea of using pre-computed visual features to identify fragments in the video and obtain identity-consistent trajectories (Rasch et al., 2016; Bai et al., 2018; Rodriguez et al., 2017). Next, we briefly review the ones we consider more relevant to contextualize the work presented in this thesis.

The tracking software ToxTrac (Rodriguez et al., 2018) integrated a marker-less animal identification algorithm, ToxId (Rodriguez et al., 2017), as an option in their tracking pipeline. ToxId uses the Hu's Seven Moments Invariants (Sivaramakrishna & Shashidharf, 1997; Q. Chen et al., 2004) of the image enclosing an animal as features for identification. These features are faster to compute than

the ones of idTracker. Using these visual features it defines a similarity measure between pairs of fragments using only the blobs that are most different between each other. ToxId identifies the fragments in parts of the video where all animals are visible using the Hungarian algorithm (Kuhn, 1955). The rest of the fragments are assigned using a greedy algorithm that minimizes the number of identity duplications. This strategy allows ToxTrac to track videos of small group sizes (≤ 11 individuals) faster and with accuracies similar to those of idTracker.

The identification methods of idTracker and ToxId do not require prior knowledge about the animals appearance, being general for any animal species. One disadvantage of using pre-computed features is that they might not capture other important aspects of the animal appearance that could be important for the identification.

Rasch et al. (2016) proposed a real-time tracking algorithm, xyTracker, that learns a representation of the appearance of the animals directly from the animal images extracted from the video. They show that xyTracker can track groups of zebrafish in real-time for long periods. To validate that the algorithm can maintain the identities of the animals over time, they compared the trajectories obtained with xyTracker with the ones obtained with idTracker in a video of 5 zebrafish. The results showed very small differences between the trajectories of both algorithms, which suggested a similar identification performance to idTracker. However, they did not report actual identification accuracies.

Importantly, though, the authors showed a comparison of the classification performance of different machine learning methods in a dataset of images of 5 zebrafish extracted from the same video. The results showed that convolutional neural networks (CNNs) obtain the best classification accuracy on test data.

While the authors claim that xyTracker can use CNNs for animal identification in videos, they do not show tracking performance results for this case. One reason could be that the long training times of the CNNs (20 minutes) are incompatible with the goal of tracking animals in real-time.[1] But most importantly, the authors do not propose any method to automatically obtain sufficient training data to train the CNNs during the tracking process. Note that for the classification accuracy tests the authors used 2000 images per animal, but these images

[1] The authors reported a training time of 20 minutes in (Rasch et al., 2016).

were obtained after tracking a video of 5 zebrafish and manually validating the trajectories.

1.2.3 Convolutional neural networks for object identification

The problem of identifying objects from images based on their appearance is a very well described problem in machine learning known as supervised image classification or image recognition (Forsyth & Ponce, 2002). This problem requires learning a functional mapping between the set of pixels in the image and the class of the object that it represents. Three decades ago, the seminal work of Yann LeCun (LeCun et al., 1989, 1989b) showed how the back-propagation algorithm (Rumelhart et al., 1986) could be used to train artificial neural networks (ANNs) (Basheer & Hajmeer, 2000) to classify single digits from images.

The most important result of LeCun's work was the use of a new architecture of neural network that exploited the fact that images typically show repeated local features. This architecture had a set of filters or receptive fields that are multiplied locally in different patches of the image generating feature maps. Since the weights of these filters are the same for all the regions of the image, this reduces the number of parameters of the ANN by a large amount and improves the generalization ability of the network (LeCun et al., 1989). Currently, the artificial neural networks that use this type of multiplication filters are known as CNNs.

Training ANNs requires performing a high number of matrix multiplications. Graphical processing units (GPUs), that were originally develop for rendering images in video games, are specially designed to perform large amounts of matrix multiplications simultaneously in a distributed way (Harris, 2008). Raina et al. (2009) showed for the first time that large ANNs could be efficiently trained using GPUs. But it was the work of Krizhevsky et al. (2012) that showed the real potential of training CNNs using GPUs to classify images of real objects and not just digits (Alom et al., 2018).

The advances in computing power provided by the new GPUs have made possible the use and development of visual pattern recognition methods based

on CNNs, which have reached top performance in most computer vision tasks, specially in object detection and image classification (Dhillon & Verma, 2020). Current CNN models consist of network architectures with high number of processing layers that allow to learn representations of data with multiple levels of abstraction. This has given the name of *deep learning* to this particular field of machine learning (LeCun et al., 2015).

An increasing fraction of top solutions for the visual object tracking and segmentation (VOTS) and multiple object tracking and segmentation (MOTS) tasks use CNNs (Yao et al., 2019; Voigtlaender et al., 2019). However, these methods are designed to perform well on benchmark datasets that include mainly sequences of pedestrians and vehicles which typically have very distinguishable features (Geiger et al., 2012; Dendorfer, Rezatofighi, et al., 2020).

Due to the large number of layers and parameters, training deep learning models from scratch requires large amounts of data. Because of this, many of the current applications using CNNs to track multiple objects rely on networks that are pre-trained with datasets of similar images, like ImageNet (Deng et al., 2009), CIFAR (Krizhevsky & Hinton, 2009) or COCO (Lin et al., 2014). But again, most of these datasets include images of objects in daily-life scenarios the appearance of which is different to the images of animals obtained when tracking animals from videos recorded under laboratory conditions.

1.2.3.1 CNNs for animal identification in multi-animal tracking

The success of CNNs in supervised image classification makes them an obvious candidate to improve the identification capacity of multi-animal video-based tracking algorithms. In fact, the results of Rasch et al. (2016) showed that CNNs can distinguish 5 zebrafish with high accuracy. We remember that to obtain these results, the authors trained the CNN with 10000 images (2000 images per individual) that they obtained after tracking a video and manually validating it against idTracker.

In real videos of animal collectives one could obtain sequences of images of the same animal by manually labelling the identity of different trajectory fragments ("tracklets"). However, this is equivalent to performing a manual validation of

the potential identity swaps, which is the problem that tracking algorithms like idTracker and ToxId tried to solve (see Section 1.2.2)

Gathering animal images from different parts of the video is possible when pre-computed visual features are available and the animals are found separated from each other in different parts of the video (Pérez-Escudero et al., 2014). However, when the goal is to train a CNN to extract the visual features that best distinguish the animals, there are no pre-computed features beforehand. Thus, until recently, the problem of automatically extracting the sufficient number of images from videos of animals collectives to train a CNN for animal identification without requiring manual annotation of the animal identities was unsolved.

1.3 Collective animal behaviour modelling

Many mathematical models have been applied to study different aspects of collective animal behaviour (Sumpter, 2010). A particular question that has caught the attention of scientists for years are the patterns that groups of animals exhibit when they move together forming swarms, flocks or schools (Okubo, 1986). Classically these phenomena have been studied with self-propelled particle models (Vicsek et al., 1995; Couzin et al., 2002, 2005; Chaté et al., 2008; Gautrais et al., 2012). These models assume simple rules by which each animal in the group updates its position based on the movement of its neighbours. Depending on the model, the update rule considers different numbers of neighbours with whom a given animal interacts.

Understanding which are the important neighbours that influence the behaviour of an animal in the collective has also caught the attention of scientists for years. In fact, different mathematical models have been proposed to study how animals aggregate information from the members of the group to make better decisions. Some of these include the many-eyes hypothesis (Lima, 1995; Roberts, 1996), weighted averages models(Conradt & Roper, 2003), the Condorcet's jury theorem (De Condorcet et al., 2014) and others (Ward et al., 2008; Sumpter et al., 2008; Pérez-Escudero & de Polavieja, 2011; Arganda et al., 2012; Hinz & de Polavieja, 2017).

A common denominator of these models is that they are of very low complexity. While model complexity can be formally defined (Grünwald & Grunwald, 2007), in practical terms one can define it to be the number of parameters of a model. Low parameter-complexity models can be written down in simple mathematical expressions which has the benefit of allowing the detailed study of the effect of each of the parameters. This typically helps to extract insight about the problem at hand and can guide designing novel experiments to test new hypotheses. However, the assumptions and simplifications required to lay these types of models out likely misses important biological components. Because of this, most of these models are used to make predictions about population level parameters of the collective and typically cannot make fine-grained predictions about the behaviour of each animal in the group (see the work by Strandburg-Peshkin et al. (2013) and Harpaz et al. (2017) for some exceptions).

Advances in computing power have allowed building and training higher complexity models based on deep neural networks that can be very accurate in a wide range of tasks (LeCun et al., 2015). The thousands or millions of parameters of these models can be adjusted to fit any possible function (Raghu et al., 2017). However, these novel models have two main drawbacks for their application. First, the higher number of parameters requires very high amounts of data in order to prevent over-fitting (Lawrence et al., 1997; Caruana et al., 2001). Second, this parameter-complexity makes deep neural network "black boxes" difficult to analyse (Benítez et al., 1997), which is a problem if we want to extract insight about how they solve the task at hand.

Despite these limitations, the high accuracy capacities of this modelling technique have encouraged scientists to investigate new ways of applying them in order to extract insight about different problems (Xie et al., 2020). In Heras et al. (2019) we showed how deep neural networks can be applied to the study of animal collectives obtaining both high accuracy predictions about the animals' movement and insight about how animals interact and aggregate information from their neighbours. Next we review the main findings of this work that motivated the new results presented on this thesis.

1.3.1 Deep attention networks reveal the rules of collective motion in zebrafish

We built a model of collective motion that could predict the future behaviour of each individual in the group on held-out data not used for training the model (like, e.g. Eyjolfsdottir et al. (2016); Harpaz et al. (2017); Hoshen (2017); Bartoli et al. (2018)). To gain insight about the behaviour of the animals, we set some assumptions about the way animals interact in groups which gave our model a very useful modular structure.

In particular, we built our model so that it could predict the future turning side of each individual in the group. Like other models before (Pérez-Escudero & de Polavieja, 2011; Arganda et al., 2012) we assumed that the behaviour of each individual depended on its own state (asocial variables, α) and the state of its neighbours (social variables, σ) (see Figure 1.2a). Also, we assumed that the interactions take place in pairs and that each individual integrates with weights these pairwise interactions (Huth & Wissel, 1992; Bode et al., 2011; Collignon et al., 2016).

Our model computes the probability of the focal fish turning right in the future as $p = 1/(1 + e^{-z})$, where

$$z = \sum_{i=1}^{n} \Pi(\alpha, \sigma_i) \frac{W(\alpha^{(w)}, \sigma_i^{(w)})}{\sum_j W(\alpha^{(w)}, \sigma_j^{(w)})} \tag{1.1}$$

is the logit that the network outputs in its readout layer, and n is the number of neighbours considered.

Instead of using simple functional expressions for the functions Π and W (e.g. Calovi et al. (2018) and Harpaz et al. (2017)), we modelled them as fully connected networks with few inputs and a single output (see Figure 1.2b and Figure 1.2c). The pair-interaction subnetwork, $\Pi(\alpha, \sigma_i)$, accounts for the interaction of the focal animal and one neighbour i, and it returns a real number. The aggregation subnetwork, $W(\alpha^{(w)}, \sigma_i^{(w)})$, returns a positive weight, w_i, depending on the focal variables, $\alpha^{(w)}$, and the variables of a relative neighbour, $\sigma_i^{(w)}$. The superscript (w), indicates that these variables can be different to the ones of the pair-interaction network.

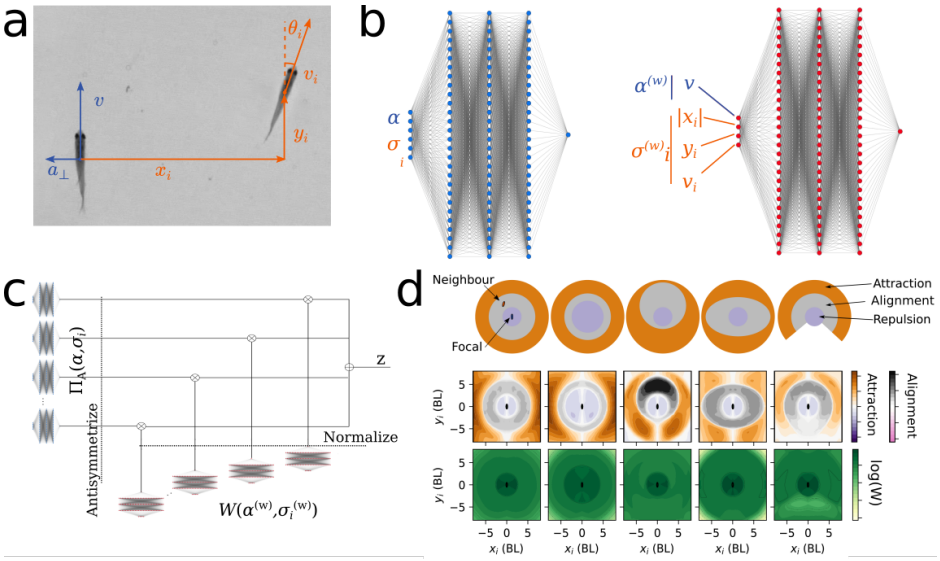

Figure 1.2. **Deep attention networks reveal the rules of interaction in animal collectives.** Figure adapted from Heras et al. (2019). **a.** Variables used to predict future turns. Asocial variables, those only involving the focal, in blue. Social variables, those involving both the focal and a neighbour, in orange. **b.** Pair-interaction subnetwork of the attention network (left) and aggregation subnetwork of the attention network (right). **c.** Attention network, showing how the inputs of the pair-interaction and aggregation subnetworks are integrated to produce a single logit z for the focal fish turning right after 1 s. **d.** Top row: interaction models used to simulate trajectories. Fish turn away from neighbours that are in the repulsion area. If there are no neighbours in the repulsion area, fish align with neighbours in the alignment area and are attracted to neighbours in the attraction area. Pair interaction scores (center row) and aggregation weights (bottom row) obtained when training using simulated trajectories generated by the interaction rules.

The final weight of each relative neighbour is the normalized aggregation weight

$$\omega_i = \frac{w_i}{\sum_j w_j} \tag{1.2}$$

which must sum 1 when considering all neighbours. Neural networks with a similar structure as the one in Equation 1.1 are known as deep attention networks (Bahdanau et al., 2014; K. Xu et al., 2015; Hoshen, 2017).

Modelling the functions Π and W as fully connected networks of few inputs and a single output allowed us to explore their shape by plotting their output for different slices of the input variables. In particular, from the pair-interaction network output we defined interaction scores that approximated the notions of repulsion, alignment and attraction described in classical models of collective motion (Couzin et al., 2002).

To validate the ability of this network of extracting insight about the behaviour of the animals in collectives we first applied it to simulated trajectories. We generated artificial trajectories of groups of agents whose interaction rules were variations of well described interaction rules (Aoki, 1982; Couzin et al., 2002). After training the deep attention networks to predict the behaviour of the agents, we could recover the interaction rules used to simulate the trajectories (see Figure 1.2d).

We then tested the predictive capacity of the model on real animal collectives data and extracted the rules of interaction in this case. We used our multi-animal tracking system idtracker.ai (Romero-Ferrero et al. (2019); see also Chapter 2) to extract highly accurate trajectories of each animal in large groups of juvenile zebrafish (*Danio rerio*). After training, the deep attention network achieved accuracies of around 85% in held-out data of groups of 100 zebrafish.

We also explored the interaction rules. In classical models of collective motion, the notions of repulsion, alignment and attraction are defined only in terms of the relative position of the neighbour. However, we found that the interaction scores in our zebrafish collectives depended also on the focal and neighbour velocities. Similarly, an exploration of the weighting function W revealed that zebrafish assigned different weights to neighbours also depending on its speed, and the relative position and speed of the neighbours.

We modelled this weighting function as a fully connected neural network so that it was flexible enough to approximate other previously proposed weighting rules. This includes the angle subtended by the neighbour fish (Strandburg-Peshkin et al., 2013; Collignon et al., 2016), the inverse of the distance to the

neighbour (Bode et al., 2011), and a decreasing function of the topological rank (Huth & Wissel, 1992). Other previously proposed rules are binary weights based on thresholds of simulated visual motion cues (Lemasson et al., 2009, 2013), first Voronoi neighbourhood (Gautrais et al., 2012) and topological ranges (Ballerini et al., 2008). A simpler aggregation rule would be to average the outputs of the pair-interaction network. Compatible with previous studies (e.g. Katz et al. (2011)), we found that this simpler rule gave a lower prediction accuracy.

From the aggregation weight we could compute the number of interacting neighbours, in both simulated and real trajectories. In simulated trajectories we found this to be consistent with the number of interacting neighbours used to generate the trajectories. In real data we found that this number changed dynamically along the video. This flexibility can be seen as allowing a smooth transition from average type of models (Vicsek et al., 1995; Couzin et al., 2002, 2005; Chaté et al., 2008; Gautrais et al., 2012) to models in which one or very few animals influence the rest as in the many-eyes model for predator detection (Lima, 1995; Roberts, 1996; Conradt & Roper, 2003; Marshall et al., 2017) and others (Hinz & de Polavieja, 2017).

Importantly, this capacity to change the attention from few animals to many might allow the group to match the changing distribution of information in the group (Laan, Madirolas, & de Polavieja, 2017). However, all animals in the experiments in Heras et al. (2019) had the same information about the environment, thus, it is unclear whether zebrafish truly take advantage of this aggregation rule to weight higher the neighbours that bring more information to the group.

1.4 Objectives of this thesis

The general goal of this thesis was to explore the applicability of deep learning tools to the study of collective behaviour.

The quantitative study of animal collectives requires highly accurate and informative measurements of their location throughout time, i.e. the animals' trajectories. With the set of trajectories of the collective, mathematical models can help to reveal the rules of interactions of the animals in the group and make

suggestions about further experiments. In this thesis, we investigate how deep learning tools can be useful to both acquire animal trajectories from video data and to reveal information aggregation rules in groups of zebrafish.

In particular, we set ourselves the goal of examining the possibility of using CNNs to identify animals in small and large collectives, and to develop a multi-animal tracking algorithm that automatically gathered the sufficient data to train a CNN to identify animals without the need of manual annotation of animal identities. Based on the highly successful tracking software idTracker (Pérez-Escudero et al., 2014), we forced ourselves to deploy the new algorithm as a ready-to-use open-source software, idtracker.ai, that worked for videos recorded under laboratory conditions and for different animal species (Chapter 2).

Using the trajectories obtained with the developed multi-animal tracking system, idtracker.ai, our work in Heras et al. (2019) proved that deep attention network models are capable of making accurate predictions and giving insight about the interaction rules of zebrafish moving in large collectives. The results of this work suggested that zebrafish aggregate information from their neighbours by dynamically weighting them depending their behaviour. We set ourselves the second goal of testing with new experiments whether zebrafish could use this flexible information aggregation rule to consider differently the neighbours that bring new information to the collective (Chapter 3).

Chapter 2

idtracker.ai: tracking all individuals in small or large collectives of unmarked animals

2.1 Introduction

Multiple tracking software packages exist to extract animal trajectories under a varying range of imaging conditions and with different levels of detail about the movement of the animals (see Section 1.2.1). Marker-less multi-animal tracking systems that maintain the identity of animals throughout the video based on pre-computed visual features are not scalable for large groups (Pérez-Escudero et al., 2014; Bai et al., 2018; Rodriguez et al., 2017). The high capacity of CNNs in image recognition tasks (LeCun et al., 2015) makes them a great candidate to improve the tracking capabilities of multi-animal tracking software. However, how to automatically extract data from animal videos to train a CNN without human supervision has been until recently an unsolved problem.

Rasch et al. (2016) tested the capacity of CNNs to identify 5 zebrafish. However, the authors trained the CNN with images (2000 per animal) obtained after tracking a video and validating the animal trajectories. In fact, they neither proposed any strategy to automatically extract the sufficient number of images to train a CNN during the tracking process nor showed tracking performance results using CNN for identification. Wang et al. (2017) and Zhiping & Cheng (2017) attempted to use to use CNNs for animal identification for multi-animal tracking. However, the algorithm by Wang et al. (2017) requires manual annotations and both of them are species-specific, and are tested on very short videos.

Our objective was to develop a multi-animal tracking system that was species-agnostic and that used CNNs for animal identification without requiring manual annotation or previous validation of the animal trajectories. In this chapter we present the multi-animal tracking system idtracker.ai, which appeared first in a preprint version (Romero-Ferrero et al., 2018) and was finally published in Romero-Ferrero et al. (2019).

idtracker.ai is an off-line video-based marker-less multi-animal 2D tracking algorithm that maintains animal identities throughout the video. We designed idtracker.ai to work under laboratory conditions for small and large groups of any animal species. After detecting the animals in the video using user-defined parameters, one CNN classifies individuals and crossings (touches or occlusions) and second CNN identifies all individuals in the collective. Importantly, we de-

signed the algorithm to automatically gather data from the video to train both CNNs in a self-supervised manner, i.e. without requiring manually labeled data from the users (Jing & Tian, 2020).

idtracker.ai can track videos of up to 100 individuals with high identification accuracy ($> 99.9\%$), which was not possible before with other multi-animal tracking systems. Similar to others (Branson et al., 2009; Pérez-Escudero et al., 2014; Rodriguez et al., 2018), we deployed idtracker.ai with an easy-to-use graphical user interface so that the rest of the scientific community could benefit from our algorithm.

In Section 2.2.1, we describe idtracker.ai's algorithm. This is basically the same algorithm already described in Romero-Ferrero et al. (2019) but with a modification in the construction of the images used to train the CNN that detects individuals and crossings (Section 2.2.1.2). In Section 2.2.2, we describe the new redesigned graphical user interface (GUI) that (i) facilitates the use of idtracker.ai by non-experienced programmers allowing high-throughput trajectories extraction (Section 2.2.2.1), (ii) enables modifying advanced parameters of the algorithm for higher flexibility (Section 2.2.2.2), and (iii) allows exploring, correcting, and validating the resulting trajectories (Section 2.2.2.3). In Section 2.2.3, we show the results of the validation of the algorithm on synthetic and real videos data. These results are the same as the ones already presented in Romero-Ferrero et al. (2019). In Section 2.2.3, we explain the working video condition of the algorithm. Finally, in Section 2.3 we summarize the main findings (Section 2.3.1), discuss the limitations of the algorithm, and propose future improvements taking into account recent advances in the field (Section 2.3.2).

2.2 Results

2.2.1 Algorithm

The idtracker.ai algorithm consists of 8 processing steps that run sequentially one after the other (see Figure 2.1).

During the *Animals detection* step (see Section 2.2.1.1), idtracker.ai extracts small images of the animals from each frame of the video using a set of parameters that the user defines with the graphical user interface, similar to the way it is done in idTracker (Pérez-Escudero et al., 2014). The *Crossing detection* step (see Section 2.2.1.2) uses a set of heuristics to select images that are either clearly a single individual or clearly several individuals and it uses them to train a CNN that classifies all the images in the video as *individuals* or *crossings*. In the *Fragmentation* step (see Section 2.2.1.3), idtracker.ai groups images of the same individual in consecutive frames into what we define as *individual fragments*. Then it detects intervals of the video where all animals are separated and creates sets of individual fragments that we call *global fragments*. The *Cascade of training and identification protocols* step (see Section 2.2.1.4) iteratively gath-

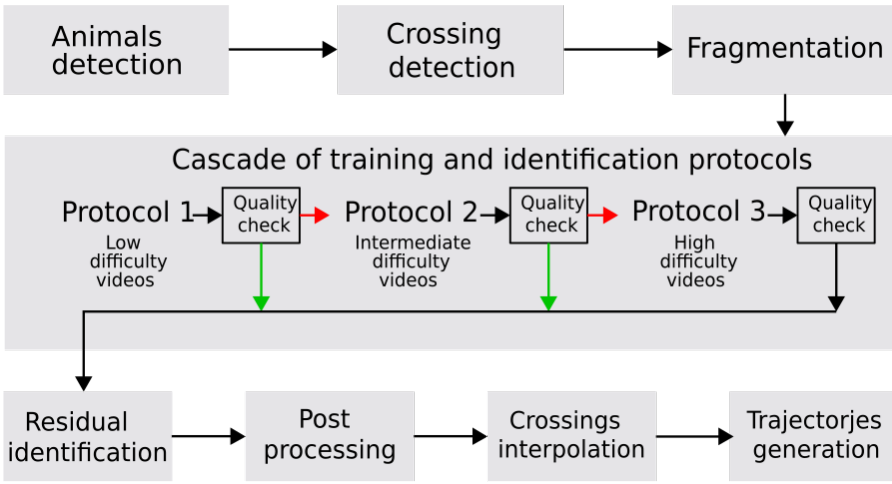

Figure 2.1. **Diagram of idtracker.ai processing steps.** Each grey box represents a processing step of the idtracker.ai algorithm.

ers images from the global fragments and trains a second CNN to identify all the individual fragments in the video. Depending on the difficulty of the video, idtracker.ai applies different protocols ensuring that the quality of the identification is high enough to jump to the next block. The *Residual identification* step (see Section 2.2.1.5) uses the Bayesian framework developed in idTracker (Pérez-Escudero et al., 2014) to identify the remaining individual fragments that the previous block could not identify with high certainty. The *Post-processing* step (see Section 2.2.1.6) corrects small identification mistakes that can be detected by unnatural changes of the speed of the individuals. The *Crossing solving* step (see Section 2.2.1.7) assigns an identity and a location in the video frame to the animals in the crossing images detected in the *Crossing detection* step using a reimplementation of the algorithm in (Pérez-Escudero et al., 2014). Finally, the *Trajectories generation* step (see Section 2.2.1.8) generates individual trajectories for each animal by grouping the locations of the identified animals in each frame of the video.

2.2.1.1 Animals detection

This processing step reads the video given by the user and detects the animals in each frame using the animal detection parameters that the user defines through the tracking GUI (see Section 2.2.2.1). We list these parameters in Table 2.1.

A video is an ordered collection of M images recorded at a given frame rate. Each image has width W, height H, and C channels. Color images have three channels, red, green and blue, so $C = 3$. idtracker.ai transforms color videos to gray-scale at the beginning of the animal detection step, so for the reminder $C = 1$. Also, if $\rho < 1$ (see Table 2.1), it reduces the resolution of each frame so that the new width and height are $W = \rho W_o$ and $H = \rho H_o$, where W_o and H_o are the original width and height of the video.

idtracker.ai assumes videos in 8-bit format, so each pixel in a frame can have values in the range $[0, 255]$. The value of a pixel encodes its color intensity, 0 encodes black pixels, 255 encodes white pixels in a gray-scale image. To remove possible fluctuations caused from overall changes in illumination, we normalize each frame dividing it by the mean intensity across all pixels. To keep the pixel values in the range $[0, 255]$ we re-scale the normalized values by multiplying them

Parameter	Required	Description
Resolution reduction, ρ	×	A number in the range $(0, 1]$ indicating the reduction factor of the video frame resolution.
Minimum intensity, $I_{\min}$	✓	An integer in the range $[0, 255]$ indicating the minimum pixel intensity of an animal in the video frame.
Maximum intensity, $I_{\max}$	✓	An integer in the range $[0, 255]$ indicating the maximum pixel intensity of an animal in the video frame.
Minimum area, $A_{\max}$	✓	An integer indicating the minimum number of pixels of a blob.
Maximum area, $A_{\min}$	✓	An integer indicating the maximum number of pixels of a blob.
Number of animals, N	✓	Number of animals that appear in the video.
Check segmentation	×	A boolean indicating to stop the tracking process if the number of blobs in a frame is $> N$.
Background subtraction	×	A boolean indicating to compute the background model and subtract it from each frame before thresholding.
Region(s) of interest (ROI)	×	List(s) of pixel coordinates that define(s) the borders of the video frame wherein the animals can be found.
Tracking interval(s)	×	List of frame intervals of the video for which the animal detection step is applied.

Table 2.1. **idtracker.ai main user-defined parameters.** List of necessary and optional user-defined parameters for the animals detection processing step. The first column indicates the name of the parameter. The second column indicates whether the parameter is necessary (✓) or optional (×) for idtracker.ai to work. The last column shows a description of the parameter.

by $\frac{255}{px_{99}}$, where px_{99} is the 99.95 percentile of the pixels' intensity after normalization.

In a given frame, each object is represented by a set of pixels. To detect animals in a frame idtracker.ai first partitions the frame into pixels representing animals and pixels representing the rest of the objects in the frame. This process is known as image segmentation (Bovik, 2010). idtracker.ai uses intensity-based segmentation, also known as thresholding. In particular, we implemented a 2-threshold method with thresholds $I_{\min}$ and $I_{\max}$, where $I_{\min} < I_{\max}$ (see Table 2.1). For each frame, pixels with intensity values in the range $[I_{\min}, I_{\max}]$ are labeled as animals and the rest are labeled as background. This results in

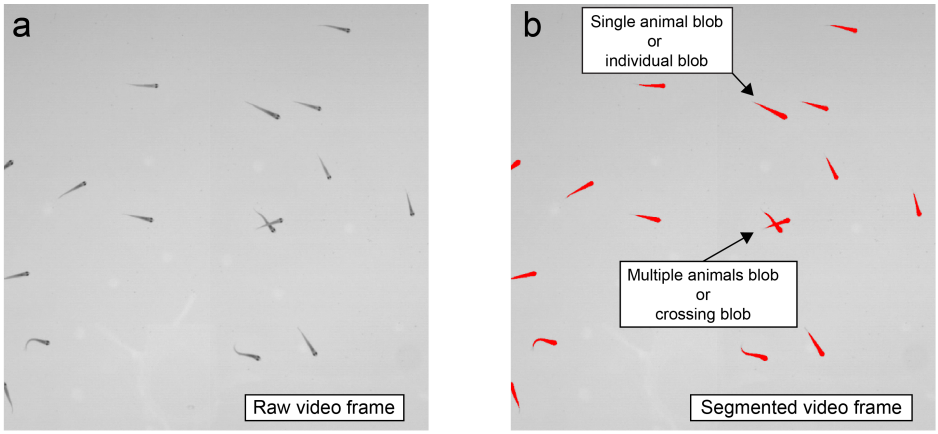

Figure 2.2. **Animals detection. a.** Cropped raw frame of a 100 juvenile zebrafish video. **b.** Same cropped video frame after applying the animals detection step with the user-defined parameters. Pixels representing detected animals are highlighted in red. White boxes with the black arrow indicate two examples of an single animal blob (top) or a multiple animals blob (bottom).

a binary image where pixels labeled as representing animals have value 255 and the rest have value 0.

Each group of pixels with value 255 completely surrounded by pixels with value 0 defines a *blob*, which stands for Binary Large OBject (Moeslund, 2012) (see Figure 2.2). idtracker.ai detects the blobs using the OpenCV library (Bradski, 2000) implementation of the algorithm described in (Suzuki et al., 1985). We define the *area* of each blob as the number of pixels that define it. idtracker.ai only considers blobs whose area is in the range $[A_{\text{min}}, A_{\text{max}}]$, where $A_{\text{min}} < A_{\text{max}}$ (see Table 2.1). At this point, for each blob in such range idtracker.ai computes the features listed in Table 2.2.

We designed idtracker.ai assuming that:

1. Each segmented blob in a frame represent a single animal or multiple animals.

2. The number of blobs in each frame, N_b, is smaller or equal to the number of animals in the video, N (given by the user, see Table 2.1).

Feature	Description
Contour	List of pixels in the border of the blob.
Pixels	List of pixels in that define the blob.
Area	Number of pixels.
Centroid	Position of the center of mass of the blob in the frame (in pixels).
Bounding box	Coordinates of the up-right bounding rectangle of the blob contour (in pixels).
Estimated body length	The length of the diagonal of bounding box.

Table 2.2. **Segmented blob features.** List of features computed for each segmented blob during the animals detection step. The left column indicates the name of the feature and the right column a short description.

idtracker.ai checks whether $N_b \leq N$ for all the frames in the video. If the parameter "Check segmentation" (see Table 2.1) is `True` and there are frames where this condition is not met ($N_b > N$), idtracker.ai finishes and outputs the corresponding frame numbers. Otherwise, the algorithm continues but its performance will be undermined if $N_b > N$ for many frames in the video.

Having more blobs than the number of animals in the video can occur if the animal detection parameters N, I_{min}, I_{max}, A_{min} and A_{max} are not properly set. However, sometimes the video conditions (see Section 2.2.4.1) make it impossible to detect the animals using only the previous parameters. Users can use the optional parameters (see Table 2.1) to increase the number of frames for which the previous condition holds. Next, we describe how these parameters affect the animal detection pipeline.

If the parameter "Background subtraction" is `True` (see Table 2.1), idtracker.ai generates a model of the background by computing an average image of the video. To do so, it selects frames of the video with a period P_{bkg}, normalizes them, and takes the mean of the intensity for each pixel across all frames.[1] The parameter P_{bkg} can be modified by the user to adapt to different video conditions (see Section 2.2.2.2 and the advanced parameter `BKG_SUB_PERIOD` in Supplementary Table A.2). The background model is subtracted from the original frame and the

[1] We followed the same algorithm as in IdTracker (Pérez-Escudero et al., 2014, Suppementary Note 2, Section 2.1). Note that taking the median instead of the mean would likely result in a better model of the background. However, taking the median requires saving all sampled frames which is less computationally efficient.

values of the pixels are re-scaled to the range $[0, 255]$ as indicated before. The 2-thresholds segmentation method is then applied to the background subtracted frame using the parameters $I_{\min}$ and $I_{\max}$.

From the ROIs drawn by the users (see Table 2.1), idtracker.ai builds a mask, i.e. a binary image where pixels inside the ROI have value 1 and pixels outside the ROI have value 0. During the normalization of each frame idtracker.ai only considers the pixels inside of the mask. After applying the thresholds $I_{\min}$ and $I_{\max}$ idtracker.ai multiplies the binary image by the mask. Then, it applies the contour detection algorithm on the new binary image resulting from such multiplication.

idtracker.ai applies all the previous steps to the frames that are contained in any of the intervals that the user defines using the "Tracking interval" parameter (see Table 2.1).

The output of this processing step is a list of blobs for each frame of the video, where each blob has the features described in Table 2.2.

2.2.1.2 Crossing detection

This processing step takes the previously computed list of blobs and classifies all the blobs as representing a single animal or multiple animals together. We refer to the these blobs as *individuals* or *individual blobs*. Blobs representing multiple animals appear mainly due to animals being occluded by each other to the view of the camera, or by animals being so close to each other that after segmentation they constitute a unique blob. We refer to these two cases as *crossings* or *crossing blobs*.

idtracker.ai uses three heuristics and a CNN to tell apart individual from crossing blobs.

The first heuristic assumes that if in a frame $N_b = N$, then all blobs in such frame must be individual blobs. The second and third heuristic work together to try to classify the rest of blobs in the video.

The second heuristic assumes that the blob area is a good feature to distinguish individual and crossing blobs. idtracker.ai computes a model of the area of a single animal. The model is defined by the median, m_A, and the standard deviation, σ_A, of the areas of all individual blobs detected by the first heuristic.

idtracker.ai labels blobs with an area higher than $m_A + 4\sigma_A$ as a *potential crossing* blobs. Otherwise, it labels them as a *potential individual* (see Supplementary Figure A.1).

The third heuristic exploits the spatial and temporal coherence of the video. It assumes that the frame rate of the video is such that the intersection between the sets of pixels representing an animal in a frame and in the immediate consecutive frame is not empty (see Figure 2.3a and 2.3b). When this happens we say that two blobs *overlap*. Under this assumption, idtracker.ai labels a sequence of overlapping blobs as potential crossings if the first (last) blob of the sequence overlaps with more than one blob of the previous (next) frame. On the other hand, it labels a sequence of overlapping blobs as potential individual blobs if the first (last) blob of the sequence overlaps with a single blob of the previous (next) frame but this blob overlaps with more than two blobs of the current frame (see Figure 2.3c).

If a blob is a potential individual or a potential crossing by the second and third heuristics, then idtracker.ai classifies the blob as an individual or crossing blob respectively. After applying the three heuristics, most of the blobs in the video will be classified as individual or crossing. However, some blobs will still remain unclassified.

Using the already classified blobs, idtracker.ai builds a dataset of square images where the blobs are aligned to the diagonal by its axis of maximum elongation (see Section 2.4.4). The width of the images is computed such that the diagonal equals the median of the estimated body lengths (see Table 2.2) of all individual blobs labeled by the first heuristic. We call each of these images an *identification image* (see Figure 2.5c). Note that the process of generating the identification images is species agnostic and is valid for any species as long as the animals are elongated (see Section 2.4.4 and Figure 2.22). Since these images are the same ones that idtracker.ai uses to identify the animals (see Section 2.2.1.4), this makes idtracker.ai a species agnostic multi-animal tracking system.[2]

[2]In a previous version of the algorithm the images used to train the cdCNN were first preprocessed and then resized to 40×40 pixels (Romero-Ferrero et al., 2019, Supplementary Notes, Section B .2.2, Preprocessing). The method here explained uses the same image to train the cdCNN and the idCNN which saves memory and computing time. We did not find negative changes in the tracking performance with the current method.

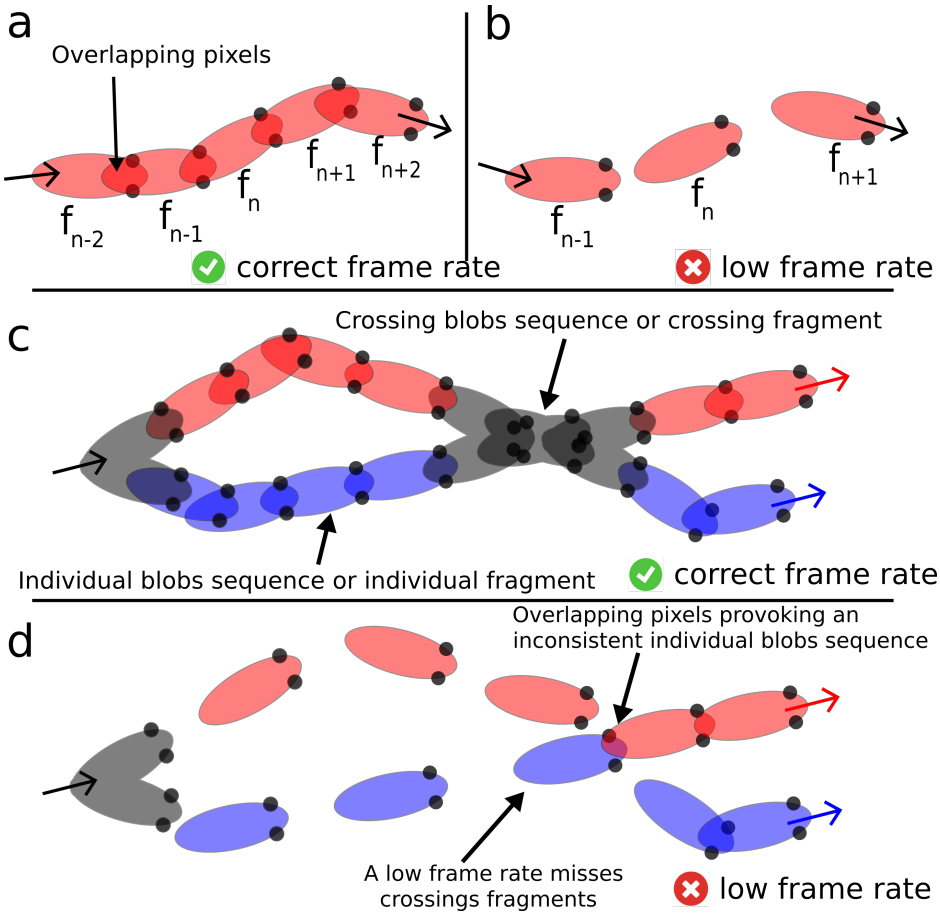

Figure 2.3. **Diagram of the blobs overlapping heuristic.** Each ellipse with two black dots represents an animal blob. Blue or red ellipses represent individual blobs. Gray ellipses represent crossing blobs. Arrows indicate the direction of movement of the animal in consecutive frames. **a.** Blobs in a sequence of frames for a video with a sufficiently high frame rate that creates an overlapping between the pixels of blobs in consecutive frames. **b.** Same as before for a video with a low frame rate that does not create an overlapping between the pixels of blobs in consecutive frames. **c.** Individual and crossing blob of two animals in a video with a sufficiently high frame rate that create individual and crossing blob sequences. **d.** Same as before for a video with a low frame rate that does not manage to capture the crossing blobs sequences and generates an inconsistent individual blobs sequence that mixes blobs from different animals.

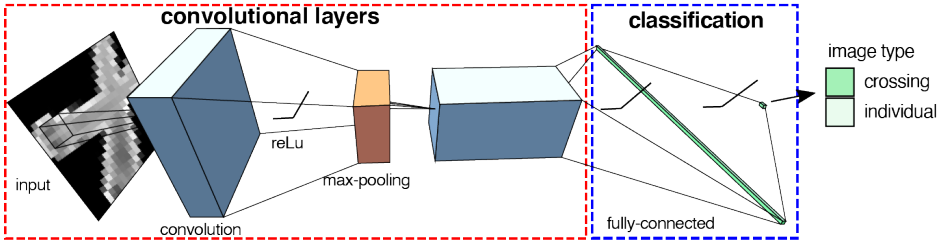

Figure 2.4. **Crossing detector convolutional neural network (cdCNN) diagram**. The red dashed rectangle contains the convolutional layers. The blue dashed rectangle contains the classification layers. The input of the crossing detector network are squared images that can contain a single animal or a crossing (the one in the diagram). The output are two neurons, the values of which encode the classification logits of the input image.

idtracker.ai trains a CNN that we call the cdCNN (crossing detector CNN) to classify images as individuals or crossings (see Figure 2.4, Supplementary Table A.8 for the architecture and Section 2.4.6.1 for details about the training). After training, it uses the cdCNN to classify the remaining blobs that were not classified by the heuristics.

At the end of this step idtracker.ai has labeled all blobs in the video as individuals or crossings (see Figure 2.5a).

2.2.1.3 Fragmentation

This processing step groups the individual and crossing blobs in sequences of consecutive overlapping blobs representing the same animal or the same group of animals (for crossing blobs). We call these sequences of blobs *fragments*. We define an *individual fragment* as a sequence of consecutive overlapping individual blobs between two crossing blobs[3] (see Figure 2.3c). An individual fragment F_i contains blobs representing the same animal in consecutive frames in the interval $[f_s, f_e]_i$. We say that two individual fragments F_1 and F_2 *coexist* if the intersection of the intervals $[f_s, f_e]_1$ and $[f_s, f_e]_2$ is not emtpy. Similarly, one can define *crossing fragments* as a sequence of consecutive overlapping blobs between two individual blobs (see Figure 2.3c). Note that idtracker.ai has previously com-

[3]Other tracking systems call the individual fragments *tracklets* (Robie et al., 2017).

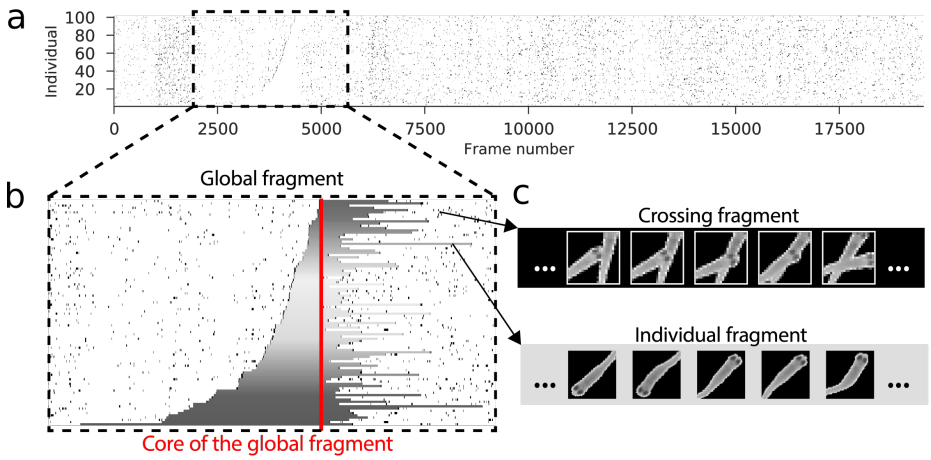

Figure 2.5. **Fragmentation diagram. a.** Raster plot of the blobs of each individual in a video of 100 juvenile zebrafish video. White segments represent individual fragments, black segments represent crossing fragments. **b.** Close-up of the raster plot around the first global fragment. The red vertical line indicates the frames of the video shared by all individual fragments in the global fragment, i.e. the core of the global fragment. **c.** Close-up diagram of a crossing fragment (top) and an individual fragment (bottom) with the respective individual and crossing images that constitute them.

puted an identification image for each blob (see 2.2.1.2). Hence, we can also understand an individual fragment as a collection of images of the same animal (see Figure 2.5a).

We define a *global fragment* as a collection of N individual fragments for which the intersection of all the intervals $[f_s, f_e]_i$ is not empty. We say that all individual fragments in a global fragment coexist. We call the frames in the intersection of all the intervals the *core of the global fragment* (see Figure 2.5b). Intuitively, lets take a frame f of the video in which the number of blobs is N. The blobs in the frame will belong to N different and unique individual fragments. These individual fragments constitute a global fragment and the frame f is part of the core of the global fragment. Note that the starting frame, f_s, and ending frame f_e, of each individual fragment in the global fragment can be different (see

Figure 2.5b). Also, a given individual fragment can belong to multiple global fragments.

At the end of this step all the individual and crossing blobs belong to a unique individual or crossing fragment, and all possible individual fragments are grouped into global fragments.

2.2.1.4 Cascade of training and identification protocols

This processing step uses the identification images in the global fragments to train a second CNN to assign an identity to each identification image in an individual fragment. We call this CNN the idCNN (identification CNN) (see Figure 2.6 and Supplementary Table A.8 for the architecture). idtracker.ai follows a series of protocols to automatically gather training data, train the idCNN, identify identification images along the video, and check the quality of the identification. For simplicity, in some parts of the text we refer to this processing steps as the *training and identification strategy*.

Before describing the different protocols we explain how each of the global fragments can be understood as a training dataset for the idCNN. Remember that a individual fragment can be seen as a collection of images of the same animal. Since a global fragment is a collection of N individual fragments, it is indeed a labeled collection of images of the N animals of the video, where each label is the identity of an animal. Since the interval of frames $[f_s, f_e]_i$ will be different for each individual fragment, the number of images in each individual fragment will be different. Hence, a global fragment is an unbalanced training dataset.

A single video can have hundreds or even thousands of global fragments (or training datsets). This depends on the number of crossings and the length of the video. A good training set for a CNN should contain as many different images of the same class as possible. Having a score that summarizes the number of images of a given animal and its variability is convenient to select the best global fragment to train the CNN.

Using the centroid of each blob (see Table 2.2) in an individual fragment, we can compute the distance traveled by an animal in the interval $[f_s, f_e]_i$. Assuming that animals move at similar speeds during the whole video, individual fragments

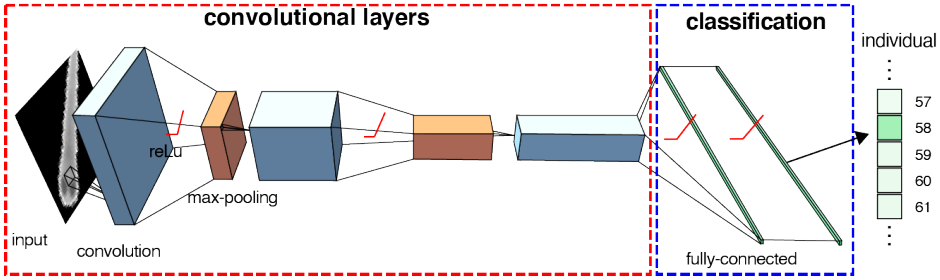

Figure 2.6. **Identification convolutional neural network (idCNN) diagram.** The red dashed rectangle contains the convolutional layers. The blue dashed rectangle contains the classification layers. The input of the identification CNN are squared images that contain individual animals. The size of the input image depends on the estimated body length of the animals in the video. The output layer has as many neurons as the number of animals in the video and they encode identification logit of the input image.

with a high distance traveled will represent animals that have spent more time without crossing with anyone. Potentially these animals will have visited more parts of the arena and have expressed a higher range of different postures. We considered that the distance traveled is a good score that summarizes the number of images in an individual fragment and the variability of the images in terms of light conditions in the arena and postures of the animal (see Supplementary Figure A.2)

The Protocol 1 starts by selecting the first global fragments to train the idCNN (see step 1 in Figure 2.7). From all the global fragments it selects the one with the maximum minimum distance traveled of all its individual fragments. We call this global fragment the *first global fragment*. Following the intuition above, this global fragment will potentially be the one with more images per animal and with higher variability. Before the first training step, idtracker.ai arbitrarily labels images of each individual fragment in the first global fragment with a number from 1 to N. This will be the identity of the animal at the end of the tracking process.

If the advanced parameter `KNOWLEDGE_TRANSFER_FOLDER_IDCNN` is set by the user (see Section 2.2.2.2 and Supplementary Table A.5) then, before training, idtracker.ai initializes the weights of the idCNN with the ones of the idCNN

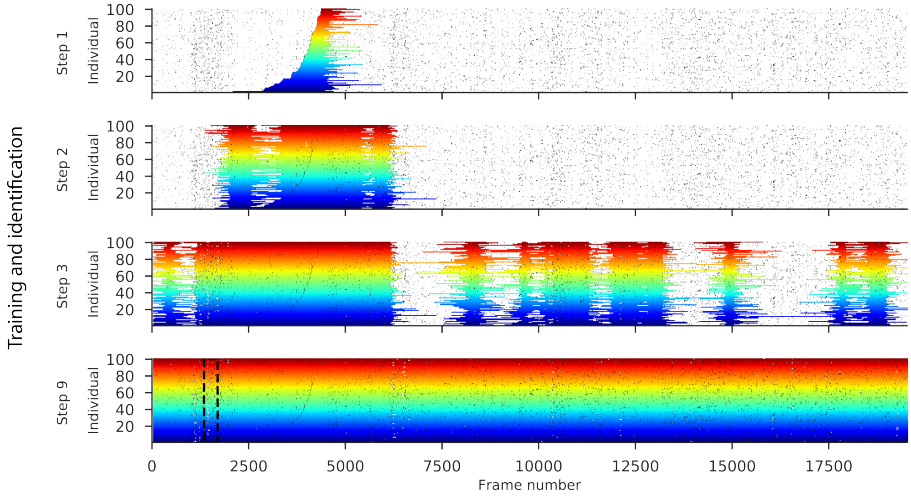

Figure 2.7. **Training and identification steps.** Accumulation of training images in a video of 100 juvenile zebrafish. White segments indicate individual fragments. Black segments indicate crossing fragments. Colored individual fragments indicate the fragments identified at each step. In step 1, the identification network trains on the starting global fragment and assigns the other global fragments. In the step 2, a subgroup of global fragments that passed the quality checks are identified. Protocol 2 increases the size of the training dataset by iterating training and quality checks, here ending at step 9. The black dashed rectangle in the step 9 represents the close-up that appears in Figure 2.8.

stored in the directory indicated by `KNOWLEDGE_TRANSFER_FOLDER_IDCNN`. In addition, if the advanced parameter `IDENTITY_TRANSFER` (see Section 2.2.2.2 and Supplementary Table A.5) is `True`, then idtracker.ai attempts to identify the individual fragments of the first global fragment using the idCNN stored in `KNOWLEDGE_TRANSFER_FOLDER_IDCNN` following the quality checks described later in the text. If the identification is of high quality (see the conditions below), idtracker.ai sets the identities of the individual fragments in the first global fragments following this identification. Otherwise, it warns the user about the inability to perform the transferring of the identities. See Section 2.2.2.2 for an explanation of the necessary conditions for identity transfer.

Then, idtracker.ai trains the idCNN using the first global fragment (see Section 2.4.6 for details about the training). After training, it uses the idCNN to identify all the identification images of all the individual fragments in the video. Under the assumption that all images in an individual fragment belong to the same animal, we can compute a series of measures for the identification of the individual fragment as a whole. Given an individual fragment with n_F images, we define the identification frequency for the identity i as $\omega_i = \frac{n_i}{n_F}$, where n_i is the number of images in the individual fragment labeled with identity i by the idCNN. Under certain assumptions,[4] we can compute the probability that the identity of fragment F is i as:

$$P(F, i) = p_i = \frac{2^{\omega_i}}{\sum_{j=1}^{N} 2^{\omega_j}} \tag{2.1}$$

Each individual fragment will have a vector $P_F = (p_1, p_2, \ldots, p_N)$ indicating the probability of each identity.

For each blob in the frame F, idCNN output a vector of softmax scores. Let S_i be the collection of softmax scores for the i – or, equivalently, the collection of i-th components of these softmax vectors – gathered across all the frames. So the k-th entry in S_i is the softmax score of the k-th image in the fragment F. Then, letting $s_i = median(S_i)$, we can define the certainty of the identification for an individual fragment as

$$cert(F) = \frac{s_a p_a - s_b p_b}{p_a + p_b} \tag{2.2}$$

where a and b are the two first identities with highest probability p_i, so $p_a \geq p_b$.

Using these measures and the fact that two animals in the same frame cannot have the same identity, idtracker.ai attempts to identify the individual fragments not used to train the idCNN. The steps followed for the identification of the individual fragments in a global fragment are the following:

[4] All the images in F are independent and that the probability to assign one image to the correct individual is twice as large as the probability to assign the image to any of the incorrect individuals (Pérez-Escudero et al., 2014).

1. **Global fragment certainty check**. Check that $cert(F) \geq 0.1$ for all individual fragments in the global fragment. If this condition is not met, all the individual fragments in this global fragment are left unidentified for now.

2. **Temporary identification**. The individual fragments in a global fragment are temporarily identified by decreasing order of $\max_i(P_F)$ with identity $i^* = \arg\max(P_F)$ if:

   (a) no other individual fragment coexists with F has been already identified with identity i^*.

   (b) $\max(P_F) < \frac{1}{n_F}$.

   If these conditions are not met for some individual fragment, all the individual fragment in the global fragment are left unidentified.

3. **Global fragment uniqueness check**. Check that the temporary identities of the individual fragments in the global fragment are unique, i.e. there are no repeated identities.

If a global fragment passes all the steps of the identification, we say that it passes the *quality check* and that the identification is of *high quality*. The temporary identities i^* are then fixed and considered definitive. Note that the certainty threshold of 0.1 can be modified by the user (see Section 2.2.2.2 and the advanced parameter `CERTAINTY_THRESHOLD` in Supplementary Table A.4).

idtracker.ai attempts to identify the global fragments iteratively by increasing order of their distance to the first selected global fragment. The distance between two global fragments is the absolute difference between the first frame of the core of each global fragment (see Figure 2.5b). Using this method, idtracker.ai attempts to identify only global fragments for which the smallest individual fragment has at least 3 frames (see Section 2.2.2.2 and the advanced parameter `MIN_NUM_OF_FRAMES_GF_TRAINING` in Supplementary Table A.4).

At the end of the identification, the global fragments that passed the quality check will have been identified, and others will have been left unidentified (see step 2 in Figure 2.7). Note that identifying a global fragment means identifying each blob in each individual fragment. idtracker.ai's goal is to identify all

individual blobs in the video, thus, in each step it computes the percentage of identified blobs in the video. If this percentage is $> 99.95\%$ the Protocol 1 ends. Otherwise, idtracker.ai starts Protocol 2. The value of this decision threshold can be modified by the user (see Section 2.2.2.2 and the advanced parameters `THRESHOLD_EARLY_STOP_ACCUMULATION` in Supplementary Table A.5).

In Protocol 2, idtracker.ai uses the images identified in Protocol 1 (see colored global fragments in step 2 in Figure 2.7) retrain again the idCNN. It uses up to 3000 images per animal randomly sampled from all the identified images (see Section 2.4.6.2). After training, it attempts to identify all global fragments following the steps described above. Then, it adds the images of the global fragments that passed the quality check to the training dataset of the idCNN (see step 3 in Figure 2.7). This process iterates until $> 99.95\%$ of the images in the global fragments are identified or if $> 90\%$ of images have been identified and there are no more global fragments that pass the quality check, i.e. no more global fragments can be identified following the steps previously described. The value of these decision thresholds can be modified by the users (see Section 2.2.2.2 and the advanced parameters `THRESHOLD_EARLY_STOP_ACCUMULATION` and `THRESHOLD_ACCEPTABLE_ACCUMULATION` in Supplementary Table A.5). In Figure 2.7, the Protocol 2 ended after 9 training steps because the percentage of images identified was $> 99.95\%$.

During the different training and identification steps, to increase the chances of reaching the 90% of identified images, once Protocol 2 reaches 50% of identified images, we allow identifying subsets of individual fragments in a global fragment following similar steps to the ones described above. In this case, we relax the previous conditions so that we consider all the individual fragments that pass the certainty check ($cert(F) > 0.1$) and do not provoke identity duplicates. The value of this decision threshold can be modified by the user (see Section 2.2.2.2 and the advanced parameter `MIN_RATIO_PARTIAL_ACCUMULATION` in Supplementary Table A.5).

If Protocol 2 fails to reach the 90% of identified images, then Protocol 3 starts.

Protocol 3 consists of two steps. First, idtracker.ai pre-trains the convolutional part of the idCNN using most of the global fragments in the video (see

Section 2.4.6.2 for details of the training). Then, it attempts to identify all global fragments following the steps in Protocol 1 and 2.

Remember that each global fragment can be considered as a training dataset by itself. To pre-train the idCNN convolutional layers, given a global fragment, idtracker.ai assigns an arbitrary label to each individual fragment and it trains the idCNN to identify the images in the global fragment. When the training finishes for a global fragment, the weights of the classification part of the idCNN are re-initialized and the pre-training continues for another global fragment. To maximize the number of images in each pre-training step, idtracker.ai selects the global fragments by increasing order of the maximum minimum distance traveled of its individual fragments. This process iterates until idtracker.ai has used 95% of the images in the global fragments to pre-train the idCNN. The value of this decision threshold can be modified by the user (see Section 2.2.2.2 and the advanced parameter `MAX_RATIO_OF_PRETRAINED_IMAGES` in Supplementary Table A.5). At the end of the pre-training the weights of the convolutional layers will have taken values more suitable for identifying the animals in the video and the identification steps described for Protocol 2 are more likely to achieve the 90% of identified images.

After the pre-training, Protocol 3 attempts to identify the global fragments in the video as in Protocol 1 and 2 but training only the classification layers of the idCNN with the weights of the convolutional layers fixed to the values of the pre-training. This time, if the training and identification process does not manage to identify 90% of the images, it starts again from the following global fragment with maximum minimum distance traveled. idtracker.ai attempts the training and identification process starting from up to three different global fragments at this point. The number of times that this training and identification process occurs can be modified by the user (see Section 2.2.2.2 and the advanced parameter `MAX_NUM_OF_PARACHUTE_ACCUMULATIONS` in Supplementary Table A.5). If none of the attempts achieves the 90% of identified images, then idtracker.ai selects the idCNN and the identities assigned during the training and identification iteration with the highest percentage of identified images. At this point the cascade of training and identification protocols finishes and the algorithm continues.

At the end of this processing step, idtracker.ai has identified a high percentage of the identification images in the individual fragments belonging to global fragments.

2.2.1.5 Residual identification

This processing step identifies all the individual fragments that are left unidentified by the previous step either because they did not pass the identification quality check, they do not belong to a global fragment, or they had a individual fragment with fewer than 3 images (see white fragments in Figure 2.7 (step 9) and lower opacity individual fragments in Figure 2.8a).

First, idtracker.ai uses the idCNN as it is at the end of the previous processing step to identify all the images from all the unidentified individual fragments. Then, it computes the P_F vector for each fragment. Considering the fact that two coexisting individual fragments cannot have the same identity, we follow the approach in (Pérez-Escudero et al., 2014, Supporting text, Section 3.1) and define the probability

$$P_2(F,i) = \frac{P(F,i)\prod_{\gamma_F}(1-P(\bar{F},i))}{\sum_{j=1}^{N} P(F,j)\prod_{\gamma_F}(1-P(\bar{F},j))} \tag{2.3}$$

where γ_F is the set of all individual fragments, $\bar{F}$, that coexist with F and N is the number of individuals in the video. For each individual fragment we compute the probability vector $P_2 = (P_2(F,1),\ldots,P_2(F,N))$ and the certainty $cert_2(F) = \frac{P_2(F,a)}{P_2(F,b)}$, where a and b are the indices of the first and second highest values in the P_2 vector.

idtracker.ai identifies all the fragments by decreasing order of $cert_2(F)$ with the identity $i = \arg\max(P_2(F))$, or $i = 0$ if the maximum value of $P_2(F)$ is the same for several identities. After identification, if $i \neq 0$ we set $P(F,i) = 1$ and $P(F,j) = 0$ for all $j \neq i$ and we recompute P_2 for all its coexisting unidentified individual fragments. Following 2.3, this will make $P_2(\bar{F},i) = 0$ for all $\bar{F}$ in γ_F which ensures that two coexisting individual fragments are not assigned with the same identity.

We fix the identity of the fragments assigned with a $P_2(F,i) > 0.9$ so that they cannot be modified during the post processing step (see Section 2.2.1.6).

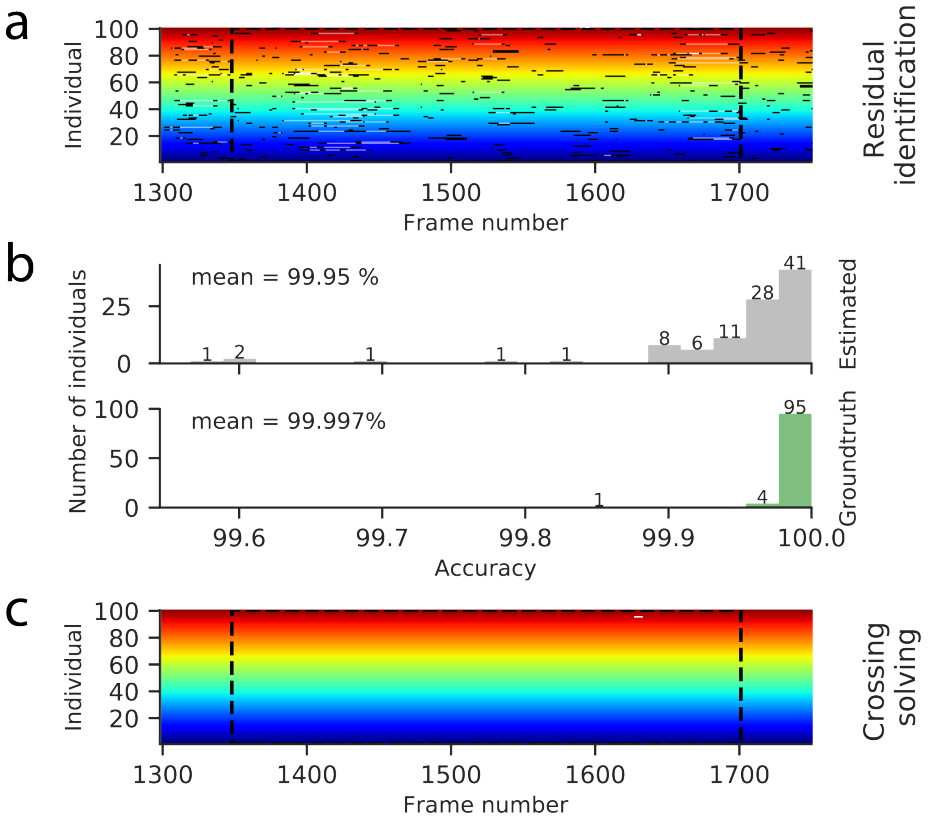

Figure 2.8. **Residual identification and crossing solving. a.** Residual identification. Colors follow the same code as in Figure 2.7. Close-up for a interval of frames of a video of 100 juvenile zebrafish. The lower transparency fragments represent the fragments identified at the end of the residual identification processing block. Dashed vertical lines indicate the edges of the same frames of the interval indicated in Figure 2.7. **b.** Histograms of estimated and manually validated identification accuracy for each individual in an interval of 3000 frames around the first global fragment. The number indicates the number of individuals in each bin. **c.** Individual fragments identified at the end of the crossing solving processing step. The white segment indicates a segmentation mistake that creates a gap of frames for which the corresponding animal is not identified. Dashed vertical lines indicate the edges of the same frames of the interval indicated in Figure 2.7.

This threshold can be modified by the user (see Section 2.2.2.2 and the advanced parameter `FIXED_IDENTITY_THRESHOLD` in Supplementary Table A.6).

At this point idtracker.ai computes an estimate of the accuracy of the identification of all individual fragments as:

$$\mathcal{A} = \frac{\sum_{F \in \mathcal{I}} \max(P_2) n_F}{\sum_{F \in \mathcal{S}} n_F} \tag{2.4}$$

where $\mathcal{I}$ is the set of all identified individual fragments, $\mathcal{S}$ is the set of all individual fragments and n_F is the number of blobs in an individual fragment.

idtracker.ai outputs this value at the end of the tracking process as a measure of the quality of the tracking. Note that $\mathcal{A} = 0$ if all individual fragments are left unidentified. In Figure 2.8b we show the estimated accuracy for each individual in a video together with the ground truth validation accuracy (see Section 2.2.3.2). In general we find the actual accuracy to be higher than the estimated accuracy when the second one is above 99.5% (see Section 2.2.3.2 and Figure 2.18).

2.2.1.6 Postprocessing

This processing step identifies and corrects identification mistakes in the individual fragments using the assumption that animals cannot move at supernatural speeds.

First, idtracker.ai computes an estimate of the maximum speed threshold at which an animal in the video can move. We recall that each individual fragment is a collection of blobs and that for each blob idtracker.ai computed a centroid during the animal detection processing step (see Table 2.2). Given an identified fragment F, we compute the speed of the animal in each frame of the fragment by taking finite differences. We set the speed threshold at which an animal can move as $v_{max} = 2 \cdot P_{99}(\mathcal{V})$, where $\mathcal{V}$ is the collection of all the speeds for all the frames of all identified individual fragments (see Section 2.2.2.2 and the advanced parameter `VEL_PERCENTILE` in Supplementary Table A.6).

We recall that an individual fragment F_i is defined by an interval of frames $[f_s, f_e]_i \equiv [f_{s,i}, f_{e,i}]$. Two individual fragments F_1 and F_2 are *consecutive fragments* if they have the same identity, $f_{e,1} < f_{s,2}$ and there is no other fragment with identity with frames in the interval $[f_{e,1}, f_{s,2}]$. Then the mean inter-fragment

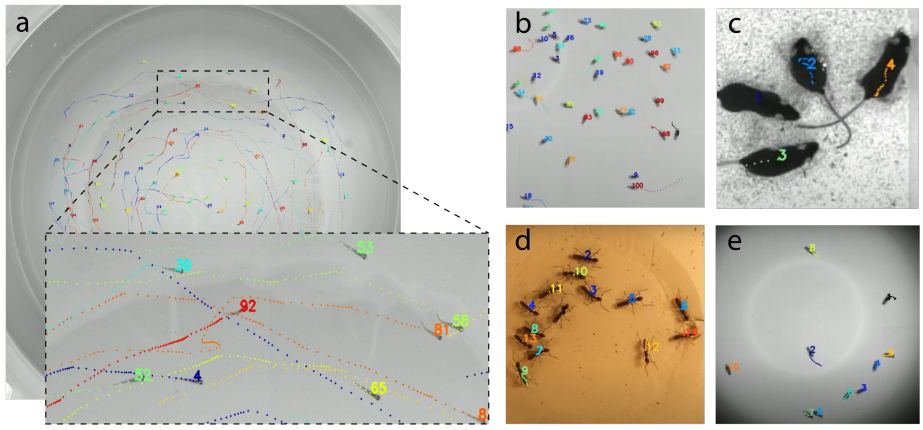

Figure 2.9. **Example video frames with resulting trajectories superimposed. a.** Frame and close-up of a 100 juvenile zebrafish video. **b.** Close-up of a frame of a 100 fruit flies video. **c.** Close-up of a frame of a 4 black mice video. **d.** Frame of a 14 ants video with illumination reflections in the body of the animals. **e.** Frame of a 10 fruit flies video that had changing light conditions. See more videos in `https://www.youtube.com/channel/UCb-0_pOXLLqGbbh3IH7AEQA`

.

speed can be defined as $v_{IF} = \frac{c_2(f_{s,2}) - c_1(f_{e,1})}{f_{s,2} - f_{e,1}}$, where c_1 and c_2 are the centroid of a blob in the fragments F_1 and F_2 respectively. Note that a given individual fragment will typically have two consecutive fragments: one in the past and another in the future. We can compute two mean inter-fragment speeds, v^p_{IF} and v^f_{IF} respectively.

For a given individual fragment F, idtracker.ai detects a possible identification mistake if $v^p_{IF} > v_{\max}$ or $v^f_{IF} > v_{\max}$. idtracker.ai attempts to correct the identity by checking the available identities within all the coexisting individual fragments of F. If the correction of the mistake generates a supernatural inter-fragment speed the individual fragment is left unidentified, i.e. its identity is set to 0.

2.2.1.7 Trajectories interpolation for crossing solving

This processing step uses the centroids of all the identified blobs to interpolate the position and identity of animals in unidentified individual and crossing

blobs. idtracker.ai assigns centroids and identities to the crossing blobs following a Python implementation of the algorithm described in (Pérez-Escudero et al., 2014, Supporting text, Section 2.12).

At the end of this process all blobs in the video are equipped with one or multiple identities and centroids (see Figure 2.8c) depending on whether they are individual or crossing blobs.

2.2.1.8 Trajectories generation

This processing step generates the individual trajectories of the animals in the video. idtracker.ai generates the trajectory of the animal with identity i by taking the centroid of all the blobs identified with such identity.

The final result is an array of dimensions $(M, N, 2)$ containing the (x, y) coordinates corresponding to the position of the N animals in each of the M frames of the video (see Figure 2.9).

2.2.2 Software

We deployed idtracker.ai with an easy-to-use graphical user interface (GUI) so that researchers with no experience with computer programming could benefit from our tracking algorithm.

In Section 2.2.2.1 we describe the main features of the idtracker.ai GUI with an emphasis on how they impact the usability and performance of the software. We also explain how users can track videos using the command line, which allows tracking multiple videos sequentially. In Section 2.2.2.2 we describe some of advanced parameters that the user can modify (or set) using a configuration file. These advanced parameters can modify default parameters of the tracking GUI, change the behaviour of some of the processing steps (see Figure 2.1), or adapt the computing performance to the hardware specifications of the computer in use. Finally, in Section 2.2.2.3 we describe the different features of a second GUI that users can use explore the resulting trajectories and validate idtracker.ai on their videos.

Note that the GUI that we present here is different to the one deployed at the publication time (Romero-Ferrero et al., 2019). In particular, we explain the

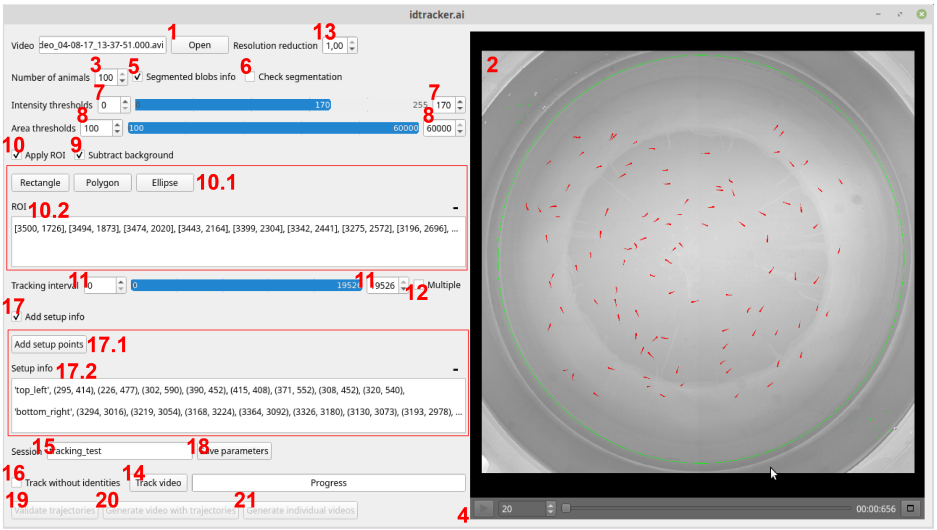

Figure 2.10. **idtracker.ai main GUI window**. The left part of the window contains text boxes, buttons, and sliders to interact with the idtracker.ai software. Each red number indicates an interactive element that is referenced in the main text. The right part of the window shows a frame of a video of 100 juvenile zebrafish with the segmented animals highlighted in red. The green circle indicates the region of interest selected by the user.

GUI as it is in the current development version that will be released in the near future.[5]

2.2.2.1 Tracking GUI

After installation,[6] users can open the idtracker.ai GUI through the command line with the command `idtrackerai` (see Figure 2.10).

To track the animals in a video, idtracker.ai requires selecting the path of a video and setting the values of the animal detection parameters that the algorithm will use to detect the animals in each frame of the video (see Table 2.1). The GUI (figure 2.10) allows the user to input this information and to visually inspect the

[5] `https://gitlab.com/polavieja_lab/idtrackerai/-/tree/v4-dev`
[6] We provide installation instructions at `https://idtracker.ai/`.

effect that the values of the animal detection parameters will have on the detection of the animals in each frame.

The users can browse through their directories and load a video using the "Open" button in the GUI (number 1 in Figure 2.10). The first frame of the video will appear in the video player panel (number 2 in Figure 2.10). In this panel, idtracker.ai highlights in red the regions in the frame that the algorithm will consider as animals with the current values of the animal detection parameters. Remember that we called each of these red regions a *segmented blob* or simply a *blob* (see Section 2.2.1.1). At this point it is important to remember two of the main assumptions under which we designed idtracker.ai.

1. All the segmented blobs in a frame correspond to one or multiple animals.

2. The number of blobs in each frame is smaller than or equal to the number of animals in the video.

Thus, the user must set the animal detection parameters such that these two assumptions are met as far as possible. When these two assumptions are met we say that the the animal detection parameters are correctly set. Note that idtracker.ai will still work if both assumptions are not fulfilled in all frames, but the tracking performance will be undermined.

As stated before, the users must also input the number of animals that appear along the video (see Table 2.1). They can do this in the in the text box "Number of animals" (number 3 in Figure 2.10). Even when the animal detection parameters are correctly set, the user can obtain a better tracking performance when the "Number of animals" matches the number of blobs in the frame for as many frames as possible along the video.[7]

On startup, idtracker.ai sets all the animal detection parameters to the default values for a good detection of the animals in the example video of the quick-start[8] (the user can modify the default values, see Section 2.2.2.2 and Supplementary Table A.1). To inspect whether the default animal detection parameters are correct for the video at hand, users can play and pause the video, or move to

[7] Assuming the sequential overlapping of blobs, this corresponds to fewer crossings fragments, and hence more global fragments (see Section 2.2.1.3 and Section 2.2.3.2).

[8] We provide a step by step quickstart guide in `https://idtracker.ai/`.

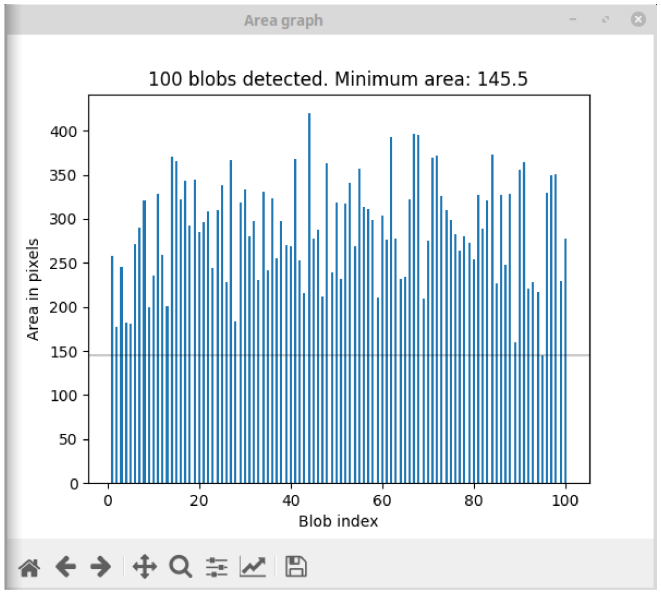

Figure 2.11. **Segmented blobs information pop-up window.** The x axis indicates the index of each segmented blob. The y axis indicates the area of the blobs in pixels. The horizontal gray line indicates the area of the smallest blob. The text on top of the plot indicates the number of blobs detected and the area of the smallest blob.

any particular frame using the text box with the frame number or the track bar (number 4 in Figure 2.10).

To check that the two previous assumptions are met and that the number of blobs in each frame matches the number of animals for as many frames as possible, the users can mark the check-box "Segmented blobs info" (number 5 in Figure 2.10). This will pop a window up with a text indicating the number of blobs in the frame, the area of the smallest blob and a bar plot with the areas of all the blobs (see Figure 2.11). The content of this window will update automatically according to the frame displayed in the video player panel.

If the user marks the check box "Check segmentation" (number 6 in Figure 2.10) and the second assumption is not met for some frames of the video, idtracker.ai will stop the tracking process and it will output the frame numbers

for which the second assumption is not met (see Section 2.2.1.1). This allows the user to set again the animal detection parameters for a better detection of the animals.

The parameters that the user can modify to improve the detection of the animals are: "Intensity thresholds", "Area thresholds", "Subtract background", "Apply ROI" and "Tracking interval" (numbers 7-11 in Figure 2.10). In the following paragraphs we explain the effect of changing these parameters and some of the most common use cases.

The "Intensity thresholds" (number 7 in Figure 2.10) are the two intensity thresholds I_{min} and I_{max} that define the range of pixel intensities that idtracker.ai uses to define the segmented blobs (see Section 2.2.1.1). Lower values of these two thresholds correspond to darker pixels in the frame, higher values correspond to brighter pixels in the frame. If the contrast between the animals and other objects in the frame (background, walls of the arena, etc.) is high, just setting the "Intensity thresholds" can be enough to detect the animals.

The "Area thresholds" (number 8 in Figure 2.10) are two values $A_{\text{min}} < A_{\text{max}}$ that define the range of areas of the segmented blobs that idtracker.ai will consider (see Section 2.2.1.1). These parameters allow to discarding objects other than animals that are in the same range of pixel intensities as the animals but are smaller or larger than the animals. This can be the case of some dust in the background of the arena, or some structural parts of the setup visible in the frame.

When the user marks the "Background subtraction" check box (number 9 in Figure 2.10), idtracker.ai computes a model of the background of the video (see Section 2.2.1.1) and subtracts from to each frame of the video. This is useful when there are static objects in the frame within the same range of pixel intensities as the animals, but with a similar size, so they cannot be discarded with the "Area thresholds".

When the user marks the "Apply ROI" check box (number 10 in Figure 2.10), idtracker.ai opens three additional buttons and a text box (see number 10.1 and 10.2 in Figure 2.10). The buttons indicate the different shapes of the regions of interest (ROI) that the user can draw in the frame. The user can draw the ROIs in the frame by clicking and dragging (for the "Rectangle), by clicking in

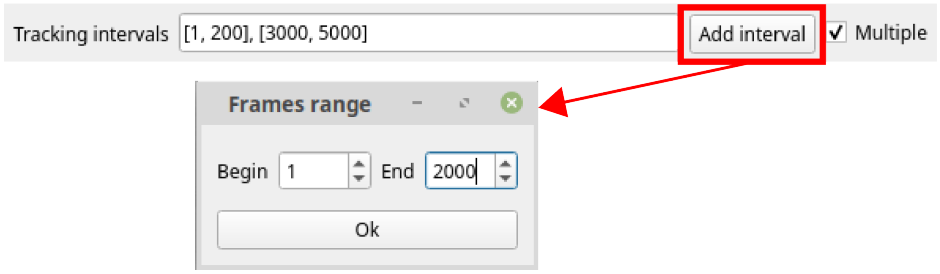

Figure 2.12. **Multiple tracking intervals GUI element.** This element of the GUI allows setting one or multiple frame intervals. Frames outside of such intervals are left out during the different processing steps of idtracker.ai.

five points in the frame (for the "Ellipse") and, by clicking in as many points as wanted (for the "Polygon"). The text box will show the pixels coordinates of the key points of the drawn ROIs (see green circle in video player window in Figure 2.10). The user can click on the list of coordinates and use the button "-" to delete a given ROI. Only segmented blobs inside of the drawn ROIs will be considered during the animals detection process (see Section 2.2.1.1). Using ROIs is useful when there are objects within the same range of pixel intensities as the animals but these objects are in parts of the frame that are never visited by the animals (e.g. the outside of the arena).

The "Tracking interval" (number 11 in Figure 2.10) are two numbers $f_1 < f_2$ that define the frames range for which idtracker.ai will apply the animal detection algorithm (see Section 2.2.1.1). Setting a "Tracking interval" is useful when the beginning or end of the video show objects other than animals that would be segmented (e.g. the hands of the experimenter introducing the animals into the arena or taking them out).

When the user marks the "Multiple" checkbox (number 12 in Figure 2.10) idtracker.ai pops a window up allowing the user to define multiple tracking intervals (see Figure 2.12). This can be useful when the user wants to discard an interval of frames that is neither at the beginning nor at the end of the video. Note that setting multiple intervals might undermine the performance of the tracking as individual fragments of different tracking intervals won't coexist (see

Section 2.2.1.4). This impact will be higher the more different the video conditions are between the different tracking intervals.

The idtracker.ai GUI allows modifying the value of another parameter that can improve the performance of the tracking in terms of time and computer memory, but is not directly related to the detection of the animals, which is the "Resolution reduction" (number 12 in Figure 2.10).

The "Resolution reduction" text box (see number 13 in Figure 2.10) allows reducing the resolution of each frame by a given factor smaller than 1 (see Section 2.2.1.1). The performance of idtracker.ai in terms of memory is directly related to the size of the segmented blobs. Very large segmented blobs will generate larger amounts of data and will slow down the tracking process. By setting the "Resolution reduction" factor below one, the user can effectively reduce the size of the segmented blobs. Note that a very aggressive resolution reduction might destroy information about the appearance of the animals which can undermine the performance of the tracking (see Section 2.2.3.4).

After the user has set the value of the different parameters, they can start the tracking process with the "Track video" button (number 14 in Figure 2.10). idtracker.ai stores the trajectories and the byproduct data resulting from the tracking process in a folder next to the video file. We call this folder the *session folder*. The user can modify the name of this folder in the "Session" text box (number 15 in Figure 2.10).

In some situations, the analysis of the behaviour of the animals might not require identifying the animals in the video. The user can mark the "Track without identities" check box (number 16 in Figure 2.10) to obtain trajectories where the identities of the animals are not maintained along the video. In this case, idtracker.ai runs much faster as this does not require training the idCNN. Note that when this check box is selected, idtracker.ai does not perform the interpolation of the trajectories during the crossings (Section 2.2.1.7). The resulting trajectories have will have missing values during the crossings. This would be equivalent to having anonymous fragments of trajectories or "tracklets" (see Section 1.2.2.1 and Robie et al. (2017)).

For some experiments, it is also useful to annotate the positions of some parts of the experimental setup. When the users mark the "Add setup info" checkbox

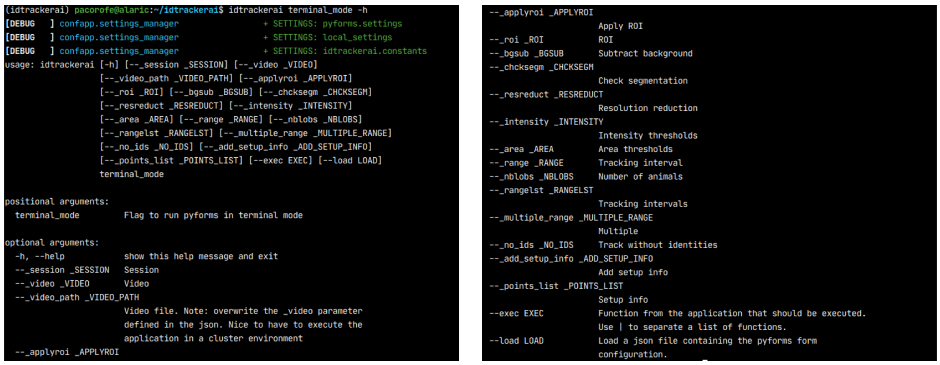

Figure 2.13. **idtracker.ai `terminal_mode` input arguments.** Screenshot of a Linux Mint terminal showing the available input parameters of idtracker.ai when executed using the command line.

(see number 17 in Figure 2.10) idtracker.ai shows the button "Add setup points" and a text box (see numbers 17.1 and 17.2 in Figure 2.10). By clicking the button "Add setup points" the user can then select points in the video player panel using the mouse. The user can give a name to each list of points. The points will appear in green in the video player panel (see top left and bottom right corners in the video player panel in Figure 2.10). The list of points and the name will also appear in the text box (see number 17.2 in Figure 2.10). This points will then be saved along with the trajectories in the `trajectories.npy` and `trajectories_wo_gaps.npy` files.

Because we coded the idtracker.ai GUI using the Pyforms framework[9] the user can input all the previously described parameters and start the tracking through the command line (see Figure 2.13). In the GUI, the "Save parameters" button (number 18 in Figure 2.10) saves all the parameters in the GUI in a JSON file next to the video file. Using the `--load` input argument, the user can pass the path of the JSON file to the `idtrackerai` command to track the video through the command line reading the parameters from the JSON file (see Figure 2.13). This feature is particularly useful when the user wants to track multiple videos sequentially without having to wait for the tracking process to finish before setting the parameters of the next video. The user can set and save

[9] `https://pyforms.readthedocs.io/en/v3.0/`.

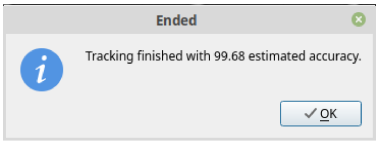

Figure 2.14. **Estimated accuracy window pop-up.** Window pop-up that appears at the end of the tracking process showing the estimated accuracy of the video at the end of the residual identification processing step (see Section 2.2.1.5).

the parameters for as many videos as they want to track and then use a Python or bash script to execute the tracking command for all of them.[10]

At the end of the tracking process idtracker.ai pops a window up showing the value of the estimated accuracy (see Figure 2.14 and Section 2.2.1.5). If the user tracked the video using the command line, idtracker.ai will display the value in the command line window.

idtracker.ai stores the files with the resulting trajectories in two folders inside the session folder. The folder `trajectories` contains a file named `trajectories.npy` with the trajectories considering the algorithm up to the post-processing processing step (see Section 2.2.1.6). The folder `trajectories_wo_gaps` contains a file named `trajectories_wo_gaps.npy` with the trajectories considering all the algorithm steps.[11]

After idtracker.ai finishes tracking the video, the user can click the "Validate trajectories" button (number 19 in Figure 2.10) to open the validation GUI which allows the user to explore, correct and validate the resulting trajectories. The user can open the validation GUI independently of the idtracker.ai GUI (see Section 2.2.2.3).

It is sometimes useful to visually inspect the trajectories superimposed on the original video. The "Generate video with trajectories" button (number 18 in Figure 2.10) will generate such video and save it inside of the session folder.

Some users can be interested on using idtracker.ai together with an animal pose tracking system like LEAP, DeepLabCut, or DeepPoseKit (Mathis et al.,

[10] In `https://idtracker.ai/` we provide a basic python script that does this.

[11] In `https://idtracker.ai/` we provide links to Jupyter Notebooks that explain how to load the trajectories using Numpy (Oliphant, 2006; Van Der Walt et al., 2011) or the Trajectorytools package (`https://github.com/fjhheras/trajectorytools`).

2018; Pereira et al., 2019; Graving et al., 2019). The "Generate individual videos" button (number 21 in Figure 2.10) generates a video for each animal identified. We center the animal in the frame using the coordinates of its trajectory. The user can use these videos to manually label the different body parts of the animals and train any their preferred animal pose tracking system.

2.2.2.2 Advanced parameters

Apart from the parameters exposed in the GUI, idtracker.ai has multiple extra parameters that we call advanced parameters (see Supplementary Tables A.2-A.7). These parameters can be modified using a settings file called `local_settings.py` which should be placed in the same directory from where the `idtrackerai` tracking command is executed. On startup, idtracker.ai reads the values of the advanced parameters defined in the `local_settings.py` file.

Some of these advanced parameters are constants of the algorithm that we left fixed to the values used during the development and validation of the algorithm (see Section 2.2.1). With enough understanding of the algorithm the users can change these parameters to modify the default behaviour of idtracker.ai algorithm. For example, they can change the training hyperparameters of the CNNs (batch size, learning rates, dropout probability, stopping criteria thresholds, etc.), or some of thresholds of the cascade of training and identification protocols (see Section 2.2.1.4). Other advanced parameters do not require such a deep understanding of the idtracker.ai algorithm and can help with improving the performance and usability of idtracker.ai for each user case.

In the following paragraphs we describe the most commonly used advanced parameters.

The default values of the animal detection parameters displayed in the GUI on startup can be modified with the advanced parameters `NUMBER_OF_ANIMALS_DEFAULT`, `MIN_THRESHOLD_DEFAULT`, `MAX_THRESHOLD_DEFAULT`, `AREA_LOWER`, `AREA_UPPER`, `MIN_AREA_DEFAULT`, `MAX_AREA_DEFAULT`, `RES_REDUCTION_DEFAULT` (see Supplementary Table A.1). Modifying these values can speed up the process of manually setting the animal detection parameters if they are set to values that result in a good segmentation for a given set of videos with similar conditions.

idtracker.ai has some routines coded to run in parallel to take advantage of the multi-threading capabilities of most modern computers. However, users might not want to use so many resources of their computer to run idtracker.ai. The advanced parameters `NUM_OF_JOBS_BKG_SUB` and `NUM_OF_JOBS_SEG` (see Supplementary Table A.2) allow the user to set the number of processes that idtracker.ai will use to compute the background model or to perform the animal detection steps (see Section 2.2.1.1).

During the tracking process, idtracker.ai generates a considerable amount of data. To prevent the RAM from filling up and blocking the computer, idtracker.ai stores the most heavy data on the disk. However, users running idtracker.ai on computers with larger RAM capacities can decide to store these objects in the RAM by changing the value of the parameters `SAVE_PIXELS` or `SAVE_SEGMENTATION_IMAGE` from `'DISK'` to `'RAM'` (see Supplementary Table A.2).

As we mentioned in the previous section, at the end of the tracking process, idtracker.ai saves all the generated data, including the resulting trajectories in the session folder. Depending on the length of the video and the area of the segmented blobs, the total size of this folder can add up to some tens of gigabytes. In most of the cases, the users will only be interested on the files containing the resulting trajectories, or on the files needed to perform the validation of the trajectories. The advanced parameter `DATA_POLICY` can be modified accordingly so that idtracker.ai saves only the necessary files at end of the tracking process (see Table 2.3 for the different options and Supplementary Table A.7).

The cascade of training and identification protocols (see Section 2.2.1.4) automatically gathers data from the video and trains the idCNN. After training, idtracker.ai saves the CNN weights in the session folder in a subfolder called "accumulation_n", where n is a counter indicating the number of idCNN instances that idtracker.ai trained. A common approach when using CNNs to classify a given set of objects is to perform knowledge transfer from a network trained to classify similar objects (Weiss et al., 2016). A user wanting to track a video with similar light conditions and appearance of the animals to a previously tracked video can use the `KNOWLEDGE_TRANSFER_FOLDER_IDCNN` advanced parameter to perform knowledge transfer from the CNN trained during the tracking of the first

DATA_POLICY option	Description
'all'	Saves all the data as it is generated during tracking
'trajectories'	Saves only the resulting trajectories
'validation'	Saves the information needed to validate the video
'knowledge transfer'	Saves the information needed to use the trained idCNN as a starting model to track the video
'identity matching'	Saves the information needed to perform identity matching between videos

Table 2.3. **Options for the advanced parameter** DATA_POLICY. The left column shows the Python strings that can be passed to the DATA_POLICY advanced parameter. The right column shows a description of the effect of using such option.

video (see Section 2.2.1.4 and see Supplementary Table A.5). The user must only set the KNOWLEDGE_TRANSFER_FOLDER_IDCNN variable to be the path to the folder where the model is stored. idtracker.ai will initialize the weights of the convolutional layers with the pre-trained weights and only train the fully connected layers of the new CNN. Using this feature can speed up the tracking process, but at the same time it can undermine the accuracy of the output trajectories (see Supplementary Table A.16).

Similar to the knowledge transfer, if the users want to track a video with exactly the same animals as a previously tracked video and obtain the same identities for the same animals, they can use the IDENTITY_TRANSFER advanced parameter together with the KNOWLEDGE_TRANSFER_FOLDER_IDCNN. When the IDENTITY_TRANSFER advanced parameter is set to True, idtracker.ai tries to identify the animals in the first global fragment of the new video using the former CNN stored at KNOWLEDGE_TRANSFER_FOLDER_IDCNN (see Section 2.2.1.4). For this kind of identity transfer to work properly the light conditions and appearance of the animals in both videos must be as similar as possible.

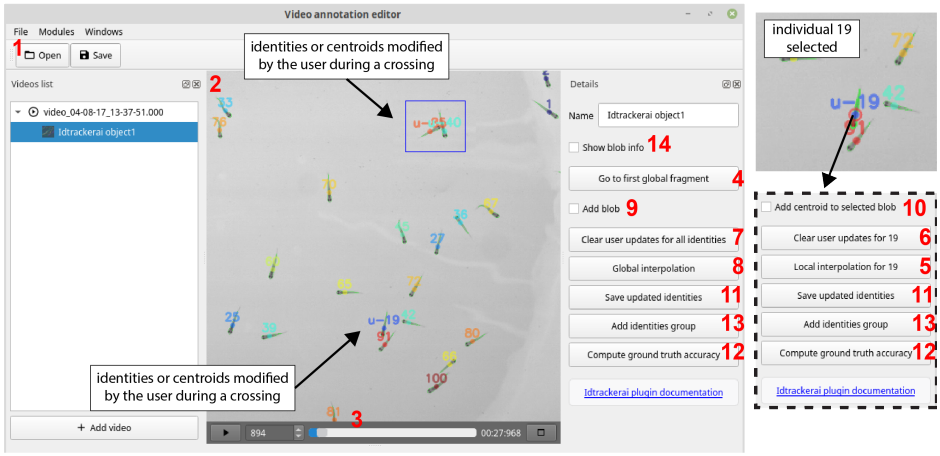

Figure 2.15. **Validation graphical user interface (GUI).** Each red colored number indicates an GUI item that is referenced in the main text. The top right close-up show a detail of the state of the centroid when an individual is selected. The dashed box shows alternative options that appear when an individual is selected.

2.2.2.3 Validation graphical user interface

Having high quality animal trajectories is important so that we can trust the analysis that we perform with them. The level of quality that users require might depend on the kind of analysis that they will perform. idtracker.ai can generate high quality trajectories (see Section 2.2.3.2) but this depends on the conditions of the video (see Section 2.2.4.1), the animal detection parameters set by the users (see Section 2.2.2.1 and 2.2.1.1) and the behaviour of the animals. Thus, users are responsible of making sure that the trajectories obtained using idtracker.ai are valid for the kind of analysis that they are going to perform. We call this process the validation of the trajectories.

To evaluate the quality of animal trajectories obtained with a given tracking software it is common to define certain performance scores to measure how different the obtained trajectories are from the ones that the user would expect. We refer to the trajectories expected by the user as *ground truth trajectories*. Manually creating ground truth trajectories is time consuming. A more efficient way of

creating ground truth trajectories is correcting a given set of trajectories that we can obtain with a tracking software. We have developed a validation GUI that allows users to explore and correct the trajectories generated with idtracker.ai and to generate ground truth trajectories for validation.

The idtracker.ai validation GUI is coded as a plug-in of a higher purpose annotation tool called Python Video Annotator.[12] Here we only describe the features of the idtracker.ai validation GUI.

As stated above, from the idtracker.ai tracking GUI (see Section 2.2.2.1) and after tracking a video, users can access the validation GUI by clicking the button "Validate trajectories" (number 17 in Figure 2.10). This opens the Python Video Annotator main window, loads the results of the tracking, and shows the main buttons of the validation GUI (see Figure 2.15). Alternatively, users can open the Python Video Annotator from the command line with the command `start-video-annotator` and then open a new Python Video Annotator project (see number 1 in Figure 2.15) by selecting the session folder where idtracker.ai stored the data generated during tracking. The Python Video Annotator will detect that the folder contains the results of idtracker.ai and load the validation GUI. The users must set the advanced parameter `DATA_POLICY` to the correct value so that idtracker.ai keeps the necessary data after tracking (see Section 2.2.2.2).

The video player panel (see number 2 in Figure 2.15) shows the video with information about the blobs superimposed. Users can move through the video using the different buttons of the video player panel as explained in Section 2.2.2.1 for the video player panel of the tracking GUI (see number 3 in Figure 2.15). For all blobs we show their contour (green line around the animals). This helps with identifying blobs representing a single animal or multiple animals. For each segmented blob representing a single animal we show the centroid of the blob, i.e. the position that defines the trajectory of the animal in that frame, and a number indicating its identity. For each segmented blob representing multiple animals, we show the centroids generated during the interpolation (see Section 2.2.1.7) and a number with its identity. In this case, the identity has the prefix "c-" to indicate that idtracker.ai assigned that centroid during the interpolation of the

[12] `https://pythonvideoannotator.readthedocs.io/en/master/`

crossings. Since identification errors occur before and after a crossing, we enclose the blobs belonging to crossing fragments with a blue rectangle so it is easy to find the crossings while exploring the video.

As explained in Section 2.2.1, idtracker.ai trains a CNN to identify single animals. It does so by automatically gathering data from different parts of the video. At the beginning of the training process, the identities of the animals are set arbitrarily in a part of the video that we called the first global fragment (see Section 2.2.1.4). Thus, it makes sense to start correcting the trajectories in the part of the video where the identities were set for the first time. The first frame that the validation GUI shows on startup is the first frame of the core of the first global fragment (see Figure 2.5b). To return to the this initial position users can click the button "Go to first global fragment" (see number 4 in Figure 2.15).

Users can correct the identity of each centroid. Double-clicking on a centroid pops a window up that prompts the user for the identity that should be set to that centroid. The user can set the desired identity. Since a blob belongs to a given individual or crossing fragment (see Section 2.2.1.3), all the centroids with the same identity of previous and future blobs belonging to the same fragment will also be modified.

Users can also correct the position of a centroid. Clicking on a centroid and dragging it to the desired position will modify the x and y coordinates of the centroid for the current frame. Modifying the centroid of a given identity for several frames can be a time consuming task. To facilitate this work, we allow the users to interpolate the centroids between two frames with modified centroids using the button "Local interpolation for ID" (see number 5 in Figure 2.15). This will use all the updated centroids between the two frames as anchor points and it will update the position of the rest using the interpolated values.

The identity label of modified identities or centroids will have the prefix "u-" to indicate that the user updated it (see Figure 2.15).

To reset an updated identity and centroid to its original value, users can click on a given centroid with the "u-" prefix and then press the button "Clear user updates for ID" (see number 6 in Figure 2.15). A window will pop up and ask the user for the interval of frames for which they want to reset the identity and the centroid. If the users did not select any centroid, they can click the button

"Clear user updates for all identities" (see number 7 in Figure 2.15) to reset all updated identities and centroids in an interval of frames.

Sometimes it is faster to only correct the identities of the animals before and after the crossings and let the crossing interpolation algorithm (see Section 2.2.1.7) solve the identities and positions of the animals during the crossings. For this, users can use the button "Global interpolation" (see number 8 in Figure 2.15).

If the user did not set correctly the animal detection parameters (see Section 2.2.1.1), it is possible that idtracker.ai did not detect some animals. Users can use the check box "Add blob" to add a centroid with an identity in any position of the frame (see number 9 in Figure 2.15).

While it is very uncommon, it is possible that idtracker.ai detected a blob with multiple animals as an individual blob. Also, in crossings blobs with multiple animals, it is possible that the crossing interpolation algorithm did not assign a centroid and an identity to some of the animals. In these situations, it is useful to add a centroid to a given blob to consider the missing identified animals. Users can use the checkbox "Add centroid to selected blob" (see number 10 Figure 2.15) to add a centroid with an identity to a selected blob.

To save the corrected identities and positions, users can press the button "Save updated identities" (see number 11 Figure 2.15). This will save the user's updates in the `blob_collection.npy` (`blob_collection_wo_gaps.npy`) file and it will generate new trajectories files `trajectories.npy` (`trajectories_wo_gaps.npy`) considering the updates. The new trajectories file name will include a time stamp indicating the date and time of the correction.

The validation GUI also allows users to compute some tracking performance scores (see Supplementary Table A.11). Users can use the button "Compute ground truth accuracy" (see number 12 in Figure 2.15). This button will prompt a window so the users can indicate whether they want to compute the accuracy for a fragment of the video, or they want to compute the individual accuracy for a subset of animals (see Supplementary Table A.11). To compute other performance scores users can use the `blob_collection.npy` and `blob_collection_wo_gaps.npy` as they include information about all the users

updates during the validation process without destroying information about the original results of idtracker.ai.

For some experiments it is useful to analyse the behaviour of the animals as subgroups depending on experimental conditions that the experimenter can identify from the video. Thus users can also generate groups of identities with the button "Add identities group" (see number 13 in Figure 2.15). This information will be saved in the `trajectories.npy` (`trajectories_wo_gaps.npy`) file together with the trajectories.

Finally, the validation GUI is not only a useful interface to explore and correct trajectories, but it is also a valuable tool to explore different information about the blobs during the debugging and development of new features. Developers can use the checkbox "Show blob info" (see number 14 in Figure 2.15) which will show information about the selected blob.

2.2.3 Algorithm validation

In this section we present the results of different experiments that we performed to validate the trajectories generated with idtracker.ai and study the performance of the algorithm and its potential limitations. These results are the same as the ones presented in Romero-Ferrero et al. (2019). In Section 2.2.3.1 we study how the accuracy of the idCNN changes with the number of animals to be identified, and with different modifications of the identification CNN architecture. In Section 2.2.3.2 we show the results of the validation of idtracker.ai trajectories for videos of different animal species and group sizes. In Section 2.2.3.3 we study how the number of images in the first global fragment affects the performance of idtracker.ai. Finally, in Section 2.2.3.4 we study the robustness of idtracker.ai to changes in the image quality of the video.

2.2.3.1 Single image identification tests

idtracker.ai uses a CNN to identify animals in each frame of the video, the idCNN (see Section 2.2.1.4). We measured the identification capacity of the idCNN, compared it with the identification capacity of the fingerprinting method used

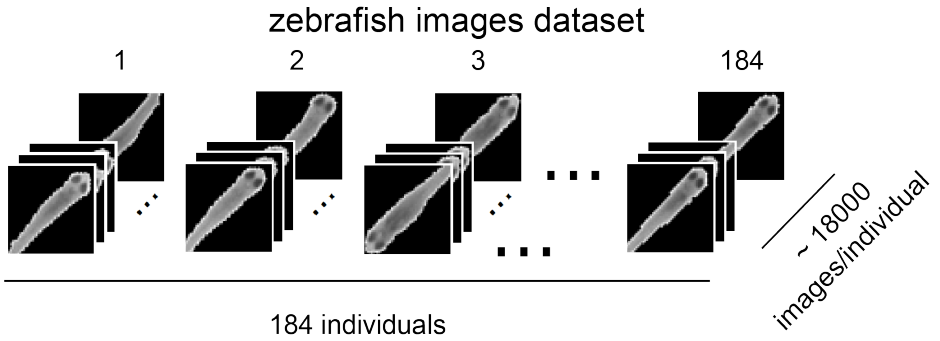

Figure 2.16. **Image dataset of individual juvenile zebrafish.** Summary of the image dataset of labeled individual juvenile zebrafish images. The dataset is composed by a total of $\approx$ 3312000 uncompressed, grayscale labelled images (52×52 pixels).

in idTracker (Pérez-Escudero et al., 2014), and explored how it changes with modifications of the CNN architecture.

To measure the identification capacity we first built a labeled dataset of $\approx 3,312,000$ images of 52 pixels of 184 different juvenile zebrafish. The dataset has $\approx$ 18000 images per animal (see Figure 2.16 and Section 2.4.5 for details of the experimental setup).

We first studied how the identification capacity changed with the number of individuals to identify. For each number of individuals N, we randomly selected 3000 images per animal for training (see Section 2.4.6). Then, we tested the network with 300 different images per animal randomly selected from the original dataset. As a measure of the identification capacity, we compute the multi-class classification accuracy of the network on the test set. We performed a 5-fold cross validation for each number of individuals to identify. For each cross validation repetition, we randomly selected N animals from the original dataset.

The results show that the multi-class classification accuracy of the identification CNN is $> 95\%$ for up to 150 individuals (see Figure 2.17). The comparison with idTracker shows that the accuracy of the idTracker fingerprinting method degrades more quickly with a multi-class accuracy of 83% for 30 animals (see also Pérez-Escudero et al. (2014, Supplementary Figure 6)).

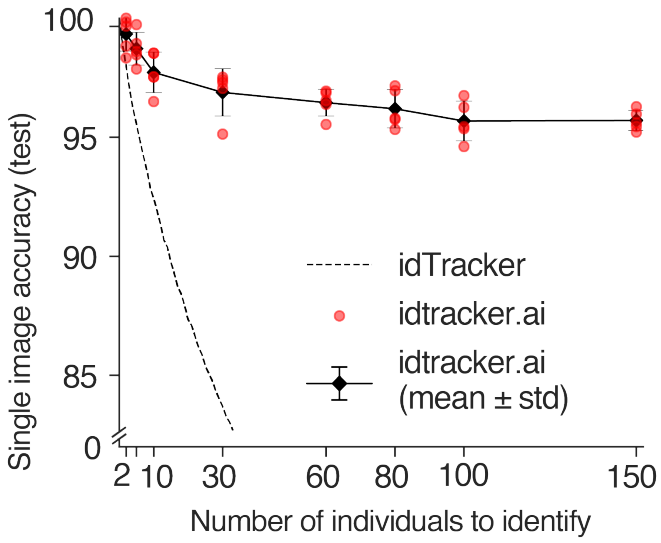

Figure 2.17. **Single-image identification accuracy.** The solid line represents the mean ± SD of the single image identification accuracy for 5 training trials (red dots). The dashed line represents the accuracy of idTracker for the same number of images.

We also tested how alternative architectures of the identification CNN impact the identification capacity (see Supplementary Tables A.9 and A.10 for the architectures). A network with only 1 or 2 convolutional layers decreased the multi-class accuracy for all the numbers of individuals to identify, while a network with 4 convolutional layers did not consistently improve the multi-class accuracy (see Supplementary Figure A.3 left). Modifications of the number of fully connected layers or the number of hidden neurons in the fully connected layers did not have a consistent change on the multi-class accuracy for all the numbers of individuals to identify (see Supplementary Figure A.3 right).

2.2.3.2 Validation on real videos

To validate that idtracker.ai is capable of tracking different group sizes of different animal species, we tracked a total of 35 videos from 5 different species and group sizes ranging from 2 to 100 animals, compared the generated trajectories with

ground truth data, and computed different performance scores (see Supplementary Table A.11). For small and intermediate group sizes ($N \leq 20$ animals) we used most videos from Pérez-Escudero et al. (2014), one video of 10 fruit flies (*D. melanogaster*) courtesy of Clara Ferreira, a video of 14 ants (*Diacamma indicum*) courtesy of Andrew Bruce, and 3 videos of 10 juvenile zebrafish (*Danio rerio*) (see Section 2.4.2). For large group sizes ($N \geq 30$ animals), we recorded new videos of fruit flies (see Section 2.4.3) and juvenile zebrafish (see Section 2.4.2).

Using the validation GUI we obtained ground truth trajectories by correcting the misidentified parts of the generated trajectories corresponding to single individuals (see Section 2.2.2.3). Identification errors typically occurred because the identity of an animal in the individual fragments before and after the a crossing fragment occlusion was not the same. We only corrected the identities of the animals in the individual fragments. For all videos we corrected the trajectories for a given interval of frames containing hundreds of crossing fragments per video (see Supplementary Tables A.12, A.13 and A.14). For videos of large groups we also corrected the trajectories of 10 randomly selected individuals for the whole duration of the video (see Supplementary Table A.13).

Supplementary Tables A.12 and A.13 show the main characteristics of the videos (duration, frames per second, area in pixels), the protocol used by idtracker.ai during the cascade of training and identification protocols (see Section 2.2.1.4), the number of crossing fragments reviewed and four performance scores (see Supplementary Table A.11) for videos of small and large groups respectively. Supplementary Table A.14 shows the same information for videos that we considered special for reasons that we will explain below.

In small groups the protocol 2 was successful for all videos (see Supplementary Table A.12). For zebrafish and medaka (*Oryzias latipes*) the accuracy was higher than 99.99% for most of the videos but for one where the accuracy was 99.969%. For fruit flies the average accuracy was $99.91\% \pm 0.1\%$ (mean $\pm$ SD; $n = 3$) and for mice the average accuracy was $99.3\% \pm 1\%$ (mean $\pm$ SD; $n = 6$).

We also tracked two videos of small groups of animals for which users reported difficulties using the idTracker software (see Supplementary Table A.12). For a compressed video of 10 fruit flies recorded in a retro illuminated arena with varying illumination conditions due to a periodical light stimulus, idtracker.ai

applied protocol 3 and achieved a 100% accuracy. For a compressed video of 14 ants with direct illumination producing multiple shadows and light reflections on the body of the animals, idtracker.ai applied protocol 2 and achieved a 99.943% accuracy.

In large groups of juvenile zebrafish, protocol 2 was always successful (see Supplementary Table A.13), giving accuracies of 99.96%±0.06% for 60 individuals and 99.99% ± 0.01% for 100 individuals (mean ± SD; $n = 3$ for both groups). With flies, idtracker.ai applied protocol 2 for a group of 38 individuals with an accuracy of 99.978%. For groups of more than 38 individuals it tracked the video applying protocol 3 and reached high accuracies (99.997%) for groups of up to 72 individuals. With groups of 80 and 100 flies, the system reached its performance limit, but still had $> 99.5\%$ accuracy.

The performance of idtracker.ai depends greatly on the behaviour of the animals. For example, in a video of 100 fruit flies where some of them spend part of the video upside down or displaying unhealthy poses due to low-humidity conditions in the setup, idtracker.ai obtained a low accuracy of 99.227% (see Supplementary Table A.14). In other videos of 60 and 100 fruit flies where animals showed very low levels of locomotor activity and some animals were dead, idtracker.ai could not track the video reporting estimated accuracies of 73.73% and 95.579% respectively.

The estimated accuracy that idtracker.ai reports at the end of the tracking process (see Section 2.2.1.5) is a conservative estimate when it is above 99.5%, i.e the actual validation accuracy is typically higher when the estimated accuracy is above 99.5%. However, this is not the case when the accuracies are lower (see Figure 2.18). Thus we could expect similar accuracies for the videos of 60 and 100 fruit flies of Supplementary Table A.14 where we only report the estimated accuracy.

For three videos of 8, 60 and 100 fish, we also computed the accuracy and saved the running time of the different processing steps of the algorithm (see Supplementary Table A.15). We can see that the different processing steps add to the accuracy and running time of the software. In particular, we can see that when the accuracy in protocol 2 is very low, the application of protocol 3 increases the accuracy from values of 33.89% and 65.64% to 89.787% and 99.315%

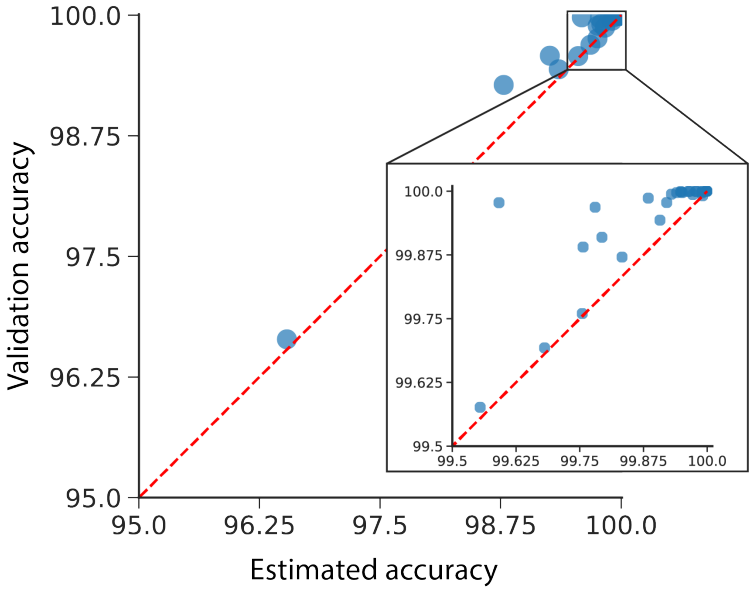

Figure 2.18. **Estimated accuracy.** Each dot represents a video used for the manual validation of the algorithm. The x axis is the accuracy estimated by the algorithm at the end of the residual identification step. The y axis is the accuracy obtained by manual validation of the identities in the video (see Section 2.2.3.2). The inset shows an expanded view for videos with accuracies higher than 99.5%.

respectively (see video of 60 zebrafish in Supplementary Table A.15, conditions G. blur. (6) and Res. red. (4) respectively). This increase in accuracy comes together with a considerable increase in time with the protocol 3 taking $\approx$ 15 hours. While not reported here, most of the time in protocol 3 is due to the pre-training of the idCNN (see Section 2.2.1.4). Large tracking times might not be desirable for most users. Using the knowledge transfer advanced parameters (see Section 2.2.2.2) can make idtracker.ai change from applying protocol 3 taking $> 60h$ to applying protocol 2 with a total tracking time of only 3.5 hours (see Supplementary Table A.16). While transferring knowledge from a previously trained idCNN can reduce the running time required to track a video, we observe that it can also have a negative impact on the accuracy.

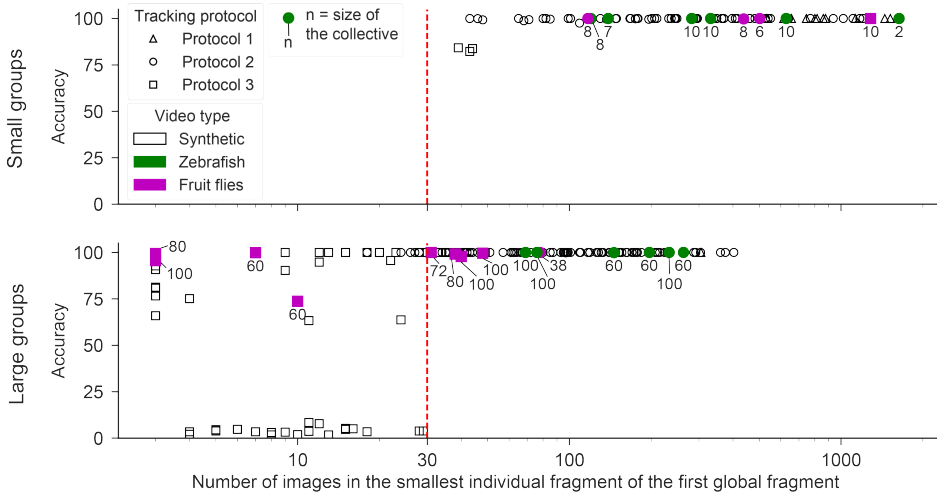

Figure 2.19. **Validation accuracy in synthetic videos**. Identification accuracy for synthetic and real videos as a function of the minimum number of images in the first global fragment. The shape of the marker indicates the protocol used to track the video. The color indicates animal species of the video. The number next to each color marker indicates the number of animals in the video. The vertical dashed line indicates the threshold of the minimum number of images in the first global fragment above which idtracker.ai showed high accuracy levels.

2.2.3.3 Validation on synthetic videos

idtracker.ai relies on the existence of multiple global fragments along the video. In particular, idtracker.ai selects the first global fragment to train the idCNN by maximizing the number of images in the smallest individual fragment and its variability (see Section 2.2.1.4). We studied how the number of images in the smallest individual fragment of the first global fragment affects the accuracy of the tracking in real and synthetic videos.

We used the same labeled image dataset that we used in Section 2.2.3.1 (see Figure 2.16) to generate synthetic videos (see Section 2.4.7). We created a total of 225 synthetic videos for groups sizes of 10, 60, and 100. We applied the cascade of training and identification protocols (see Section 2.2.1.4) and the residual identification (see Section 2.2.1.5) steps of the algorithm to identify the images in the individual fragments of the synthetic videos and computed the accuracy of

the identification. Figure 2.19 shows how the accuracy changes with the number of images in the smallest individual fragment of the first global fragment. Fruit flies can display very low motor activity periods likely due to low-humidity levels. Images of individual fragments during these periods will potentially encode less useful information to distinguish the animals since the repertoire of postures and changes in illumination will be smaller. To correct for this, we only considered images in which the animals were moving at a speed higher than $> 0.75BL/s$ (see Supplementary Figure A.4).

The tests show that the pipeline is typically successful for both real and synthetic videos with > 30 images in the smallest individual fragment of the first global fragment. Real videos of large groups of zebrafish followed this condition by a large margin, while real videos of large groups of fruit files were closer to the 30 images limit with some of them being way below the limit. Still, idtracker.ai managed to track some of these videos with high accuracy.

2.2.3.4 Image quality tests

Another factor to consider when studying the performance of the tracking is the image quality of the video. We selected two videos of 8 and 60 zebrafish and applied different artificial manipulations to decrease the image quality of the video, we tracked them, and compared the resulting trajectories with the corresponding ground truth data.

We recommend using videos with at least 300 pixels per animal at segmentation (see Section 2.2.4.1). Fewer pixels per animal imply less information available to distinguish the animals between each other. Reducing the resolution of the video has a direct impact on the number of pixels per animals. To study how the number of pixels per animal affects the tracking performance we applied 5 different resolution reduction factors to the videos (see Section 2.4.8.1). In these videos idtracker.ai shows high accuracies for up to ≈ 63 pixels per animal in segmentation which corresponded to ≈ 193 pixels per images of the identification image used by the idCNN (see Supplementary Figure A.5 and Supplementary Table A.17).

Intuitively, image definition can be a key factor to distinguish animals since sharper images can make more evident the high frequency features that the net-

work can use to identify the animals. We applied 6 different levels of Gaussian blurring to both videos by changing the standard deviation of the Gaussian kernel applied to each frame (see Section 2.4.8.2). idtracker.ai showed to be robust with high accuracies for up to a standard deviation of 3 pixels (see Supplementary Figure A.6 and Supplementary Table A.18).

Storing raw videos can be expensive and unaffordable for some users due to the large size of the files. Compressed videos occupy much less space making it cheaper to store them. Because of this, video compression is an important factor against which to test the performance of idtracker.ai. We compressed both videos using two standard video compression algorithms (see Section 2.4.8.3). We obtained above 99.8% accuracies in all cases (see Supplementary Table A.19).

Finally, we showcase how idtracker.ai can be resistant to inhomogeneous light conditions in the video. To do so, we recorded a new video of 60 animals where part of the setup illumination was switched off. This produced larger gradient of light between different parts of the setup than one would obtain with the normal light conditions (see Supplementary Figure A.7). The accuracy of this new video was 99.96%, comparable to the accuracies obtained with similar videos with more homogeneous lighting conditions.

2.2.4 Working conditions

2.2.4.1 Video conditions

In this section we describe the recommended video conditions to obtain a successful tracking of the animals in the video with high identification accuracy. Users should bear these conditions in mind when recording their videos if they plan to track the animals using idtracker.ai. Note that the following comments assume that animals will appear in the video separated from each other so that sufficient global fragments can be detected along the video.

Resolution. The resolution of the video (width and height) will affect the number of pixels of the segmented blobs that represent animals. The larger the area (in pixels) of a blob, the more information idtracker.ai will have to distinguish the animal from the rest. Notice that, on the downside, very large

areas will impact the performance in terms of running time. Blobs with large areas imply a large list of pixels and large identification images. Large list of pixels slow down the process checking whether two blobs overlap in consecutive frame (see Section 2.2.1.2). Large identification images slow down the process of training the idCNN (see Section 2.2.1.4). Check Supplementary Tables A.12, A.13 and A.14 for the average number of pixels per animal in each of the videos that we used for the validation of the software. See also Supplementary Figure A.5 and Supplementary Table A.17 for the change in accuracy with the area of the segmented blobs.

Frame rate. The frame rate of the video must be high enough so that the blobs associated with the same individual overlap in consecutive frames when moving at average speed (see Section 2.2.1.2). A low frame rate with respect to the average speed of the animals will reduce the number of consecutive frames for which the segmented blobs of a given animal overlap (see Figure 2.3b). This will result in shorter individual fragments and global fragments (see Section 2.2.1.3) impacting the performance of the tracking (see Figure 2.19). This can also generate inconsistent individual fragments, with blobs in the individual fragment belonging to different animals (see Figure 2.3d). Conversely, excessively high frame rates will make the images in the individual fragments be highly correlated without adding extra information. Since idtracker.ai will have to analyse more frames, this will increase the total running time necessary to track the video, without guaranteeing an improvement of the identification of the individuals. In the examples provided in this paper, the frame rate ranges from 25fps to 50fps (see Supplementary Tables A.12, A.13 and A.14).

Duration. The length of the video for which the system works depends on the number of animals, the distribution of images per individual fragment and the number of pixels per animal. For 8 zebrafish we can track videos as short as ≈ 18 sec (≈ 500 frames at 28 fps).[13] For large groups we can track videos as short as 1 min (≈ 1950 frames at 32 fps).[14] The system works for longer videos as long

[13] This is the example video of the quick-start step-by-step guide in `https://idtracker.ai/`

[14] This is the large group example video of the quick-start step-by-step guide in `https://idtracker.ai/`

as the overall conditions do not change abruptly in different parts of the video. Very long videos and with many animals will take longer to track, will require a high amount of RAM and could block the user's computer. The user can split the video in overlapping intervals and track each of the intervals independently. Check Supplementary Tables A.12, A.13 and A.14 for the duration of the videos that we used in the validation of the software.

Video format. The system works with any video format compatible with OpenCV (Bradski, 2000). We recommend uncompressed or loss-less video formats. Some compression algorithms work by deleting pieces of information that could be crucial for the identification of the individuals. However, we have successfully tracked videos with compressed formats: .avi (FPM4 video codec) and .MOV (avc1 video codec) (see Supplementary TablesA.14 and A.19).

Illumination. Illumination has to be as uniform as possible in each frame and through the video, so that the appearance of the animals is as homogeneous as possible. We recommend using indirect light either by making the light reflect on the walls of the setup, or by covering the setup with a light diffuser as we show in Supplementary Figures 2.20 and 2.21. Although we have also tracked videos with retro illuminated arenas (see Supplementary Table A.14), recall that the tracking systems relies on visual features of the animals that this type of illumination could hide.

Definition and focus. Images of individuals should be as sharp and focused as possible for their features to be clearly displayed throughout the entire video. When using wide apertures on the camera, the depth of field can be quite narrow. The user should ensure that the plane of the camera sensor is parallel to the plane of the arena so that animals are in focus in everywhere in the arena. In addition, exposition time (or shutter speed) should be high enough so that animals do not appear blurred when moving at average speed. Blurred and out of focus images of animals are more difficult to be identified correctly. In any case, we proved that idtracker.ai is robust under some levels of blurring (see Supplementary Figure A.6 and Supplementary Table A.18).

Background. The background of the video (objects that are not animals) should be as uniform as possible through the video. To facilitate the detection of the animals during the segmentation process the background color has to be chosen in order to maximise the contrast with the animals (see Section 2.2.1.1). Small background inhomogeneity or noise are acceptable and can be removed by the user with an adequate selection of the the animal detection parameters "Area", "Background subtraction", and "Apply ROI" in the tracking GUI (see Section 2.2.2.1).

Shadows. Shadows projected by the individuals on the background of the tank can make the selection of parameters difficult for a correct detection of the animals generating more crossing fragments and make the tracking process more difficult. Shadows can be diffused by using a transparent base separated from an opaque background (see Supplementary Figures 2.20 and 2.21) or by using retro illuminated arenas.

Reflections. Reflections of individuals on the walls of the arena with similar intensity color as the animals will count as animals during the animal detection step. This will make the number of segmented blobs be higher than the number of animals in the video. Users can reduce reflections in opaque walls by using either very diffused light or matte walls. For aquatic arenas with transparent walls, users can soften reflections by having water at both sides of the walls (see Supplementary Figures 2.20 and 2.21). Furthermore, in the tracking GUI users can select an appropriate ROI to discard reflections on the walls.

Variability in the area of the segmented blobs . The blob area is one of the features used to distinguish individual animals from crossings. idtracker.ai might encounter difficulties tracking videos with animals of very different sizes as the crossing detection step (see Section 2.2.1.2) could label some large individuals as crossings generating fewer individual fragments. Videos should fulfill the two following conditions. First, the area of the blobs representing single individuals should vary as little as possible through the video. Second, the size of an individual should vary as little as possible depending on its position in the arena.

2.2.4.2 Hardware requirements

We developed idtracker.ai in desktop computers running GNU/Linux Mint 18.1 64-bit (Intel Core i7-6800K or i7-7700K, 32 or 128 GB RAM, Titan X or GTX 1080 Ti GPUs, and 1 TB SSD disk). However, idtracker.ai is cross-platform and can be executed in Windows and MacOs computers, and with lower hardware specifications. Due to the use of CNNs, it is preferable to run idtracker.ai in computers mounting a dedicated GPU with Cuda capability 3.0 or higher. In the opposite case, idtracker.ai will train the CNNs on the CPU which can have a big impact on the tracking time. The requirements in terms of RAM and storage memory depend highly on the length of the video and the number of animals in the video.

2.3 Discussion

2.3.1 Summary of main findings

Our main goal was to examine the possibilities of convolutional neural networks (CNNs) as an identification method for a species-agnostic marker-less multi-animal tracking system. In particular, we aimed to solve the problem of automatically extracting enough images from the video to train an CNN in a self-supervised manner, i.e. without requiring manual labels of the animal identities.

In this chapter, we presented idtracker.ai, a marker-less multi-animal tracking system that maintains identities throughout the video using a CNN to identify the animals, the idCNN. idtracker.ai is species-agnostic and does not require manual annotation of animal identities as it trains the idCNN by automatically gathering data from the video. This allows idtracker.ai to identify up to 100 animals in real videos without propagation of identification errors, which was not possible before with other multi-animal tracking software that used marker-less animal identification (Pérez-Escudero et al., 2014; Bai et al., 2018; Rasch et al., 2016; Rodriguez et al., 2017; Wang et al., 2017; Zhiping & Cheng, 2017).

2.3.1.1 Algorithm

We designed idtracker.ai based on idTracker (Pérez-Escudero et al., 2014), the state-of-the-art at the time. Thus, the animals detection step is very similar to the one in idTracker (see Section 2.2.1.1). However, we redesigned the rest of the algorithm as idTracker uses pre-computed visual features to detect animal crossings and to identify the individuals in the video. Instead, we wanted to learn the optimal features for each task using CNNs.

idtracker.ai uses two different CNNs, one to detect animal crossings, the cdCNN, and another one to identify animals throughout the video, the idCNN. Both CNNs are trained without manually annotated data. The training dataset of the cdCNN is automatically constructed using a set of heuristics (see Section 2.2.1.2). For the idCNN, idtracker.ai uses parts of the video where all animals are separated from each other to define global fragments (see Section 2.2.1.3). A single global fragment typically does not have sufficiently many images to fully train

the idCNN. Thus, we designed a cascade of protocols that automatically gathers images from all global fragments in the video to train the idCNN and identifies the animals in the video. This cascade of protocols adapts to the video difficulty (see Section 2.2.1.4).

We validated the identification capacity of the idCNN with a labeled image dataset of individual juvenile zebrafish (see Section 2.2.3.1). Rasch et al. (2016) did a similar test but only showed results for 5 individuals. Our results showed that the idCNN can distinguish up to 150 individuals with $> 95\%$ accuracy in a labeled images dataset. In addition, we also showed that this accuracy was much higher than the one of idTracker identification paradigm, which uses pre-computed feature maps.

We also explored and characterized the limits of our automatic training and identification strategy in synthetic videos (see Section 2.2.3.3). In these conditions, we showed that idtracker.ai can track videos with high identification accuracy if animals appear separated from each other during intervals as short as 30 frames (see Figure 2.19). Note that while idtracker.ai can use up to 3000 images per animal to train the idCNN, these images are automatically accumulated during the cascade of training and identification protocols. Thus, idtracker.ai does not require animals to be separated from each other for 3000 frames as misinterpreted by Bozek et al. (2021), but only for at least 30 frames.

Since the animals detection step is species-agnostic idtracker.ai can track videos of any species as far as the body shape is elongated and all animals are separated from each other at some point in the video (see Figure 2.9). We validated the algorithm in videos of different group sizes and different species (see Section 2.2.3.2) We obtained similar accuracies to the ones of idTracker for small groups sizes of zebrafish, medaka, fruit flies, ants and mice. These results showed that CNNs can distinguish animals other than fish (Rasch et al., 2016; Wang et al., 2017; Zhiping & Cheng, 2017). Importantly, idtracker.ai could track large groups of zebrafish and fruit flies of up to 100 individuals with $> 99.99\%$ and $> 99.5\%$ identification accuracy respectively, being the first marker-less multi-animal tracking system capable of tracking these sizes of collectives without propagation of identification errors.

idTracker identification paradigm does not scale well for large groups of animals (see Section 1.2.2.2). However, with idtracker.ai we could track videos of 100 fish and 10 minutes of duration in less than 14 hours. We also tested idtracker.ai under different manipulations of the video quality conditions showing that it is robust under changes in resolution, image blurring, video compression and inhomogeneous light conditions (see Section 2.2.3.4).

2.3.1.2 Software

We coded idtracker.ai in the free and open-source language Python, documented the code (see `https://idtracker.ai`) and used version control so that other members of the scientific community could contribute to update it and improve it (see `https://gitlab.com/polavieja_lab/idtrackerai`). idtracker.ai itself is also an open-source and free software (license GPL v3.0).

Like others before (Branson et al., 2009; Pérez-Escudero et al., 2014; Rodriguez et al., 2017), we designed idtracker.ai to be a useful multi-animal tracking software for the rest of the scientific community. We deployed it as an easy-to-install Python package with a graphical user interface (GUI). The GUI allows the user to set the necessary animal detection parameters, start the tracking, and explore, correct, and validate the resulting trajectories (see Sections 2.2.2.1 and 2.2.2.3). In the idtracker.ai web page[15] we show, among other things, information about the installation, a quick-start step-by-step guide to track an example video, the GUI manual instructions, and guidelines to generate videos that are adequate for idtracker.ai.

For greater flexibility, we enable modifying any advanced parameter of the algorithm. This can allow the users to obtain a better performance for their use case (see Section 2.2.2.2). However, note that in all our validation videos we only modified the animals detection parameter (see Table 2.1) and all the advanced parameters were fixed to the same values.

Since the study of animal behaviour typically requires high-throughput systems, idtracker.ai can be executed directly from the command line, making easier to track multiple videos sequentially with a simple script using personal computers and institutional or cloud computing servers.

[15] `https://idtracker.ai/`

idtracker.ai outputs by default the 2D trajectories of the animals in the video (see Section 2.2.1.8 and Figure 2.9) and it does not output direct information about the posture of the animal. However, users can access the data generated during the tracking (e.g. blobs contours or bounding boxes; see Table 2.2) and apply their customized pose estimation algorithm to obtain more information about the body posture of the animal.

As suggested by Graving et al. (2019), the trajectories of a multi-animal tracking system can be used in combination with animal pose tracking systems to obtain both animal location and animal body parts trajectories of each animal in the collective. To facilitate the integration of idtracker.ai with the recently developed deep learning-based animal pose tracking systems (Mathis et al., 2018; Pereira et al., 2019; Graving et al., 2019), our system can output single animal-centered videos that users can use to annotate body parts and train their preferred pose estimation method (see Section 2.2.2.1).

All in all, we showed that it is possible to automatically train CNNs to identify and track small and large groups of unmarked animals of any species from videos without requiring manual annotation or correction. We validated and characterized the limitations of the algorithm in synthetic and real videos of different species. Finally, we deployed the algorithm in a software that is easy to install and other researchers can use it to track their videos.

As any software, idtracker.ai has limitations and it can be improved in different directions. In the next section, we first review idtracker.ai's main limitations and then discuss possible solutions.

2.3.2 Limitations and future improvements

We acknowledge that all the processing steps in idtracker.ai (see Figure 2.1) can be improved in different directions. Since our main goal was to change the identification paradigm with respect to idTracker, here we focus on the limitations that affect directly the idCNN training and identification strategy (see Section 2.2.1.4).

The main limitation is the pair of assumptions under which we designed it. idtracker.ai uses these two assumptions to build the idCNN and to robustly accumulate training samples along the video.

The first assumption is that the number of animals in the video must be known in advance. This is ensured by asking the user to input the number of animals as an parameter to the algorithm (see Table 2.1). The second one is that the video must have at least one interval where all the animals can be detected separated from each other as individual blobs (see Figures 2.5 and 2.7). This second condition depends on the behaviour of the animals and the density of animals in the group. These two factors affect the number of touches or occlusions (i.e. crossings) in the video and hence the number of parts in the video where all animals can be detected as separated (i.e. global fragments). Note that idtracker.ai uses the first assumption to check that the second one is met.

When the second assumption is not met, i.e. there are no global fragments, idtracker.ai cannot track the video. When the individual fragments in the global fragments are very short (< 30 frames), idtracker.ai can track the video but typically falls into the protocol 3 (see Figure 2.19). This requires pre-training the idCNN which is a very slow process. In these cases, idtracker.ai can take up to 2 days to track a video (see Supplementary Table A.16).

We present two main directions of improvement that can solve these limitations and can highly increase the tracking capabilities of future multi-animal tracking systems.

2.3.2.1 Reducing the number of crossings

Reducing the number of crossings (touches or occlusions) directly impacts on the length of the individual fragments and consequently it results in fewer but longer global fragments (see Section 2.2.1.3). In idtracker.ai this will very likely avoid the protocol 3 resulting in a faster tracking (see Figure 2.19). This would not only avoid protocol 3 but it could also result in more videos tracked with protocol 1, or with faster executions of protocol 2 (i.e. fewer accumulation steps, see Section 2.2.1.4).

Using adaptive thresholding techniques, like in the Tracktor tracking system (Sridhar et al., 2019), or automatic methods for determining the optimal segmentation parameters, like in the MAT tracking system (Itskovits et al., 2017), can provide a more accurate segmentation threrby reducing the number of touches. Some visual programming systems, like the UMATracker (Yamanaka

& Takeuchi, 2018) or Bonsai (Lopes et al., 2015), allow building custom video segmentation pipelines that users can adapt to different video conditions. Other alternatives consist of attempting to split the crossings by applying more aggressive segmentation thresholds on the crossings blobs, like TRex (Walter & Couzin, 2021), or by eroding it, like idTracker (Pérez-Escudero et al., 2014).

As explained by Robie et al. (2017), methods that incorporate the appearance of the animals can also help to reduce the number of touches. For example, Wang et al. (2017) and Zhiping & Cheng (2017) detect only the head of the fish which reduces the number of touches provoked by the tails of the fish. However, these methods require previous knowledge about the appearance of the animals, which would make idtracker.ai a species-specific tracking software.

Videos of highly dense groups in which the animals cannot be easily distinguished from their environment can require training specific animal detection algorithms that learn the appearance of the animals from annotated data (Bozek et al., 2021). This could also be a solution for videos of sparse groups of animals with heterogeneous backgrounds where classical background subtraction and thresholding methods would fail (see for example Laradji et al. (2020)). However, we believe that in the future animal tracking methods will be able to automatically segment animals from the background using self-supervised learning strategies (Pathak et al., 2017; Wichert, 2019).[16]

While reducing the number of touches certainly results in fewer but longer global fragments, animals in groups are still likely to occlude each other in ways that cannot be resolved using the previous methods. In the worst case scenario, there might even exist situations for which finding global fragments is just not possible. This can occur when not all animals are visible to the camera view, for example in videos where animals can go in and out of the video frame. In other cases, animals could hide under shelters or structures for long periods of time. Also, some species like honeybees, ants, or cockroaches live in very dense groups where multiple and frequent occlusions are likely to occur. In these cases, the current idtracker.ai training and identification strategy for the identification CNN cannot be used.

[16] `https://gitlab.com/polavieja_lab/flowseg`

2.3.2.2 Better CNN training strategies

Under the assumption that the video includes at least one global fragment, the current training and identification strategy of idtracker.ai can be considerably improved. Recently, Walter & Couzin (2021) have shown that this is the case with the TRex tracking software. Overall, the authors highly improved the training strategy making it much more time and memory efficient, which is a requirement for high-throughput data extraction. However, note that these improvements did not result in a higher identification performance with respect to idtracker.ai. In addition, the TRex strategy is also limited to videos that must show all animals separated from each other at some point in the video, like idtracker.ai.

Both idtracker.ai and TRex use a classification approach to learn the features that distinguish animals between themselves. This necessarily forces the systems to require at least a global fragment to act as a starting training dataset in which the identities of the animals are arbitrarily set for the first time. Most recent works on video-based identification are focused on using a metric learning approach (Bellet et al., 2013; Vidal et al., 2021). The two main methods for metric learning on images are the siamese networks (Bromley et al., 1994; Chopra et al., 2005) and the triplet loss (Schultz & Joachims, 2004; Hoffer & Ailon, 2015). Both methods try to learn a metric on the image features that is small for pairs of images of the same class and large pairs of images of different classes. However, these approaches require choosing wisely the triplets (or pairs) of images for an efficient training, a problem known as *triplet mining* (Xuan et al., 2020).

In very dense groups, Bozek et al. (2021) showed that CNNs can be trained to learn a metric representation of honeybees appearance that can then be used to distinguish them from each other and match their identities after touches or occlusions from frame to frame. They solved the triplet mining problem by sampling images of honeybees that were close in time and space, as they are potential sources of identity swaps. With this approach they manage to distinguish ≈ 1000 honeybees. However, the training of both the network used for animal detection and the network used for animal identification required manual annotation and validation of the trajectories at different stages. This is a very arduous task that systems like idtracker.ai try to avoid. Tracking highly dense groups of animals

where global fragments might be nonexistent is more challenging problem and some sort of manual annotation might be unavoidable.

While metric learning strategies are highly capable of distinguishing same or different animals (Schneider et al., 2020; Vidal et al., 2021), they do not learn to assign a unique label to each animal in the group. Typically a clustering algorithm over the extracted features must be used for this task. New strategies are capable of learning both a metric representation of the images and a probability of the image class at the same time (Hsu et al., 2019).

Preliminary results suggest that in systems like idtracker.ai, this approach should be flexible enough to allow training an idCNN in videos where there are a mix of multi-class labels (i.e. there are some global fragments in the video) and pair-wise labels (i.e. same/different pairs of images can be created); and also, when there are no multi-class labels at all but pair-wise labels can still be constructed (i.e. no global fragments at all but many individual fragments). Thus, multi-animal tracking algorithms can benefit from this type of training strategy to use information from the whole video to train CNNs without relying on global fragments, but taking advantage of them when they exists. Note that this would relax the second assumption under which we constructed idtracker.ai, i.e. that there should be at least one global fragment in the video.

Interestingly, Hsu et al. (2019) suggested that their training strategy can be used to perform unsupervised learning of the image classes with an upper bound on the number of classes in the dataset. This feature would weaken the first assumption under which we designed idtracker.ai, i.e. the fact that the number of animals in the video must be known in advance. In this scenario, the users would only need to provide an estimate of the maximum number of animals in the video. However, whether this type of unsupervised learning approach can be implemented in a multi-animal tracking system remains to be tested.

We showed how CNNs can be trained in a self-supervised manner to identify all animals in collectives of small and large animals of different species. Our results showed high identification accuracy for groups of up to 100 animals, however, it is clear that idtracker.ai does not solve the problem of tracking animal collectives in all possible scenarios. We suggest that the incorporation of new

methods developed in the rapidly evolving field of deep learning will result in more robust and capable multi-animal tracking algorithms that maintain the animals identities throughout the whole video under more challenging scenarios.

2.4 Materials and methods

2.4.1 Animal rearing and handling

All fish were raised at the Champalimaud Foundation Fish Platform, according to the housing and husbandry methods integrated in the zebrafish welfare program fully described in Martins et al. (2016). Animal handling and experimental procedures were approved by the Champalimaud Foundation Ethics Committee (CF internal reference 2015/007) and the Portuguese Direcção Geral Veterinária (DGAV reference 0421/000/000/2016) and were performed according to European Directive 2010/63/EU13.

2.4.2 Zebrafish videos experimental setup

For zebrafish videos we used the wild-type TU strain at 31 days post fertilization (dpf). Animals were kept in 8 L holding tanks at a density of 10 fish/L and a 14 h/10 h light/dark cycle in the main fish facility.

The experimental setup consisted of a main tank placed inside a box built with matte white acrylic walls (see Figure 2.20a). The lighting was based on infrared and RGB LED strips attached to the walls of the box. A custom made cylindrical retractable light diffuser made of plastic ensured homogeneous illumination in the central part of the main tank. A 20 MP monochrome camera (Emergent Vision HT-20000M) with a 28-mm lens (ZEISS Distagon T* 28-mm f/2.0 Lens with ZF .2) was positioned approximately 70 cm from the surface of the arena. To prevent reflections of the room ceiling, we used black fabric to cover the top of the box (see Figure 2.20b). We used this setup to record zebrafish in groups and in isolation (see Section 2.4.5). Videos of groups of 10, 60 and 100 fish were recorded in a custom-made one-piece circular tank of 72 cm diameter, designed in-house (see Supplementary Figure A.8). The tank was filled with fish system water to a depth of 2.5 cm, making a circumference of $\approx$ 65 cm of diameter in the surface. The circular tank was held in contact with the water of the main tank approximately 10 cm above a white background to improve the contrast between animals and background (see Figure 2.20c). A water-recirculating system equipped with a filter and a chiller ensured a constant water temperature of 28°C. For each video,

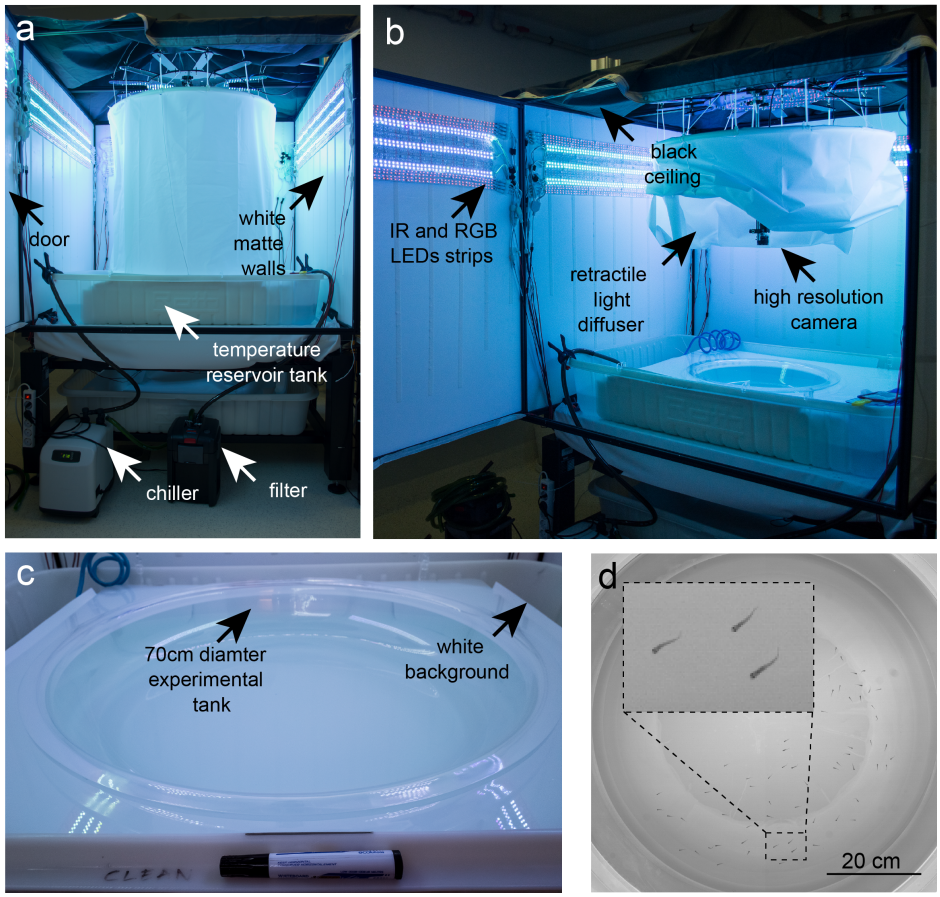

Figure 2.20. **Juvenile zebrafish video experimental setup. a.** Front view of the experimental setup used to recording zebrafish in groups and in isolation (see Section 2.4.5). **b.** Side view of the same setup with the light diffuser rolled up. **c.** Close-up view of the custom made circular tank used to record the groups of 10, 60 and 100 juvenile zebrafish. **d.** Sample frame from a video of 60 juvenile zebrafish.

a holding tank with the necessary number of fish was transported from the fish facility room to the experimental room. Then fish were carefully transferred to the experimental arena using a standard fish net appropriate for their age. Figure 2.20d shows a raw frame of a video recorded using this setup.

2.4.3 Fruit flies videos experimental setup

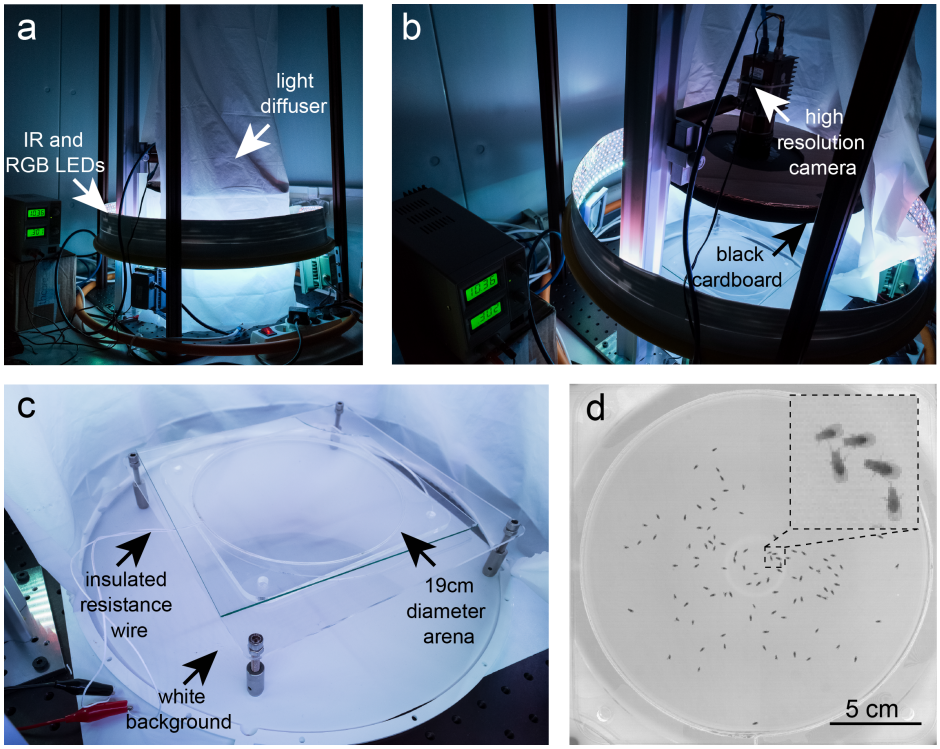

Figure 2.21. **Fruit flies videos experimental setup. a.** Exterior view of the setup used to recording flies in groups. **b.** Top view of the same setup with the diffuser rolled up. **c.** Close-up view of one of the two arenas used (arena 1). **d.** Sample frame of a 100 flies video.

For the fruit fly videos we used adults from the Canton S wild-type strain at 2-4 days post-eclosion. Animals were reared on a standard fly medium and kept on a 12-h light-dark cycle at 28°C.

The setup was placed in a dedicated experimental room with controlled humidity (60%) and temperature (25°C). RGB and IR LEDs placed on a ring around a cylindrical light diffuser guaranteed homogeneous light conditions in the central part of the setup (see Figure 2.21a). Videos were recorded with the same camera as in the zebrafish setup. Black cardboard around the camera reduced reflec-

tions of the ceiling on the glass covering the arena (see Figure 2.21b). We used two different arenas made of transparent acrylic, both built to prevent animals from walking on the walls. The arena 1 (diameter, 19 cm; height, 3 mm) had vertical walls that were heated with a white insulated resistance wire (Pelican Wire Company; 28 AWG solid (0.0126 inch), Nichrome 60, 4.4 Ω /ft, 0.015-inch white TFE tape). At 10 V, 0.3 A, the temperature at the walls reached 37°C (see Figure 2.21c). The arena 2 (diameter, 19 cm; height, 3.4 mm) had conical walls (angle of inclination, 11°; width of conical ring, 18 mm) (see Supplementary Figure A.9). The best results were obtained with standard top-view recording (see Supplementary Table A.13). Arena 1 was also used for bottom-view recordings. The top of both arenas was a sheet of glass covered with Sigmacote SL2 (Sigma-Aldrich), which prevented the flies from walking on the ceiling. A white plastic sheet was placed below the arena to increase the contrast between flies and background, at a distance of 5 cm below the arena to avoid shadows. Flies were placed in the arena either by anesthetizing them with CO_2 or ice, or by using a suction tube. We found the last method to have the least negative effect on the flies' health and to provide better activity levels.

2.4.4 Generation of identification images

During the crossing detection step of the algorithm (see Section 2.2.1.2), idtracker.ai generates an identification image for each blob. idtracker.ai uses these images to train the cdCNN (see Figure 2.4) and the idCNN (see Figure 2.6). Here we explain how these images are generated.

Remember that each segmented blob has the features described in Table 2.2 (see Section 2.2.1.1). Figure 2.22 illustrates the steps to generate an individual image for a single fish. For a given blob, we first expand the bounding box by 45 pixels on each side and extract an image from the video frame using the extended bounding box. Then, we expand the original contour of the blob using a kernel of size 3×3 and set to 0 (black color) all the pixels that do not belong to the contour, i.e. the pixels corresponding to the background. We compute the axis of maximum elongation by applying a PCA (Principal Component Analysis, (Wold et al., 1987)) on the set of pixel coordinates inside of the expanded contour. We rotate the image with the background removed so that the first principal

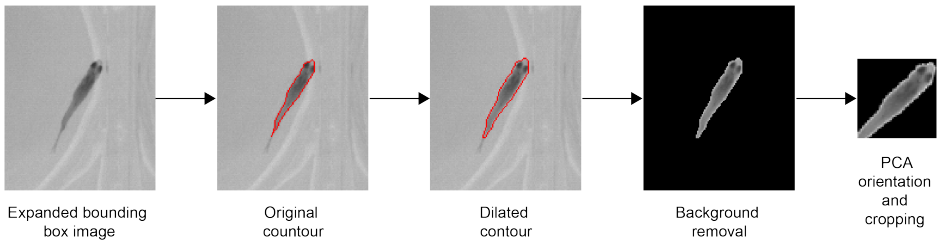

Figure 2.22. **Processing steps to generate the identification images.**

component vector is aligned to the 45° diagonal. Finally we crop the image to get a square image with diagonal the median estimated body length of all the individual blobs labeled by the first heuristic of the crossing detection processing step (see Section 2.2.1.2).

2.4.5 Image dataset of individual juvenile zebrafish

To test the identification capacity of the idCNN (see Section 2.2.3.1) and to test the cascade of training and identification protocols in synthethic videos (see Section 2.2.3.3) we generated a dataset of labeled images of juvenile zebrafish. Here we explain how we constructed this dataset.

We recorded 184 juvenile zebrafish (TU strain, 31 dpf) in separate chambers (60 mm diameter Petri dishes), using and earlier version of the experimental setup described in Figure 2.20. A holding grid with transparent acrylic walls allowed equal spacing between the chambers while granting visual access to the neighbouring dishes (see Figure 2.23a). To increase image contrast, we used a white acrylic floor placed 5 cm below the holding grid, which acted as a light diffuser to prevent shadows. We recorded four individuals at a time for 10 min. Along the outer borders we placed additional dishes containing fish to act as social stimuli (see Figure 2.23b). This made the recorded fish swim more than they would have in isolation. In total we recored 46 videos with 4 fish each. We extracted images of 52×52 pixels for each fish following the procedure described in Section 2.4.4 (see Figure 2.16). The dataset comprised a total of ≈ 3312000 uncompressed, grayscale, labeled images.

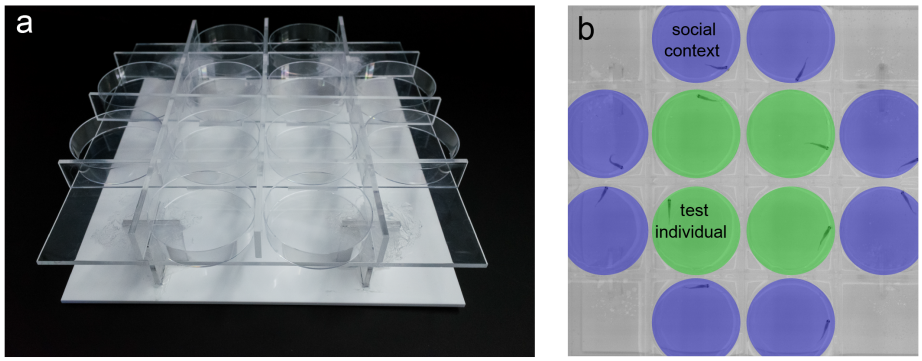

Figure 2.23. **Image dataset of individual juvenile zebrafish. (a)** Holding grid used to record 184 juvenile zebrafish (TU strain, 31dpf) in separated chambers (60 mm diameter Petri dishes). **(b)** Sample frame showing the individuals used to create the dataset and the inidividuals used as social context ($n = 46$ videos, ≈ 18000 per video).

2.4.6 Convolutional neural networks

2.4.6.1 Crossing detector convolutional neural network (cdCNN) training

See Supplementary Table A.8 for details of the cdCNN architecture. idtracker.ai initializes the weights of the convolutional and fully connected layers using Xavier initialisation (Glorot & Bengio, 2010). Biases are initialised to 0. Since there are typically more individual blobs than crossing blobs, idtracker.ai trains the cdCNN to minimize a multi class weighted cross entropy loss. idtracker.ai trains the cdCNN using the algorithm and hyperparameters in Kingma & Ba (2014) with mini batches of 50 images. Each image was standardised by subtracting its mean and dividing by its standard deviation. The learning rate is set to 0.001. To prevent the overfitting problem we used an early stopping strategy (Morgan & Bourlard, 1990) that checks the evolution of the loss on a validation dataset (10% of the total number of images for training). See Supplementary Table A.3 for the advanced parameters that can be modified to affect the training process.

2.4.6.2 Identification convolutional neural network (idCNN) training

See Table A.8 for details of the idCNN architecture. idtacker.ai initializes the weights of the convolutional and fully connected layers using Xavier initialisation (Glorot & Bengio, 2010). Biases are initialised to 0. In the first steps training and identification loop there are typically different numbers of images for the different individuals in the video, thus, idtracker.ai trains the idCNN to minimize a multi-class weighted cross entropy loss. idtracker.ai trains the idCNN using stochastic gradient descend with mini batches of 50 images. The learning rate is set at the initial value of 0.005. To avoid overfitting with use an early stopping strategy (Morgan & Bourlard, 1990) that checks the evolution of the loss on a validation dataset (10% of the total number of images for training). See Supplementary Tables A.4 and A.5 for the advanced parameters that can be modified to affect the training process.

During the cascade of training and identification protocols (see Section 2.2.1.4) idtracker.ai trains the idCNN with images from the identified global fragments with a limit of up to 3000 images per animal. This value is an advanced parameter (`MAXIMAL_IMAGES_PER_ANIMAL`) and it can be modified by the user (see Section 2.2.2.2 and Supplementary Table A.4). If in the first training and identification step (Protocol 1) an individual fragment has > 3000 images, idtracker.ai randomly samples 3000 to be used for training. At each training and identification step (Protocol 2), new individual fragments will be identified which images can be added to the bag of images for training. In long videos it is normal that the total number of images available for training for a given animal identity exceeds the threshold of 3000 in the first training and identification steps. In this case, idtracker.ai randomly select up to 1800 (60%) of the images already identified in previous steps and at least 1200 (40%) of the new images identified in the latest training and identification step. This values can be modified with the `RATIO_OLD` and `RATIO_NEW` advanced parameters (see Section 2.2.2.2 and Supplementary Tables A.4).

2.4.7 Generation of synthetic videos

To study the limits of the cascade of training and identification protocols (see Section 2.2.1.4) with respect to the number of images in the smallest individual fragment of the first global fragment, we generated synthetic videos using the labeled image dataset of juvenile zebrafish (see Section 2.4.5). Here we explain how we generated these synthetic videos.

We set the length of the synthetic videos to 10000 frames and generated videos for groups sizes of 10, 60 and 100 animals. We generated the videos by simulating lists of individual fragments for each individual in the group. Note that the images in the individual fragments were chosen from consecutive frames of the videos used to generate the labeled image dataset. To have a systematic way of generating the videos, we studied the distribution of the number of images in individual fragments of juvenile zebrafish videos and we observed that such distribution could be described using a gamma distribution (see Figure 2.24). Thus, the number of images in every individual fragment was drawn from a gamma distribution and the crossings fragments lasted three frames. We used the following parameters of the gamma distribution: $\theta = [2000, 1000, 500, 250, 100]$, $k = [0.5, 0.35, 0.25, 0.15, 0.05]$. For every combination of the θ, k parameters and for each group size we created three videos. In total we generated 225 synthetic videos.

2.4.8 Image quality manipulation conditions

To test the robustness of the idtracker.ai algorithm, we tracked videos with different manipulations of the image quality conditions (see Section 2.2.3.4). Here we explain how these manipulations were constructed.

2.4.8.1 Resolution

To test the performance of the system as a function of the number of pixels per animal, we artificially reduced the resolution of every frame by resizing it in width and height by a factor $\rho = [0.75, 0.5, 0.35, 0.25, 0.15]$. Moire effects were reduced by resampling using pixel area relation (Bradski, 2000).

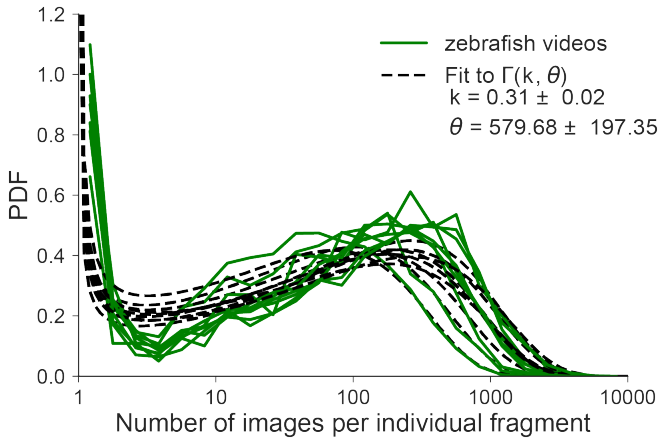

Figure 2.24. **Distribution of the number of images in the individual fragments of juvenile zebrafish.** The green lines represent the experimental distribution for each video. The black lines represent the fit to a gamma distribution.

2.4.8.2 Blurring

To test the performance of the system under different levels of image blurring, we artificially blurred every frame using a Gaussian kernel with a standard deviation $\sigma = [0.5, 1, 2, 3, 4, 5]$. The kernel size was computed automatically from the standard deviation (Bradski, 2000).

2.4.8.3 Compression

To test the robustness of the system under the effects of compression algorithms, we encoded the raw videos using the MPEG-4 and H.264 video codecs. Videos were encoded using the multimedia framework FFmpeg.[17] For the MPEG-4 video codec we used the maximum quality/file size level which corresponds to the value 1 in *qscale* parameter of the FFmpeg framework. For the H.264 video codec we used a constant rate factor of 0 which corresponds to a lossless compression.

[17] `https://ffmpeg.org/`

2.5 Author contributions and acknowledgements

This chapter is an adapted version of the manuscript Romero-Ferrero et al. (2019) from which Francisco Romero Ferrero is the first author.

Francisco Romero Ferrero (F.R.F), Mattia G. Bergomi (M.G.B.) and Gonzalo G. de Polavieja (G.G.d.P.) devised the project, designed the algorithms and analysed the data. G.G.d.P. supervised the project. F.R.F. and M.G.B. wrote the code with help from Francisco J.H. Heras. (F.H.). M.G.B. wrote the first version of the GUI. Ricardo Ribeiro (R.R.) coded the current version of the GUI with help from F.R.F. F.R.F. Robert Hinz (R.H.) built the zebrafish setup and performed the zebrafish experiments. R.H. built the fruit flies setup and performed the fruit flies experiments with help from F.R.F. F.R.F. managed the performance tests and wrote the code for analysis. F.R.F. wrote this chapter with comments from F.H. and G.G.d.P.

We thank Alfonso Pérez-Escudero and Andres Laan for discussions during the development of idtracker.ai. Isabel Campos, Catarina Certal, Sandra Martins and Joana Monteiro from the Champalimaud Fish and Fly platforms and for animal husbandry. Paulo Cariço from the Champalimaud Scientific Hardware platform for the assistance with the design and construction of the zebrafish and fruit flies arenas. Ricardo Riberio from the Scientific Software platform for the migration of idtracker.ai GUI to Pyforms. Tomás Cruz for assistance in the design of the fruit flies arena. We thank Andrew Bruce (Monash University, Melbourne, Australia) and N. Blüthgen (Technische Universität Darmstadt, Darmstadt, Germany) for videos of ants and, C. Ferreira (Champalimaud Foundation, Lisbon, Portugal) for videos of flies. We also thank all the idtracker.ai user that tested idtracker.ai and provided feedback for improvements.

This study was supported by Congento LISBOA-01-0145-FEDER-022170, three NVIDIA GPU grants (to M.G.B., F.H. and G.G.d .P.), the grant PTDC/NEU-SCC/0948/2014 (to G.G.d.P.) and the fellowship PD/BD/105946/2014 (to F.R.F.) from the Portuguese Fundação para a Ciência e a Tecnologia, and by the Champalimaud Foundation (G.G.d.P., M.G.B., F.H. and R.H.).

Chapter 3

Deep learning tools to study information aggregation in zebrafish collectives

3.1 Introduction

Few collective behaviour models try to predict the movement of each fish in the group, and most collective behaviour models only compare group level statistics (see Section 1.3). Heras et al. (2019) showed that deep attention networks can be both predictive and insightful about the behaviour of juvenile zebrafish in groups of $60-100$ individuals. However, it is unclear how well this models perform under different conditions and whether they can also provide insight about the animals behaviour in other experimental paradigms.

Also, many models of collective behaviour consider the information of all neighbours equally important (see Section 1.3). In groups of juvenile zebrafish Heras et al. (2019) found an information aggregation rule that is flexible and allows considering differently each neighbour depending on its behaviour. In particular, this rule suggests that zebrafish weigh higher a neighbour if it moves at faster speeds with respect to the rest of the collective. This flexibility permits potentially weighting higher the neighbours that bring new information to the group. However, whether this aggregation rule is truly useful in a real scenario where members of the group have different information about the environment has not been tested.

While the modelling approach followed by Heras et al. (2019) has advantages and poses new questions about how zebrafish interact, it has two related problems. First, in the experiments in Heras et al. (2019) all fish had the same information about the environment. In these conditions, the behaviour of all fish is highly correlated. The movement of a fish at each time point depends on the behaviour of its neighbours, and the behaviour of each of the neighbours depends at the same time on the behaviour of such fish. Second, it is well known that deep neural networks used in a supervised learning paradigm learn correlations from the given dataset used for training. In this particular case, the deep attention networks capture all correlations that might help in predicting the future turning side of the fish, including both causal correlation and spurious correlations. Because of this, it us unclear whether the rules inferred by training the deep attention networks in these experiments are real, i.e. not spurious correlations.

To answer these questions, we designed an experiment similar to the one in Strandburg-Peshkin et al. (2013) where animals have different information about the environment. We used this to provoke events where the behaviour of some animals was at least partially caused by the movement of others members of the group. In particular, we trained a group of fish to move fast towards a localized area in the arena that we illuminated with visible light. We called this group the *informed* fish. We also desensitized a different group of fish to be as indifferent as possible to the presentation of the same visual cue. When mixed together in the same arena, after the presentation of the visual cue, the informed fish moved fast towards it and the uniformed fish followed them. We then used deep attention networks to study how the uninformed fish aggregated information from their neighbours.

In Section 3.2.1, we summarize the response of the fish to the visual cue during the training and desensitization sessions preceding the experiment. In Section 3.2.2, we show how the presence of informed fish influence the response of the uninformed fish after the presentation of visual cue. In Section 3.2.3, we dissect in more detail the reactions of the informed and uninformed fish. In Section 3.2.4, we show results of the performance of different models on the prediction of the future turning side of the fish and discuss the impact of the different input variables of the models. In Section 3.2.5, we describe the functional form of the aggregation submodule of a deep attention network model from the previous section. In Sections 3.2.6 and 3.2.7, we use the same aggregation submodule to study how the uninformed fish aggregated information from their neighbours and to check whether the uninformed fish paid more attention to the informed neighbours that reacted before them. Finally, in Section 3.3 we summarize the main results and discuss some open-ended questions of the project.

3.2 Results

3.2.1 Training and desensitization of informed and uninformed fish

We designed an experimental procedure with two parts (see Section 3.4.3). The goal of the first part, the training and desensitization procedure (see Section 3.4.3.1, and Figure 3.1a for a diagram), was to have two groups of fish with different information about the environment, the *informed* and *uninformed* fish. In particular, we trained the informed fish to move fast towards a localized area of the experimental arena after shining visible light from an LED in such region (see Section 3.4.2 for details about the experimental setup). For the uninformed fish, we desensitized them so they would be as indifferent as possible to the presentation of the same light and they would not move towards it. Here, we present evidence to support that with our experimental design we were able to give informed and uninformed fish different information about such visual cue.

For simplicity, in the rest of the text we will refer to the training and desensitization procedure simply as the *training procedure*. Also, we will call each training or desensitization session (see Section 3.4.3.1) a *training session*. Note that the labels *informed* and *uninformed* already indicate whether it corresponded to a training or desensitization session.

We video-recorded the fish during 10 training sessions and extracted their trajectories using idtracker.ai (Romero-Ferrero et al., 2019) (see Section 3.4.4). In addition, in each video we manually annotated the area of the arena shone with the visible light LED (see Sectoin 3.4.5), and automatically detected the light onset for all trials in each training session (see Section 3.4.6). In the rest of the text we will refer to the center of the region of the arena that we shone with the visible light LED simply as the *light*. Figure 3.1b shows the trajectories for a group of informed (top) and uninformed (bottom) in an example trial of the last training session 1 second around the light onset.

From each fish trajectory, we computed its velocity vector, $\vec{v}$, and the distance from its position to the light, d_{light} (see Figure 3.1c for a diagram of these variables and Section 3.4.8). To have a sense of how fast animals moved with respect to their body size, we computed distances in body lengths (BL). One BL was ≈ 1.3

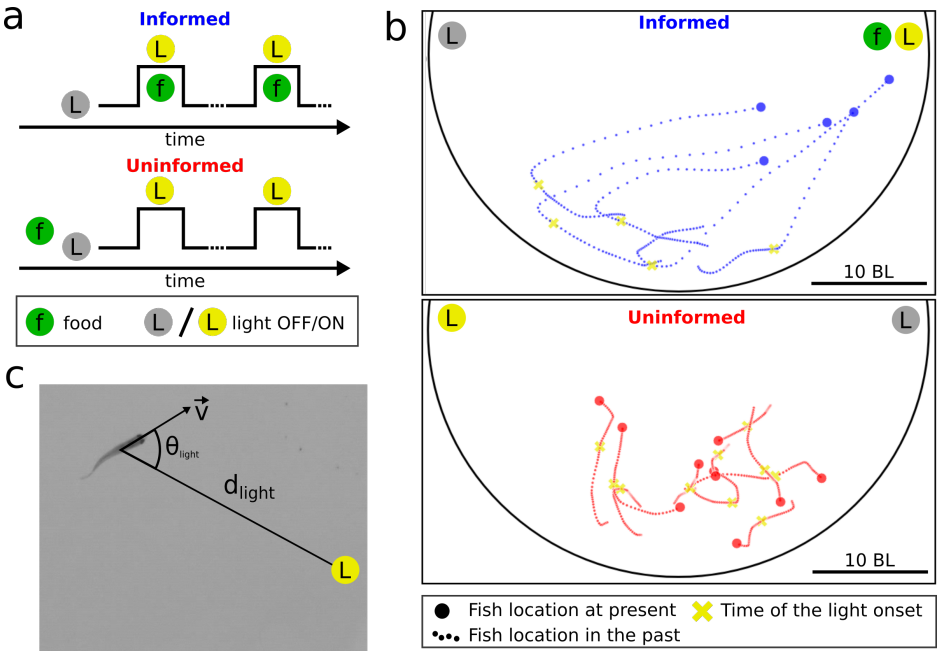

Figure 3.1. **Training procedure and trajectories of an example trial of the last training session. a.** Diagram of the training and desensitization procedures time line (see Section 3.4.3.1 for details). For the informed fish, in each trial we activated a visible LED light (yellow circle with an "L"), and paired it with the deposition of food (green circle with an "f") in the same area. For the uninformed fish, we spread food across the arena before the desensitization session. Then, in each trial we activated a visible LED light but without depositing any food. **b.** Diagram of the experimental arena with animal trajectories for a trial of the last training session. The black circular line indicates the border of the arena. We only show the part of the arena where the fish were present. The two lights were placed at opposite extremes of the arena. Top: trajectories of a group of 5 informed fish 1 s around the light onset. Bottom: trajectories of group of 10 uninformed fish 1 s around the light onset. **c.** Diagram of the variable used for analysis. $\vec{v}$ is the velocity vector of the fish. d_{light} is the distance between the location of the fish and the center of the region where the light was presented. θ_{light} is the angle between the velocity vector and the line that goes from the location of the fish to the location of the light.

cm, and the radius of the arena was $\approx$ 25 BL ($\approx$ 32.5 cm). Supplementary Figure A.11 shows the speed, $v = \|\vec{v}\|$, and distance to the light, d_{light}, around the light onset for an example trial of the last training session.

Here we only show data for the last training session. However, to support the idea that the following results are an effect the training procedure, we show in Supplementary Figure A.10 the mean of v and d_{light} across individuals and trials for each training session.

3.2.1.1 Informed fish got close to the light at high speeds.

First, we wondered whether in the last training session informed fish got closer to the light after the onset than before. Figure 3.2 (left) shows a clear decrease of the mean distance to the light for the informed fish $\approx$ 250 ms after the light onset. In fact, in the interval of 3 s to 4 s after the onset informed fish were or average closer to the light than before the onset ($\overline{D(\langle d_{\text{light}}\rangle)} = -20.24$ BL, $p < 0.001$, $N = 900$; see Figure 3.2 right; see Section 3.4.10.1 for details about the statistics). In particular, the mean distance to the light of the informed fish in such interval was 5 ± 7 BL (mean $\pm$ SD) BL and the median was 3 BL.

Considering the fact that the feeder devices deposited the food in a circular area of $\approx$ 4 BL of radius, this result suggests that informed fish learned to move towards the light after the onset.

Next, we wondered whether informed fish changed their speed in order to move towards the light, or kept similar speeds to the ones they had before the onset. Figure 3.3 (left) shows that the mean speed of the informed fish increased strongly $\approx$ 250 ms after the light onset, reaching a peak at $\approx$ 500 ms and then decreased. After $\approx$ 2 s from the onset the mean speed took values below the baseline value that it had before the onset. In the first 2 s after the onset, the mean speed of the informed fish was higher than before the onset ($\overline{D(\langle v\rangle)} = 3.35$ BL/s, $p < 0.001$, $N = 900$; see Figure 3.3 right). In the interval of 2 s to 4 s after the onset, the mean speed of the informed fish was smaller than before the onset ($\overline{D(\langle v\rangle)} = -1.91$ BL/s, $p < 0.001$, $N = 900$).

These results suggest that, after the light onset, informed fish first moved fast towards the light and then decelerated when getting close to it.

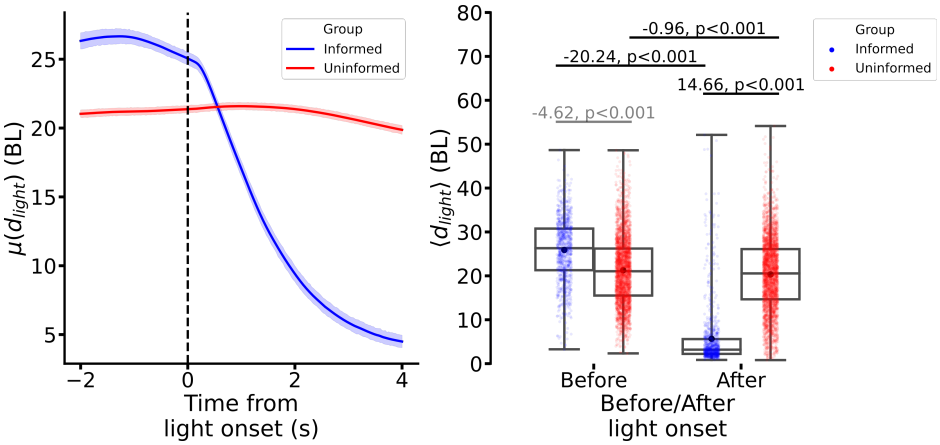

Figure 3.2. **Distance to the light of informed and informed fish in the last training session. Left.** Mean distance to the light around the light onset. The mean is computed across all fish and trials of the last training session. The shaded area around the mean represents the 95% confidence interval of the mean. This information is the same for all the future similar figures unless indicated otherwise. **Right.** Mean distance to the light in the intervals $[-1, 0)$ s (before) and $[3, 4)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session. The whiskers of the boxes represent the minimum and maximum values of the data points. The boxes represent the 25 and 75 percentiles. The line inside the box represents the median. The black dot represent the mean. The numbers on top of the horizontal line indicate the statistic and p-value for the corresponding permutation test (see Section 3.4.10). Higher opacity values indicate the tests that we refer to in the text. This information is the same for all the future similar figures unless indicated otherwise.

3.2.1.2 Uninformed fish stayed away from the light

Regarding the uninformed fish, Figure 3.2 (left) shows that the mean distance to the light of the uninformed fish was ≈ 21 BL both before and after the onset. However, we can observe a slow decreasing trend ≈ 2 s after the onset. In fact, we found that the mean distance to the light in the interval of 3 s to 4 s after the onset was statistically smaller than the one before the onset ($\overline{D(\langle d_{\text{light}} \rangle)} = -0.96$ BL, $p < 0.001$, $N = 3090$; see Figure 3.2 right). Considering the fact that the radius of the arena was ≈ 20 BL, this result suggests that, on average, the uninformed fish stayed away from the light after the onset.

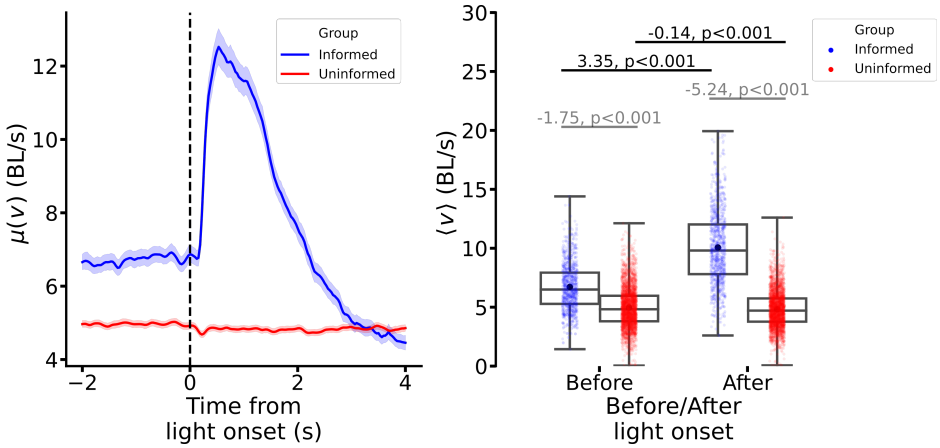

Figure 3.3. **Speed of informed and uninformed fish in the last training session. Left.** Mean speed around the light onset. The mean is computed across all fish and trials of the last training session. **Right.** Mean speed in the intervals $[-2, 0)$ s (before) and $[0, 2)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

We also wondered whether the activation of the light provoked any change in the speed of the uninformed fish. We observed a statistically significant decrease on the mean speed during the first 2 s after the onset ($\overline{D(\langle v \rangle)} = -0.14$ BL/s, $p < 0.001$, $N = 3090$; see Figure 3.3 right). Together with the small decrease of mean distance to the light, this result suggests that uninformed fish were not completely indifferent to the light. We discuss in Section 3.3.2.2 the possible cause of this decrease of speed and distance to the light.

It is worth noticing that the change of the mean speed of the uninformed fish was ≈ 20 times smaller and of the opposite sign than the one of the informed fish in the same interval. In addition, after the onset informed fish got closer to the light than the uninformed fish ($\overline{D(\langle d_{\text{light}} \rangle)} = 14.64$ BL, $p < 0.001$ $N_{\text{informed}} = 900$, $N_{\text{uninformed}} = 3090$; see Figure 3.2 right; see Section 3.4.10.1 for details about the statistics).

Altogether, these results support the idea that in the last training session informed and uninformed fish had a different response to the light after the onset.

Informed fish moved fast towards light and got close to it. Conversely, while the uninformed fish were not completely indifferent to the the light, they stayed generally away from it.

3.2.2 Experiment with mixed groups of informed and uninformed fish

The second part of our experimental procedure aimed to produce a different response by the uninformed fish after the light onset due to the presence of informed fish in the same experimental arena. Since our goal was to study how the uninformed fish aggregated information from their neighbours, we call this part the *information aggregation experiment* (see Section 3.4.3.2 for details, and Figure 3.4a for a diagram of the experiment time line). In the remainder we refer to the information aggregation experiment simply as the *experiment*.

The experiment had two parts, one where the uninformed fish swam alone, and a second part where they swam together with the informed fish. In the second part, we sequentially activated the light in a series of trials. As in the training procedure, we video recorded each experiment, annotated the location of the light, and detected the onset for each trial (see Figure 3.4a). Using idtracker.ai, we extracted the trajectories of all individuals in both parts of the experiment. With the fish images and identification networks resulting from tracking the two experiment parts with idtracker .ai (see Section 2.2.1.4), we automatically and unequivocally determined the experimental group of each fish, informed or uninformed, for 20 out of 22 experiments (see Section 3.4.7 for details). Figure 3.4b shows the trajectories of a mixed group for 2 s around the light onset for an example trial of an experiment. See Supplementary Figure A.12 for the fish distance to the light and speed in an example trial of an experiment.

3.2.2.1 Informed fish moved fast towards the light when mixed with uninformed fish

Here we show evidence to support the idea that in the experiment informed fish had a similar response to the light activation than in the last training session.

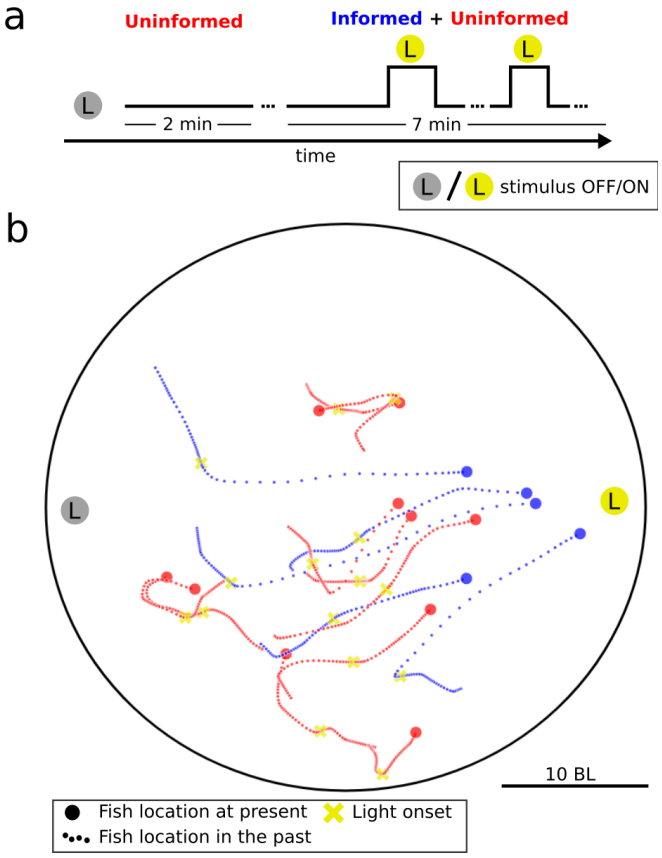

Figure 3.4. **Information aggregation experiment and example trajectories in a trial. Left.** Diagram of the experiment time line (see Section 3.4.3.2 for details). First, the uninformed fish were allowed to swim freely in the arena for 2 min. During this period we neither activate the light nor we gave them any food. Then we introduced the informed fish in the arena. We next proceeded with a sequence of trials for 7 min. In each trial we activated a visible LED light in an extreme of the arena but note that we did not deposit any food. **Right.** Informed and uninformed fish trajectories in a trial of an experiment 2 s around the light onset.

First, we checked whether after the light onset informed fish got as close to the light as they did in the last training session. In the experiment, the mean distance to the light in the interval of 3 s to 4 s after the onset was smaller than before

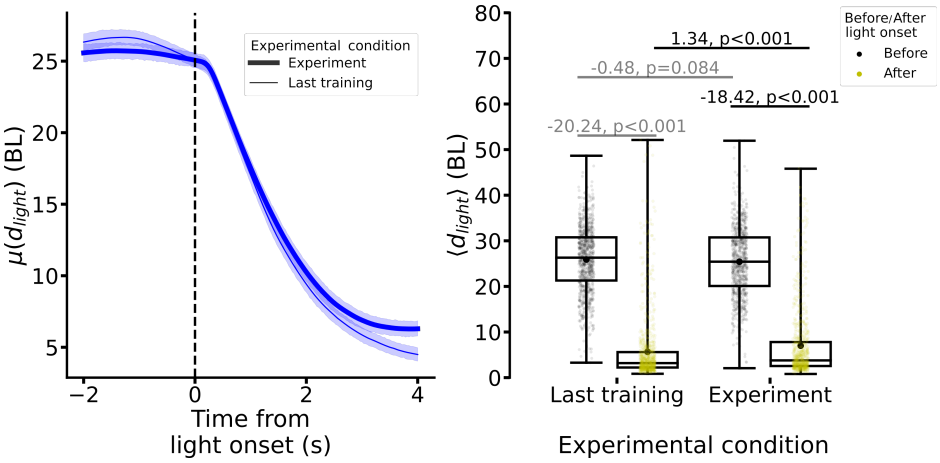

Figure 3.5. **Distance to the light of the informed fish in last training session and in the experiment. Left.** Mean distance to the light around the light onset. The mean is computed across all fish and trials of the last training session and of the experiment respectively. **Right.** Mean distance to the light in the intervals $[-1, 0)$ s (before) and $[3, 4)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session or in the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

the onset ($\overline{D(\langle d_{stim} \rangle)} = -18.42$ BL, $p < 0.001$, $N = 795$; see Figure 3.5 right). However, after the onset the mean distance to the light was higher than in last training session ($\overline{D(\langle d_{\text{light}} \rangle)} = 1.34$ BL, $p < 0.001$, $N_{\text{training}} = 900$, $N_{\text{experiment}} = 795$; see Figure 3.5 right). A possible cause for this difference is the fact that in the experiment we did not deposit food in the area where we shone the visible light, so informed fish might have spent less time around it. In any case, the mean distance to the light in such interval after the onset was 7 ± 7 BL (mean $\pm$ SD) and the median was 4 BL. This suggests that informed fish got close to the light after the onset when they were mixed with uninformed fish.

We also checked whether in the experiment the informed fish also moved fast towards the light. We did not find a significant difference of the mean speed in the first 2 s after the onset when comparing the last training session and the experiment ($\overline{D(\langle v \rangle)} = -0.16$ BL/s, $p = 0.161$, $N_{\text{last training}} = 900$, $N_{\text{experiment}} =$

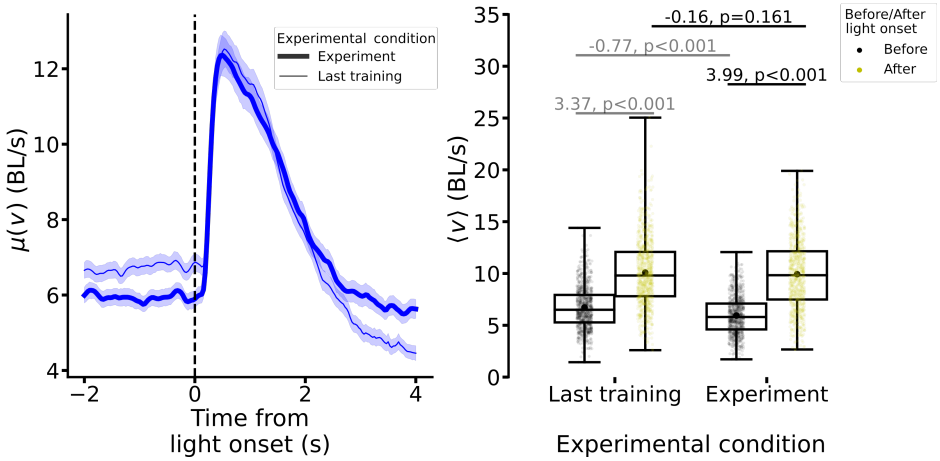

Figure 3.6. **Speed of informed fish in the last training session and in the experiment. Left.** Mean speed around the light onset. The mean is computed across all fish and trials of the last training session and of the experiment respectively. **Right.** Mean speed in the intervals $[-2, 0)$ s (before) and $[0, 2)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session or in the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

795; see Figure 3.6 right). This suggests that on average informed fish moved as fast towards the light as they did in the last training session.

All these results support the idea that on average informed fish had a similar response to the activation of the light in the experiment than in the last training session.

3.2.2.2 Uninformed fish got closer to the light when mixed with informed fish

Next, we show evidence to support the idea that uninformed fish got closer to the the light after the onset in the experiment than in the last training session.

Figure 3.7 (right) shows the mean distance to the stimulus of the uninformed fish in the last training session and in the experiment. In the experiment, the mean distance to the light of the uninformed fish in the interval of 3 s to 4 s after the onset was smaller than before the onset ($D(\overline{\langle d_{\text{light}} \rangle}) = -7.05$ BL,

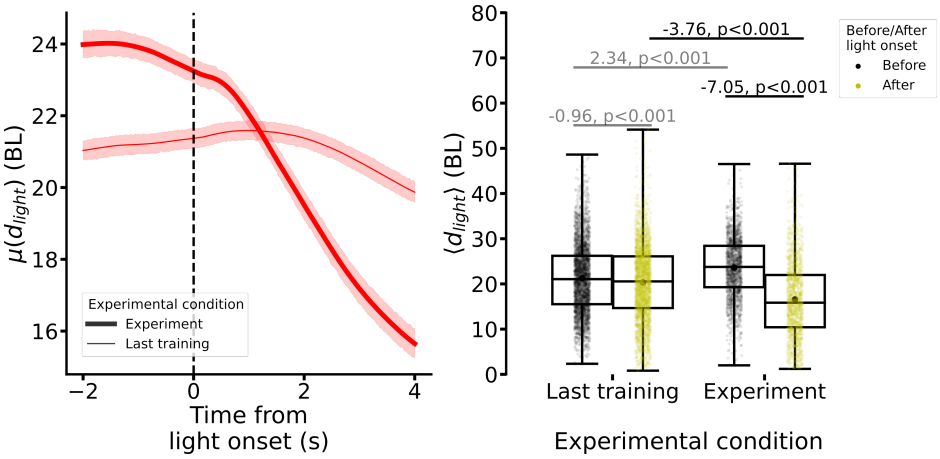

Figure 3.7. **Distance to the light of uninformed fish in the last training session and in the experiment. Left.** Mean distance to the light around the light onset. The mean is computed across all fish and trials of the last training session and of the experiment respectively. **Right.** Mean distance to the light in the intervals $[-1, 0)$ s (before) and $[3, 4)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session or in the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

$p < 0.001$, $N = 1590$; see Figure 3.7 right). In addition, after the onset the mean distance to the light was smaller in the experiment than in the last training session ($D(\overline{\langle d_{\text{light}} \rangle}) = -3.76$ BL, $p < 0.001$, $N_{\text{last training}} = 3090$, $N_{\text{experiment}} = 1590$; see Figure 3.7 right). This suggests that uninformed fish got closer to the light after the onset when swimming with informed fish.

Note that the progress of the mean distance to the light in the experiment (see Figure 3.7 left, thick line) shows that, on average, uninformed fish were getting closer to the light already before the onset. Conversely, in the last training session the mean distance to the light before the onset suggests that uninformed fish were moving away from it (see Figure 3.7 left, thin line). This could suggest that the mean distance to the light after the onset was smaller in the experiment because uninformed fish had the light in front and were already moving in that direction. In Section 3.3.2.3 we discuss how a bias during the experiment can have caused this effect before the light onset.

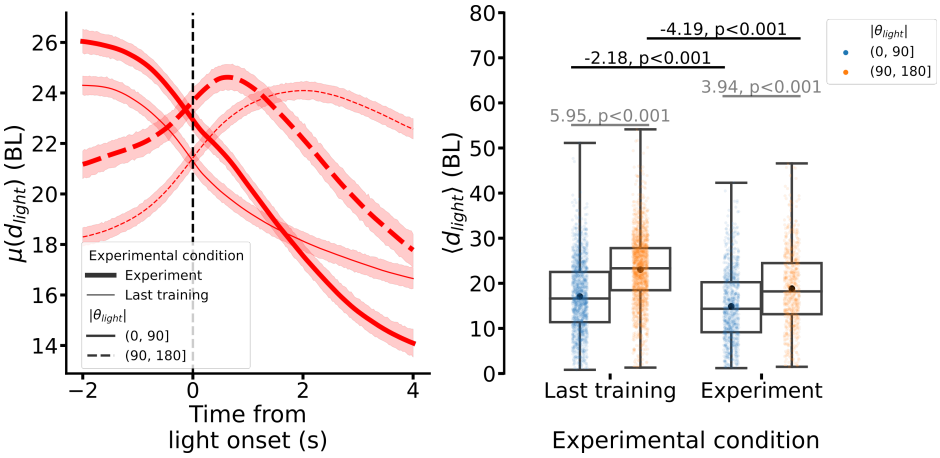

Figure 3.8. **Distance to the light of uninformed fish conditioned to the orientation towards the light. Left.** Mean distance to the light around the light onset for two different ranges of the angle θ_{light} in the last training session (thin line) and in the experiment (think line). The mean is computed across all fish and trials of the last training session and of the experiment respectively. **Right.** Mean distance to the light in the $[3, 4)$ s after the light onset for two different ranges of the angles θ_{light}. Each dot represent the mean for a fish in a trial of the last training session or of the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

Here, we checked whether the distance to the light after the onset was smaller in the experiment than in the last training session regardless of the orientation towards the light prior to the onset. We computed the orientation of the animals towards the light as $|\theta_{\text{light}}|$ (see Figure 3.1c). We looked at the mean distance to the light for uninformed fish that had the light in front ($|\theta_{\text{light}}| < 90$) or behind ($|\theta_{\text{light}}| \geq 90$) at the onset time (see Figure 3.8 right). The mean distance to the light in the interval of 3 s to 4 s after the onset was smaller in the experiment than in the last training session both for fish that had the light in front ($D(\overline{\langle d_{\text{light}} \rangle}) = -2.18$ BL, $p < 0.001$ $N^{\text{front}}_{\text{last training}} = 1408$, $N^{\text{front}}_{\text{experiment}} = 918$) and fish that had the light behind ($D(\overline{\langle d_{\text{light}} \rangle}) = -4.19$ BL, $p < 0.001$, $N^{\text{behind}}_{\text{last training}} = 1682$, $N^{\text{behind}}_{\text{experiment}} = 672$) at the onset time (see Figure 3.8 right). These results suggest that regardless of the orientation towards the light at the onset time, uninformed fish got generally closer to it in the experiment than in the last training session.

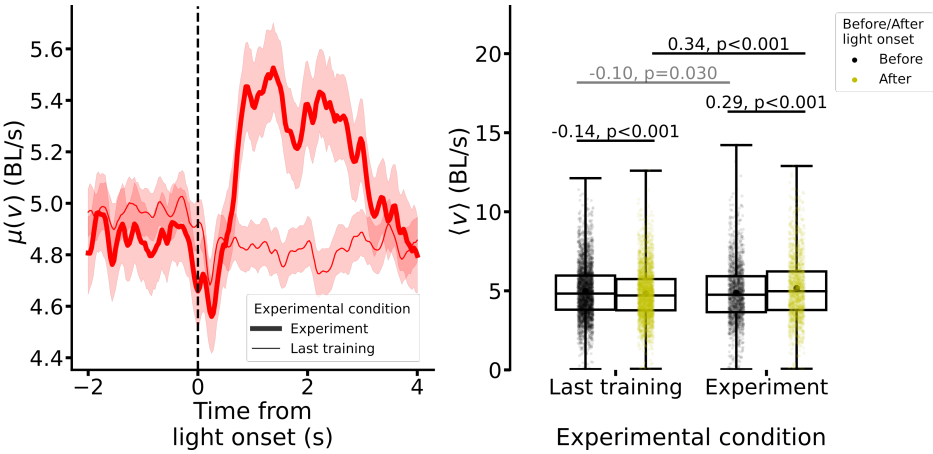

Figure 3.9. **Speed of uninformed fish in the last training session and in the experiment. Left.** Mean speed around the light onset for the last training session (thin line) and for the experiment (thick line). The mean is computed across all fish and trials of the last training session and of the experiment respectively. **Right.** Mean speed in the intervals $[-2, 0)$ s (before) and $[0, 2)$ s (after). Each dot represents the mean for a fish in a given trial of the last training session or of the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

All considered, these results support the idea that on average uninformed fish got closer to the light when mixed with informed fish than when swimming only with uninformed fish.

3.2.2.3 Uninformed fish moved faster after the light onset when mixed with informed fish

Since the previous results suggest that in the experiment uninformed fish copied the behaviour of the informed fish getting closer to the light, we wondered whether uninformed fish also changed their speed after the onset as the informed fish did.

Figure 3.9 (left) shows higher values of the mean speed of the uninformed fish ≈ 500 ms after the onset. In the experiment, the mean speed of the uninformed fish in the first 2 s after the onset was higher than 2 s before ($D(\overline{\langle v \rangle}) = 0.29$ BL/s, $p < 0.001$, $N = 1590$; see Figure 3.9 right). Note that the small change of speed

that we found in the last training session had the opposite sign (see Section 3.2.1 and Figure 3.3 right). In fact, the mean speed of the uninformed fish in the first 2 s after the onset was higher in the experiment than in the last training session ($D(\overline{\langle v \rangle}) = 0.34$ BL/s, $p < 0.001$, $N_{\text{last training}} = 3090$, $N_{\text{experiment}} = 1590$; see Figure 3.9 right).

These results suggest that some uninformed fish increased slightly their speed after the onset when swimming together with informed fish.

All the results so far suggest that the experiment worked as expected. A group of only uninformed fish did not move towards the light when we turned it on, but they did so in the presence of informed fish. This necessarily implies that uninformed fish followed the informed fish to some extent.

3.2.3 Informed and uninformed fish reactions after the light onset

The response of the uninformed fish after the light onset did not seem as strong as the one of the informed fish. However, it is important to note that the observed differences are on the mean of the variables in a interval of time. Thus, these differences could come from a decrease in the magnitude of the response of the uninformed fish, or from a higher variability in the latency of the response after the light onset, or both. Here we show results that clarify the nature such differences.

First, we took a closer look at the speed and distance to the light in some trials of the experiment. Supplementary Figure A.12 shows the speed and distance to the light in an example trial of the experiment. The majority of the informed fish moved fast towards the light shortly after the onset by increasing their speed and getting close to it. Conversely, fewer uninformed fish seemed to change their speed after the onset and they did so after the informed fish. This suggests that the smaller change of the mean speed that we observed in Section 3.2.2.3 for the uninformed fish (see Figure 3.9) could be an effect of them responding with a lower probability and with varying delays after the informed fish movement towards the light. To understand whether this was the case we decided to automatically detect the times when the informed and uninformed fish behaviour changed after the

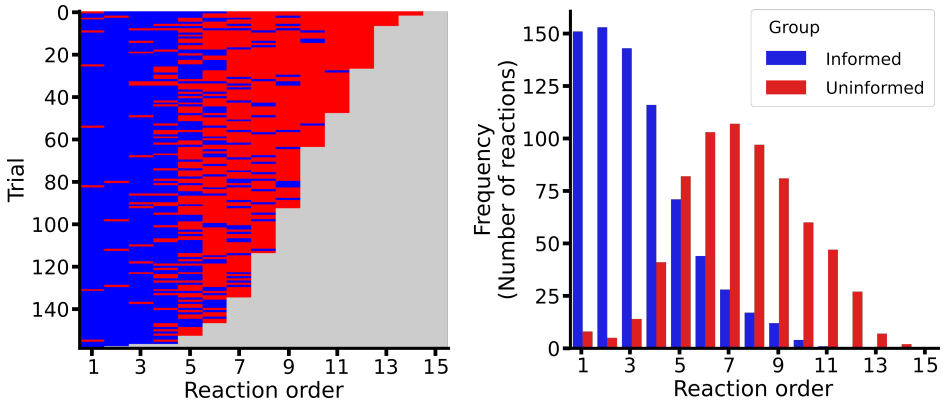

Figure 3.10. **Sequence of reactions after the light onset for all trials in the experiments.** Only reactions detected in the first 4 s after the light onset are considered. **Left.** Raster plot representing the reaction order of the informed and uninformed fish in each trial of all experiments. **Right.** Bar plot indicating the frequency of the reaction order for informed and uninformed fish.

stimulus onset. We called the moment when a fish stated showing a different behaviour than the one it had before the onset a *reaction*.

We trained a recurrent neural network to estimate the reaction probability of each fish and computed the reaction time, τ, with a threshold on such probability (see Section 3.4.9 for details). Note that we expected most of the informed fish reactions to be direct responses to the light onset, and the uninformed fish reactions to be indirect responses due to the influence of the informed fish. We assigned a *reaction order* to each fish in each trial by ordering the reaction times, τ, of all the fish that reacted in such trial. Thus, in each trial, the fish with the smallest reaction time has a reaction order of 1, and the one with the highest reaction time will have the highest reaction order. Note that fish that did not react do not have a reaction order assigned.

3.2.3.1 Uninformed fish reacted with a lower probability and had more variable reaction times

First, we wondered whether in each trial informed fish reacted more often than uninformed fish. Figure 3.10 (left) summarizes the sequence of reactions for

each trial in the experiments. We computed the fraction of fish that reacted in each trial for the informed and uninformed fish as $\phi_{\text{group}} = n_{\text{reacted}}/n_{\text{group}}$, where n_{reacted} is the number of fish that reacted within the first 4 s after the onset and n_{group} is the number of fish of each experimental group ($n_{\text{informed}} = 5$, $n_{\text{uninformed}} = 10$). The mean fraction of informed fish that reacted in each trial was 0.9±0.2 (mean ± SD). The one of the uninformed fish was 0.3±0.2. We found ϕ_{informed} to be higher than $\phi_{\text{uninformed}}$ ($\overline{D(\phi)} = 0.63$, $p < 0.001$, $N_{\text{trials}} = 159$). This suggests that on average 4.5 out of 5 informed fish and 3 out of 10 uninformed fish reacted in each trial.

Next, we checked whether uninformed fish reacted after the informed fish. Figure 3.10 (right) shows the frequency of the reaction order for the informed and uninformed fish considering all trials in our experiments. The number of times that a fish reacted in the first, second, third or fourth positions of the sequence was higher for informed than for uninformed fish. Conversely, the number of times that fish reacted after the fifth position in the sequence of reactions was higher for uninformed than for informed fish. In particular, the median of the reaction order of the first uninformed fish that reacted across trials was 5. This means that generally around 4 informed fish reacted before the first uninformed fish reacted.

We also wondered how much earlier the reaction of the informed fish was compared to the one of the uninformed fish. Figure 3.11 (left) shows the distribution of the reaction times, τ, for informed and uninformed fish. We can observe that most informed fish reacted shortly after the onset, while uninformed fish reacted after a longer and varying delay. In particular, $\approx$ 90% (663) of the informed fish that reacted did it during the first 2 s while only $\approx$ 53% (361) of the uninformed fish that reacted did so in the same interval. The mean reaction time of the informed fish was 0.82 ± 0.79 s (mean ± SD) (median(τ_{informed}) = 0.5 s), and the one of the uninformed fish was 1.98 ± 0.96 s (median($\tau_{\text{uninformed}}$) = 1.9 s). We found the uninformed fish to have longer reaction times ($D(\overline{\tau}) = -1.17$, $p < 0.001$, $N_{\text{informed}} = 740$, $N_{\text{uninformed}} = 681$). To get a sense of how fast the earliest informed and uninformed fish reacted after the onset, we looked at the reaction time of the first informed and uninformed fish that reacted in each trial, τ_{first}. The mean reaction time of the first informed fish was 0.3 ± 0.16 s (

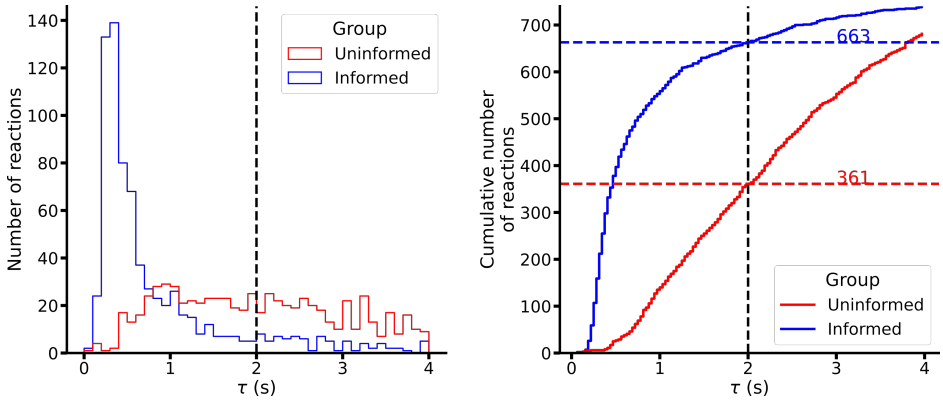

Figure 3.11. **Informed and uninformed fish reaction times (τ) distribution.** Only reactions detected in the first 4 s after the light onset are considered. The vertical dashed line indicates the threshold selected to remove second order effects in the cascade of reactions. **Left.** Histogram showing the frequency of reaction times for informed and uninformed fish. **Right.** Cumulative frequency of reaction times for informed and uninformed fish. The horizontal dashed lines with a number indicate the number of reactions that occurred within the first 2 s after the light onset, i.e. $\tau < 2$ s.

median($\tau_{\text{first informed}}$) = 0.28 s). The one of the first uninformed was 1.2 ± 0.7 s (median($\tau_{\text{first uninformed}}$) = 1.03 s), which was longer than the one of the informed fish ($D(\overline{\tau_{\text{first}}}) = 0.9$, $p < 0.001$, $N_{\text{informed}} = 159$, $N_{\text{uninformed}} = 154$).

Altogether, these results support the idea that the uninformed fish that reacted after the onset did so with a lower probability and with longer and more variable latencies than the informed fish.

Note that fish with shorter reaction times are more likely to have reacted as a direct response to the light, if they were informed, or to the informed fish that reacted before, if they were uninformed. However, fish with longer reaction times could have reacted to other uninformed fish that reacted before them. We are interested in understanding the first instants of the cascade of reactions. Thus, for the rest of the analysis, from all the fish that reacted we will only consider those that did so during the first 2 s after the onset (see Figure 3.11b). Also, we will consider all the fish that did not react at all during the first 4 s after the onset as a control for comparisons.

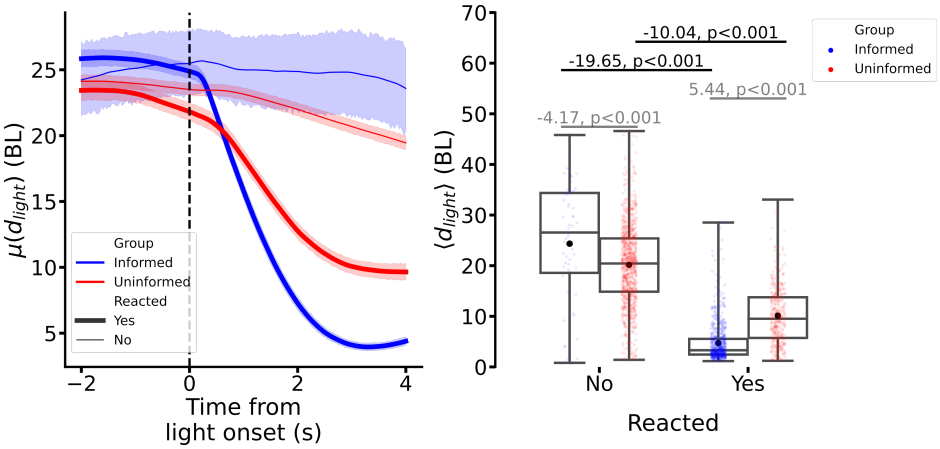

Figure 3.12. **Mean distance to the light for informed and informed fish depending on whether they reacted or not after the light onset.** Only fish that reacted during the first 2 s or fish that did not react at all in the first 4 s after the light onset are considered. **Left.** Mean distance to the light around the onset in the experiment for fish that reacted (thick line) or did not react (thin line) after the light onset. The mean is computed across all trials and fish for each condition. The shaded area around the mean represents the 95% confidence interval of the mean. **Right.** Mean distance to the light in the interval $[3, 4)$ s after the light onset. Each dot represents the mean for a fish in a given trial of the experiment depending on whether it reacted (Yes) or it did not react (No) after the light onset. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

3.2.3.2 Fish that reacted after the light onset got closer to it and with higher speeds

We wondered whether there were differences in the behaviour of the fish that reacted after the light activation and of the ones that did not. For this, we looked again at the distance to the light and the speed of the informed and uninformed fish.

First, we checked whether informed and uninformed fish that reacted after we turned the light on approached it. Figure 3.12 (left) shows the mean distance to the light around the onset time for the informed and uninformed fish depending on whether they reacted after the light onset. For the informed fish that reacted,

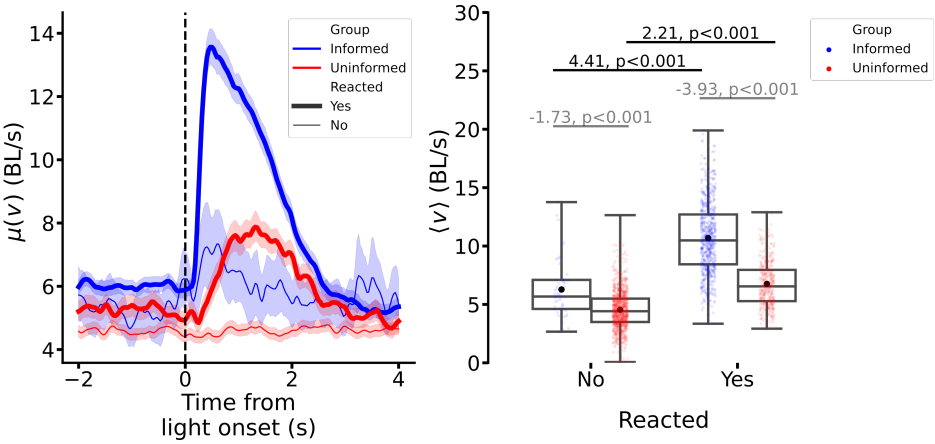

Figure 3.13. **Speed of informed and informed fish depending on whether or not they reacted after the light onset.** Only fish that reacted during the first 2 s or fish that did not react at all in the first 4 s after the light onset are considered. **Left.** Mean speed around the light onset in the experiment for fish that reacted (thick line) or did not react (thin line) after the light onset. **Right.** Mean speed in the interval $[0, 2)$ s after the light onset. Each dot represents the mean for a fish in a given trial of the experiment depending on whether it reacted (Yes) or it did not react (No) after the light onset. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

the mean distance to the light in an interval of 3 s to 4 s after the onset was 5 ± 4 BL (median$(d_{\text{light}}) = 3$ BL), while the one of the informed fish that did not react was 24 ± 12 BL (median$(d_{\text{light}}) = 26$ BL). The difference of the means was statistically significant ($D(\overline{\langle d_{\text{light}} \rangle}) = -19.65$ BL, $p < 0.001$, $N_{\text{no reacted}} = 55$, $N_{\text{reacted}} = 663$; see Figure 3.12 right). For the uninformed fish that reacted, the mean distance to the light in an interval of 3 s to 4 s after the onset was 10 ± 6 BL (median$(d_{\text{light}}) = 10$ BL), while the one of the uninformed fish that did not reacted at all was 20 ± 8 BL (median$(d_{\text{light}}) = 20$ BL). The difference of the means was statistically significant ($D(\overline{\langle d_{\text{light}} \rangle}) = -10.04$ BL, $p < 0.001$, $N_{\text{no reacted}} = 909$, $N_{\text{reacted}} = 361$; see Figure 3.12 right).

These results suggest that informed and uninformed fish that reacted got on average closer to the light than fish that did not react.

Next, we checked whether informed and uninformed fish that reacted moved faster than the ones that did not react. Figure 3.13 (left) shows the mean speed around the onset for the informed and uninformed fish depending on whether they reacted after the light onset. For the informed fish that reacted, the mean speed in the first 2 s was 11 ± 3 BL/s. For the ones that did not react was it 7 ± 2 BL/s. The difference between the means was statistically significant ($D(\overline{\langle v \rangle}) = 4.41$ BL/s, $p < 0.001$; $N_{\text{no reacted}} = 55$, $N_{\text{reacted}} = 663$; see Figure 3.13 right).

In the same interval, the mean speed of the uninformed fish that did not react was 5 ± 2 BL/s and the one for the ones that reacted was 7 ± 2 BL/s. The difference between the means was also statistically significant in this case ($D(\overline{\langle v \rangle}) = 2.21$ BL/s, $p < 0.001$; $N_{\text{no reacted}} = 909$, $N_{\text{reacted}} = 361$; see Figure 3.13 right).

These results suggest that, in general, informed and uninformed fish that reacted after the light onset moved faster after the onset than the ones that did not react.

Altogether, these results support the idea that fish that reacted during the first 2 s after the onset displayed a different behaviour than the fish that did not react at all.

3.2.3.3 Informed fish moved faster towards the light than uninformed fish after reacting

Figure 3.12 (left) and Figure 3.13 (left) suggest that there could be differences in the behaviour of the informed and uninformed fish that reacted after the light onset. However, part of these difference could be due to the fact that uninformed fish have more variable reaction times than the informed fish (see Section 3.2.3). To check to what extent these differences would be a result of averaging, we looked at the distance to the light and the speed of the fish around the reaction time.

Figure 3.14 (left) shows that the mean distance to the light was higher before the reaction for the informed fish than for the uninformed ones. However, in an interval of 1 s to 2 s after the reaction, the mean distance to the light of the informed fish was 7 ± 5 BL ($\text{median}(d_{\text{light}}) = 6$ BL), while the one of the uninformed fish was 11 ± 5 BL ($\text{median}(d_{\text{light}}) = 11$ BL), higher than of the

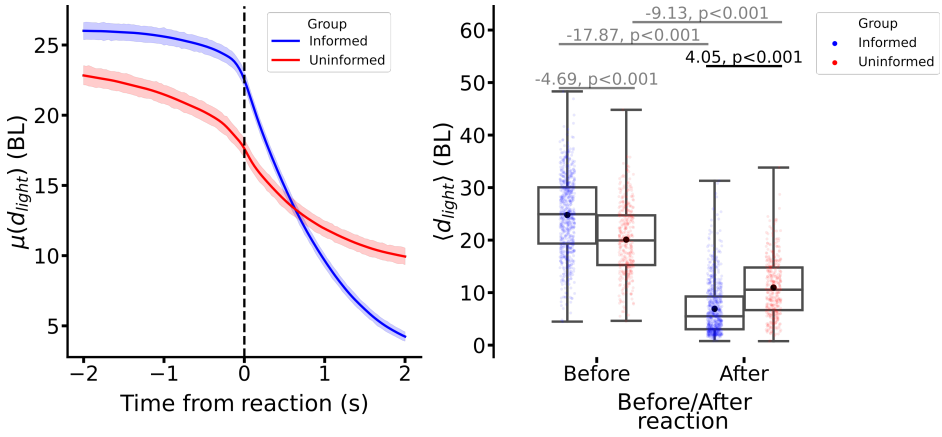

Figure 3.14. **Distance to the light around the reaction time for informed and uninformed fish.** Only fish that reacted during the first 2 s are considered. **Left.** Mean distance to the light around the reaction time for informed and uninformed fish. **Right.** Mean distance to the light in the intervals $[-1, 0)$ s (before) and $[1, 2)$ after the light onset. Each dot represents the mean for a fish in a given trial of the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

informed ones ($D(\overline{\langle d_{\text{light}} \rangle}) = 4.05$ BL, $p < 0.001$, $N_{\text{informed}} = 663$, $N_{\text{uninformed}} = 361$; see Figure 3.14 right). These results confirm that, after reacting, informed fish got on average closer to the light than the uninformed ones.

Figure 3.15 (left) shows a fast increase of the mean speed for both the informed and uninformed fish at the reaction time. The maximum of the mean speed of the informed fish was ≈ 18 BL/s while the one of the uninformed fish was ≈ 12 BL/s. In the first 2 s after the reaction time, the mean speed of both informed and uninformed fish decreased until values similar to the ones before the reaction. However, the mean speed of the informed fish seemed to decrease slower than the ones of the uninformed fish. After reacting the mean speed of the informed fish was higher than the one of the uninformed fish ($D(\overline{\langle v \rangle}) = 4.13$ BL/s, $p < 0.001$; $N_{\text{informed}} = 663$, $N_{\text{uninformed}} = 361$; see Figure 3.15 right). This suggests that after reacting the informed fish moved in general faster than the uninformed ones.

These results suggest that while both informed and uninformed fish increased their speed at the reaction time, there are on average qualitative and quantitative

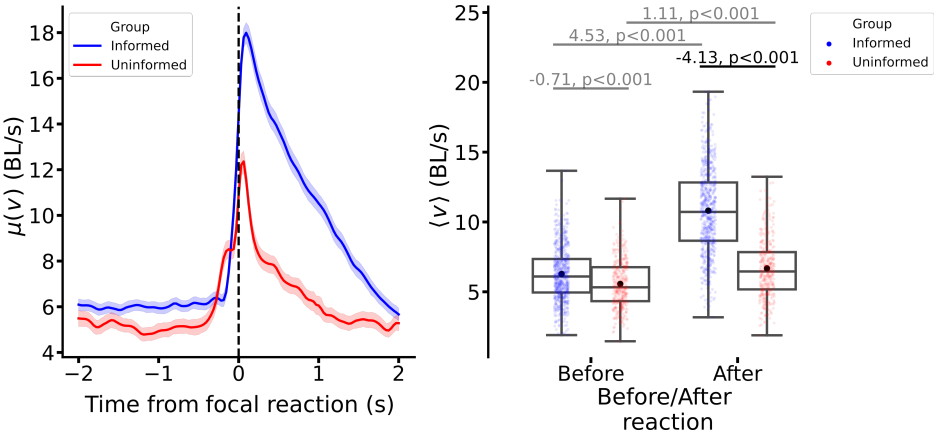

Figure 3.15. **Speed around the reaction time for informed and uninformed fish.** Only fish that reacted during the first 2 s are considered. **Left.** Mean speed around the reaction time for informed and uninformed fish. **Right.** Mean speed in the intervals $[-2, 0)$ s (before) and $[0, 2)$ s after the light onset. Each dot represents the mean for a fish in a given trial of the experiment. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

difference in the reactions. In general, informed fish reacted with higher speeds than the uninformed fish. As a consequence, informed fish got generally closer to the light than the uninformed ones after reacting.

3.2.4 Deep attention networks predict the uninformed fish behaviour

All results so far indicate that some of the uninformed fish moved towards the light after the onset same as the majority of the informed fish that reacted before them. This suggests that the uninformed fish that reacted copied to some extend the behaviour of the informed fish that reacted before.

Our goal is to understand the interaction rules that allow uninformed fish to follow the informed fish. To do that, we will use deep attention networks as described by Heras et al. (2019). Remember that, similar to other models, the deep attention networks assume that interactions occur in pairs and are aggregated as a weighted sum across neighbours. However, by modeling the pairwise interac-

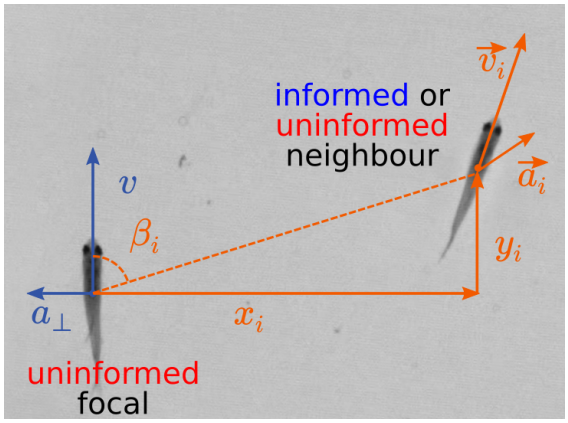

Figure 3.16. **Diagram of the variables used for predicting future turns.** Only uninformed fish are considered as focals. The neighbour fish can be either an informed or uninformed fish. Asocial variables, in blue. Social variables, those involving both the focal and neighbour fish, in orange.

tions and the weights of the sum as fully connected networks, these deep attention network models are more flexible and can potentially capture more details about the real interactions (see also Section 1.3.1).

Since the number of animals and our experimental paradigm is different than the one in Heras et al. (2019) we decided to check which variables from the focal fish and its neighbours are more relevant to predict the behaviour of the uninformed fish in our experiment. We followed a similar approach as in Heras et al. (2019) and used interaction networks (Battaglia et al., 2016) to check the relevance of the different variables in the prediction. Then, we used the set of variables of the two best performing models to check the performance of the deep attention networks.

We trained interaction networks and deep attention networks to predict the turning side of the uninformed fish after 500 ms (see Section 3.4.11 for details about the models and the training). Note that we are not trying to predict again whether a fish is going to react or not at a given time after the light onset, like others before (Strandburg-Peshkin et al., 2013; Rosenthal et al., 2015; MacGregor et al., 2020). Here we want to be able to predict the future turning side of the

uninformed fish for any moment of the experiment. In particular, the training data includes data before and after the light onset and also data from the first 2 min interval of the experiment where the uninformed fish were swimming without the informed fish and without any light being activated (see Section 3.4.3.2 and Figure 3.4). The idea behind this it to build a model of the uninformed fish behaviour as general as possible and to use it to study how they interacted with informed and uninformed neighbours after the light onset.

We considered basic dynamical variables from the focal and the neighbours that could be computed directly from the trajectories of the fish. In particular, the set of all variables that we considered included the focal speed $v = \|\vec{v}\|$ and its normal acceleration $a_\perp$, the relative position of each neighbour, (x_i, y_i), its velocity, $\vec{v}_i$, and acceleration, $\vec{a}_i$, vectors (see Figure 3.16).

In test data, the best performing interaction network model had an accuracy of 72.55% for turning angles between 20° and 160°, and the second best performing model had an accuracy of 72.34% for the same angles range (see Table 3.1). The best performing model included the focal speed v, and normal acceleration $a_\perp$, and the neighbours relative position (x_i, y_i), velocity $\vec{v}_i$ and acceleration $\vec{a}_i$. The second best performing interaction network considered the same variables but ignored the neighbours acceleration. The accuracies of all models were higher for turning angles between 30° and 100°, in this case the two best performing models had more similar accuracies, 73.57% and 73.59% respectively.

The accuracy increases by 20 points with the addition of the normal acceleration, $a_\perp$, because fish most often keep on turning in the same direction. Adding the neighbour relative position increases another 2 points to the accuracy (see Table 3.1). However, the accuracy is the highest when considering also the neighbour's velocity and acceleration. This suggests that knowing the velocities and accelerations of the neighbours helps to obtain a better prediction, and thus that they most likely play a role in the interaction rules.

Next, we trained deep attention network models with the same set of variables as the ones in the two best interaction network models. The deep attention networks have two submodules that can have different sets of input variables (see Section 3.4.11 and Equation 3.3). For the pair-interaction submodule (Π_A) we chose the same variables as the interaction network models. As in Heras et

interaction network (Π_I) variables	test accuracy (%) $(20°, 160°)$	$(30°, 100°)$
v, $a_\perp$, (x_i, y_i), $\vec{v}_i$, $\vec{a}_i$	72.55	73.57
v, $a_\perp$, (x_i, y_i), $\vec{v}_i$	72.34	73.59
v, $a_\perp$, (x_i, y_i), $\vec{a}_i$	72.11	73.15
v, $a_\perp$, (x_i, y_i)	72.15	73.40
v, $a_\perp$, $\vec{v}_i$	70.11	71.38
v, $a_\perp$, $\vec{a}_i$	70.06	71.22
v, $a_\perp$	70.05	71.28
v	51.34	50.02

Table 3.1. **Interaction network models test accuracy.** Accuracy in test trails in an interval of 0.5 s to 2 s after the light onset for interaction network models trained to predict the turning side 500 ms in the future. The left column indicates the input variables of each model. The i subscript indicates neighbour variables. The two right columns show the accuracy for two different ranges of turning angles for each of the models. Models are ranked by the accuracy in the angle range of $(20°, 160°)$.

al. (2019), for the aggregation submodule (W) we considered the focal speed v, and the neighbours relative position (x_i, y_i), speed $v_i = \|\vec{v}_i\|$ and norm of the acceleration $a_i = \|\vec{a}_i\|$.

Similar to the interaction network, the best performing deep attention model on the test data included the neighbours acceleration, the vector in the pair-interaction submodule and the module in the aggregation submodule (see Table 3.2). This model had an accuracy of 72.42% while the model that ignored the neighbour's acceleration had an accuracy of 72.33%. The accuracies of both models for turning angles between 30° and 100° were also higher and more similar between them.

Altogether, these results show that deep attention networks can predict well the behaviour of the uninformed fish after the light onset. The small differences between considering or ignoring the neighbour acceleration suggest that this variable might be only important in very particular cases. For simplicity, in the rest of the chapter we will consider only the deep attention model that considers the neighbour's relative position and velocity.

aggregation (W) variables	pair-interaction (Π_A) variables	test accuracy (%) $(20°, 160°)$	$(30°, 100°)$
v, (x_i, y_i), v_i, a_i	v, $a_\perp$, (x_i, y_i), $\vec{v}_i$, $\vec{a}_i$	72.35	73.30
v, (x_i, y_i), v_i	v, $a_\perp$, (x_i, y_i), $\vec{v}_i$	72.33	73.31

Table 3.2. **Attention network models performance.** Accuracy in test trails in an interval of 0.5 s to 2 s after the light onset for attention network models trained to predict the turning side 500 ms in the future. The two left columns indicate the input variables for the aggregation and pair-interaction submodules respectively. The i subscript indicates neighbour variables. The two right columns show the accuracy for two different ranges of turning angles for each of the models. Models are ranked by the accuracy in the angle range of $(20°, 160°)$.

3.2.5 The deep attention network aggregation function

The final impact of each neighbour on the probability of the focal turning right (see Equation 3.5) depends on the normalized aggregation weight $\omega_i = \frac{w_i}{\sum_j w_j}$, where w is the output of the aggregation function W for each neighbour. To understand how the uninformed fish aggregated information from its neighbours it is then important to understand how ω_i changes with its input variables.

In the selected model (second row in Table 3.2), the aggregation weight, $w_i = W(v, x_i, y_i, v_i)$, depends on 4 variables. In our experiment, each uninformed fish has 14 neighbours, thus ω_i is a function of 43 variables. While it is impossible to plot a function of 43 variables, we can take a look at the shape of the weighting function $W(v, x_i, y_i, v_i)$ for each focal-neighbour pair.

Each panel in Figure 3.17 shows w_i for different neighbour positions relative to the focal with the neighbour and focal speeds fixed. In general, w_i is higher close to the focal fish, and higher in front than behind, with intermediate values on the sides. As the focal speed increases the front-back difference accentuates (see upper row in Figure 3.17). This implies as the focal fish moves faster, the neighbours in front of it carry more weight in the aggregation. As the neighbour speed increases, w_i increases for all neighbours' positions (see lower row in Figure 3.17). This implies that faster neighbours carry more weight in the aggregation than slower ones.

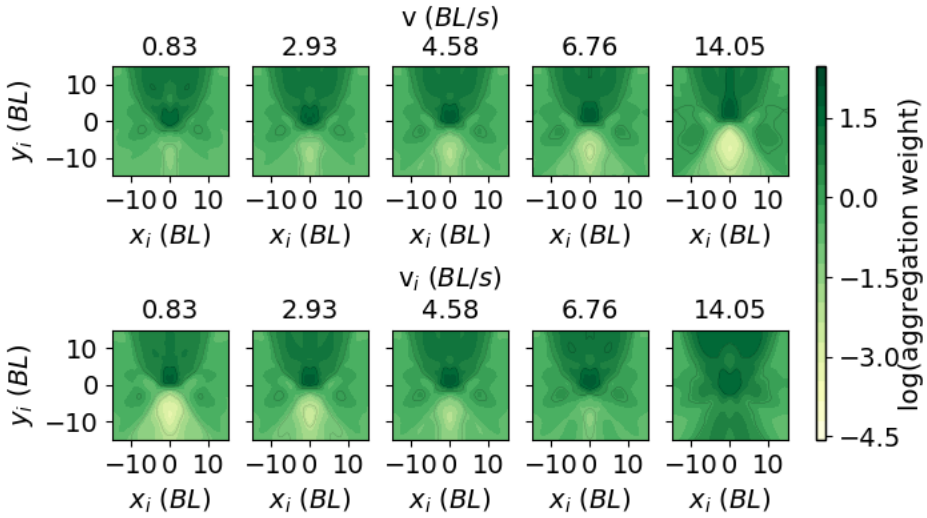

Figure 3.17. **Deep attention network aggregation weight w_i.** Logarithm of the aggregation weight $\log(w_i)$, as a function of the neighbour position (x_i, y_i). **Top.** Neighbour speed fixed to the median speed of all individuals in the dataset, 4.58 BL/s. Each subplot corresponds to a different focal speed, v. The value is indicated on the top. **Bottom.** Same as top row but with the focal speed fixed to 4.58 BL/s and varying the speed of the neighbour v_i.

In Section 3.2.3.3 we showed that at reaction time the mean speed of both informed and uninformed fish increases, with the informed fish moving faster than the uninformed fish after the reaction time (see Figure 3.15).

To be able to give a more complete intuition of how w_i changes with the movement of the neighbours relative to the focal uninformed fish, we also looked at the change of the relative angular position of the neighbours around their reaction time. To this end, we computed the relative angular position of the neighbour i with respect to the orientation of the focal fish (its unity velocity vector), β_i (see Figure 3.16). Under this definition, neighbours in the back of the focal have $|\beta_i| > 90°$, and neighbours in the front have $|\beta_i| < 90°$.

Figure 3.18 shows the absolute value of the mean and the standard deviation of β_i for the informed and uninformed neighbours that reacted before each uninformed focal, $|\mu(\beta_i)|$ and $\sigma(\beta_i)$ respectively. We see that $|\mu(\beta_i)|$ is smaller

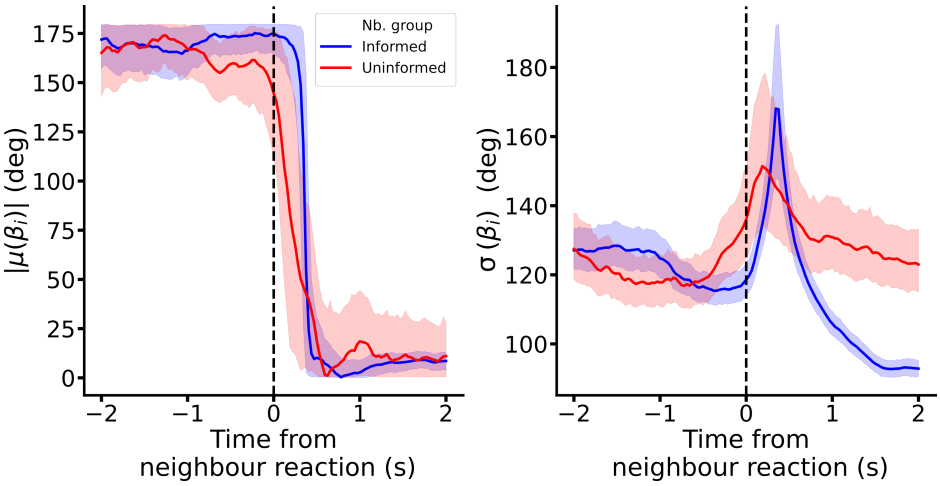

Figure 3.18. **Relative angular position of informed and uninformed neighbours around an uninformed focal fish. Left.** Absolute value of the circular mean of the angle β_i around the light onset for informed and uninformed neighbours of the uninformed focal fish across all trials in the experiment. **Right.** Circular standard deviation of the angle β_i around the light onset for informed and uninformed neighbours of the uninformed focal fish across all trials in the experiment.

after the neighbours reaction time than before for both informed and uninformed fish. Before the reaction, $\sigma(\beta_i)$ takes similar values for both informed and uninformed neighbours. However after the reaction, $\sigma(\beta_i)$ of the informed neighbours increases fast and then decreases to values below the ones that it had before the reaction. The mean of $\sigma(\beta_i)$ of the uninformed neighbours also increases after the reaction and after decreasing but it takes similar values to the ones that it had before the reaction.

The smaller values of $\sigma(\beta_i)$ 1 s after the reaction suggest that informed neighbours occupied positions in front of uninformed focal fish more often than the uninformed neighbours did.

Now we can give an intuition of how w_i changes when a neighbour reacts. Lets suppose an uninformed focal fish has not reacted but a neighbour around it just did. After the reaction the neighbour will generally occupy positions in front of the focal uninformed (see Figure 3.18), as a result w_i will be higher after

the neighbour reaction. This will happen more often if the neighbour was an informed fish. Also, since informed neighbours move faster after reacting than uninformed neighbours (see Figure 3.15), w_i will be higher if the neighbour that reacted was informed than if it was uninformed. This gives an intuition how w_i dynamically changes for neighbours that react around an uninformed focal fish.

Since ω_i depends on the w_i, we can imagine that ω_i will also change dynamically for each neighbour. However, since ω_i is normalized by $\sum_j w_j$ (where j runs for all the neighbours), the change of ω_i does not only depend on the movement of the neighbours itself, but also on the movement of all the neighbours around the focal. In a simplified scenario, imagine that at time t all neighbours around the focal have the same weight. As a consequence, since there are 15 individuals in the group, $\omega_i = 1/14$ for all the neighbours of a given uninformed focal fish. Now lets assume that only one of the neighbours reacts and moves fast in front of the focal fish. The weight w_i of this neighbour will be higher than the one of the other neighbours and as a consequence ω_i will also be higher for such neighbour and smaller for the rest of the neighbours.

This explanation illustrated how w_i and therefore ω_i can change dynamically for each neighbour of a focal uninformed fish.

3.2.6 The normalized aggregation weight of the neighbours was higher after they reacted

In our experiment, we saw that most of the informed fish reacted at the beginning of the cascade when most of the uninformed fish had not reacted yet (see Section 3.2.3.1). Reacting when nobody else has reacted makes your ω_i higher because of the the normalizing denominator. Thus, we expect that ω_i will be disproportionately higher for the informed neighbours than for the uninformed ones.

Figure 3.19 (left) shows the mean aggregation weight around the reaction time considering all neighbours that reacted before any uninformed focal fish. Before reacting, the mean of ω_i for the informed neighbours was around 1/14. At the reaction the mean of ω_i increased fast and afterwards it decreases taking values higher than before the reaction time. In fact, the mean aggregation weight of the

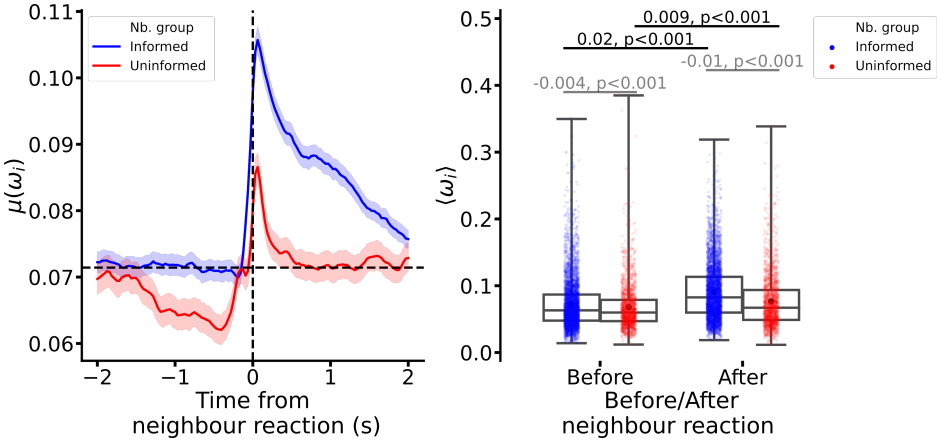

Figure 3.19. **Normalized aggregation weight (ω_i) for neighbour fish around their reaction time.** Only neighbours that reacted within the first 2 s after the light onset and before any focal fish are considered. **Left.** Mean of ω_i across all trials in the experiment around their reaction time. The dashed horizontal line indicates a model in which all neighbours have the same weight $\omega_i = 1/14$. **Right.** Mean of ω_i in the intervals $[-2, 0)$ s (before) and $[0, 2)$ s (after) around the reaction time. Each dot represents the mean for a neighbour. See Figure 3.2 for a description of the boxes and the numbers on top of the boxes.

informed neighbours was higher after the reaction time than before ($\overline{D(\langle\omega_i\rangle)} = 0.02$, $p < 0.001$, $N = 4638$, see Figure 3.19 right). For the uninformed neighbours, the mean aggregation weight was below 1/14 before reacting. At the reaction time, the mean increased and then decreased taking values closer to 1/14. In particular, for the uninformed neighbour the mean aggregation weight was also higher after the reaction than before ($\overline{D(\langle\omega_i\rangle)} = 0.009$, $p < 0.001$, $N = 1876$, see Figure 3.19 right).

In our experiment each focal fish has 14 neighbours. If the uninformed focal fish gave the same weight to all neighbours at all times, the aggregation weight of each neighbour would be 1/14. We saw that the mean of the all informed neighbours before the onset was $\approx 1/14$, however Figure 3.19 (right) shows that the actual mean of the aggregation weight of each informed neighbour after the reaction time is not 1/14 for all of them. This is because our model assigns different aggregation weights depending on each neighbour's behaviour. However, the fact

that before reacting the mean of all informed neighbours was $\approx 1/14$ suggests that, on average, informed neighbours did not carry a higher weight in the final aggregation than other neighbours around the focal. Conversely, the higher mean aggregation weight after the reaction time suggests that informed neighbours carried on average a higher weight in the aggregation after reacting. Presumably this happened because they moved fast in the direction of the light more often occupying positions in front of uninformed focal fish (see Section 3.2.5).

In Section 3.2.3 we showed that on average uninformed fish reacted after the informed ones. Because of this, before an uninformed neighbour reacts the focal fish will be already assigning high values of to w_i weight to the informed neighbours that reacted before, as we saw above. Since the aggregation weights ω_i must sum 1 when considering all neighbours, it is to be expected that the mean aggregation weight of the uninformed neighbours is below 1/14 already before reacting (see Figure 3.19 left, red line). This suggests that in general the uninformed neighbours carried a lower weight in the final aggregation before reacting. After reacting, the mean aggregation weight of the uninformed neighbours was higher than before and closer to 1/14, suggesting that after reacting, uninformed neighbours did not carry on average a higher weight on the aggregation than other fish around the focal.

All considered, we showed evidence to support the idea that on average informed neighbours that had a higher normalized aggregation weight after reacting.

3.2.7 Uninformed reacting fish assigned higher normalized aggregation weights to informed fish that reacted before them

If the aggregation weight of the neighbours is related somehow to the reaction of the uninformed focal fish, the share of the aggregation weight that a focal assigns to neighbours that have already reacted should be higher if the focal fish is reacting than if it has not reacted yet. Here we tested whether this hypothesis is actually true.

For each uninformed focal fish and at each time t we defined the share of the aggregation weight assigned to the neighbours that reacted before t as

$$\Omega(t) = \sum_{\forall i:\tau_i<t} \omega_i(t) \tag{3.1}$$

where τ_i is the reaction time of the neighbour i and ω_i is the normalized aggregation weight that the focal assigned to neighbour i. We computed $\Omega(t)$ for the informed and uninformed neighbours separately, so we define Ω_{un} for the uninformed neighbours, and Ω_{in} for the informed neighbours. Figure 3.20 (left) shows that both the mean of Ω_{in} and Ω_{un} increases in time. This was to be expected since more fish react with time (see Figure 3.11). Figure 3.20 (left) also shows that the mean of Ω_{in} increases faster than Ω_{un}. This also makes sense since informed neighbours have on average smaller reaction times than the uninformed one, and the uninformed neighbours also have more variable reaction times (see Section 3.2.3.1).

To test our hypothesis we labeled the uniformed focal fish at each time t depending on whether they were reacting or they had not reacted yet. Based on the median reaction time to the light of the earliest reacting informed fish (≈ 250 ms; see Section 3.2.3.1), we expected that if an uninformed focal fish reacted at time t, the cause should be in a short interval before the reaction. We set a minimum threshold of 125 ms and labeled each uninformed focal fish at each time t depending on whether it was *reacting* at time t ($\tau - 125\text{ms} < t < \tau$ ms) or it had *not reacted* yet ($0 < t < \tau - 125$ ms, or the fish had not reacted within the first 4 s after the onset, i.e. no τ defined). We ignored the focal fish after they had reacted ($t > \tau$). We call this label the *instantaneous reaction condition* as it can change for each time t.

Figure 3.20 (right) shows that Ω_{in} and Ω_{un} are in general higher for the uninformed focal fish that were reacting than for the ones that had not reacted. However, we will show next that this difference mixes two effects.

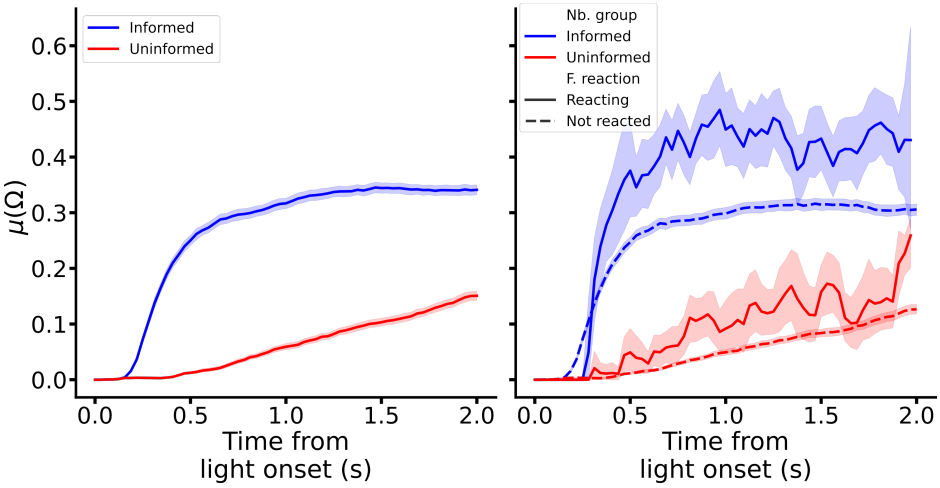

Figure 3.20. **Share of the aggregations weight (Ω) for informed and uninformed neighbours. Left.** Mean of Ω for informed and uninformed neighbours regardless of the reaction condition of the focal uninformed fish. **Right.** Mean of Ω for informed and uninformed neighbours (Nb. group) depending to the instantaneous reaction condition of the uninformed focal fish (F. reaction) at each time step after the light onset.

For each uninformed focal fish we can compute the fraction of neighbours that reacted before a given time t as

$$\Phi_{\text{nb}}(t) = \frac{1}{14} \sum_{\forall i : \tau_i < t} 1 \tag{3.2}$$

where i considers all the neighbours of a given group (informed or uninformed). Thus, we define $\Phi_{\text{nb,in}}$ and $\Phi_{\text{nb,un}}$ for each neighbour group respectively.

Figure 3.21 shows that the mean of $\Phi_{\text{nb,in}}$ and $\Phi_{\text{nb,un}}$ is also higher for uninformed focal fish that were reacting than for uninformed focal fish that had not reacted. Since $\Omega(t)$ depends on the number of neighbours that reacted before a time t, this suggests that the differences observed in Figure 3.20 (right) are partially due to a difference in Φ_{nb}.

To eliminate this effect, instead of using the time from the light onset as a measure of the evolution of the sequence of reactions, we decided to look at the

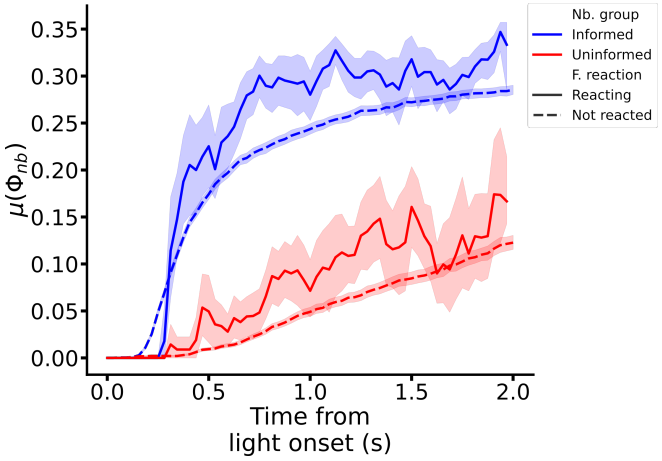

Figure 3.21. **Fraction of neighbours that reacted ($\Phi_{\mathbf{nb}}$) after the light onset.** Mean of Φ_{nb} depending on the neighbour group (Nb. group) and on the instantaneous reaction condition of the focal fish (F. reaction) at each time step after the light onset. We considered as focals the fish that reacted within the first 2 s and fish that did not react at all during the first 4 s after the light onset. We only considered neighbours that reacted within the first 2 s after the light onset and before any focal fish at any time step. Note that the oscillations in the "reacting" curve arise from the time-to-time difference of the number of individuals in the "reacting" group.

fraction of fish that had already reacted, Φ_{reacted}. Note that, after all, Φ_{reacted} is a different way of looking at the progress of the sequence of reactions but it ignores the reaction times. To show that this choice in fact eliminates this effect, in Figure 3.22 (left) we plotted Φ_{nb} as a function of Φ_{reacted}. We can see that in this case, we do not see differences between the reacting focals and the ones that had not reacted yet.

Reassuringly, Figure 3.22 (left) shows an important feature of our experiment, this is the fact that the majority of the informed neighbours reacted before the uninformed ones (see Figures 3.10 and 3.11). In our experiment, each uninformed focal fish has 14 neighbours, 5 of which are informed and 9 are uninformed. If all informed neighbours had reacted before the uninformed ones, $\Phi_{\mathrm{nb,in}}$ should grow linearly up to the value of $5/14 \approx 0.36$ and then stay constant (see blue dot-dashed line in Figure 3.22 left). Conversely, $\Phi_{\mathrm{nb,un}}$ should be 0 until $\Phi_{\mathrm{reacted}} =$

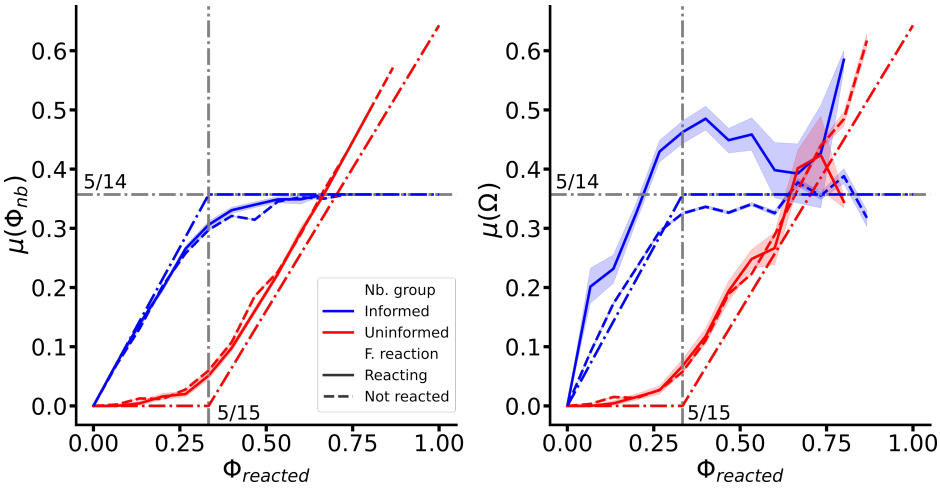

Figure 3.22. **Fraction of neighbours that reacted, $\Phi_{\mathbf{nb}}$ and share of the normalized aggregation weight, Ω. Left.** Mean of Φ_{nb} for informed and uninformed neighbours that reacted before the focal fish for any value of the total fraction of fish that had already reacted, Φ_{reacted}. The continuous and dashed lines indicate the instantaneous reaction condition of the focal fish. The dot-dashed line indicate the same results for a model in which all informed fish react before all the uninformed fish. **Right.** Mean of Ω for informed and uninformed neighbours that reacted before the focal fish for any value of Φ_{reacted}. The continuous and dashed lines indicate the instantaneous reaction condition of the focal fish. The dot-dashed line indicates the same results for a model in which all informed fish react before all the uninformed fish and the normalized aggregation weight, ω_i of all neighbours is the same, i.e. $1/14$.

$5/15 \approx 0.33$ and then grow linearly until the value $9/14 \approx 0.64$ (see red dot-dashed line in Figure 3.22 left) We can see that in our experimental data, $\Phi_{\mathrm{nb,in}}$ increases fast until ≈ 0.3 and then it asymptotically saturates to the value of $5/14 \approx 0.36$. Conversely, $\Phi_{\mathrm{nb,un}}$ grows slowly until $\Phi_{\mathrm{reacted}} = 5/15 \approx 0.33$ and the grows linearly. Note that in the our experimental data Φ_{nb} does not take values after $\Phi_{\mathrm{reacted}} \approx 0.8$. This is due to the fact that in our experiment, not all fish react after the light onset.

At this point, we are ready to explain the differences of Ω for the different reaction condition of the uninformed focal fish. Figure 3.22 (right) shows the mean of Ω for the informed and uninformed neighbours depending on the reac-

tion condition of the focal fish. The mean of Ω_{in} increases faster and it reaches higher values if the focal fish was reacting than if the focal fish had not reacted. This implies that, generally, the aggregation weight, ω_i, of the informed neighbours that had already reacted was on average higher if the uninformed focal was reacting than if it had not reacted yet. Conversely, we do not observe clear differences in the mean of Ω_{un} regardless of the reaction condition of the focal fish.

To further understand that the differences that we observed are a consequence of the dynamically changing aggregation weight of the neighbours, we compared the results with a model that assigns the same weight ω_i to all neighbours. In such model, $\omega_i = 1/14$ for all neighbours, and Ω is exactly the same as Φ_{nb}. So, we only need to compare the left and right panels in Figure 3.22. Such comparison reveals that Ω_{un} is similar to $\Phi_{\text{nb,un}}$, suggesting that the aggregation weight of the uninformed neighbours after their reaction was on average close to 1/14. Indeed, we already saw in Section 3.2.6 that this was the case (see Figure 3.19).

The comparison also shows that the uninformed focal fish that had not reacted yet generally assigned weights closer to 1/14 to the informed neighbours that had already reacted. Conversely, the uninformed focal fish that were reacting assigned higher aggregation weights to the neighbours that reacted before them.

All considered, these results suggest that reacting uninformed focal fish payed more attention to the informed fish that reacted before them.

3.3 Discussion

3.3.1 Summary of main findings

In Heras et al. (2019) we showed that deep attention networks are useful for revealing the rules of interaction in animal collectives. In large groups of juvenile zebrafish we found the rule of information aggregation to be flexible and change dynamically in time. This suggested that zebrafish could potentially used it to weight higher the neighbours that bring new information to the group in a given moment. However, the behaviour of the animals in the data used to discover such rule was highly correlated as we allowed them to swim freely in the experimental arena. In addition, we know that in general deep neural networks capture both spurious and causal correlations. Because of this, it was unclear whether the rules of information aggregation that we found were real or a mix of causal and spurious correlations.

To help separate this and to understand whether zebrafish use this aggregation rule, we designed an experiment with a clearer causal direction (see Section 3.4.3). Before the experiment, we trained fish to move fast towards an area of the experimental arena where we presented a visible light (see Section 3.2.1.1). Conversely, we desensitized uninformed fish so that they would stay away from the light after the onset (see Section 3.2.1.2). In the experiment, we mixed informed and uninformed fish in the same group. After the light onset, the informed ones had a similar behaviour to the one they had during the training sessions (see Section 3.2.2.1). Conversely, the uninformed fish generally got closer to the light (see Section 3.2.2.2) and moved on average at higher speeds (see Section 3.2.2.3), which they did not do in groups of only uninformed fish. Moreover, in the experiment, most of the informed fish reacted before the uninformed fish (see Section 3.2.3). This suggests that the change of behaviour of the uninformed fish was at least partially caused by the earlier reaction of the informed fish to the light.

To study how uninformed fish aggregated information from their neighbours we took the following steps. First, we used idtracker.ai to extract trajectories and animal identities of all fish in the training sessions and in the experiment (see Section 3.4.4). Second, we developed a new tool called idmatcher.ai that

matched the fish identities across videos allowing us to assign each fish to its corresponding experimental condition in the videos where they swam together (see Section 3.4.7). Note that a similar tool was already available for fruit flies (*D. Melanogaster*) (Murali et al., 2019), but idmatcher.ai works directly with the output of idtracker.ai. Third, we designed a recurrent neural network, ReactNet, to detect reactions of the fish after the light onset (see Section 3.4.9). Fourth, we trained a deep attention network model to predict the turning side of the uninformed fish under general conditions (see Sections 3.4.11 and 3.2.4). Finally, we looked at the aggregation submodule of the deep attention networks to study how uninformed fish aggregated information from their neighbours in the experiment (see Section 3.2.5).

Similar to what we found in Heras et al. (2019), the aggregation rule of the uninformed fish was also flexible and changed dynamically along the experiment. In particular, in the model selected for analysis, the neighbour aggregation weight changed with the speed of the focal fish, and the relative position and speed of the neighbours (see Figure 3.17). This weight was higher the faster the neighbour moved and in general it was higher if the neighbour was in front of the focal fish than if it was behind. This front-back difference was more accentuated the faster the focal fish moved.

In our experiments, after we turned the light on, the majority of the informed neighbours moved fast towards it and more often occupied positions in front the uninformed focal fish than uninformed neighbours. As a result, the normalized aggregation weight, ω_i, was generally higher for the informed neighbours that reacted to the light onset before them than for the uninformed neighbours (see Section 3.2.6). Finally, we found that the share of the aggregation weight assigned to informed fish that had already reacted after the light onset, Ω_{in}, was higher if the uninformed focal fish was reacting than if it had not reacted (see Section 3.2.7). This confirms the flexible information aggregation rule found with the deep attention networks allows the uniformed fish to match the changing distribution of information in the group.

Altogether, these results show that deep attention networks could predict well the behaviour of fish in other experimental paradigms and could give insight about how animals might aggregate information in collectives. We are aware that

the experimental methods and the choices made during our analysis have some limitations and left some open-ended questions. Considering that the results presented here are part of a work in progress we next discuss these limitations and propose further analysis that should be considered in the future.

3.3.2 Limitations and open-ended questions

3.3.2.1 Information aggregation experimental paradigm

We showed that in the information aggregation experiments uninformed fish got on average closer to the light than during the last training session (see Section 3.2.2.2). This approach towards the light occurred after the reaction of the majority of the informed fish, which could suggest that uninformed fish moved towards the light only because they copied the behaviour of the informed fish. However, animals make decisions based on their social and asocial information (Pérez-Escudero & de Polavieja, 2011; Arganda et al., 2012). In our experiment all fish could see the light in the extreme of the arena, which would be the asocial information. Because of this, we cannot completely say that uninformed fish moved towards the light only because of social factors. For example, the uninformed fish could have a mild attraction towards the light that gets amplified by the reaction of the informed fish.

In that sense, our experimental design shares some limitations with the experiment in Strandburg-Peshkin et al. (2013). Similar to our experimental procedure, Strandburg-Peshkin et al. (2013) trained fish to move towards a light in a localized area of the arena and paired such visible light with food. However, in their case the uninformed fish did not undergo any desensitization procedure and were reused in a haphazard order in each experimental session. We went a step further by controlling the information about the light that the uninformed fish had before being mixed with the informed fish, and did not reuse fish for each experiment. We showed data supporting the idea that our uninformed fish did not have a strong attraction towards the light (see Section 3.2.1.2) when swimming in groups of only fish. However, as we will see next the uninformed fish were not completely unaware about the light.

3.3.2.2 Uniformed fish prior asocial information about the light

The analysis of the uninformed fish behaviour during the training session gives us an intuition about their prior information about the light in the absence of other contradicting social information, i.e. the informed fish (see Section 3.2.1.2).

Zebrafish typically display sensitivity to visible light, positive phototaxis in larval zebrafish and light avoidance in adult zebrafish (Kalueff et al., 2013). Steenbergen et al. (2011) reported that juveniles at 70 days post fertilization (dpf) display strong dark-avoidance behaviours in a standardised light/dark preference test. In the last training session our uninformed fish ($34 - 35$ dpf) stayed generally away from the light and slightly decreased their mean speed after the onset. However, we also observed a small decrease of the mean distance to the light 3 s after the onset. We considered that due to the differences in the experimental procedure a direct comparison between a standardised light/dark preference test and our design can be misleading. Here we briefly described the behaviour of the uninformed fish in all training sessions to get a sense of their priors about the light.

On the one hand, the decrease of the mean speed after the light onset in the last training session (see Figures 3.3 and 3.9) suggests that the uninformed fish might have an aversive response to the light. In fact, the mean acceleration towards the light of the uninformed fish shows a fast decrease right after the onset in all training sessions and also in the experiment (see Supplementary Figure A.14 left). This suggests that juvenile zebrafish could have some fast negative aversion to the light onset that existed even before the training procedure. For example, some fish could have shown some degree of freezing behaviour, commonly an indicator of alarm reactions and also associated with anxiety-like behaviour (Kalueff et al., 2013).

On the other hand, the decrease of the mean distance to the light 3 s after the onset in the last training session (see Figures 3.2 and 3.7) might suggest that uninformed fish might also have a mild attraction to the light at longer times after the onset. In fact, we observe the same decrease of the mean distance to the light in all training sessions (see Supplementary Figure A.14 right). This might suggest that juvenile zebrafish had on average a small positive attraction to the light at longer times after the onset prior to the training procedure. However,

note that this could also be an effect of regression to the mean. Uninformed fish were on average away from the extreme of the arena that we illuminated with the visible light and with time they are more likely to get closer to the light. Further analysis should investigate whether this small attraction is real or an effect of regression to the mean.

In any case, this suggests that uninformed fish are not completely indifferent to the light. However, remember that the uninformed fish response after the light onset in the presence informed fish was in many ways different to the one they had during the training procedure. Generally, they showed higher accelerations towards the light (see Supplementary Figure A.14 left), higher speeds (see Figure 3.9), and got closer to it (see Figure 3.7) than in the training sessions. In addition, in the experiment they reacted generally after the majority of the informed fish, and did so with higher and more variable reaction times (see Section 3.2.3.1).

Because of all this, we can say that the response of the uninformed fish after the light onset in the experiment was at least partially a consequence of the social influence of the informed fish. However, we cannot discard that some part of their response after the light onset is also due to a potentially innate response to an unpredictable visual cue.

3.3.2.3 Experimental bias before the light onset.

In Section 3.2.2.2 we pointed out that the mean distance to the light across all the uninformed fish slightly increased before the onset in the last training session. Conversely, in the experiment the same mean had a decreasing tendency before the onset (see Figure 3.7). Here we explain how this could be an effect of an experimental bias caused by the strategy followed to turn the light on.

During the training sessions, for the uninformed fish we activated the stimulus at pseudo-randomized times (see Section 3.4.3.1). However, in the experiment, we followed a strategy similar to the one during the training sessions of the informed fish (see Section 3.4.3.2). This strategy was to turn the light on when some of the fish were slightly oriented towards the it. This experimental bias appears clearly in the mean and standard deviation of the angle θ_{light} (see Figure 3.1c and

Supplementary Figure A.15). Uninformed fish were more often oriented towards the light before the onset in the experiment than in the last training session.

Besides this experimental bias, we showed in Section 3.2.2.2 that the uninformed fish got closer to the light in the experiment than in the last training regardless of their orientation towards the light at the onset time (see Figure 3.8).

In any case, future experiments following the same experimental paradigm should avoid this bias. A possible improvement on the experimental procedure is to use a second camera that measures different population level variables with respect to the light (distance, orientation, speed...) in real time. Then the light could be turned on under the same conditions in the last training sessions and in the experiment. This improvement could be implemented with the Bonsai framework (Lopes et al., 2015) which interacts seamlessly with the current HARP Behaviour controller used to turn the light on and activate the feeder device (see Section 3.4.2).

3.3.2.4 Reaction detection using ReactNet

To detect the times when a fish had reacted after the light onset we used ReactNet (see Section 3.4.9). ReactNet estimates the probability that the light was active given the behaviour of the fish. We chose a threshold on the probability above which we considered that the fish had reacted. The results that we show in this chapter are for a threshold of 0.8. We obtained similar results with lower thresholds (0.75 and 0.7). However, note that lower thresholds are prone to detect more false positives, while higher thresholds are prone to detect more false negatives. Because of this, we decided to choose the higher threshold so that in the worst case scenario we would be missing some fewer evident reactions.

Also, note that the input to ReactNet included information about the experimental group of each fish. This gives the opportunity to the network to consider differently the variables of the informed and uninformed fish. However, this higher flexibility has the added problem that, in principle, the optimal threshold could be different for the informed and uninformed fish. For simplicity, we considered the same thresholds for both informed and uninformed fish. Further analysis should consider a sensitivity analysis to quantify the impact of these thresholds on the results.

Previous similar experiments used a combination of thresholds on the speed and acceleration, or on the speed and turning rate of the fish to detect visually-apparent reactions (Strandburg-Peshkin et al., 2013; Rosenthal et al., 2015; MacGregor et al., 2020). We remember that the dynamical variables used as input in ReactNet are the past locations, speeds and acceleration relative to the light. Therefore, our method is more flexible and it requires fewer assumptions on the variables that determine whether a fish has reacted or not.

Strandburg-Peshkin et al. (2013); Rosenthal et al. (2015) did not use real ground truth data but they specified that the thresholds were chosen so that the detected responses showed visually-apparent response points. We built a reaction exploration dashboard were we could inspect whether the reaction points detected using ReactNet corresponded to visually-apparent reactions. However, further analysis on the performance of the ReactNet should include a manually annotated ground truth dataset of reactions (see for example MacGregor et al. (2020)).

3.3.2.5 Turning side prediction accuracy

In Section 3.2.4 we showed that the prediction accuracy of the deep interaction and deep attention networks was $\approx 72-73\%$. The accuracy obtained in Heras et al. (2019) for videos of 100 fish was $\approx 85\%$. A possible reason of this difference can be due to the level of organization of the collectives.

In fact, in the experiments in Heras et al. (2019) the local polarization was higher in the videos of 100 fish than in videos of 60 fish (Heras et al., 2019, Figures S1 and S3). In line with our hypothesis, the prediction accuracy for the videos of 60 fish was $\approx 76\%$, lower than the one for 100 fish and closer to the one in the results of this chapter.

Smaller groups usually show higher levels of polarization, however habituation to the environment typically decreases the group polarization (Miller & Gerlai, 2012b). In the experiments presented here we used the same experimental arena as in Heras et al. (2019). While our groups were smaller (15 fish), the fish were already habituated to the arena due to the many days of exposure to it. This could have potentially resulted in lower polarization levels, that could explain the lowr accuracies.

Even more, in our experiments the activation of the light provokes a perturbation of the collective behaviour. First, the informed animals moved towards the light and after ≈ 1 second the uninformed fish followed them (see Section 3.2.3). We expect the group organization to change as the cascade of reactions advances. Further analysis should look more carefully at the prediction accuracy considering the temporal changes of the collective organization along the experiment.

3.3.2.6 The role of the acceleration in the prediction

We showed in Section 3.2.4 that the best performing model included as social variables the relative position, the velocity and the acceleration of the neighbours. However, for simplicity we decided to only look at the second best performing model, which did not include the neighbours' acceleration but had a similar performance. The small differences in performance between both models might suggest that the contribution of the acceleration to the interaction rule and to the final prediction is small. Here we give an intuition of why these small differences might occur and suggest situations in which the neighbours' acceleration might play an important role in the final prediction.

Zebrafish movements follow a typical burst-and-glide biphasic pattern that can be easily identified from its speed (Calovi et al., 2018; Laan, Gil de Sagredo, & de Polavieja, 2017; Harpaz et al., 2017; Bod'ová et al., 2018). During the first phase fish accelerate and reach high speeds. In the gliding phase they decelerate due to the drag of the water moving typically in a straight line. In normal swimming conditions the speed and acceleration are quite correlated, especially during the gliding phase. Thus, a predictive model that includes both the speed and the acceleration might not find extra information in the acceleration to improve the prediction.

However, under the presence of an external stimulus, fish could accelerate more strongly than usual or break on purpose decelerating more strongly than they would just by the drag of the water. These situations can make the speed and acceleration be more decorrelated. In these cases, the acceleration could add extra information to the prediction that cannot be extracted from the the speed. In our experiment, this occurs right after the light onset. On the one hand, the mean acceleration of the uninformed fish decreases after the light onset (see

Supplementary Figure A.14). On the other hand, the mean acceleration of the informed fish increases strongly right after the light onset (see Supplementary Figure A.16 left).

We propose that further analysis should look more closely into the role of the acceleration in the final prediction and in the aggregation of information rule, especially shortly after the light onset.

We also want to note that our best performing model includes the norm of the acceleration, $\|\vec{a}\|$, as a variable of the aggregation submodule. However, the norm does not distinguish whether the neighbour is braking, accelerating forward or turning. We suggest that considering the tangential and normal components of the neighbour acceleration should capture better the dynamics of the acceleration (see Supplementary Figure A.16 center and right) and potentially have a higher impact on the prediction accuracy.

3.3.2.7 Uninformed focal fish that reacted at futures more than 125 ms also had high aggregation weights

In Section 3.2.7 we set the hypothesis that the share of the aggregation weight for neighbours that already reacted, Ω, should be higher for reacting focal fish than for focal fish that had not reacted. We showed that reacting focal fish had higher values of Ω_{in} than focal fish that had not reacted (see Figure 3.22).

However, note that the instantaneous reaction condition "not reacted" included fish that reacted at a *future* time t more than 125 ms and fish that *never* reacted during the first 4 s. If the cause of the reaction of the uninformed focal fish was 125 ms before their reaction, as we hypothesize, one would expect that Ω_{in} for the focal fish that reacted at a future time more than 125 ms should be lower than for the fish that were reacting. In favor of our hypothesis, the mean of Ω_{in} is lower for the fish labeled reacted in a future longer than 125 ms than for the fish that were reacting (see Supplementary Figure A.17) However, we noted that these differences are small. Further analysis should investigate whether these differences persist for different values of the time threshold that we set at 125 ms.

3.3.2.8 Second order effects of the uninformed neighbours' reactions

All the results indicate that the informed neighbours play a major role on the reaction of the uninformed focals. In addition, the results in Section 3.2.7 suggest that the uninformed neighbours that reacted before an uninformed focal had little impact on the reaction (see Figure 3.22 right). However, we cannot completely rule out the possibility that other second order interactions exist between the uninformed focals and the uninformed neighbours that reacted before them. In fact, the mean aggregation weight of the neighbours around the focal reaction time shows that the uninformed neighbours that reacted before had a higher aggregation weight than those that reacted after (see Supplementary Figure A.18).

All in all, our results showed that the combination of different deep learning tools (CNNs in idtracker.ai and idmatcher.ai, a RNN in ReactNet, and deep attention networks) can help to extract high-quality data from experiments and to obtain insight about information aggregation rules in animal collectives. In particular, we confirmed that zebrafish seem to use a flexible rule to aggregate information from their neighbours which depends on the neighbours' relative position and speed, and potentially also the neighbours' acceleration. This rule allows the uninformed fish to pay more attention to neighbours that bring new information to the group.

As results of a work in progress, we acknowledge that further analysis must be carried out to understand more closely this information aggregation rule and the impact of other variables that might relevant to predict the fish behaviour.

3.4 Materials and methods

3.4.1 Animal rearing and handling

All fish were raised at the Champalimaud Foundation Fish Platform, according to the housing and husbandry methods integrated in the zebrafish welfare program fully described in Martins et al. (2016). Animal handling and experimental procedures were approved by the Champalimaud Foundation Ethics Committee (CF internal reference 2020/002) and the Portuguese Direcção Geral Veterinária (DGAV reference 0421/000/000/2020) and were performed according to European Directive 2010/63/EU13. We performed all experimental procedures with juvenile zebrafish (28-34 days post fertilization, dpf) of the Tubingen (TU) strain. All experiments presented in this chapter were performed in the months of October to December of 2019 at the Champalimaud Foundation (Lisbon, Portugal).

3.4.2 Experimental setup

The experimental setup that we used to perform the experiments presented in this chapter is a modified version of the one already described in Chapter 2 (see Section 2.4.2). Figure 3.23 shows a zenithal view diagram of the experimental setup. Here we describe the most important modifications required for the experimental procedure of this chapter (see Section 3.4.3)

The experimental box had inside a rectangular water tank that contained holding tanks and the experimental arena. The main tank included a recirculating water system with ≈ 300000 cm^3 of water from the same holding system as the one in which the fish are housed in the fish facility (pH 7.2-7.4; 1300 μS, 28°C). A thermostat and a filter with a water pump ensured adequate water conditions in the main tank. In addition, to ensure adequate water conditions in the holding tanks we expanded the recirculating water system to include them. We used smaller water pumps placed inside of the tank to flow water from the main tank to the holding tank. Each holding tank had a baffle that made water flow back to the main tank keeping the water level in the holding tank constant.

The illumination of the experimental box was the one explained in Section 2.4.2. The fish were hedl in the experimental setup for 7 days. To ensure a

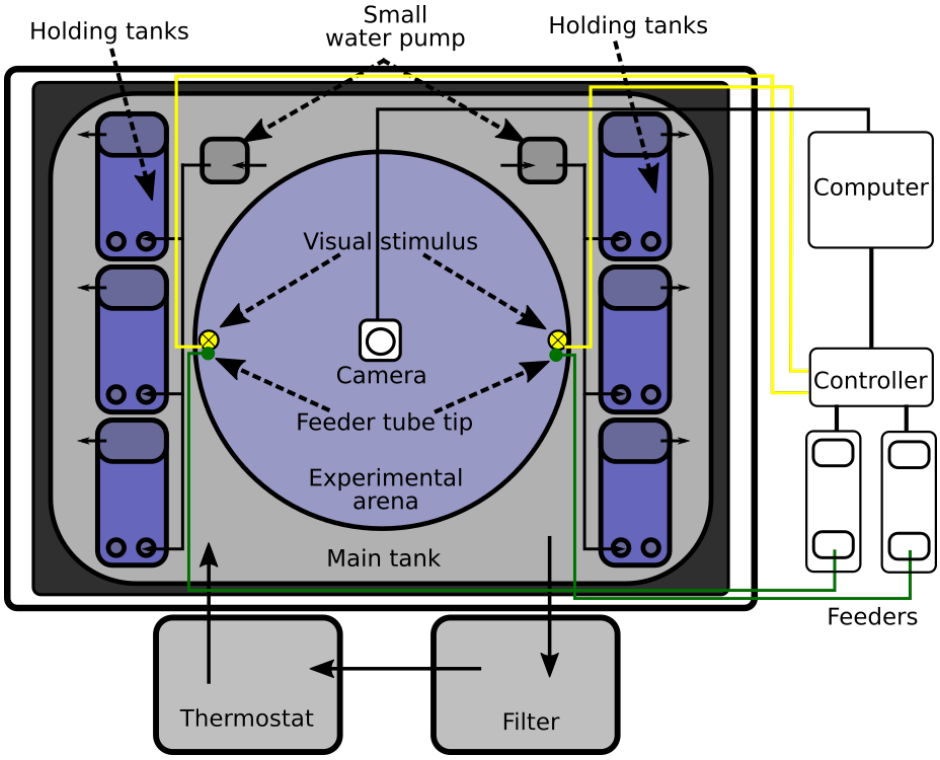

Figure 3.23. **Zenithal view of the experimental setup.** Text and dashed arrows indicate the elements of the setup. Solid black lines with arrows indicate the water flow direction.

14 h/10 h light/dark cycle, we connected an automatic timer to the power source of the RGB and IR strips.

Our experimental procedure required presenting a visible light in localized areas of the arena (see Section 3.4.3). We located two visible white LED in opposite sites of the experimental arena at ≈ 1 cm above the water surface. We placed a second white matte circular tank of the same size as the experimental arena ≈ 3 cm below to provide an opaque surface where we projected the light coming from the visible light LED.

The projection of the white LED provoked heterogeneous light conditions in the video. To avoid them, we connected an IR LED in parallel with the LED so

that they activated at the same time. This IR LED was placed outside of the swimming region of the fish and facing towards the camera. We attached a IR pass filter (MIDOPT LP715 Near-IR Longpass Filter) to the lens of the camera so that only the light with wavelengths below $\approx$ 715 nm would be captured by the camera during video recording. The IR LED also allowed us to automatically synchronize the presentation of the light with the frames of the video.

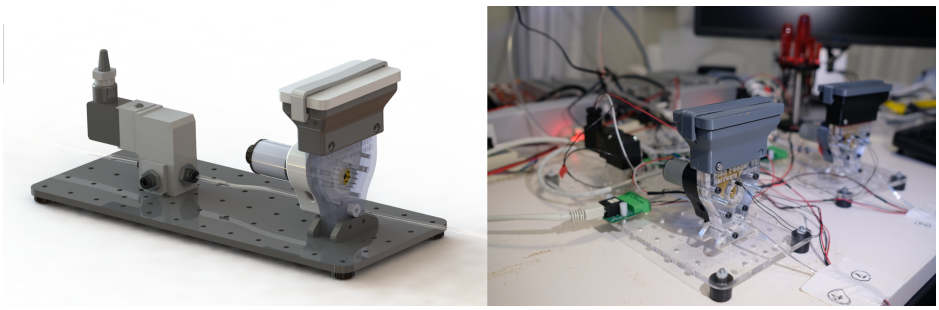

Figure 3.24. **Feeder device. Left.** Model of the feeder device use to deliver small amounts of food during the training procedure. **Right.** Photo of the two feeder devices outside of the experimental box.

To facilitate the deposition of the food in the water during the training of the informed fish (see Section 3.4.3.1), we included in the setup two custom-built automatic feeders (see Figure 3.24). Each feeder consisted of a chamber where the food (GEMMA Micro 300) was stored, a motor with a cogwheel, a valve and a set of tubes. The feeders were placed outside experimental box and a set of tubes directed the food from the feeder to the area in the experimental arena next to where the visible light LED was located (see Figure 3.23). When a feeder was activated, the motor rotated, depositing a small amount of food in the tube. Then, the valve released a puff of compressed air that moved the food until the tip of the tube deposited the food on the surface of the water in the experimental arena. We made sure that the food was deposited in the same area where the light coming from the LED was projected.

We controlled the visible light LED, the IR LED and the feeder using the HARP Behaviour controller[1] through the Bonsai software (Lopes et al., 2015).

[1] `https://www.cf-hw.org/harp/behavior`

3.4.3 Experimental procedure

3.4.3.1 Training and desensitization procedure

Here we describe the procedures that we followed to train groups of juvenile zebrafish ($28-35$ dpf) to move fast in the direction of a visible light in the arena, the informed group, or desensitize them so that stay away from it, the uninformed group.

The duration of the training and desensitization procedure lasted 7 days. The procedure includes 3 days of food deprivation and 4 days of training. All procedures started on Friday, fish were food deprived during the weekend and the training and desensitization sessions started on Monday. For simplicity we call a training and desensitization session just a *training session*. The group condition (i.e. informed or uninformed) determines if it was a training or a desensitization session respectively.

On the first day (Friday), at ≈ 2 pm, we prepared $4-6$ groups of 10 juvenile zebrafish ($28-29$ dpf) in standard 3500 cm^3 holding tanks. We introduced these holding tanks into the main tank of the experimental setup (see Section 3.4.2) and we deprived the fish of food for ≈ 60 hours.

On the fourth day (age was $31-32$ dpf) we started the training and desensitization respectively. We performed 3 training session per day in the following intervals: 9am-11am, 1pm-3pm and 5pm-7pm. We chose these intervals to match the same feeding schedule of fish of the same age held at the Champalimaud Foundation Fish Platform. Each group underwent a total of 10 training sessions. This corresponds to 3 full days (Monday, Tuesday and Wednesday) with 3 sessions each day and a last training session on the seventh day (Thursday) of the procedure. In each of the intervals, we trained $4-6$ groups. We randomized the training order of the groups in each of these intervals.

The final purpose of this training procedure was to create groups of 5 informed and 10 uninformed fish to perform the information aggregation experiment described in Section 3.4.3.2. With this purpose, before the last training session, we separated each group of 10 informed fish into two subgroups of 5 informed fish. Thus, in the last training session each group of informed fish had 5 animals.

At the end of each training session we returned the fish back into their corresponding holding tank. The fish stayed in the holding tank until the following training session or until the information aggregation procedure (see Section 3.4.3.2). To remove any possible bias due to the position of the animals in the main tank, we randomized the position of the holding tanks in the main tank after each training session.

In total we desensitized 22 groups of 10 uninformed fish, and trained 22 groups of 5 informed fish.

Next, we explain the details of the training and desensitization procedures for the informed and uninformed groups. Before each training session, we rinsed the experimental arena with clean water from the same holding system where the fish are housed in the Champalimaud Foundation Fish Platform. Afterwards, we filled the tank with ≈ 5000 cm^3 of the same water, which corresponded to ≈ 2.5 cm of depth in the center of the experimental arena (the body height of a juvenile fish at this age was ≈ 0.3 cm). Then we introduced the fish of the corresponding group in the experimental arena with a standard fish net.

For the uninformed groups, we manually spread food ($\approx 32 \pm 3$ mg of GEMMA Micro 300; mean $\pm$ SD) as uniformly as possible across the arena using a fish-food plastic spoon. We observed that right after spreading the food, fish started to look for the food and consumed it. After ≈ 1 min, fish stopped displaying such movements and started swimming together.

At this point we started the desensitization session. During 8 min, we sequentially presented the visible lights (see Section 3.4.2), one at a time. We call the presentation of one of the visible lights a *trial*. The trials sequence followed a pseudo-randomized order that ensured that the same light was not activated more than 2 trials in a row. Each session contained 14 ± 3 (mean $\pm$ SD) trials and in each trial the light was presented for 5 ± 3 s. Consecutive trials were spaced by 23 ± 16 s.

For the informed groups, we did not spread any food at the beginning the training session. Before the training session we allowed animals to habituate for ≈ 1 min. Then, we started the training session. During 8 min, we sequentially presented the visible lights one at a time in a sequence of trials. The trial sequence was pseudo-randomized in the same way as during the desensitization of the

uninformed fish. Each session contained 9 ± 3 trials. In each trial, if during the first ≈ 5 s after the light onset some of the animals moved towards the part of the arena where the active light was, we rewarded such movement by depositing food ($\approx 5 \pm 2$ mg of GEMMA Micro 300) in such region of the arena using custom feeder devices (see Figure 3.24) and turned the light off after ≈ 10 s. Otherwise, if no animals moved towards the light during the first ≈ 5 s we turned it off. On average, in each trial the light was on for 12 ± 6 s. Consecutive trials were spaced by 33 ± 17 s. To shape the behaviour of the animals to the wanted behaviour, during the first 2 training sessions we activated the visible light and deposited the food when the animals were already swimming towards one of the regions where we presented the light and deposited food.

3.4.3.2 Information aggregation experiment

Here we describe the procedure that we followed to perform an information aggregation experiment with a mixed group of 5 informed and 10 uninformed animals. In what follows, we will refer to an information aggregation experiment simply as an experiment.

We performed the experiment on the same day as the last training session (Thursday), in the interval between 12pm-3pm. At this point the age of the fish was $34-35$ dpf. In this time interval we performed $4-6$ experiments, one for each mixed group of 5 informed and 10 uninformed. The duration of an experiment was 10 min.

We first introduced in the arena a group of 10 uninformed fish and allowed them to swim freely in the arena for ≈ 2 min. Note that we did not spread food anywhere in the arena nor did we switch any of the lights on during this period. After these ≈ 2 min we introduced in the arena a group of 5 informed fish and allowed them to swim freely together with the 10 min fish for ≈ 1 min.

During the next 7 min we sequentially activated the lights one at a time in a sequence of trials. Note that opposite to the training trials, in this case we did not deposit any food in the region of the arena where the visible stimuli was located. The trials sequence was pseudo-randomized as in the training procedure. To ensure that at least some of the informed animals in the group perceived the light, we activated it when some fish were slightly oriented towards it. We did

this by visual inspection of the group. Note that at this point we could not distinguish informed from uninformed fish. Each experiment consisted of 8 ± 2 (mean $\pm$ SD) trials. In each trial we presented the light for 5 ± 2 s. Consecutive trials were spaced by 44 ± 17 s.

In total we performed 22 experiments of mixed groups of 5 informed fish and 10 uninformed fish.

3.4.4 Animal tracking

We video recorded all the training sessions and all the experiments. We extracted the trajectories of the animals in all the videos using idtracker.ai (Romero-Ferrero et al. (2019); see also Chapter 2). To track the 10 min videos of the experiment we used the tracking interval feature of idtracker.ai and tracked the first 2 min (only 10 uninformed fish) and the last 7 min (10 uninformed and 5 informed fish) independently but with the same animal detection parameters (see Table 2.1).

If the estimated accuracy reported by idtracker.ai at the end of the tracking was below 99.5% we re-adjusted the animal detection parameters parameters (see Table 2.1) and tracked the video again. The average estimated accuracy of the tracking for the videos corresponding to the training procedures was $99.9 \pm 0.2\%$. The average estimated accuracy of the tracking for the first 2 minute video interval was $99.98 \pm 0.01\%$. The average estimated accuracy of the tracking for the last 7 minute videos interval was $99.99 \pm 0.01\%$.

3.4.5 Light area presentation annotation

When designing the experimental setup we made sure that the area of the white background where the visible light was presented was in the same region where the feeders deposited the food (see Section 3.4.2). In each video, we manually annotated ≈ 10 points around the area where the feeder devices deposited the food. We computed the coordinates of the center of the region as the mean of the pixel coordinates of the annotated points. We later used these computed coordinates to calculate variables such as the distance of the fish to the stimulus, d_{light}, and the angle ψ_{light}.

3.4.6 Light onset detection

As explained in Section 3.4.2 the experimental arena included an IR LED that was activated at the same time as the visible light LED. The position of the IR LED was easily recognizable in all the videos. We manually annotated the location of each IR LED in each video. Then we automatically created a region of interest (ROI) of ± 50 pixels around the location of the IR LED. We computed the average intensity of the pixels in such ROI for all frames in the video. The time series of the average intensity showed clear changes when the IR LED was active for all videos. To remove background noise, we convolved each time series with a kernel of 3 frames. Then we computed the derivative of such signal by taking finite differences. The distribution of the derivative had 3 modes. Very high positive values indicated the onsets of the IR LED, very low values negative indicated the moment when the IR LED was turned off. Small values around 0 indicated the changes due to background noise. To automatically detect the high transients we set 2 thresholds. The minimum threshold was the minimum edge of the first zero valued bin of a 10 bins histogram of the derivatives. The maximum threshold was the maximum edge of the last zero valued bin of a 10 bins histogram of the derivatives. We detected the onset of the IR LED and hence the onset of the visible light LED as the frames where the derivative of the average intensity time series was higher than the maximum threshold. Note that we properly corrected the length of the time series after the convolution and the computation of the derivative so that the detected frames matched the first frame where the IR LED was active in the video for each trial. We found this method to correctly detect the onset of all trials for all videos.

3.4.7 Informed and uninformed fish identification

The result of tracking one of the last 7 min intervals of the videos of a mixed group of 5 informed and 10 uninformed fish (see Section 3.4.3.2) is a set of identified trajectories for each of the animals in the video. However, the identities assigned by idtracker.ai to each of the animals do not have information about their experimental group (informed or uninformed).

Our goal was to match the identities assigned by idtracker.ai to the 10 uninformed fish in the last 7 min interval of the experiment video, with the ones of the first 2 min interval of the experiment video, where there were only 10 uninformed fish. Note that a unique match of the identities of the first interval with the identities of the second interval immediately gives the identity of the 10 uninformed animals in the mixed group. Consequently, the remaining identities of the second interval video that do not match any of the identities of the first interval video correspond to the informed fish.

To solve this identity matching problem we developed idmatcher.ai, a Python package with tools to match the identities of the animals in two videos tracked with idtracker.ai.[2]

The system uses the trained idCNN (see Figure 2.6) and the set of labeled images resulting from tracking each of the videos with idtracker.ai (see Chapter 2). For each video it constructs the set of labeled images from the individual fragments that idtracker.ai used for training during the training and identification protocol cascade (see Section 2.2.1.4). Given two videos A and B, idmatcher.ai uses the idCNN of video A to identify the set of labeled images of the video B and vice versa. The result are two matrices $\mathbf{M}^{AB}$ and $\mathbf{M}^{BA}$. Each element M_{ij}^{AB} is the number of images of the label j in video B labeled with the identity i by the idCNN of video A, and the same but reversed for elements M_{ij}^{BA}. For each matching matrix we compute the cost matrix $C_{ij}^{AB} = 1 - M_{ij}^{AB} / \sum_j M_{ij}^{AB}$. To find the optimal match of identities we solve the generalized linear assignment problem (with different numbers of rows and columns) with the two cost matrices $\mathbf{C}^{AB}$ and $\mathbf{C}^{BA}$ independently. We used the implementation of the Kuhn-Munkres algorithm or Hungarian algorithm(Kuhn, 1955) from the Scipy Python library.[3] If the assignment was exactly equivalent and unique in both directions we considered that the matching was successful.

idmatcher.ai found a successful match for 20 out of 22 experiments. We only used these 20 videos for the analysis performed with the trajectories of the mixed groups.

[2] `https://gitlab.com/polavieja_lab/idmatcherai`

[3] `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.linear_sum_assignment.html`

3.4.8 Dynamical variables from trajectories

We computed all dynamical variables from the idtracker.ai trajectories files using the Python package Trajectorytools developed by Francisco J.H. Heras and the author of this thesis.[4] Trajectorytools is a Python package that allows computing different measures of 2D trajectories of multiple animals.

We interpolated linearly the very little missing data in the trajectories due to the video-tracking. To reduce noise while preventing contamination by any future information, we smoothed the trajectories using a 5-frame half-Gaussian kernel with $\sigma = 1$ frame. We computed each fish velocity $\vec{v}$ and acceleration $\vec{a}$ by finite differences, using only current and past frames. Considering the pixel coordinates of the center of the light regions (see Section 3.4.5) we computed the distance to the stimulus, d_{light}, and the angle θ_{light} (see Figure 3.1c). We computed the neighbour relative location (x_i, y_i) ((d_i, β_i) in polar coordinates) in a coordinate system where the focal fish location is at (0, 0) and the focal velocity vector is oriented with the positive y axis (see Figure 3.16).

As the spatial unit we used the body length, BL, of the fish. The body length for each set of trajectories was the one estimated by idtracker.ai to compute the size of the identification images (see Section 2.2.1.1 and Table 2.2). One BL was on average ≈ 1.3 cm. As the time unit we used seconds (s). We estimated the time step from the frame rate of the videos, which was 32 frames per second in all videos.

See Section 3.4.11.6 for details about the pre-processing of the trajectories for the training of the deep neural network models.

3.4.9 Animal reaction detection

We trained a neural network, ReactNet, to determine whether, and if so, when, each animal reacted during the experiment. The input to the network is a 2D array representing the trajectory and class of one animal during a time interval. The output is a 1D array whose values are a monotonically increasing function of the probability of the animal having reacted in each frame of such interval. We use a threshold to the output to decide if (and when) the animal reacted.

[4] `https://github.com/fjhheras/trajectorytools`

3.4.9.1 Input and output

The input to ReactNet is a 2D array. In each frame (the first dimension is a temporal one), the array encodes the location (2 components), velocity (2 components) and acceleration (2 components), together with a binary flag representing whether the animal is informed or uninformed (1 component). To facilitate learning, the trajectories are rotated and displaced so the active light is placed in (0,0) and the centre of the arena is on the positive x semiaxis. Training is performed using a minibatch of such 2D arrays, each sampled from a different animal, possibly from different trials and taken at different times during an interval of -4 s to 4 seconds around the light onset. Data augmentation is performed during training by rotating the trajectories by a small random angle around the origin.

3.4.9.2 Architecture

The structure of ReactNet was a multi-layer gated recurrent unit (GRU; Cho et al. (2014), a type of recurrent neural network), followed by a cumulative max layer and a sigmoid. This structure forces the network to have some useful properties. The first useful property is that each component of the output depends only on past points of the trajectory. The second useful property is that the output of the network increases in the temporal dimension.

3.4.9.3 Training

To train ReactNet, we used a total of 454 trials. The trials were of two classes. The first class were trials from training sessions, where all fish were informed and had already learned to react to the light (last two training sessions). The second were trials from the experiment, where informed and uninformed fish were together in the arena (second 7 min interval of the experiment). Note that we avoid cases where only uninformed fish are in the arena (first 2 min interval of the experiment), or early during the training of the informed fish, because we did not expect any meaningful reactions in these cases.

Among the trials of the experiments, we selected 4 trials for validation and 4 trails for testing. This leaves 446 trials to build the training dataset.

During training, we force the output to approximate a target 1D array. The target array contains zeros until two frames after light turns on and ones afterwards. We chose this margin (two frames, 62.5 ms), because we do not expect any meaningful reaction to be faster. We forced the ReactNet output to approximate the target by defining a loss and using stochastic gradient descent to modify the parameters, as is standard in deep learning. The loss is the sum of the cross-entropy losses in each frame between the output of the network and the target. This sum is weighted, with a different weight in each frame, to counterbalance the bias of the training data (e.g. in most examples light had not turned on yet in the first frame but had already turned on in the last frame).

As it is standard, we monitor loss on the validation dataset to detect overfitting. We performed early-stopping if validation loss has not decreased for 500 consecutive epochs. In any case, we stopped the training after 2500 epochs. We selected the model that minimised the validation loss.

3.4.9.4 Building a classifier from the trained model

After training, the output of ReactNet approximates the probability of the light being on at each frame. To calculate each prediction, the network can only use information from the past of the trajectory. Therefore, if ReactNet outputs that it is likely that the light is on at a given frame, it is because the trajectory up to that point shows evidence of the light being on; i.e. some reaction (direct or indirect) to the light by the animal.

We built classifiers from the output of ReactNet by choosing a threshold.

Any threshold that we choose classifies the frames into frames where the future trajectory shows that the animal reacted ("having reacted") and frames where the past trajectory shows that the animal did not react ("not reacted"). As the output grows with the frame, any frame classified as "having reacted" is followed only by frames with the same classification. We consider the first of such frames as the frame where the animal reaction happened. If no such frame exists, or if it occurs later than 128 frames (4 s) after the light onset, we conclude that the animal did not react during the trial.

3.4.9.5 Using the classifier on the information aggregation experiments

To obtain the reaction times used in the results of this chapter, we applied the trained ReactNet to all trials of the experiments. To obtain data for this application, we cut all trajectories using a fixed window starting 15 frames before the light onset and with a total length of 143 frames. Even though only reactions detected after the light onset were considered, we still fed ReactNet data from some frames before the light onset as they have information that can be accumulated by the GRU to improve performance. Supplementary Figure A.13 shows the reaction probabilities for all fish and all trials of the experiments. In this case we chose a probability threshold of 0.8. We discarded the reactions that occurred before the light onset.

3.4.10 Statistics

We looked at the behaviour of the fish along time by taking the mean and 95% confidence interval of the mean of a given variable x. We denote the mean of the variable x for all animals in a given time as $\mu(x)$. We computed the 95% confidence intervals by bootstrapping with replacement with 1000 repetitions. We used the Seaborn (v0.11.1) Python library (Waskom et al., 2017) implementation for the computation of the mean and the confidence intervals. Note that in Figures 3.22 the mean and confidence interval are computed across all fish and time steps with the same fraction of fish that reacted, Φ_{reacted}.

To compare the behaviour of the fish at different moments in the video or in different experimental conditions we computed the mean of a given dynamical variable in a time interval in the video. We denoted the mean of the variable x in a time interval as $\langle x \rangle$. We used the Pandas (v1.2.1) python library (McKinney et al., 2010) for the computation of such means.

To test the significance of the differences that we encountered we performed standard permutation tests (Ernst, 2004). Depending on whether the data points were matched-pairs or not we used paired permutation tests or no-paired permutation tests respectively. We always reported the tested statistic, one-tailed

p-values, and the number of data points. We describe below the details of these tests.

3.4.10.1 Paired permutation tests

To test whether the mean of the differences of a set of paired data points was statistically different than 0 we performed the following permutation test.

Let $X = \{(x_a, x_b)\}$ be a set of pairs of observations of the variable x measured in two different conditions a and b (or two different time intervals). The difference of the variable for each pair is $D(x) = x_b - x_a$. We define the statistic $\overline{D(x)}$, i.e. the mean difference of the variable x between the two conditions.

The null hypothesis assumes that the value of $\overline{D(x)} = 0$ for the set of pairs of observations in X. The alternative hypothesis assumes that the true value of $\overline{D(x)}$ is not zero. The alternative hypothesis can take one of two forms depending on whether $\overline{D(x)} > 0$ or $\overline{D(x)} < 0$.

Let $n(X)$ be the number of points in the set X. We define X^* as a set of pairs of observations of size $n(X^*) = n(X)$ computed by exchanging the order of each element with probability 0.5. Lets define $\overline{D(x^*)}$ as the mean difference of the variable x for a set of pairs X^*.

To test the null hypothesis we generated M different realizations of X^* and defined the p-value of the test the fraction of realizations whose value for the statistic $\overline{D(x^*)}$ was more extreme than the value of the statistic for X, $\overline{D(x)}$.

For the tests in Section 3.2 we used $M = 10000$ and we always report $\overline{D(x)}$, p, and $N_a = n(X_a)$ and $N_b = n(X_b)$ for each case. We report values of $p < 0.001$ as such and give the actual value when $p >= 0.001$.

3.4.10.2 No paired permutation tests

To test whether the means of two sets of data points of the same variable for two conditions were statistically different we designed the following permutation test.

Let $X_a = \{x_a\}$ be a set of observations of the variable x measured in the condition a and $X_b = \{x_b\}$ a set of observations of the same variable measured in the condition b. We define the statistic $D(\overline{x}) = \overline{x_a} - \overline{x_b}$, i.e. the difference of the means of the variable x between the two conditions.

The null hypothesis assumes that $D(\overline{x}) = 0$ for the sets of observations X_a and X_b. The alternative hypothesis assumes that the true value of $D(\overline{x})$ is not zero. The alternative hypothesis can take one of two forms depending on whether $D(\overline{x}) > 0$ or $D(\overline{x}) < 0$.

Let X^* be the set of observations $X_a \cup X_b$. We define the sets of observations X_a^* and X_b^* as random partitions of X^* of sizes $n(X_a^*) = n(X_a)$ and $n(X_b^*) = n(X_b)$ respectively. And consequently, we define $D(\overline{x^*})$ as the difference of the mean for the sets of observations X_a^* and X_b^*.

To test the null hypothesis we generated M different realizations of X_a^* and X_b^* and computed the p-value of the permutation test as the fraction of realizations whose statistic $D(\overline{x^*})$ was more extreme than the statistic of X, $D(\overline{x})$.

For the tests in Section 3.2 we used $M = 10000$ and we reported $D(\overline{x})$, p, and $N_a = n(X_a)$ and $N_b = n(X_b)$ for each case. We report values of $p < 0.001$ as such and give the actual value when $p >= 0.001$.

3.4.11 Deep neural networks models training and testing

3.4.11.1 Prediction problem

As in (Heras et al., 2019), we aimed to solve the following classification problem: Given the dynamical variables of the focal fish and its n nearest neighbours ($\mathcal{I}$), does the focal fish turns left or right after 500 ms? We trained deep neural network models to solve this problem.

We distinguish two types of input variables in our models, asocial and social variables. As asocial variables, α, we considered the focal speed, $v = \|\vec{v}\|$ and its tangential and normal accelerations, $a_{\parallel}$ and $a_{\perp}$. As social variables, σ_i, we considered the relative location, (x_i, y_i), the velocity $\vec{v}_i$, and acceleration $\vec{a}_i$ of each neighbour $i \in \mathcal{I}$. Note that v_i is the norm of the velocity vector of the neighbours, $\|\vec{v}_i\|$. In the deep attention network models we also used the norm of the neighbours acceleration vector, $a_i = \|\vec{a}_i\|$. Note that since we are only interested in understanding the uninformed fish behaviour, the asocial variables α will be only of the uninformed fish, however the social variables σ can be of informed or uninformed neighbours.

The final prediction was the probability of the focal fish turning right, and we computed it using the logistic function $p = 1/(1+e^{-z})$, where z is the output logit of the networks.

3.4.11.2 Deep networks

We considered two deep networks models, the *interaction* and *attention* networks as defined by Heras et al. (2019). Here we briefly explain the functional form of the logit of both networks.

The interaction network logit was calculated from asocial, α, and social variables of the n closest neighbours, $\{\sigma_i, i \in \mathcal{I}\}$, as

$$z = I\left(\alpha, \{\sigma_i\}\right) = \Gamma\left(\sum_{i\in\mathcal{I}} \Pi_I\left(\alpha\sigma_i\right)\right), \tag{3.3}$$

where Π_I is a 3-layer fully connected network with 128 neurons in its output layer. Γ is a 1-layer fully connected network with a single neuron in its output layer (see Heras et al. (2019) for details). Note that the weights of the network Π_I are shared for all n neighbours, which effectively multiplies the training data by n. Also, to multiply the training data by 2 we forced the output of the network to be anti-symmetrical with respect to reflections along the y axis, i.e. the body axis of the focal fish. We did so by anti-symmetrization of I,

$$z = I(\alpha, \{\sigma_i\}) - I(\alpha^*, \{\sigma_i^*\}), \tag{3.4}$$

where the star superscript indicates the reflection along the y axis, which corresponds to a switch of sign of the x components.

The logit of the attention network can be written as

$$z = \sum_{i=1}^{n} \Pi_A(\alpha, \sigma_i) \frac{W(\alpha^{(w)}, \sigma_i^{(w)})}{\sum_j W(\alpha^{(w)}\sigma_j^{(w)})}. \tag{3.5}$$

The function Π_A has the same structure as Π_I but it has a single neuron in the output layer and we anti-symmetrize it. The function W has the same structure as Π_A except that we symmetrize it with respect to reflections on the y axis and it has an exponential function after the output layer. Note that the

superscript (w) in the input variables of the function W indicates that the set of asocial and social variables can be different for the function Π_A and the function W.

3.4.11.3 Model training

To train the models we followed standard procedures in binary classification problems. We trained the models to estimate the probability p_i of turning right of each uninformed focal fish i by minimizing the cross-entropy loss,

$$\mathcal{L} = -\frac{1}{N_b}\sum_{i=1}^{N_b} log(p_i^*), \tag{3.6}$$

where N_b is the number of data points in the minibatch, and $p_i^* = p_i$ if the real turn was also to the right, or $p_i^* = 1 - p_i$ if the real turn was to the left. We minimized the loss using Adam (Kingma & Ba, 2014). We stopped training if validation loss did not reach a new minimum for 10 epochs and did increase 25% from the current minimum, or after 100 training epochs. In the attention network, we annealed the learning rate from 10^{-4} to 10^{-5}, using a batch size of 500. In the interaction network, we annealed the learning rate from 5×10^{-5} to 10^{-5} and trained with a batch size of 200.

Note that in our analysis we were interested in the behaviour of the uninformed fish. Thus, we trained the networks to predict only the turning side of the uninformed fish.

3.4.11.4 Model testing

We wanted to find the set of input variables that better predicted the turning side of the uninformed fish in test data. To test the models we followed three steps for each choice of the input variables. First, we trained 6 models with different initializations of the networks weights. We call this step the training step. Second, we selected the 3 best performing models on a different data set to the one used in the previous step. We call this one the ranking step. Finally, to improve the models' calibration, we created an ensemble of the 3 best performing models and computed its mean loss and accuracy in a different test dataset. We

call this last step the test step. The losses and accuracies reported in Tables 3.1 and 3.2 are the mean losses and accuracies of the ensemble in the test step.

3.4.11.5 Data splits

We used trajectories from the experiments (see Section 3.4.3.2). In particular, we used trajectories from the first 2 min interval, where the uninformed fish were swimming without informed fish, and trajectories 5 s around the onset for each trial in the second 7 min interval, where the uninformed fish were swimming mixed with the informed fish. In total we used trajectories of 20 videos of 2 min and 159 trials 10 s around the onset.

For the training step we used the trajectories from 100 trials and all the trajectories from the 2 min videos. Since the amount of data in the 2 min videos was higher than in the 100 trials, we up-sampled the data in the 100 trials 5 times. This ensured that enough data points were from frames where the uninformed fish were swimming together with the informed fish. For the ranking step we used 10 different trials not used in the training set. Finally, for the test step we used the remaining 49 trials not used in the two previous steps.

3.4.11.6 Data pre-processing

We interpolated linearly the very little missing data in the trajectories due to the video-tracking. We normalized all the trajectories by translating them so that the center of the experimental arena was at (0, 0), and scaled them so that the radius of the arena was 1. To reduce noise while preventing contamination by any future information, we smoothed the trajectories using a 5-frame half-Gaussian kernel with $\sigma = 1$ frame. We computed the velocity and acceleration by finite differences, using only current and past frames. To avoid direct border effects due to the limited size of the experimental arena, we removed all data points where the focal fish were further away than 80% for the radius. In the training step, we divided each trajectories set of a 2 min video or of a trial into two parts to obtain the training and validation sets (90%/10%). We used the validation set for the early stopping of the training.

For all the steps, in each frame, and for each focal fish we found the n nearest neighbours. Then, we computed the focal fish velocity and acceleration, and the relative position, velocity and acceleration of each of the n neighbours. Note that while we compute all the variables during the data pre-processing, we only pass to the network the necessary variables depending on the input variables choice. Finally, we computed the turning side of the focal fish, left or right, after N_f frames in the future. In this chapter we only show results for models trained with $n = 9$ and $N_f = 16$, which corresponds to 500 ms. Note that during the ranking and test steps we used $n = 14$ to consider all the neighbours. This was possible thanks to the modular structure of the deep attention networks.

3.5 Author contributions and acknowledgements

The work of this chapter is inspired by the results published in Heras et al. (2019) of which Francisco Romero Ferrero is an author. The data and results of this chapter are unpublished results of a project in progress.

Francisco Romero Ferrero (F.R.F.), Francisco J.H. Heras (F.H.) and Gonzalo G. de Polavieja (G.G.d.P.) devised the project presented in this chapter and analysed the data. G.G.d.P. supervised the project. F.R.F. and G.G.d.P. designed the experiments with input from F.H. F.R.F. built the zebrafish setup with help from P. Cariço (P.C.) from the Champalimaud Scientific Hardware platform. F.R.F. designed the experimental procedures and performed the experiments. F.R.F. tracked the videos and curated the data. F.R.F. wrote the code of the idmatcher.ai tool. F.H. and F.R.F. wrote the code of the TrajectoryTools. F.H. and F.R.F wrote the code for the deep attention networks. F.H. wrote the code for the ReactNet network. F.R.F wrote the code for the analysis of the results of this chapter with help from F.H. F.R.F wrote this chapter with comments from F.H. and G.G.d.P.

We thank P.C., F. Carvalho and A. Silva from the Champalimaud Scientific Hardware platform for the design and construction of the feeder device and assistance with the HARP Behaviour controller; and I. Campos, J. Moteiro, A.C. Cabrera and M. Sampaio from the Champalimaud Fish platform for animal husbandry.

This study was supported by Congento LISBOA-01-0145-FEDER-022170, three NVIDIA GPU grants (to F.H. and G.G.d .P.), the grant PTDC/NEU-SCC/0948/2014 (to G.G.d.P.) and the fellowship PD/BD/105946/2014 (to F.R.F.) from the Portuguese Fundação para a Ciência e a Tecnologia, and by the Champalimaud Foundation (G.G.d.P.).

Chapter 4

General discussion

4.1 Brief summary of main findings

In this thesis we presented some applications of deep learning tools to study of collective animal behaviour.

In Chapter 2, we presented idtracker.ai (Romero-Ferrero et al., 2019), a multi-animal video-based tracking software that identifies unmarked animals in small and large groups using convolutional neural networks (CNNs).

idtracker.ai extracts small images of animals from the video using standard computer vision techniques. Then, it trains a CNN to distinguish images that show a single animal from images that show multiple animals (i.e. crossings). With the images that show single animals it trains a second CNN to identify all the individuals in the video. Importantly, we showed that both CNNs can be trained in a self-supervised manner by automatically extracting training data from the video without manual annotation. We showed that idtracker.ai can track with high accuracy videos of juvenile zebrafish and fruit flies of up to 100 individuals as long as they appear separated from each other in some parts of the video. This proves that CNN can be used to track by identification different animal species from videos recorded in laboratory conditions. Our results pushed forward the applicability of deep learning methods to extract high-quality data from animal collectives.

In Chapter 3 we built on previous results obtained by applying deep attention networks to the trajectories of 100 juvenile zebrafish obtained with idtracker.ai (Heras et al., 2019). These results suggested that zebrafish use a flexible rule to aggregate information from their neighbours, which could allow fish to weigh higher the neighbours that bring new information to the group.

To test this hypothesis, we designed an experiment in which fish in a group had different information about the environment. In particular, we first trained some fish to move fast towards a light when we turned it on (informed fish) and desensitized other fish to ignore the same light (uninformed fish). Then, we performed experiments where informed and uninformed fish swam together and study how the informed fish influenced the behaviour of the uninformed ones after we turned the light on.

We used idtracker.ai to track all videos during training and in the experiments. Then, we used a new tool called idmatcher.ai to detect the experimental group of the fish in the experiments. We also designed a recurrent neural network, ReactNet, to detect the reactions of the fish after the activation of the light, and found that most informed fish reacted before the uninformed ones. Finally, we trained deep attention networks to predict the behaviour of the uninformed fish in general conditions and found that they can predict well the behaviour of the uninformed fish.

We then used the aggregation submodule of the deep attention network to study how the uninformed fish aggregated information from their neighbours. We found that neighbours that reacted after we turned the light on carried a higher weight in the aggregation. In addition, we found that uninformed fish that reacted payed more attention to the informed neighbours that reacted before them, while the uninformed fish that did not react payed less attention to the same informed neighbours. Our results confirmed that a flexible aggregation rule is useful to weigh higher the neighbours that bring new information to the group, and that deep attention networks models can help to extract insight about animal collective behaviour.

Overall, our work proved the study of animal collective can benefit highly from the use of deep learning tools.

4.2 Future perspectives on deep learning for multi-animal tracking

We have shown that CNNs can be used to track by identification animals in small and large collectives of different species. While the location of the animals throughout time can be sufficient to answer many questions of collective behaviour in species like fish, the movement of their body parts can reveal new information about the nature of the interactions. Even in the small zebrafish, tracking the eyes movement and tail bouts could be used to make better predictions on their behaviour and understand how they interact (Heras et al., 2019).

Inspired by deep learning advances of human pose estimation (see Y. Chen et al. (2020) for a review), recent methods have been proposed to track the body parts of single animals, like DeepLabCut, LEAP and DeepPoseKit (Mathis et al., 2018; Pereira et al., 2019; Graving et al., 2019). Newer versions of these methods can also track the animal body parts of small groups of animals (Pereira, Tabris, et al., 2020). The combination of these methods with multi-animal tracking systems like idtracker.ai can provide location and posture information for all animals in large groups (Graving et al., 2019).

idtracker.ai uses standard computer vision algorithms to segment the animals from the background that require the users to adjust few parameters using a graphical user interface. While this is sufficient for most videos recorded under laboratory conditions in quasi-planar environments, when animals move in more complex environments (e.g. with heterogeneous backgrounds or 3D environments) detecting each animal along the video with these methods is not reliable. Deep learning network architectures have been proposed to detect, track and identify multiple objects (Y. Xu et al., 2020). Typically the different modules of these network architectures are pre-trained on popular image datasets like ImageNet (Deng et al., 2009), CIFAR (Krizhevsky & Hinton, 2009) or COCO (Lin et al., 2014), that include objects similar to the ones to be tracked. Then, with few labeled data these networks can be trained to track multiple objects from videos. However, it is unclear how well they can work for videos recorded under laboratory conditions, due to the differences in the appearance of the video frames. Future research should explore more closely how new advances using deep learning to solve the multiple objects tracking and segmentation task can be applied to track animals in videos (Luo et al., 2020). We believe that in videos of moving animals self-supervised learning strategies will allow to segment the animals from the background without the need of manual annotations or user-defined parameters (Pathak et al., 2017; Wichert, 2019).[1]

Identifying animals from image-data extracted from quasi-planar environments, typical in videos recorded under laboratory conditions, is simpler than in more complex environments. In the second case animals can show different views of their body and the appearance might change due to variations in the

[1] `https://gitlab.com/polavieja_lab/flowseg`

illumination conditions. Recently, Vidal et al. (2021) extensively reviewed different situations in which animal identification is necessary and explained in detail the state-of-the-art deep learning methods used for animal identification. The authors points out that typically metric learning approaches work well for animal identification (see for example Ferreira et al. (2019) and Bozek et al. (2021)). As we suggested in Section 2.3.2.2, new multi-animal tracking systems could incorporate a combination of metric learning and classification (Hsu et al., 2019) to identify the animals in the video even when not all animals are visible during the video.

The great success of deep learning in computer vision (LeCun et al., 2015; Liu et al., 2020), together with the publicly available deep learning computing libraries (Abadi et al., 2015; Paszke et al., 2019), and multiple ready-to-use neural network implementations, are encouraging scientists and software developers to create new all-in-one algorithms that can provide scientists with more and better data of the animals behaviour. For example, Marks et al. (2020) recently proposed a system that combines multiple standard deep computer vision methods to segment, identify, and estimate the pose of multiple animals from a single-camera-view video, the SIPEC system. However, this method still requires manual annotations for both segmentation and animal identification.

In our work, we forced idtracker.ai to be independent of manual annotations by using a self-supervised approach. We noted that the current state-of-the-art solution for the off-line Multiple Object Tracking and Segmentation challenge (MOTSChallenge2020; Dendorfer, Osep, et al. (2020)), ReMOTS, also proposed a self-supervised approach (Yang et al., 2020). Similar to idtracker.ai, ReMOTS automatically extracts data from the video by using spatiotemporal constraints of the video, i.e. (i) objects in the same frame must have different identities, and (ii) images in the same individual fragment belong to the same object., to train the identification CNN. Interestingly, ReMOTS automatically sets the optimal hyperparameters of the algorithm to track the multiple objects using the statistics of the video. In addition, differently than idtracker.ai, ReMOTS does not require having all individuals visible during the video as it uses a triplet loss to train the identification CNN.

We believe that in the future off-line multi-animal tracking systems will segment, track, and identify animals from videos in a completely self-supervised manner without the need of user-defined parameters or manual annotations.

With the increasing development of different tracking methods it is becoming more and more difficult for scientists to evaluate which system is more suitable for extracting animal trajectories from their videos. Some works showed extensive reviews on existing tracking systems that solve similar or related problems (Werkhoven et al., 2019; Tadres & Louis, 2020). Others showed detailed comparisons on computational resources and accuracy performances (Graving et al., 2019; Walter & Couzin, 2021) against similar state-of-the-art systems that were developed before (Mathis et al., 2018; Pereira et al., 2019; Romero-Ferrero et al., 2019). Note that we also did so by comparing idtracker.ai with most of the videos used to validate the previous state-of-the-art on marker-less multi-animal tracking, idTracker (Pérez-Escudero et al., 2014). However, we believe that more standardised benchmarks are needed to perform fairer comparisons among the different systems.

Recently some more standardised benchmarks have been proposed for different animal tracking challenges (Muller et al., 2018; Pedersen et al., 2020; Gallois & Candelier, 2021), however, this is not common. Note that we also provide access to the videos generated in idtracker.ai and, in addition, we created an image dataset of individual juvenile zebrafish that can be used to develop and benchmark new training and identification strategies.[2]

As well as the animal behaviour field has taken benefit from the advances in deep computer vision for the development of new animal tracking methods, it is necessary that we also adopt the best practices in code[3] and data sharing[4]. On the one hand, this will allow developers to more easily detect the tracking challenges that need to be solved for animal behaviour. On the other hand, users will more easily be able to select the animal tracking software that best fits their needs.

[2] `https://idtrackerai.readthedocs.io/en/latest/data.html`
[3] `https://paperswithcode.com/`
[4] `https://computervisiononline.com/datasets`

4.3 Future directions of the project

4.3.1 Other measures of social influence

The analysis in Chapter 3 suggests that the normalized aggregation weight of the deep attention networks (Heras et al., 2019) could be a potential measure of social influence. Others before have proposed different ways to measure causal relationships between the social interactions of animals in collectives (Strandburg-Peshkin et al., 2018). Some of them include temporal cross-correlations (Nagy et al., 2010; Jiang et al., 2017; Lecheval et al., 2018), information theoretical measures (Pilkiewicz et al., 2020), or extreme-event synchronization (Quiroga et al., 2002). Mwaffo et al. (2017) do a detailed review of these methods and combine them to detect leaders in Vicsek self-propelled particle models (Vicsek et al., 1995) and with data driven models (Gautrais et al., 2012). Further analysis could explore whether the results about information aggregation obtained with the deep attention network are comparable to these other measures.

Recently, Nauta et al. (2019) proposed a Temporal Causal Discovery Framework (TCDF) that uses attention-based CNNs in combination with a causal validation step for discovering causal relationships in observational time series data. The authors test their method on financial and neuroscience benchmarks. Future work could also explore the applicability of this framework on collective behaviour experiments similar to ours.

4.3.2 The impact of the pairwise interactions

While the aggregation submodule of the deep attention network can provide insight about how animals aggregate information from their neighbours, we must remember that the final turning probability of the focal fish depends also on the pair-interaction submodule, i.e. on how much each neighbour indicates that the focal fish will turn left or right. Further analysis should investigate the role of the pair-interaction submodule to understand more completely the interaction rules that allow the uninformed fish to follow the informed fish that reacted before them.

As shown in (Heras et al., 2019), from the output of the pair-interaction submodule we can define attraction and alignment scores. An analysis of these scores over time could provide more insight about how the uninformed focal fish moves in the group in order to get closer to the light. However, computing these scores from real data poses some challenges. First, the scores are computed by varying the orientation (unit velocity vector) of the neighbour at each time step instead of using the actual orientation of the animal. This requires multiple runs of the network for the computation of the score for each neighbour of each focal, which becomes computationally very expensive to compute in real trajectories. Second, it is unclear how other variables, like the neighbour acceleration, should be considered as currently the scores are defined using the unit velocity vector. Assuming that these scores depend only on the orientation of the neighbour fish, a simple solution would be to compute the scores for a generic neighbour, like done in Heras et al. (2019) and then use interpolation techniques to determine the score for each neighbour of each focal when using real trajectories.

Future work could explore other definitions for the notions of attraction and alignment that can be efficiently computed in real trajectories. In our experiments, the exploration of the score for each neighbour of each uninformed focal fish could provide more insight about how the uninformed fish positioned themselves in the group to move towards the light.

4.3.3 Generative models of fish trajectories

In our work, we have focused on training models to predict the future turning side of the fish. This is a simplified version of the real animals' behaviour that has the benefit of being very low dimensional (1 output), which still allows defining classical notions of interaction rules. However, this approach is likely to miss other relevant aspects of the fish behaviour. For example, in our experiment fish sometimes accelerate forward without turning. A predictive model of the future turning side might not be capturing the interaction rules that lead to these types of behaviours.

The deep attention networks that we used can be easily adapted to predict the future position of the fish. This would allow potentially capturing finer details about the interaction rules. In addition, predicting the future position of the

fish has the benefit of being able to generate trajectories. Costa et al. (2021) followed this approach and trained similar models using reinforcement learning. They trained the models to simulate some patterns of fish schools observed in the nature. The analysis of the submodules of the deep attention network allowed them to get insight about the interaction rules of the simulated agents and found similar rules to the ones previously obtained (Heras et al., 2019).

A step further would be to use these models to generate fish trajectories that resemble the real trajectories of the animals in our videos. In the deep learning field this approach is known as imitation learning (Hussein et al., 2017).

Most approaches in deep imitation learning focus on minimizing point-wise metrics, e.g. predicting the behaviour n-steps into the future (similar to our approach). Im et al. (2020) explored this idea to create models of fruit fly behaviour (*D. melanogaster*) and also proposed other metrics that compare the distribution of scalars about the animal behaviour. The authors point out that n-step errors are high for all their models, likely due to the inherent unpredictability of the animal behaviour (Honegger & de Bivort, 2018). Note that while fruit files show different modalities of social behaviour, they do not show a clear self-organized behaviour as schooling fish do. During schooling all fish are very polarized and the behaviour of one animal in the group can be more predictable. Because of this, one could expect lower n-steps errors when predicting the fish behaviour in collectives than when predicting behaviour of fruit flies. In fact, we remember that with we obtained accuracies of $73\% - 86\%$ depending on the video, and suggested that these differences could be due to the different levels of organization of the collective.

Future work should investigate the possibilities of training deep attention networks (Heras et al., 2019) to generate real fish trajectories using imitation learning with distribution metrics and compare whether the interaction rules found with this method are consistent with the ones obtained with point-wise metrics.

4.4 Concluding remarks

In this thesis we have explored the applicability of deep learning tools to the study of collective animal behaviour. We have shown that with a proper understanding of the problem at hand, deep neural networks (DNNs) can be highly beneficial for the extraction of data in animal collectives (Chapter 2 and Romero-Ferrero et al. (2019)) and for the study of their interactions (Chapter 3 and Heras et al. (2019)). Since the projects presented in this thesis started, the rapidly evolving field of deep learning has not stopped providing solutions to a wide variety of problems in many disciplines (LeCun et al., 2015; Alom et al., 2018). Deep learning and other machine learning techniques are continuously being applied to the study of collective phenomena at different scales (Cichos et al., 2020). While DNNs have a super-human learning capacity, their black-box nature makes understanding the decision taken by a DNN in a given task a very challenging problem. This challenge has started a whole new field of research that focuses on making DNNs more interpretable so that we can extract more about a given problem by using them (Xie et al., 2020). Interpretability has been and continues being an important challenge in machine learning (Rudin et al., 2021). We believe that the field of animal behaviour must follow closely the advances in interpretable artificial intelligence (Linardatos et al., 2021). Interpretable DNN-based models can be very complex, highly predictive and, at the same time, they can provide insight about the behaviour of a given system. This makes them a very promising tool for studying the highly complex system that an animal collective is.

Appendix A

Supplementary Materials

A.1 Chapter 2 Supplementary Tables

Advanced parameter	Description
`GUI_MINIMUM_HEIGHT`	Minimum height of the GUI.
`GUI_MINIMUM_WIDTH`	Minimum width of the GUI.
`NUMBER_OF_ANIMALS_DEFAULT`	Default number of animals in GUI text box.
`AREA_LOWER`	Minimum area threshold allowed in GUI slider.
`AREA_UPPER`	Maximum area threshold allowed in GUI slider.
`MIN_AREA_DEFAULT`	Default minimum area threshold.
`MAX_AREA_DEFAULT`	Default maximum area threshold.
`RES_REDUCTION_DEFAULT`	Default resolution reduction factor in GUI.
`MIN_THRESHOLD`	Minimum intensity threshold allowed in GUI slider.
`MAX_THRESHOLD`	Maximum intensity threshold allowed in GUI slider.
`MIN_THRESHOLD_DEFAULT`	Default minimum intensity threshold.
`MAX_THRESHOLD_DEFAULT`	Default maximum intensity threshold.

Table A.1. **List of advanced parameters for the GUI.** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
`SIGMA_GAUSSIAN_BLURRING`	Standard deviation for the gaussian kernel applied during the image blurring robustness tests.
`FRAMES_PER_EPISODE`	Number of frames processed in each thread in the parallel processes.
`STD_TOLERANCE`	Number of standard deviations used for the model area threshold.
`BKG_SUB_PERIOD`	Period with which a frame is selected to compute the background model (P_{bkg}).
`NUM_OF_JOBS_BKG_SUB`	Number of Python worker processes for the background subtraction parallel loop.
`NUM_OF_JOBS_SEG`	Number of Python worker processes for the segmentation parallel loop.
`NUMBER_OF_JOBS_ID_IMAGES`	Number of Python worker processes for the computation of the identification images parallel loop.
`SAVE_PIXELS`	String indicating where to save the pixels, wither `'DISK'` or `'RAM'`.
`SAVE_SEG_IMAGE`	String indicating where to save the segmentation bounding box image, either `'DISK'` or `'RAM'`.

Table A.2. **List of advanced parameters of the animals detection processing step.** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
MIN_NUM_CROSSINGS	Minimum number of crossing blobs to train the cdCNN.
MAX_IMAGES_PER_CLASS	Maximum number of images per class for the training the cdCNN.
LEARNING_RATE_DCD	Learning rate to train the crossing detector cdCNN.
VAL_PROPORTION	Percentage of images used for validation from the whole training dataset.
BATCH_SIZE_DCD	Batch size used during the training of the cdCNN.
BATCH_SIZE_PRED_DCD	Batch size used during evaluation with the cdCNN.
LEARNING_DIFFERENCE_DCD	Fraction of the validation loss difference in consecutive epochs.
OVERFITTING_COUNTER_DCD	Threshold for the number of epochs of overfitting after which the training of the cdCNN stops.
NUM_OF_EPOCHS_DCD	Threshold for number of epochs after which the training of the cdCNN stops.

Table A.3. **List of advanced parameters for the crossing detection processing step.** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
MIN_NUM_OF_FRAMES_GF_TRAINING	Number of images in the smallest individual fragment of a global fragment to be considered for training.
MAXIMAL_IMAGES_PER_ANIMAL	Maximum number of images per animal to be used for the training of the idCNN.
RATIO_OLD	Fraction of images identified in past accumulation steps that will be used for training.
RATIO_NEW	Fraction of images identified in the current accumulation step that will be used for training.
CERTAINTY_THRESHOLD	Threshold of the certainty above which an individual fragment is considered correctly identified.
MIN_RATIO_ PARTIAL_ACCUMULATION	Threshold of the percentage of images used for training above which the training and identification loop can accumulate global fragments using the partial strategy.
LAYERS_TO_OPTIMISE_ ACCUMULATION	String indicating whether to train all the idCNN or only the fully connected layers when using knowledge transfer.
LEARNING_RATE_ IDCNN_ACCUMULATION	Learning rate for training the idCNN.
LEARNING_RATE_ IDCNN_PRETRAINING	Learning rate for training the idCNN during pre-training.
BATCH_SIZE_IDCNN	Batch size during training.
BATCH_SIZE_PREDICTIONS_IDCNN	Batch size during prediction.
LEARNING_DIFFERENCE_1_IDCNN	Fraction of the validation loss difference between consecutive epochs (protocol 1).
LEARNING_DIFFERENCE_2_IDCNN	Fraction of the validation loss difference in consecutive epochs (protocol 2).
OVERFITTING_COUNTER_ IDCNN_FIRST_ACCUM	Threshold for the number of epochs for which overfitting is allowed after which the training of the idCNN stops in Protocol 1.
OVERFITTING_COUNTER_IDCNN	Threshold for the number of epochs for which overfitting is allowed before stopping the training.

Table A.4. **List of advanced parameters for the cascade of training and identification protocols processing step (1).** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
NUM_EPOCHS_IDCNN	Threshold for number of epochs after which the training of the idCNN stops.
THRESHOLD_EARLY_STOP_ ACCUMULATION	Threshold for the percentage of images used for training after which the training of the id-CNN stops.
THRESHOLD_ACCEPTABLE_ ACCUMULATION	Threshold for the percentage of images used for training after which the training of the id-CNN is considered to be successful.
MAX_NUM_OF_PARACHUTE_ ACCUMULATIONS	Maximum number of times that the training and identification loop is stated in the Protocol 3.
KNOWLEDGE_TRANSFER_ FOLDER_IDCNN	Path to a idCNN model folder to use for knowledge transfer.
IDENTITY_TRANSFER	Flag indicating whether the identities from the knowledge transfer model should be used to infer the identities of the animals in the current video.
IDENTIFICATION_IMAGE_SIZE	Size of the identification images (optional).
MAX_RATIO_OF_ PRETRAINED_IMAGES	Maximum fraction of images that should be used for pre-training.

Table A.5. **List of advanced parameters for the cascade of training and identification protocols processing step (2).** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
FIXED_IDENTITY_THRESHOLD	Probability threshold above which an identity is fixed and it cannot be modified during the post-processing.
VEL_PERCENTILE	Percentile of the velocity used to computer the velocity threshold to detect impossible velocity jumps.

Table A.6. **List of advanced parameters for the residual identification and post-processing processing steps.** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Advanced parameter	Description
`DATA_POLICY`	String indicating the data stored after the tracking process ends.
`SAVE_AREAS`	Boolean indicating whether to save the areas of the segmented and identified blobs in the file output.
`INDIVIDUAL_VIDEO_WIDTH_HEIGHT`	Number indicating the width and height of the the individual videos generated with the button "Generate Individual Videos" of the GUI.
`CONVERT_TRAJECTORIES_DICT_TO_CSV_AND_JSON`	Boolean indicating whether to save the trajectories also with CSV and JSON files.

Table A.7. **List of advanced parameters for the output data.** The left column shows the name of the parameter as it should be in the `local_settings.py` file. The right column shows a brief description of the parameter.

Crossing detector (cdCNN)

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	max pooling	-	-	(2,2)
3	convolution - ReLu	64	(5,5)	1
4	max pooling	-	-	(2,2)
5	fully connected - ReLu	100	-	-
6	fully connected - softmax	2	-	-

Identification convolutional neural network (idCNN)

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	max pooling	-	-	(2,2)
3	convolution - ReLu	64	(5,5)	1
4	max pooling	-	-	(2,2)
5	convolution - ReLu	100	(5,5)	1
6	fully connected - ReLu	100	-	-
7	fully connected - softmax	group size	-	-

Table A.8. **Convolutional neural networks used in idtracker.ai.** The cdCNN is used to classify images as belonging to a single animal or to multiple animals crossing or touching. The identification convolutional neural network (idCNN) is used to identify individual images.

1 convolutional layer

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	fully connected - ReLu	100	-	-
3	fully connected - softmax	group size	-	-

2 convolutional layers

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	max pooling	-	-	(2,2)
3	convolution - ReLu	64	(5,5)	1
4	fully connected - ReLu	100	-	-
5	fully connected - softmax	group size	-	-

4 convolutional layers

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	max pooling	-	-	(2,2)
3	convolution - ReLu	64	(5,5)	1
4	max pooling	-	-	(2,2)
5	convolution - ReLu	100	(5,5)	1
6	max pooling	-	-	(2,2)
7	convolution - ReLu	100	(5,5)	1
8	fully connected - ReLu	100	-	-
9	fully connected - softmax	group size	-	-

Table A.9. **Identification CNN architectures with variations in the number of convolutional layers part**. Several architectures with variable numbers and shape of convolutional layers have been tested in order to assess the stability of the accuracy in single-image identification.

fc 50 units

Layer	operation	number of units	kernel size	stride
1	convolution - ReLu	16	(5,5)	1
2	max pooling	-	-	(2,2)
3	convolution - ReLu	64	(5,5)	1
4	max pooling	-	-	(2,2)
5	convolution - ReLu	100	(5,5)	1
6	fully connected - ReLu	50	-	-
7	fully connected - softmax	group size	-	-

fc 200 units

Layer	operation	number of units	kernel size	stride
1-5	(see first model)			
6	fully connected - ReLu	200	-	-
7	fully connected - softmax	group size	-	-

fc 100 units → fc 50 units

Layer	operation	number of units	kernel size	stride
1-5	(see first model)			
6	fully connected - ReLu	100	-	-
7	fully connected - softmax	50	-	-
8	fully connected - ReLu	group size	-	-

fc 100 units → fc 100 units

Layer	operation	number of units	kernel size	stride
1-5	(see first model)			
6	fully connected - ReLu	100	-	-
7	fully connected - softmax	100	-	-
8	fully connected - ReLu	group size	-	-

fc 100 units → fc 200 units

Layer	operation	number of units	kernel size	stride
1-5	(see first model)			
6	fully connected - ReLu	100	-	-
7	fully connected - softmax	200	-	-
8	fully connected - ReLu	group size	-	-

Table A.10. **Identification CNN architectures with variations in the number and size of the fully connected layers (fc)**. Several architectures with variable numbers and shape of fully connected layers has been tested in order to assess the stability of the accuracy in single-image identification. The notation *fc n* → *fc m* characterises each architecture according to its fully connected layers before the output layer.

Score	Description
Identification accuracy in protocol cascade (Acc. prot.).	Percentage of images correctly identified considering only the images used for training the idCNN during the cascade of training and identification protocols.
Identification accuracy (Acc.)	Percentage of all individual blobs correctly identified in a given interval of the video. In Supplementary Table A.15 the accuracy at each step of the algorithm considers all the individuals blobs identified in that processing step over all the individual blobs in the video.
Individual identification accuracy (Indv. acc.)	Average ($n = 10$) of the percentage of individual blobs correctly identified for an animal during the whole duration of the video.
% no identified (% no-id.)	Percentage of all individual blobs not identified (the identity was set to 0).
% misidentified (% misid.)	Percentage of identified blobs incorrectly identified.

Table A.11. **List of tracking performance scores.** The right column indicates the name of the score. Between brackets we show the abbreviation it appears in the Supplementary Tables A.12-A.19. The right column describes each score.

Video	Duration	Fps.	Area	Prot.	Crossings	Acc. prot.	**Acc.**	% No-id.	% misid.
2 nacre zebrafish (1) †	15'58"	24	764	2	90	100	**99.991**	0	0.009
7 inbred WIK zebrafish †	08'19"	31	670	2	393	100	**99.997**	0.003	0
8 zebrafish † ∗	05'22"	28	331	2	842	100	**99.969**	0.028	0.003
10 zebrafish	10'03"	32	528	2	311	100	**100**	0	0
10 zebrafish	10'03"	32	551	2	877	100	**99.999**	0	0.001
10 zebrafish	10'10"	32	534	2	667	100	**100**	0	0
5 medaka fish †	21'41"	28	232	2	124	100	**100**	0	0
10 medaka fish †	12'20"	28	199	2	89	100	**100**	0	0
20 medaka fish †	08'20"	29	273	2	597	100	**99.999**	0.001	0
6 Drosophila [4 ♀, 2 ♂] †	07'57"	42	1048	2	306	100	**99.994**	0.006	0
8 Drosophila (♀) (1) †	03'07"	28	819	2	199	100	**99.986**	0	0.014
8 Drosophila (♀) (2) †	14'58"	28	750	2	298	99.882	**99.760**	0.104	0.136
2 agouti mice †	05'19"	49	12855	2	93	99.932	**99.910**	0	0.090
2 black mice (1) †	12'24"	49	11986	2	200	99.582	**99.576**	0	0.424
2 black mice (2) †	07'06"	49	11401	2	41	99.738	**99.693**	0	0.307
2 black mice (3) †	07'06"	49	12124	2	44	100	**100**	0	0
4 black mice (2)†	33'58"	25	7398	2	351	100	**99.993**	0.002	0.005
4 black mice (3)†	52'54"	24	6440	2	287	97.974	**96.641**	0.032	3.327

Table A.12. **Results of manual validation for videos of small group size.** The first column indicates the number of animals and the animal species. †: video from (Pérez-Escudero et al., 2014). (n): number indicated in (Pérez-Escudero et al., 2014) tables. ∗: video used for the algorithm development. ♀/♂: sex of the animals. Duration: video duration in minutes and seconds. Fps.: Frame rate of the video. Area: mean area of the blobs in pixels. Prot.: protocol used for tracking. Crossings: number of crossings analysed for manual validation. The remaining columns correspond to performance scores as described in Table A.11.

Video	Duration	Fps.	Area	Prot.	Crossings	Acc. prot.	**Acc.** / Indv. acc.	% No-id.	% misid.
60 zebrafish	10'29"	31	325	2	338	99.921	**99.871** / 99.880	0.024	0.106
60 zebrafish	10'30"	31	308	2	277	100	**99.998** / 99.999	0.002	0
60 zebrafish	10'09"	32	303	2	213	100	**100** / 99.999	0	0
100 zebrafish	10'16"	32	316	2	2552	100	**99.977** / 99.989	0.016	0.007
100 zebrafish	10'12"	32	273	2	750	100	**99.999** / 99.999	0.001	0
100 zebrafish ∗	10'10"	32	309	2	628	100	**99.997** / 100	0.002	0.001
38 fruit flies ↓ (2)	10'07"	36	602	2	753	99.996	**99.978** / -	0.002	0.020
60 fruit flies ♀↑ (1)	10'00"	51	372	3	699	100	**99.999** / -	0.001	0
72 fruit flies ↓ (2)	10'00"	36	602	3	165	100	**99.997** / -	0.003	0
80 fruit flies ↓ (1)	10'02"	34	589	3	514	99.925	**99.439** / -	0.420	0.141
80 fruit flies ↑ (2)	10'02"	51	273	3	222	100	**99.891** / 99.046	0.109	0
100 fruit flies ↓ (1)	10'00"	28	558	3	107	99.909	**99.580** / -	0.327	0.093

Table A.13. **Results of manual validation for videos of large group size.** The first column indicates the number of animals and the animal species. (*n*): experimental arena used to record the video (see Section 2.4.3). ∗: video used for the algorithm development. ♀/♂: sex of the animals. ↓ / ↑: whether the camera recorded the animals from a bottom view or a top view. Duration: video duration in minutes and seconds. Fps.: Frame rate of the video. Area: mean area of the blobs in pixels. Prot.: protocol used for tracking. Crossings: number of crossings analysed for manual validation. The remaining columns correspond to performance scores as described in Table A.11.

Video	Duration	Fps.	Area	Prot.	Crossings	Acc. prot.	**Acc.**	% No-id.	% misid.
Compressed videos with bad illumination conditions (difficulties with idTracker Pérez-Escudero et al. (2014))									
10 fruit flies	10'12"	60	420	3	288	100	**100**	0	0
14 ants	13'10"	59	573	2	498	99.989	**99.943**	0.001	0.056
Very low locomotor activity levels for most animals or dead animals (low humidity)									
60 fruit flies	10'14"	51	292	3	-	-	73.720 †	-	-
100 fruit flies (1) ♀	10'14"	50	334	3	-	-	95.579 †	-	-
Some animals show atypical behavior with poses very different to healthy ones (low humidity)									
100 fruit flies (2)	10'00"	34	538	3	217	99.905	**99.277**	0.546	0.177

Table A.14. **Results of manual validation for videos that do not fulfill some of the general video conditions listed in Section2.2.4.1.** The column headers are defined in Table A.12. †: the value is computed as the estimated accuracy after the residual identification block (see Section 2.2.1.5). The first two videos were recorded in a compressed lossy video formats, .avi (FMP4 compression code) and .MOV (avc1), respectively. The video of 10 fruit flies was recorded in a retro-illuminated circular arena with conic walls. During the video a LED turns on and off at regular time intervals. The video of 14 ants has multiple shadows of animals which make the segmentation difficult. Moreover, it is illuminated with direct light that creates reflections on the bodies of the animals. The animals in the video of 60 flies have very low locomotor activity levels and the contrast between the animals and the background is poor. The first video of 100 female flies contains dead animals due to the after effects of the anaesthesia with CO_2. In the second video of 100 flies some animals show atypical behaviours, like rolling on their back, which changes the appearance the animals along the video.

Video	Cond.	Prot.	Prep. time	Prot. 1 time/acc.	Prot. 2 time/acc.	Prot. 3 time/acc.	Postp. time/acc.	Total time
8 zebrafish	raw	2	0:04:18	0:01:09/98.110	0:15:14/99.490	-/-	0:01:48/99.891	0:22:31
	G. blur. (3)	2	0:04:40	0:02:22/90.514	0:16:31/98.876	-/-	0:01:42/99.870	0:25:17
	G. blur.(6)	2	0:04:57	0:01:30/ 52.649	0:16:25/96.772	-/-	0:02:06/99 .191	0:24:59
	Res. red. (3)	2	0:03:40	0:01:16/96.454	0:14:19/99.513	-/-	0:01:45/99.907	0:21:02
	Res. red. (5)	2	0:02:32	0:01:30/95.182	0:08:40/ 98.163	-/-	0:01:36/99.526	0:14:20
60 zebrafish	raw	2	3:11:17	0:10:09/95.494	1:37:34/99.982	-/-	0:22:50/99.997	5:21:52
	G. blur. (4)	2	6:58:23	0:22:12/94.828	1:49:25/99.821	-/-	0:22:15/99 .995	9:32:16
	G. blur. (6)	3	3:54:03	0:12:46/33.890	0:05:48/33.890	15:20:18/89.787	1:29:28/90.629	21:02:24
	Res. red. (2)	2	3:19:34	0:11:36/93.126	1:44:13/99.996	-/-	0:20:24/99.998	5:35:49
	Res. red. (4)	3	2:15:41	0:11:51/65.643	0:04:41/65.643	14:50:19/99.315	0:31:55/99.991	17:54:28
100 zebrafish (3)	raw	2	6:39:32	0:26:30/98.516	4:53:46/99.948	-/-	1:19:47/99.978	13:19:37

Table A.15. **Running times and accuracies at different stages in the algorithm.** See Figure 2.1 for the different processing steps. Video: number of animals and animal species. Cond.: video quality manipulation condition. The number in between parenthesis corresponds to the row number in Tables A.17 and A.18. Prot.: protocol used to track the video. Prep. time: running time up to the fragmentation processing step. Prot. 1 time: running time up to the end of protocol 1. Prot. 2 time: running time up to the end of protocol 2. Prot. 3 time: running time up to the end of protocol 3. Postp. time: running time up to the crossing detection processing step. Total time: total running time of the tracking. The accuracy (acc.) is computed as the percentage of correctly identified blobs at the end of a given processing step considering all the individual blobs in the video.

Video	Knowledge transfer	Prot. time	Prot.	Acc.
8 zebrafish	no	0:24:48	2	99.969
	yes	0:07:52	2	99.904
60 zebrafish	no	2:08:27	2	99.998
	yes	1:18:49	2	99.962
72 fruit flies	no	2 days, 12:08:58	3	99.997
	yes	3:30:25	2	95.584

Table A.16. **Results of manual validation for videos tracked using kowledge transfer**. Video: number of animals and animal species. Knowledge transfer: whether knowledge transfer was used to track the video (see Section 2.2.1.4). Prot. time: running time during the cascade of training and identification protocols. Prot.: protocol used to track the video. Acc.: accuracy as defined in Table A.11. For the zebrafish videos, the knowledge transfer was done from an idCNN of a tracked video of 100 zebrafish. For the fruit flies video, the knowledge transfer was performed from an idCNN of a tracked video of 100 fruit flies. Only the last two fully connected layers of the network were trained when performing knowledge transfer.

Video	Area 1	Area 2	File size (Gb)	Total time	Prot.	Acc.
8 zebrafish	331	599	9.14	0:30:51	2	99.969
	168	361	5.14	0:19:38	2	99.893
	63	193	2.29	0:19:04	2	99.890
	23	100	1.12	0:29:34	2	99.618
	9	61	0.57	0:23:42	2	96.690
	2	28	0.20	4:15:31	3	48.366
60 zebrafish	303	578	228.02	5:18:41	2	100
	163	367	128.26	5:14:47	2	99.998
	64	193	57.00	7:05:52	2	99.998
	28	104	27.92	15:18:29	3	99.989
	12	63	14.25	12:42:58	3	89.187
	3	28	5.13	1 day, 22:33:39	3	19.572

Table A.17. **Results of manual validation for 6 different image resolutions.** Video: number of animals and animal species. Area 1: average number of pixels in an individual blob after the segmentation (see Table 2.2). Area 2: average number of non-black pixels in the identification image. File size (Gb): size of the video file. Total time: running time of the whole tracking process. Prot.: protocol used to track the video. Acc.: accuracy as defined in Table A.11.

Video	σ gaussian kernel	Total time	Prot.	Acc.
8 zebrafish	0.0	0:30:51	2	99.969
	0.5	0:39:00	2	99.876
	1.0	0:23:59	2	99.902
	2.0	0:21:44	2	99.865
	3.0	0:34:09	2	99.684
	4.0	0:28:12	2	99.300
	5.0	2:31:30	3	81.215
60 zebrafish	0.0	5:18:41	2	100
	0.5	8:48:47	2	99.997
	1.0	6:41:33	2	99.998
	2.0	11:23:53	2	99.996
	3.0	11:32:12	2	99.995
	4.0	1 day, 11:39:10	3	98.367
	5.0	22:22:50	3	85.359

Table A.18. **Results of manual validation for 7 values of standard deviation of a Gaussian blurring**. Video: number of animals and animal species. σ gaussian kernel: standard deviation of the Gaussian kernel used to generate the blurring (see Section 2.4.8.2). Total time: running time for the whole tracking process. Prot.: protocol used to track the video. Acc.: accuracy as defined in Table A.11.

.

Video	Compression	File size (Gb)	Total time	Prot.	Acc.
8 zebrafish	raw video	9.14	0:30:51	2	99.969
	H264	3.83	0:34:36	2	99.895
	MPEG4	0.82	0:22:02	2	99.896
60 zebrafish	raw video	228.02	5:18:41	2	100
	H264	95.48	6:55:03	2	99.997
	MPEG4	12.92	6:20:37	2	99.998

Table A.19. **Results of manual validation for 2 video compression encodings formats**. Video: number of animals and animal species. Compression: compression encoding format. File size (Gb): size of the video file. Total time: running time of the whole tracking process. Prot.: protocol used to track the video. Acc.: accuracy as defined in Table A.11.

A.2 Chapter 2 Supplementary Figures

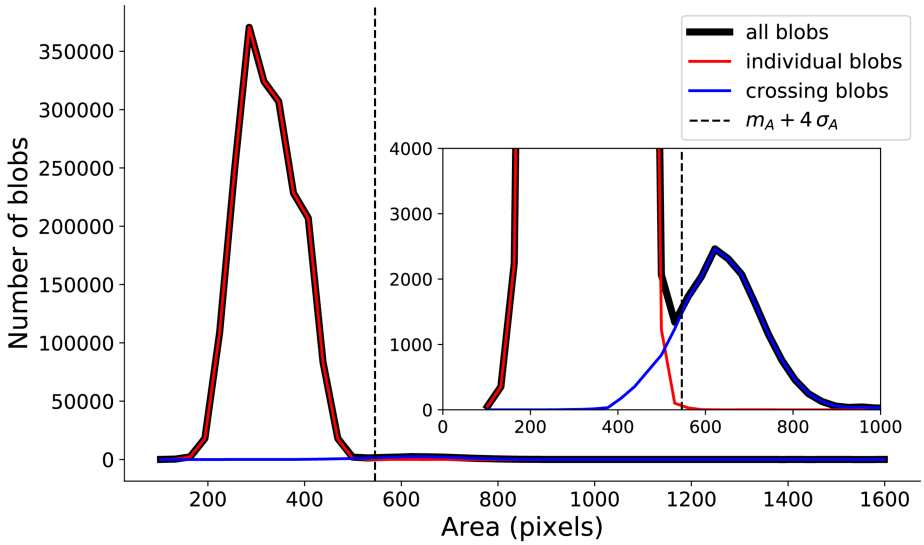

Figure A.1. **Model area heuristic.** Histogram of the area of all blobs in a 100 juvenile zebrafish video (black thick line). The vertical dashed line represents the area threshold above which the heuristic considers a blob to be a potential crossing blob. The red and blue thinner lines are the histogram of the area of the individual and crossing blobs, respectively, as classified at the end of the crossing detection processing step.

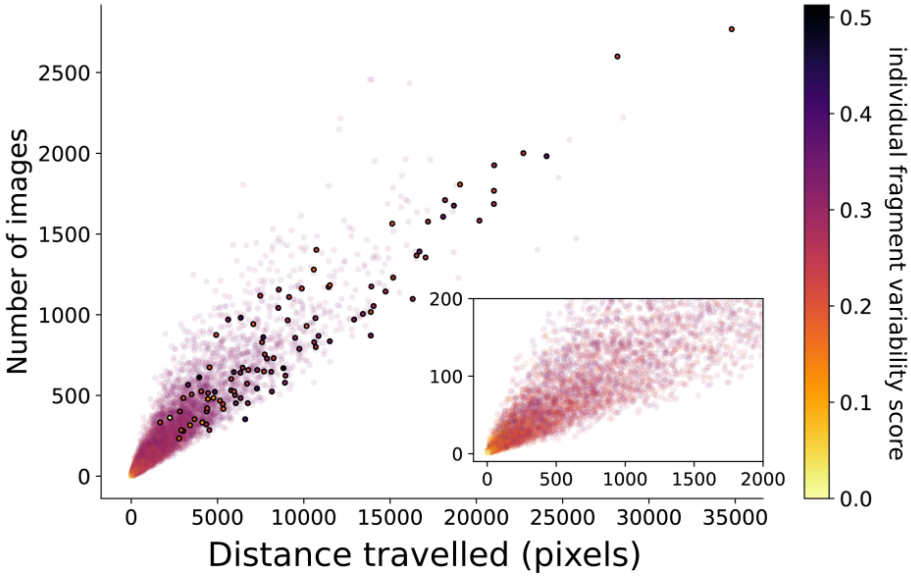

Figure A.2. **Individual fragments distance traveled heuristic.** Each dot represents an individual fragment in a video of 100 juvenile zebrafish. The higher opacity dots represent the individual fragments of the first global fragments selected to start the training identification protocols cascade. The x axis is the distance traveled by the animal in the individual fragment. The y axis is the number of images in the individual fragments. The color encodes the mean of the standard deviation of each pixel across all images in the individual fragment. The inset shows a close up for the individual fragments with less than 200 images and a distance traveled smaller than 2000 pixels.

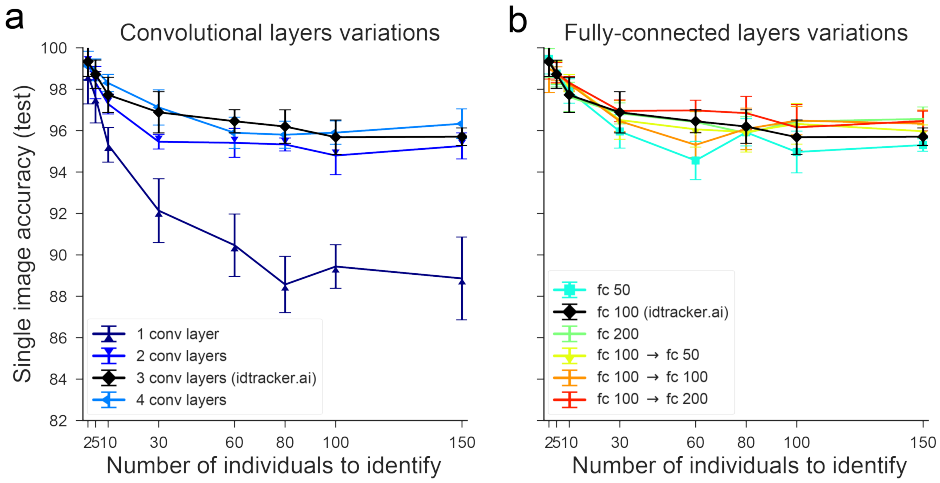

Figure A.3. **Single image identification accuracy for different group sizes and different variations of the identification CNN architectures.** Colored lines with markers represent single image accuracies (mean ± SD, n = 5) for networks architectures with different numbers of convolutional layers (a) (see Supplementary Table A.9) and different sizes and numbers of fully connected layers (b) (see Supplementary Table A.10). The black solid line with diamond markers shows the accuracy for the idCNN of idtracker.ai.

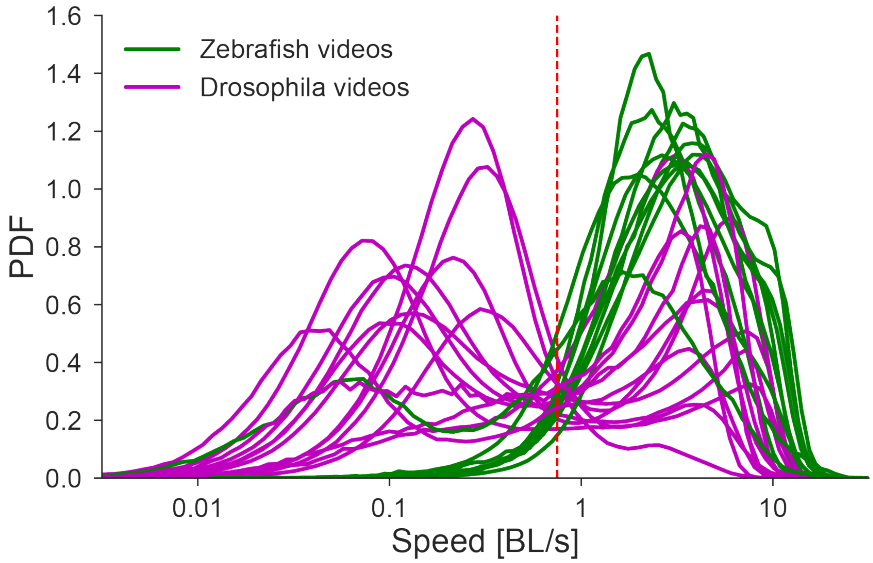

Figure A.4. **Zebrafish and fruit flies locomotor activity.** Speed probability distributions for zebrafish and fruit flies videos. The vertical line indicates the threshold selected to consider that animals were moving.

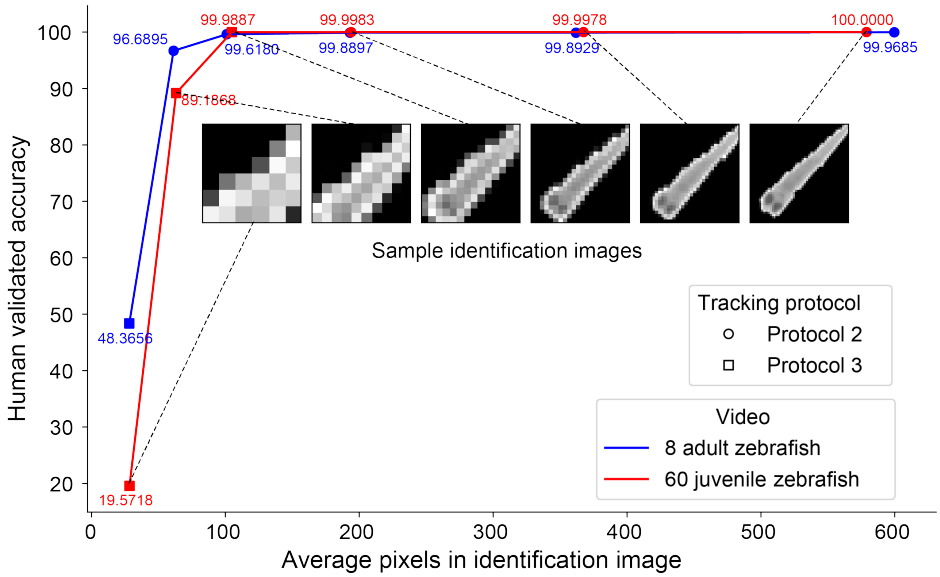

Figure A.5. **Validation accuracy for 6 manipulations of the video resolution.** The x axis indicates the mean number of pixels of the images at the identification stage, i.e. all pixels that were not black in the identification image. See also Supplementary Table A.17.

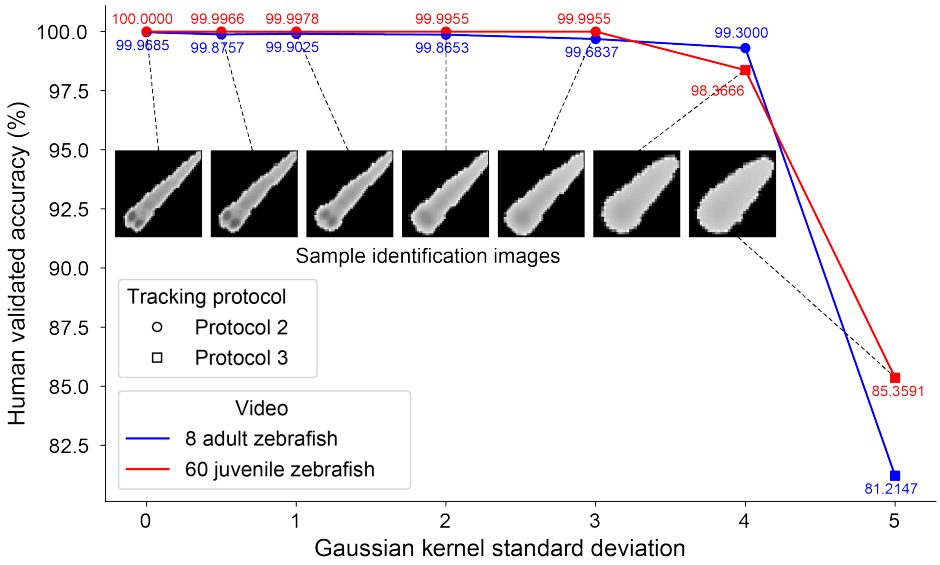

Figure A.6. **Validation accuracy for 7 gaussian blurring manipulations.** The x axis indicates the standard deviation of the gaussian kernel used to perform the image blurring of the video frames. See also Supplementary Table A.18.

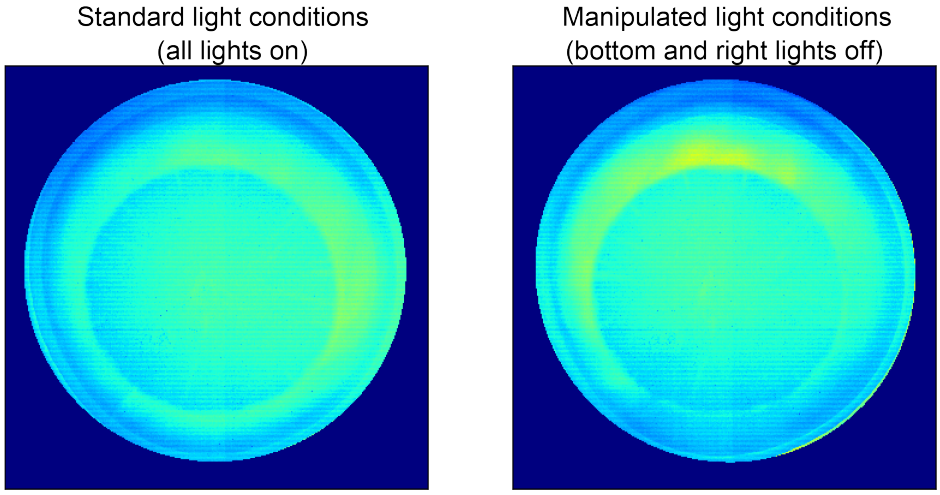

Figure A.7. **Inhomogeneous lighting**. Estimated model of the background corresponding to two different videos 60 juvenile zebrafish. Left: our standard setup (99.99% accuracy). Right: our standard setup switching off the IR LEDs in two walls and covering the light diffuser in the same side with a black cloth (99.96% accuracy).

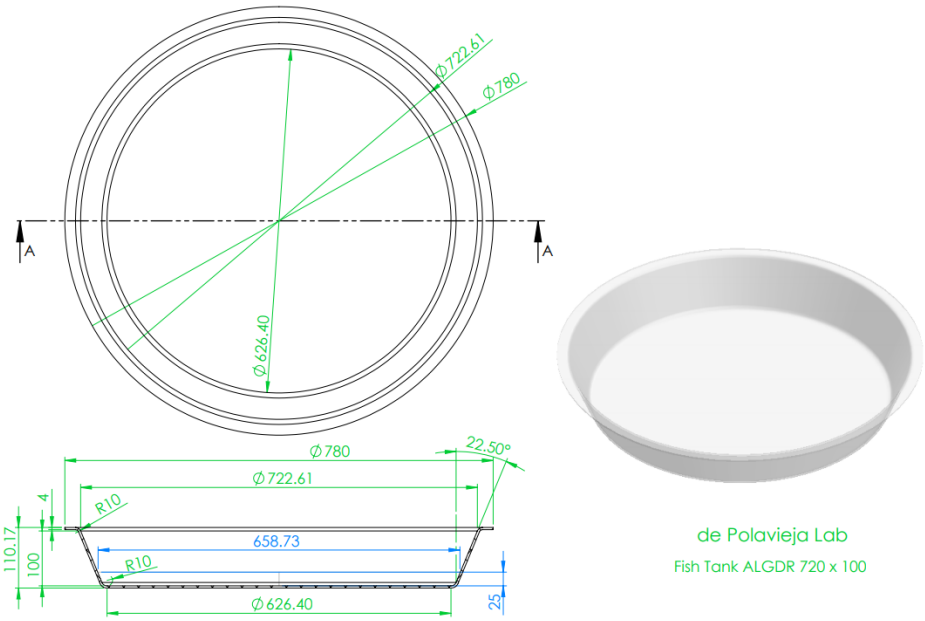

Figure A.8. **Fish tank model diagram.** Model used for the construction of the acrylic tank used for the videos of juvenile zebrafish. Dimensions are given in millimeters.

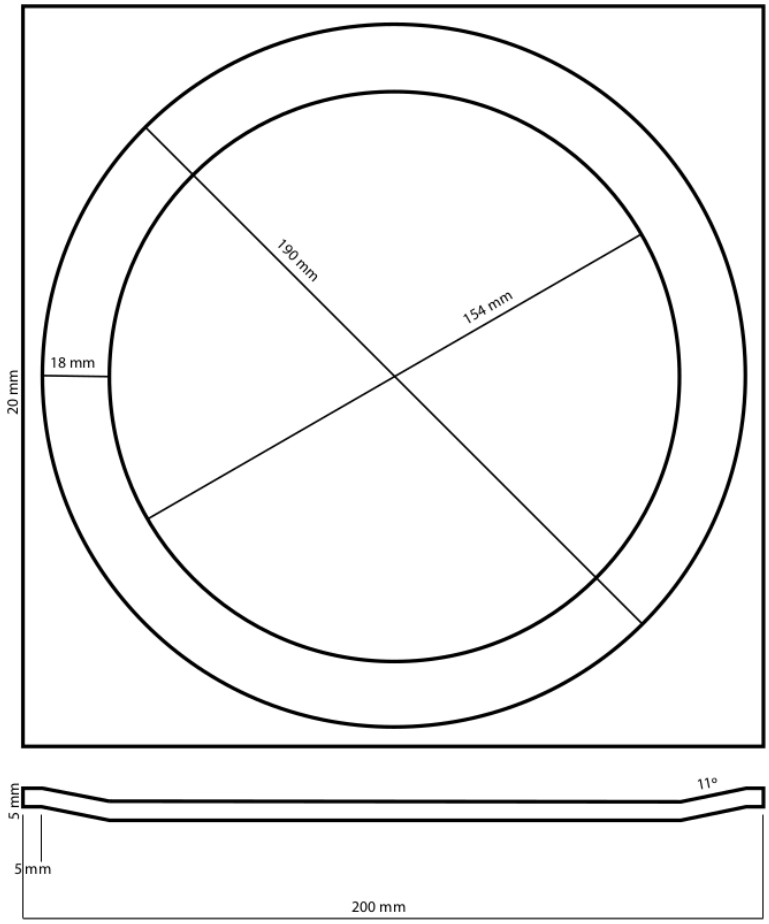

Figure A.9. **Fruit flies arena model diagram.** Model used for the construction of the acrylic arena used for the videos of fruit flies. Dimensions are in millimeters.

A.3 Chapter 3 Supplementary Figures

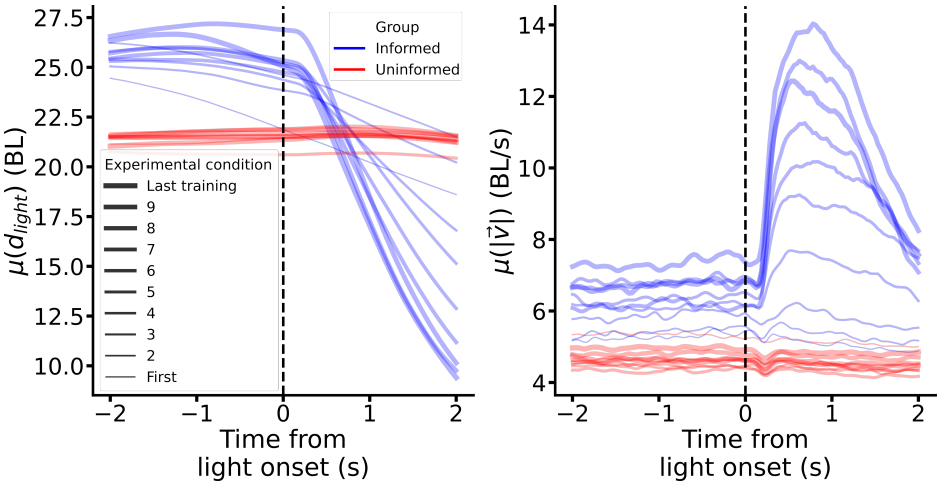

Figure A.10. **Distance to the light and speed for all training sessions. Left.** Mean distance to the stimulus for informed and uninformed fish in each training session. The mean is computed across all animals and trials. **Right.** Mean speed for informed and uninformed fish in each training session. The mean is computed across all animals and trials.

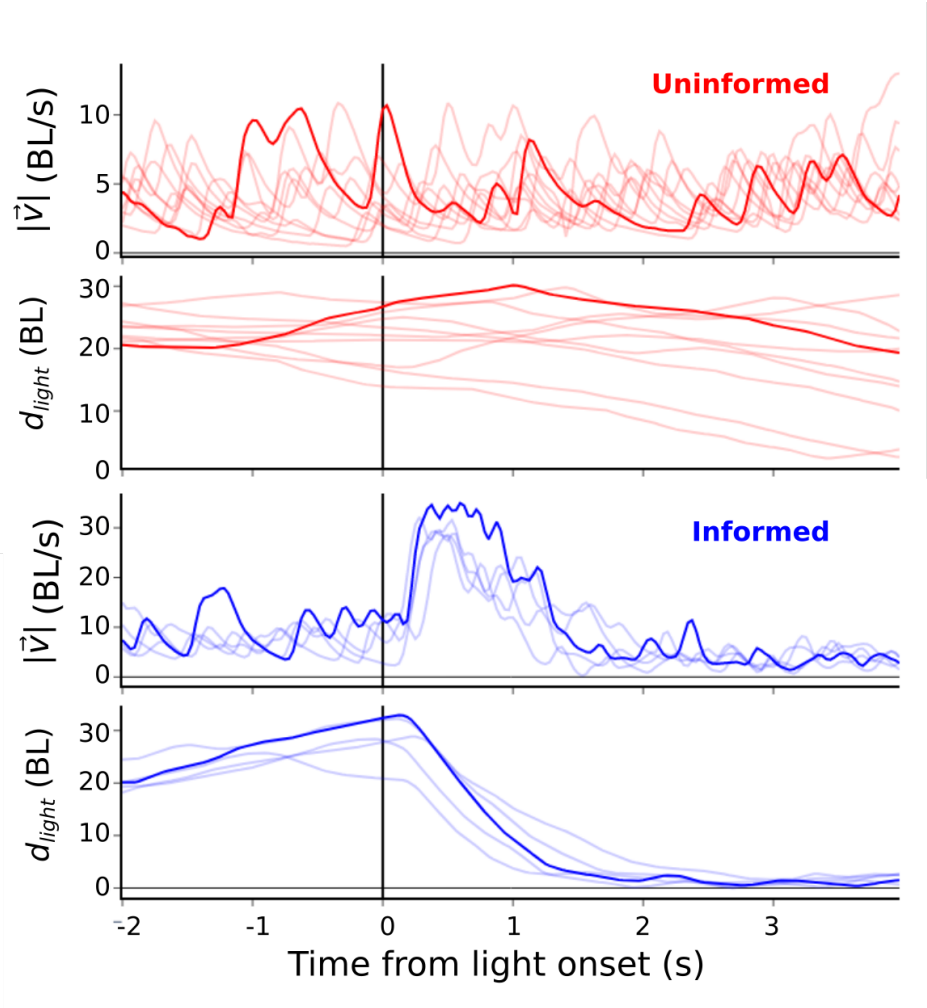

Figure A.11. **Distance to the light and speed in a training trial**. **Top.** Speed and distance to the stimulus around the light onset of a group of 10 uninformed fish in an example trial of the last training session. **Bottom.** Speed and distance to the stimulus around the light onset of a group of 5 informed fish in an example trial of the last training session.

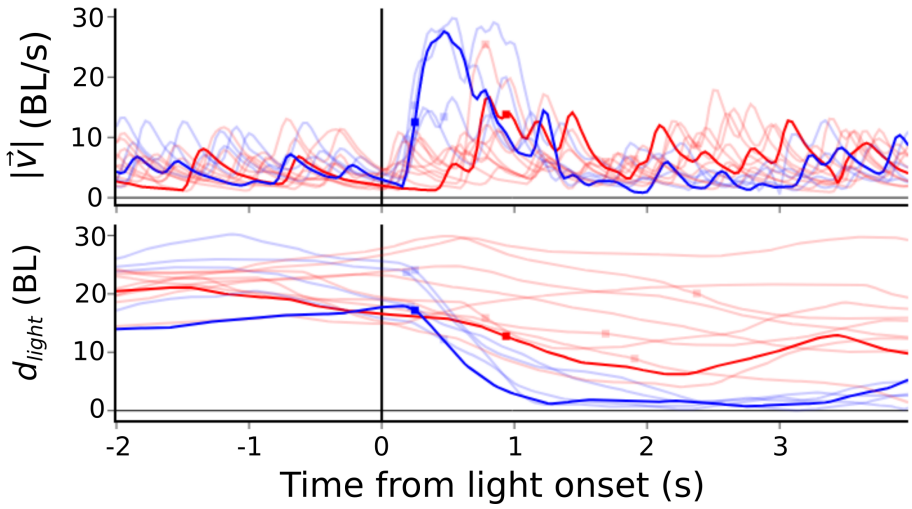

Figure A.12. **Distance to the light and speed in an experiment trial**. Speed (top) and distance to the light (bottom) around the light onset for a mixed group of 5 informed fish and 10 uninformed fish for an example trial of the information aggregation experiment. The square markers indicate the reaction time as detected by the ReactNet network (see Section 3.4.9)

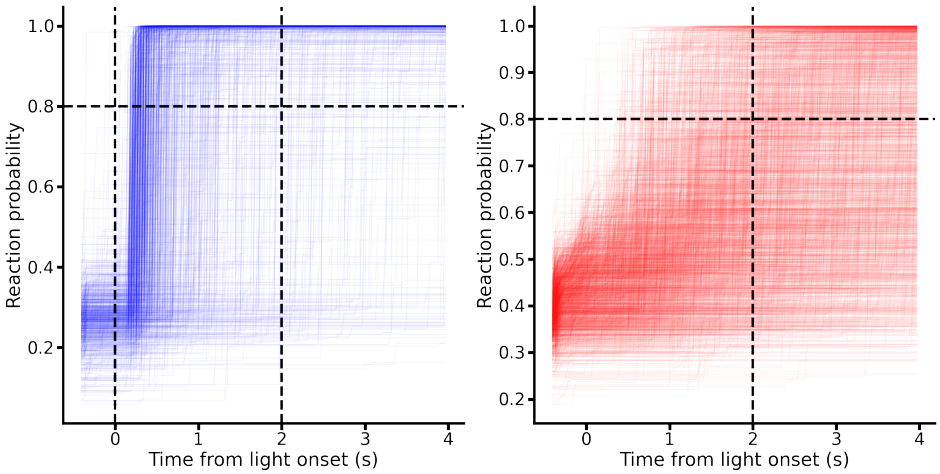

Figure A.13. **Estimated reaction probabilities**. Raw reaction probabilities as output by the ReactNet network (see Section 3.4.9). Each line represents the estimated reaction probability for a fish in a trial of an experiment. All trials of all experiments are considered here. The horizontal dashed line indicates the threshold selected to determine when a fish had reacted. When the probability was higher than the threshold already before the light onset we considered that the fish had not reacted.

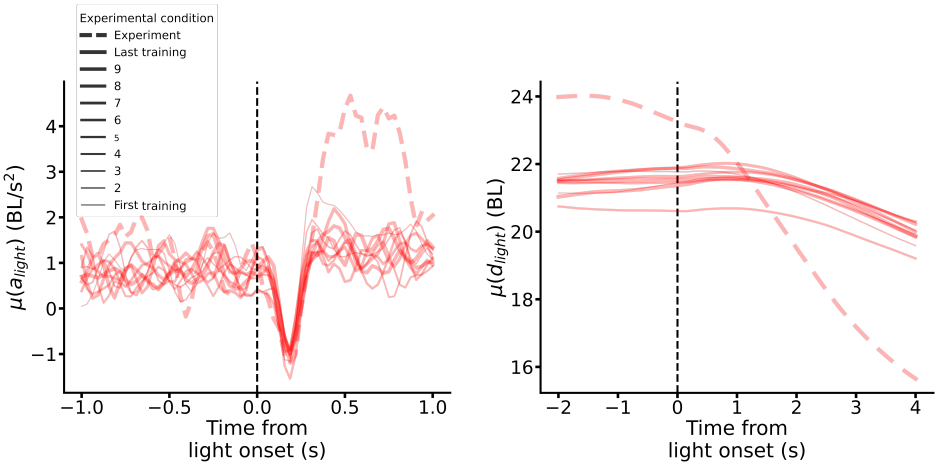

Figure A.14. **Uninformed fish acceleration towards the light and distance to the light. Left.** Mean acceleration towards the light across fish and all trials for each training session (solid lines) and in the information aggregation experiment (dashed line). **Right.** Mean distance to the light across fish and all trials for each training session (solid lines) and in the information aggregation experiment (dashed line).

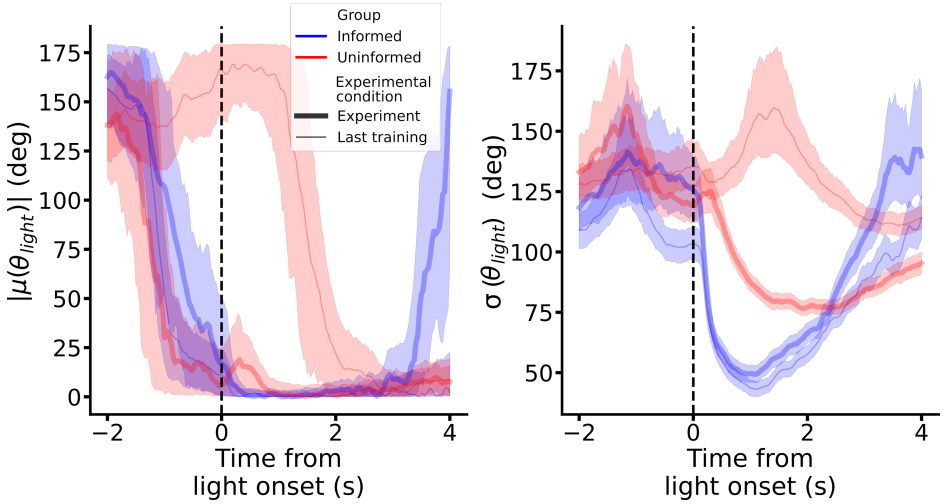

Figure A.15. **Orientation towards the light. Left.** Absolute value of the circular mean of the angle θ_{light} in the last training session (thin line) and in the experiment (think line) for informed (blue) and uninformed (red) fish. The mean was computed across all fish and trials in each experimental condition. The shaded areas represent the 95% confidence interval of the absolute value of the circular mean. **Right.** Circular standard deviation of the angle θ_{light} in the last training session (thin line) and in the experiment (think line) for informed (blue) and uninformed (red) fish. The standard deviation was computed across all fish and trials in each experimental condition. The shaded areas represent the 95% confidence interval of the standard deviation.

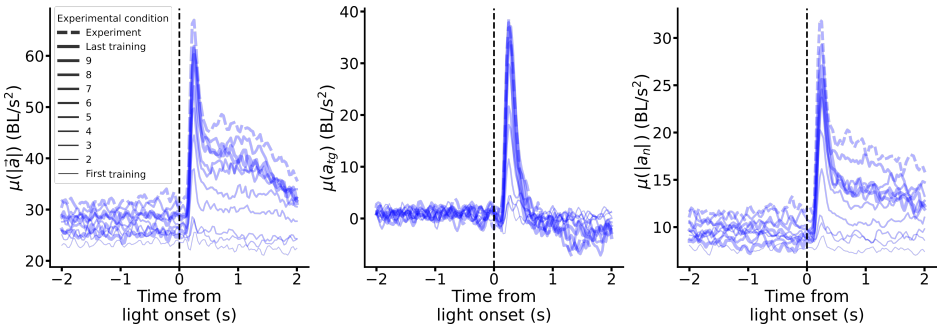

Figure A.16. **Informed fish acceleration. Left.** Mean of the norm of the acceleration vector in all training sessions (solid line) and in the experiment (dashed line) around the light onset. The mean is computed across all fish and trials in each experimental condition. **Center.** Same but for the tangential acceleration. **Right.** Same but for the absolute value of the normal acceleration.

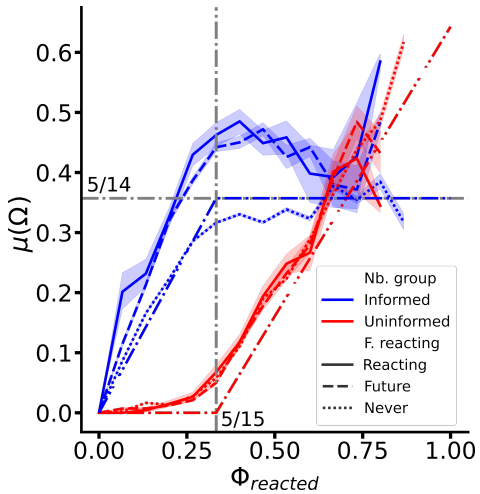

Figure A.17. **Share of the normalized aggregation weight, Ω.** Mean of Ω as a function of the fraction of fish that had already reacted, Φ_{reacted}. The continuous, dashed and dotted lines indicate the instantaneous reaction condition of the focal fish. The dot-dashed line indicates the same results for a model in which all informed fish reacted before all the uninformed fish and the normalized aggregation weight, ω_i, of all neighbours is the same, i.e. 1/14.

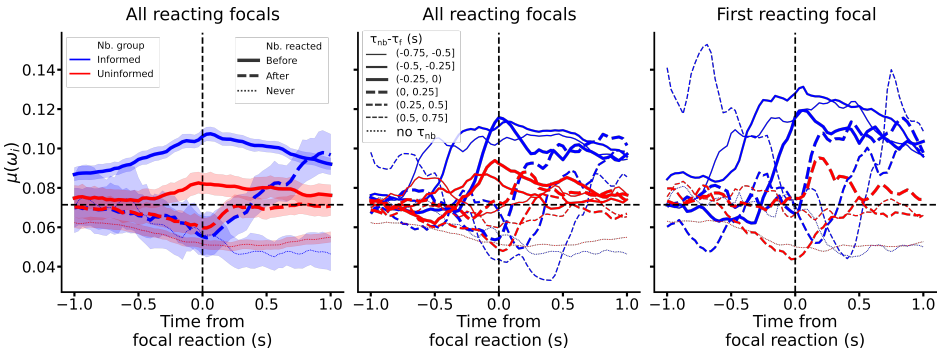

Figure A.18. **Normalized aggregation weight (ω_i) around the uninformed focal fish reaction time. Left.** Mean of ω_i around the uninformed focal fish reaction time. The shaded area represents the 95% confidence interval. The different line styles indicate whether the neighbour reacted before (solid line) or after the focal (dashed line) or the neighbour did not react at all (dotted line). **Center.** Mean of ω_i around the uninformed focal fish reaction time. The different line styles indicate how earlier or later the neighbour fish reacted with respect to the focal fish. Negative values of $\tau_{nb} - \tau_f$ indicate that the neighbour reacted before the focal (solid line), positive values of $\tau_{nb} - \tau_f$ indicate that the neighbour reacted after the focal (dashed line). The dotted line represent the mean of neighbours that did not react. **Right.** Same as the central panel but only considering the first uninformed focal fish that reacted after the light onset.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Alexander, R. M. (1976). Estimates of speeds of dinosaurs. *Nature*, *261*(5556), 129–130.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.

Altmann, J. (1974). Observational study of behavior: sampling methods. *Behaviour*, *49*(3-4), 227–266.

Anderson, D. J., & Perona, P. (2014). Toward a science of computational ethology. *Neuron*, *84*(1), 18–31. doi: 10.1016/j.neuron.2014.09.005

Angilletta Jr, M., Roth, T., Wilson, R., Niehaus, A., & Ribeiro, P. (2008). The fast and the fractalous: speed and tortuosity trade off in running ants. *Functional Ecology*, 78–83.

Aoki, I. (1982). A simulation study on the schooling mechanism in fish. *NIPPON SUISAN GAKKAISHI*, *48*(8), 1081–1088.

Arganda, S., Pérez-Escudero, A., & de Polavieja, G. G. (2012). A common rule for decision making in animal collectives across species. *Proceedings of the National Academy of Sciences*, *109*(50), 20508–20513.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, Y.-X., Zhang, S.-H., Fan, Z., Liu, X.-Y., Zhao, X., Feng, X.-Z., & Sun, M.-Z. (2018). Automatic multiple zebrafish tracking based on improved hog features. *Scientific reports*, *8*(1), 1–14.

Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., ... others (2008). Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the national academy of sciences*, *105*(4), 1232–1237.

Bartoli, F., Lisanti, G., Ballan, L., & Del Bimbo, A. (2018). Context-aware trajectory prediction. In *2018 24th international conference on pattern recognition (icpr)* (pp. 1941–1946).

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, *43*(1), 3–31.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., & Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*.

Bayne, B., & Scullard, C. (1978). Rates of feeding by thais (nucella) lapillus (l.). *Journal of Experimental Marine Biology and Ecology*, *32*(2), 113–129.

Bellet, A., Habrard, A., & Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*.

Benítez, J. M., Castro, J. L., & Requena, I. (1997). Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, *8*(5), 1156–1164.

Berman, G. (2018). Measuring behavior across scales. *BMC Biology*, 1–11. doi: 10.1016/S0733-8619(05)70212-7

Bode, N. W., Franks, D. W., & Wood, A. J. (2011). Limited interactions in flocks: relating model simulations to empirical data. *Journal of The Royal Society Interface*, *8*(55), 301–304.

Bod'ová, K., Mitchell, G. J., Harpaz, R., Schneidman, E., & Tkačik, G. (2018, mar). Probabilistic models of individual and collective animal behavior. *PLOS ONE*, *13*(3), e0193049. Retrieved from `http://dx.plos.org/10.1371/journal.pone.0193049` doi: 10.1371/journal.pone.0193049

Boenisch, F., Rosemann, B., Wild, B., Dormagen, D., Wario, F., & Landgraf, T. (2018). Tracking all members of a honey bee colony over their lifetime using learned models of correspondence. *Frontiers in Robotics and AI*, *5*, 35.

Bovik, A. C. (2010). *Handbook of image and video processing.* Academic press.

Bozek, K., Hebert, L., Portugal, Y., & Stephens, G. J. (2021). Markerless tracking of an entire honey bee colony. *Nature Communications*, *12*(1), 1–13.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Branson, K., Robie, A. A., Bender, J., Perona, P., & Dickinson, M. H. (2009). High-throughput ethomics in large groups of drosophila. *Nature methods*, *6*(6), 451–457.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 737–737.

Brown, A. E., & de Bivort, B. (2018). Ethology as a physical science. *bioRxiv*, 220855. Retrieved from `https://www.biorxiv.org/content/early/2018/02/02/220855` doi: 10.1101/220855

Buske, C., & Gerlai, R. (2011). Shoaling develops with age in zebrafish (danio rerio). *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, *35*(6), 1409–1415.

Buske, C., & Gerlai, R. (2012). Maturation of shoaling behavior is accompanied by changes in the dopaminergic and serotoninergic systems in zebrafish. *Developmental psychobiology*, *54*(1), 28–35.

Butail, S., & Paley, D. A. (2012). Three-dimensional reconstruction of the fast-start swimming kinematics of densely schooling fish. *Journal of the Royal Society Interface*, *9*(66), 77–88.

Calovi, D. S., Litchinko, A., Lecheval, V., Lopez, U., Escudero, A. P., Chaté, H., ... Theraulaz, G. (2018). Disentangling and modeling interactions in fish with burst-and-coast swimming reveal distinct alignment and attraction behaviors. *PLoS computational biology*, *14*(1), e1005933.

Caruana, R., Lawrence, S., & Giles, L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 402–408.

Cavagna, A., Cimarelli, A., Giardina, I., Parisi, G., Santagati, R., Stefanini, F., & Viale, M. (2010). Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, *107*(26), 11865–11870.

Chaté, H., Ginelli, F., Grégoire, G., Peruani, F., & Raynaud, F. (2008). Modeling collective motion: variations on the vicsek model. *The European Physical Journal B*, *64*(3), 451–456.

Chen, Q., Petriu, E., & Yang, X. (2004). A comparative study of fourier descriptors and hu's seven moment invariants for image recognition. In *Canadian conference on electrical and computer engineering 2004 (ieee cat. no. 04ch37513)* (Vol. 1, pp. 103–106).

Chen, Y., Tian, Y., & He, M. (2020). Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, *192*, 102897.

Cheng, X. E., Du, S. S., Li, H. Y., Hu, J. F., & Chen, M. L. (2018). Obtaining three-dimensional trajectory of multiple fish in water tank via video tracking. *Multimedia Tools and Applications*, *77*(18), 24499–24519.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 1, pp. 539–546).

Cichos, F., Gustavsson, K., Mehlig, B., & Volpe, G. (2020). Machine learning for active matter. *Nature Machine Intelligence*, *2*(2), 94–103.

Collignon, B., Séguret, A., & Halloy, J. (2016). A stochastic vision-based model inspired by zebrafish collective behaviour in heterogeneous environments. *Royal Society open science*, *3*(1), 150473.

Colwill, R. M., Raymond, M. P., Ferreira, L., & Escudero, H. (2005). Visual discrimination learning in zebrafish (danio rerio). *Behavioural Processes*, *70*(1), 19–31.

Conradt, L., & Roper, T. J. (2003). Group decision-making in animals. *Nature*, *421*(6919), 155–158.

Costa, T., Laan, A., Heras, F. J., & de Polavieja, G. G. (2021). Automated discovery of local rules for desired collective-level behavior through reinforcement learning. *Fundamentals and Applications of AI: An Interdisciplinary Perspective*.

Couzin, I. D., Krause, J., Franks, N. R., & Levin, S. A. (2005). Effective leadership and decision-making in animal groups on the move. *Nature*, *433*(7025), 513–516.

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, *218*(1), 1–11.

Crosato, E., Jiang, L., Lecheval, V., Lizier, J. T., Wang, X. R., Tichit, P., ... Prokopenko, M. (2018). Informative and misinformative interactions in a school of fish. *Swarm Intelligence*, *12*(4), 283–305.

Dahlbom, S. J., Lagman, D., Lundstedt-Enkel, K., Sundström, L. F., & Winberg, S. (2011). Boldness predicts social status in zebrafish (danio rerio). *PloS one*, *6*(8), e23565.

Danchin, E., Giraldeau, L.-A., Valone, T. J., & Wagner, R. H. (2004). Public information: from nosy neighbors to cultural evolution. *Science*, *305*(5683), 487–491.

Daniel Kissling, W., Pattemore, D. E., & Hagen, M. (2014). Challenges and prospects in the telemetry of insects. *Biological Reviews*, *89*(3), 511–530.

Darwin, C., & Prodger, P. (1998). *The expression of the emotions in man and animals*. Oxford University Press, USA.

Davis, S., Lukeman, R., Schaerf, T. M., & Ward, A. J. (2017). Familiarity affects collective motion in shoals of guppies (poecilia reticulata). *Royal Society open science*, *4*(9), 170312.

De Condorcet, N., et al. (2014). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press.

Dell, A. I., Bender, J. A., Branson, K., Couzin, I. D., de Polavieja, G. G., Noldus, L. P., ... others (2014). Automated image-based tracking and its application in ecology. *Trends in ecology & evolution*, *29*(7), 417–428.

Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., ... Leal-Taixé, L. (2020). Motchallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 1–37.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., ... Leal-Taixé, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dennis, R., Newberry, R., Cheng, H.-W., & Estevez, I. (2008). Appearance matters: artificial marking alters aggression and stress. *Poultry science*, *87*(10), 1939–1946.

Dhillon, A., & Verma, G. K. (2020). Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, *9*(2), 85–112.

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-agent systems: A survey. *Ieee Access*, *6*, 28573–28593.

Easley, D., Kleinberg, J., et al. (2010). *Networks, crowds, and markets* (Vol. 8). Cambridge university press Cambridge.

Egnor, S. E., & Branson, K. (2016). Computational Analysis of Behavior. *Annual Review of Neuroscience*, *39*, 217–236. doi: 10.1146/annurev-neuro-070815-013845

Elliott, J. P., Cowan, I. M., & Holling, C. (1977). Prey capture by the african lion. *Canadian Journal of Zoology*, *55*(11), 1811–1828.

Engeszer, R. E., Patterson, L. B., Rao, A. A., & Parichy, D. M. (2007). Zebrafish in the wild: a review of natural history and new notes from the field. *Zebrafish*, *4*(1), 21–40.

Ernst, M. D. (2004). Permutation methods: A basis for exact inference. *Statistical Science*, *19*(4), 676–685. doi: 10.1214/088342304000000396

Eyjolfsdottir, E., Branson, K., Yue, Y., & Perona, P. (2016). Learning recurrent representations for hierarchical behavior modeling. *arXiv preprint arXiv:1611.00094*.

Faria, J. J., Dyer, J. R., Clément, R. O., Couzin, I. D., Holt, N., Ward, A. J., ... Krause, J. (2010). A novel method for investigating the collective behaviour of fish: introducing 'robofish'. *Behavioral Ecology and Sociobiology*, *64*(8), 1211–1218.

Ferreira, A. C., Silva, L. R., Renna, F., Brandl, H. B., Renoult, J. P., Farine, D. R., ... Doutrelant, C. (2019). Deep learning-based methods for individual recognition in small birds. *bioRxiv*, 862557.

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Fortin, D., Beyer, H. L., Boyce, M. S., Smith, D. W., Duchesne, T., & Mao, J. S. (2005). Wolves influence elk movements: behavior shapes a trophic cascade in yellowstone national park. *Ecology*, *86*(5), 1320–1330.

Francisco, F. A., Nührenberg, P., & Jordan, A. (2020). High-resolution, non-invasive animal tracking and reconstruction of local environment in aquatic ecosystems. *Movement ecology*, *8*(1), 1–12.

Gal, A., Saragosti, J., & Kronauer, D. J. (2020). antrax: high throughput video tracking of color-tagged insects. *bioRxiv*.

Gallois, B., & Candelier, R. (2021). Fasttrack: An open-source software for tracking varying numbers of deformable objects. *PLoS computational biology*, *17*(2), e1008697.

Gautrais, J., Ginelli, F., Fournier, R., Blanco, S., Soria, M., Chaté, H., & Theraulaz, G. (2012). Deciphering interactions in moving animal groups.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 ieee conference on computer vision and pattern recognition* (pp. 3354–3361).

Geissmann, Q., Garcia Rodriguez, L., Beckwith, E. J., French, A. S., Jamasb, A. R., & Gilestro, G. F. (2017). Ethoscopes: An open platform for high-throughput ethomics. *PLoS biology*, *15*(10), e2003026.

Gerlai, R. (2019). Reproducibility and replicability in zebrafish behavioral neuroscience research. *Pharmacology Biochemistry and Behavior*, *178*, 30–38.

Giraldeau, L.-A., & Caraco, T. (2018). *Social foraging theory* (Vol. 73). Princeton University Press.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Gomez-Marin, A., Paton, J. J., Kampff, A. R., Costa, R. M., & Mainen, Z. F. (2014). Big behavioral data: Psychology, ethology and the foundations of neuroscience. *Nature Neuroscience*, *17*(11), 1455–1462. doi: 10.1038/nn.3812

Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., & Couzin, I. D. (2019). Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *Elife*, *8*, e47994.

Grünwald, P. D., & Grunwald, A. (2007). *The minimum description length principle*. MIT press.

Guille, A., Hacid, H., Favre, C., & Zighed, D. A. (2013). Information diffusion in online social networks: A survey. *ACM Sigmod Record*, *42*(2), 17–28.

Haalck, L., Mangan, M., Webb, B., & Risse, B. (2020). Towards image-based animal tracking in natural environments using a freely moving camera. *Journal of neuroscience methods*, *330*, 108455.

Harpaz, R., Tkačik, G., & Schneidman, E. (2017). Discrete modes of social information processing predict individual behavior of fish in a group. *Proceedings of the National Academy of Sciences of the United States of America*, *114*(38), 10149–10154. doi: 10.1073/pnas.1703817114

Harris, M. (2008). Many-core gpu computing with nvidia cuda. In *Proceedings of the 22nd annual international conference on supercomputing* (pp. 1–1).

Haslam, S. A., Reicher, S. D., & Platow, M. J. (2020). *The new psychology of leadership: Identity, influence and power*. Routledge.

Heras, F. J., Romero-Ferrero, F., Hinz, R. C., & de Polavieja, G. G. (2019). Deep attention networks reveal the rules of collective motion in zebrafish. *PLoS computational biology*, *15*(9), e1007354.

Herbert-Read, J. E., Perna, A., Mann, R. P., Schaerf, T. M., Sumpter, D. J., & Ward, A. J. (2011). Inferring the rules of interaction of shoaling fish. *Proceedings of the National Academy of Sciences*, *108*(46), 18726–18731.

Hinz, R. C., & de Polavieja, G. G. (2017). Ontogeny of collective behavior reveals a simple attraction rule. *Proceedings of the National Academy of Sciences*, *114*(9), 2295–2300.

Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition* (pp. 84–92).

Honegger, K., & de Bivort, B. (2018). Stochasticity, individuality and behavior. *Current Biology*, *28*(1), R8–R12. Retrieved from `http://dx.doi.org/10.1016/j.cub.2017.11.058` doi: 10.1016/j.cub.2017.11.058

Hoshen, Y. (2017). Vain: Attentional multi-agent predictive modeling. *arXiv preprint arXiv:1706.06122*.

Hsu, Y.-C., Lv, Z., Schlosser, J., Odom, P., & Kira, Z. (2019). Multi-class classification without multi-class labels. *arXiv preprint arXiv:1901.00544*.

Hughey, L. F., Hein, A. M., Strandburg-Peshkin, A., & Jensen, F. H. (2018). Challenges and solutions for studying collective animal behaviour in the wild. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *373*(1746), 20170005.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, *50*(2), 1–35.

Huth, A., & Wissel, C. (1992). The simulation of the movement of fish schools. *Journal of theoretical biology*, *156*(3), 365–385.

Im, D. J., Kwak, I., & Branson, K. (2020). Evaluation metrics for behaviour modeling. *arXiv preprint arXiv:2007.12298*.

Itskovits, E., Levine, A., Cohen, E., & Zaslaver, A. (2017). A multi-animal tracker for studying complex behaviors. *BMC biology*, *15*(1), 1–16.

Jeon, W., Kang, S.-H., Leem, J.-B., & Lee, S.-H. (2013). Characterization of fish schooling behavior with different numbers of medaka (oryzias latipes) and goldfish (carassius auratus) using a hidden markov model. *Physica A: Statistical Mechanics and Its Applications*, *392*(10), 2426–2433.

Jiang, L., Giuggioli, L., Perna, A., Escobedo, R., Lecheval, V., Sire, C., ... Theraulaz, G. (2017). Identifying influential neighbors in animal flocking. *PLoS computational biology*, *13*(11), e1005822.

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kalueff, A. V., Gebhardt, M., Stewart, A. M., Cachat, J. M., Brimmer, M., Chawla, J. S., ... others (2013). Towards a comprehensive catalog of zebrafish behavior 1.0 and beyond. *Zebrafish*, *10*(1), 70–86.

Kane, G. A., Lopes, G., Saunders, J. L., Mathis, A., & Mathis, M. W. (2020). Real-time, low-latency closed-loop feedback using markerless posture tracking. *bioRxiv*.

Kareiva, P., & Shigesada, N. (1983). Analyzing insect movement as a correlated random walk. *Oecologia*, *56*(2-3), 234–238.

Katz, Y., Tunstrøm, K., Ioannou, C. C., Huepe, C., & Couzin, I. D. (2011). Inferring the structure and dynamics of interactions in schooling fish. *Proceedings of the National Academy of Sciences*, *108*(46), 18720–18725.

Kays, R., Crofoot, M. C., Jetz, W., & Wikelski, M. (2015). Terrestrial animal tracking as an eye on life and planet. *Science*, *348*(6240).

Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the ieee international conference on computer vision* (pp. 4696–4704).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Laan, A., Gil de Sagredo, R., & de Polavieja, G. G. (2017). Signatures of optimal control in pairs of schooling zebrafish. *Proceedings of the Royal Society B: Biological Sciences*, *284*(1852), 20170224.

Laan, A., Iglesias-Julios, M., & de Polavieja, G. G. (2018). Zebrafish aggression on the sub-second time scale: evidence for mutual motor coordination and multi-functional attack manoeuvres. *Royal Society open science*, *5*(8), 180679.

Laan, A., Madirolas, G., & de Polavieja, G. G. (2017). Rescuing collective wisdom when the average group opinion is wrong. *Frontiers in Robotics and AI*, *4*, 56.

Laradji, I., Saleh, A., Rodriguez, P., Nowrouzezahrai, D., Azghadi, M. R., & Vazquez, D. (2020). Affinity lcfcn: Learning to segment fish with weak supervision. *arXiv preprint arXiv:2011.03149*.

Lawrence, S., Giles, C. L., & Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. In *Aaai/iaai* (pp. 540–545).

Lecheval, V., Jiang, L., Tichit, P., Sire, C., Hemelrijk, C. K., & Theraulaz, G. (2018). Social conformity and propagation of information in collective u-turns of fish schools. *Proceedings of the Royal Society B: Biological Sciences*, *285*(1877), 20180251.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989b). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, *19*, 143–155.

Lemasson, B. H., Anderson, J. J., & Goodwin, R. A. (2009). Collective motion in animal groups from a neurobiological perspective: the adaptive benefits of dynamic sensory loads and selective attention. *Journal of theoretical biology*, *261*(4), 501–510.

Lemasson, B. H., Anderson, J. J., & Goodwin, R. A. (2013). Motion-guided attention promotes adaptive communications during social navigation. *Proceedings of the Royal Society B: Biological Sciences*, *280*(1754), 20122003.

Lennox, R. J., Aarestrup, K., Cooke, S. J., Cowley, P. D., Deng, Z. D., Fisk, A. T., ... others (2017). Envisioning the future of aquatic animal tracking: technology, science, and application. *BioScience*, *67*(10), 884–896.

Levin, E. D., & Cerutti, D. T. (2009). Behavioral neuroscience of zebrafish. *Methods of Behavior Analysis in Neuroscience. 2nd edition*.

Lima, S. L. (1995). Back to the basics of anti-predatory vigilance: the group-size effect. *Animal Behaviour*, *49*(1), 11–20.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2021). Explainable ai: A review of machine learning interpretability methods. *Entropy*, *23*(1), 18.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International journal of computer vision*, *128*(2), 261–318.

Lopes, G., Bonacchi, N., Frazão, J., Neto, J. P., Atallah, B. V., Soares, S., ... others (2015). Bonsai: an event-based framework for processing and controlling data streams. *Frontiers in neuroinformatics*, *9*, 7.

Lorbach, M., Poppe, R., van Dam, E. A., Noldus, L. P., & Veltkamp, R. C. (2015). Automated recognition of social behavior in rats: The role of feature quality. In *International conference on image analysis and processing* (pp. 565–574).

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2020). Multiple object tracking: A literature review. *Artificial Intelligence*, 103448.

MacGregor, H. E., Herbert-Read, J. E., & Ioannou, C. C. (2020). Information can explain the dynamics of group order in animal collective behaviour. *Nature communications*, *11*(1), 1–8.

Machado, A. S., Darmohray, D. M., Fayad, J., Marques, H. G., & Carey, M. R. (2015). A quantitative framework for whole-body coordination reveals specific deficits in freely walking ataxic mice. *Elife*, *4*, e07892.

Marks, M., Qiuhan, J., Sturman, O., von Ziegler, L., Kollmorgen, S., von der Behrens, W., ... Yanik, M. F. (2020). Sipec: the deep-learning swiss knife for behavioral data analysis. *bioRxiv*.

Marques, J. C., Lackner, S., Félix, R., & Orger, M. B. (2018). Structure of the zebrafish locomotor repertoire revealed with unsupervised behavioral clustering. *Current Biology*, *28*(2), 181–195.

Marshall, J. A., Brown, G., & Radford, A. N. (2017). Individual confidence-weighting and group decision-making. *Trends in ecology & evolution*, *32*(9), 636–645.

Martins, S., Monteiro, J. F., Vito, M., Weintraub, D., Almeida, J., & Certal, A. C. (2016). Toward an integrated zebrafish health management program supporting cancer and neuroscience research. *Zebrafish*, *13*(S1), S–47.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, *21*(9), 1281–1289.

McKinney, W., et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th python in science conference* (Vol. 445, pp. 51–56).

Miao, Z., Gaynor, K. M., Wang, J., Liu, Z., Muellerklein, O., Norouzzadeh, M. S., ... others (2019). Insights and approaches using deep learning to classify wildlife. *Scientific reports*, *9*(1), 1–9.

Miller, N., & Gerlai, R. (2012a). From Schooling to Shoaling: Patterns of Collective Motion in Zebrafish (Danio rerio). *PLoS ONE*, *7*(11), 8–13. doi: 10.1371/journal.pone.0048865

Miller, N., & Gerlai, R. (2012b). From schooling to shoaling: patterns of collective motion in zebrafish (danio rerio). *PloS one*, *7*(11), e48865.

Moeslund, T. B. (2012). Blob analysis. In *Introduction to video and image processing* (pp. 103–115). Springer.

Morgan, N., & Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in neural information processing systems* (pp. 630–637).

Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., & Ghanem, B. (2018). Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the european conference on computer vision (eccv)* (pp. 300–317).

Murali, N., Schneider, J., Levine, J., & Taylor, G. (2019). Classification and re-identification of fruit fly individuals across days with convolutional neural networks. In *2019 ieee winter conference on applications of computer vision (wacv)* (pp. 570–578).

Muybridge, E. (1887). *Animal locomotion.* Da Capo Press.

Mwaffo, V., Butail, S., & Porfiri, M. (2017). Analysis of pairwise interactions in a maximum likelihood sense to identify leaders in a group. *Frontiers in Robotics and AI*, *4*, 35.

Nagy, M., Ákos, Z., Biro, D., & Vicsek, T. (2010). Hierarchical group dynamics in pigeon flocks. *Nature*, *464*(7290), 890–893.

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019). Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, *14*(7), 2152–2176.

Nauta, M., Bucur, D., & Seifert, C. (2019). Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, *1*(1), 312–340.

Newton, I., & Chittenden, N. (1850). *Newton's principia: The mathematical principles of natural philosophy.* Geo. P. Putnam.

Nicolis, G. (1977). Self-organization in nonequilibrium systems. *Dissipative Structures to Order through Fluctuations*, 339–426.

Okubo, A. (1986). Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. *Advances in biophysics*, *22*, 1–94.

Oliphant, T. E. (2006). *A guide to numpy* (Vol. 1). Trelgol Publishing USA.

Oliveira, R. F. (2013). Mind the fish: zebrafish as a model in cognitive social neuroscience. *Frontiers in neural circuits*, *7*, 131.

Orger, M. B., & de Polavieja, G. G. (2017). Zebrafish behavior: opportunities and challenges. *Annual review of neuroscience*, *40*, 125–147.

Parrish, J. K., Viscido, S. V., & Grunbaum, D. (2002). Self-organized fish schools: an examination of emergent properties. *The biological bulletin*, *202*(3), 296–305.

Partridge, B. L. (1982). The structure and function of fish schools. *Scientific american*, *246*(6), 114–123.

Partridge, B. L., Johansson, J., & Kalish, J. (1983). The structure of schools of giant bluefin tuna in cape cod bay. *Environmental biology of fishes*, *9*(3), 253–262.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Pathak, D., Girshick, R., Dollár, P., Darrell, T., & Hariharan, B. (2017). Learning features by watching objects move. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2701–2710).

Pedersen, M., Haurum, J. B., Bengtson, S. H., & Moeslund, T. B. (2020). 3d-zef: A 3d zebrafish tracking benchmark dataset. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 2426–2436).

Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S.-H., Murthy, M., & Shaevitz, J. W. (2019). Fast animal pose estimation using deep neural networks. *Nature methods*, *16*(1), 117–125.

Pereira, T. D., Shaevitz, J. W., & Murthy, M. (2020). Quantifying behavior to understand the brain. *Nature neuroscience*, 1–13.

Pereira, T. D., Tabris, N., Li, J., Ravindranath, S., Papadoyannis, E. S., Wang, Z. Y., ... others (2020). Sleap: multi-animal pose tracking. *bioRxiv*.

Pérez-Escudero, A., & de Polavieja, G. (2011). Collective animal behavior from bayesian estimation and probability matching. *Nature Precedings*, 1–1.

Pérez-Escudero, A., & de Polavieja, G. G. (2017). Adversity magnifies the importance of social information in decision-making. *Journal of The Royal Society Interface*, *14*(136), 20170748.

Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S., & De Polavieja, G. G. (2014). idtracker: tracking individuals in a group by automatic identification of unmarked animals. *Nature methods*, *11*(7), 743–748.

Pilkiewicz, K., Lemasson, B., Rowland, M., Hein, A., Sun, J., Berdahl, A., ... others (2020). Decoding collective communications using information theory tools. *Journal of the Royal Society Interface*, *17*(164), 20190563.

Qian, Z.-M., & Chen, Y. Q. (2017). Feature point based 3d tracking of multiple fish from multi-view images. *PloS one*, *12*(6), e0180254.

Quiroga, R. Q., Kreuz, T., & Grassberger, P. (2002). Event synchronization: a simple and fast method to measure synchronicity and time delay patterns. *Physical review E*, *66*(4), 041904.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., & Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *international conference on machine learning* (pp. 2847–2854).

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning* (pp. 873–880).

Rasch, M. J., Shi, A., & Ji, Z. (2016). Closing the loop: tracking and perturbing behaviour of individuals in a group in real-time. *bioRxiv*, 071308.

Roberts, G. (1996). Why individual vigilance declines as group size increases. *Animal behaviour*, *51*(5), 1077–1086.

Robie, A. A., Seagraves, K. M., Egnor, S. R., & Branson, K. (2017). Machine vision methods for analyzing social interactions. *Journal of Experimental Biology*, *220*(1), 25–34.

Rodriguez, A., Zhang, H., Klaminder, J., Brodin, T., & Andersson, M. (2017). Toxid: an efficient algorithm to solve occlusions when tracking multiple animals. *Scientific reports*, *7*(1), 1–8.

Rodriguez, A., Zhang, H., Klaminder, J., Brodin, T., Andersson, P. L., & Andersson, M. (2018). Toxtrac: a fast and robust software for tracking organisms. *Methods in Ecology and Evolution*, *9*(3), 460–464.

Romero-Ferrero, F., Bergomi, M. G., Hinz, R., Heras, F. J., & de Polavieja, G. G. (2018). idtracker. ai: Tracking all individuals in large collectives of unmarked animals. *bioRxiv*, 280735.

Romero-Ferrero, F., Bergomi, M. G., Hinz, R. C., Heras, F. J., & de Polavieja, G. G. (2019). Idtracker. ai: tracking all individuals in small or large collectives of unmarked animals. *Nature methods*, *16*(2), 179–182.

Rosenthal, S. B., Twomey, C. R., Hartnett, A. T., Wu, H. S., & Couzin, I. D. (2015). Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion. *Proceedings of the National Academy of Sciences*, *112*(15), 4690–4695.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Schneider, S., Taylor, G. W., & Kremer, S. C. (2020). Similarity learning networks for animal individual re-identification-beyond the capabilities of a human observer. In *Proceedings of the ieee/cvf winter conference on applications of computer vision workshops* (pp. 44–52).

Schultz, M., & Joachims, T. (2004). Learning a distance metric from relative comparisons. *Advances in neural information processing systems*, *16*, 41–48.

Schulze, L., Henninger, J., Kadobianskyi, M., Chaigne, T., Faustino, A. I., Hakiy, N., ... others (2018). Transparent danionella translucida as a genetically tractable vertebrate brain model. *Nature methods*, *15*(11), 977–983.

Sivaramakrishna, R., & Shashidharf, N. (1997). Hu's moment invariants: how invariant are they under skew and perspective transformations? In *Ieee wescanex 97 communications, power and computing. conference proceedings* (pp. 292–295).

Spence, R., Gerlach, G., Lawrence, C., & Smith, C. (2008). The behaviour and ecology of the zebrafish, danio rerio. *Biological reviews*, *83*(1), 13–34.

Sridhar, V. H., Roche, D. G., & Gingins, S. (2019). Tracktor: Image-based automated tracking of animal movement and behaviour. *Methods in Ecology and Evolution*, *10*(6), 815–820.

Steenbergen, P. J., Richardson, M. K., & Champagne, D. L. (2011). Patterns of avoidance behaviours in the light/dark preference test in young juvenile zebrafish: a pharmacological study. *Behavioural brain research*, *222*(1), 15–25.

Stowers, J. R., Hofbauer, M., Bastien, R., Griessner, J., Higgins, P., Farooqui, S., ... others (2017). Virtual reality for freely moving animals. *Nature methods*, *14*(10), 995–1002.

Strandburg-Peshkin, A., Papageorgiou, D., Crofoot, M. C., & Farine, D. R. (2018). Inferring influence and leadership in moving animal groups. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *373*(1746), 20170006.

Strandburg-Peshkin, A., Twomey, C. R., Bode, N. W., Kao, A. B., Katz, Y., Ioannou, C. C., ... others (2013). Visual sensory networks and effective information transfer in animal groups. *Current Biology*, *23*(17), R709–R711.

Straw, A. D., Branson, K., Neumann, T. R., & Dickinson, M. H. (2011). Multi-camera real-time three-dimensional tracking of multiple flying animals. *Journal of The Royal Society Interface*, *8*(56), 395–409.

Sumpter, D. J. (2006). The principles of collective animal behaviour. *Philosophical transactions of the royal society B: Biological Sciences*, *361*(1465), 5–22.

Sumpter, D. J. (2010). *Collective animal behavior*. Princeton University Press.

Sumpter, D. J., Krause, J., James, R., Couzin, I. D., & Ward, A. J. (2008). Consensus decision making by fish. *Current Biology*, *18*(22), 1773–1777.

Suriyampola, P. S., Shelton, D. S., Shukla, R., Roy, T., Bhat, A., & Martins, E. P. (2016). Zebrafish social behavior in the wild. *Zebrafish*, *13*(1), 1–8.

Suzuki, S., et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, *30*(1), 32–46.

Tadres, D., & Louis, M. (2020). Pivr: an affordable and versatile closed-loop platform to study unrestrained sensorimotor behavior. *PLoS biology*, *18*(7), e3000712.

Toledo, S., Shohami, D., Schiffner, I., Lourie, E., Orchan, Y., Bartan, Y., & Nathan, R. (2020). Cognitive map–based navigation in wild bats revealed by a new high-throughput tracking system. *Science*, *369*(6500), 188–193.

Valente, A., Huang, K.-H., Portugues, R., & Engert, F. (2012). Ontogeny of classical and operant learning behaviors in zebrafish. *Learning & memory*, *19*(4), 170–177.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., & Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical review letters*, *75*(6), 1226.

Vidal, M., Wolf, N., Rosenberg, B., Harris, B. P., & Mathis, A. (2021). Perspectives on individual animal identification from biology and computer vision. *arXiv preprint arXiv:2103.00560*.

Viscido, S. V., Parrish, J. K., & Grünbaum, D. (2004). Individual behavior and emergent properties of fish schools: a comparison of observation and theory. *Marine Ecology Progress Series*, *273*, 239–249.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., & Leibe, B. (2019). Mots: Multi-object tracking and segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7942–7951).

Walter, T., & Couzin, I. D. (2021). Trex, a fast multi-animal tracking system with markerless identification, and 2d estimation of posture and visual fields. *Elife*, *10*, e64000.

Wang, S. H., Zhao, J. W., & Chen, Y. Q. (2017). Robust tracking of fish schools using cnn for head identification. *Multimedia Tools and Applications*, *76*(22), 23679–23697.

Ward, A. J., Sumpter, D. J., Couzin, I. D., Hart, P. J., & Krause, J. (2008). Quorum decision-making facilitates information transfer in fish shoals. *Proceedings of the National Academy of Sciences*, *105*(19), 6948–6953.

Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., ... Qalieh, A. (2017, September). *mwaskom/seaborn: v0.8.1 (september 2017).* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.883859` doi: 10.5281/zenodo.883859

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 1–40.

Werkhoven, Z., Rohrsen, C., Qin, C., Brembs, B., & de Bivort, B. (2019). Margo (massively automated real-time gui for object-tracking), a platform for high-throughput ethology. *PloS one*, *14*(11), e0224243.

Westley, P. A., Berdahl, A. M., Torney, C. J., & Biro, D. (2018). *Collective movement in ecology: from emerging technologies to conservation and management.* The Royal Society.

Wichert, I. (2019). *Machine learning based video segmentation for animal tracking in behavioral experiments* (Unpublished master's thesis). Bernstein Center for Computational Neuroscience Berlin. (Supervisor: de Polavieja, G. G., co-supervisors: Romero-Ferrero F., Heras, F.J.H., Romanczuk, P.)

Williams, F. E., White, D., & Messer Jr, W. S. (2002). A simple spatial alternation task for assessing memory function in zebrafish. *Behavioural Processes*, *58*(3), 125–132.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Xie, N., Ras, G., van Gerven, M., & Doran, D. (2020). Explainable deep learning: A field guide for the uninitiated. *arXiv preprint arXiv:2004.14545*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Xu, Y., Osep, A., Ban, Y., Horaud, R., Leal-Taixé, L., & Alameda-Pineda, X. (2020). How to train your deep multi-object tracker. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 6787–6796).

Xuan, H., Stylianou, A., & Pless, R. (2020). Improved embeddings with easy positive triplet mining. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 2474–2482).

Yamanaka, O., & Takeuchi, R. (2018). Umatracker: an intuitive image-based tracking platform. *Journal of Experimental Biology*, *221*(16).

Yang, F., Chang, X., Dang, C., Zheng, Z., Sakti, S., Nakamura, S., & Wu, Y. (2020). Remots: Self-supervised refining multi-object tracking and segmentation. *arXiv e-prints*, arXiv–2007.

Yao, R., Lin, G., Xia, S., Zhao, J., & Zhou, Y. (2019). Video object segmentation and tracking: A survey. *arXiv preprint arXiv:1904.09172*.

Zhiping, X., & Cheng, X. E. (2017). Zebrafish tracking using convolutional neural networks. *Scientific reports*, *7*, 42815.

ITQB-UNL | Av. da República, 2780-157 Oeiras, Portugal
Tel (+351) 214 469 100 | Fax (+351) 214 411 277

www.itqb.unl.pt