Deep Classifiers for Robust Larval Zebrafish Behavior Classification

Alexandre Domingues Duhau Laborde

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Jacinto Carlos Marques Peixoto do Nascimento
Prof. Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Jacinto Carlos Marques Peixoto do Nascimento
Member of the Committee: Prof. David Manuel Martins de Matos

May 2019
Abstract

Zebrafish larvae as young as five days post-fertilization already exhibit a rich repertoire of innate behaviors enable them to explore the world, maintain their position in a changing environment, hunt, escape from threatening stimuli and more, making them attractive models to study behavior in the lab. Current state-of-the-art zebrafish behavior classification techniques depend on identifying complex kinematic features in order to classify the data, and thus cannot be applied in experiments where such features cannot be calculated. Taking advantage of an existing large dataset of millions of zebrafish movements across a wide range of stimuli, this project will focus on studying how deep learning techniques can be used to improve current state-of-the-art zebrafish behavior analysis methods. The aim of this thesis is to create a set of behavior classifiers that achieve robust classification of the zebrafish locomotor repertoire using only some of the tracking data without the need for kinematic feature calculation. Based on different types of input information, several deep classification networks are proposed. The proposed networks are optimized evaluating their capacity for achieving a robust behavior classification across different input types, architectures and activation functions without requiring any feature computation. Results show that it possible to achieve robust featureless behavior classification of zebrafish movements using all the different input types. In addition, results reveal that in this context using more complex network architectures does not increase the classification performance but combining different information sources helps improve the classification accuracy. Finally, the results also expose possible labeling inconsistencies in the dataset used to train the networks.

Keywords

Zebrafish, Behavior, Supervised machine learning, Deep learning
Resumo

Larvas de peixe zebra, com apenas cinco dias de idade, já exibem um reportório rico de comportamentos inatos que lhes permitem explorar o mundo, manter a mesma posição num ambiente dinâmico, caçar, fugir de predadores e de outros peixes, o que faz com que sejam um modelo interessante para estudar comportamento no laboratório. Os melhores modelos actuais de classificação de movimentos do peixe zebra dependem de uma infra-estrutura complexa para classificar os dados necessitando de calcular parâmetros cinemáticos do movimento, o que faz que estes modelos não possam ser usados em situações onde os parâmetros do movimento não podem ser calculados. Usando um conjunto de milhões de movimentos de larvas de peixe zebra em variadas situações, este projecto tem como objectivo estudar em que medida, técnicas de deep learning podem ser usadas para melhorar estes métodos de classificação de movimento. O objectivo é então criar um conjunto de classificadores de comportamento que sejam capazes de obter a mesma performance que os métodos actuais de classificação, mas recorrendo apenas a dados de tracking não calculando qualquer parâmetro cinemático do movimento. Neste projecto, são propostas vários arquitecturas de redes diferentes, considerando diferentes inputs possíveis. Estas redes foram optimizadas tendo em conta a sua capacidade de obter uma classificação robusta sem recurso a parâmetros calculados manualmente, entre tipos de dados variados, diferentes arquitecturas e funções de activação. Os resultados mostram que é possível obter classificadores precisos independentemente do tipo de input. Além disso, os resultados revelam que neste contexto, aumentar a complexidade da rede não aumenta a sua capacidade de classificação, mas sim que, combinar várias fontes de dados contribui para um aumento ligeiro da precisão na classificação. Finalmente, os resultados de trabalho apontam para a existência de possíveis erros de identificação das amostras
nos dados usados para treinar as redes deste projecto.

**Palavras Chave**

Peixe zebra, Comportamento, supervised machine learning, deep learning
# Contents

1 Introduction .............................................. 1
   1.1 Motivation ........................................... 3
   1.2 Goals of the Project ................................. 4
   1.3 Contributions ....................................... 4
   1.4 Document Structure ................................ 5

2 Zebrafish, an animal model for genetics and neuroscience .......... 6
   2.1 Zebrafish as an animal model ....................... 7
   2.2 Study of zebrafish larvae locomotion .............. 8

3 Artificial Neural Networks .................................. 11
   3.1 Training a Neural Network .......................... 13
   3.2 Activation Functions ................................ 15
      3.2.1 Sigmoid ....................................... 15
      3.2.2 Hyperbolic Tangent ............................ 15
      3.2.3 Rectified Linear Unit .......................... 16
   3.3 Splitting the dataset ................................ 17
   3.4 Regularization ...................................... 17
   3.5 Deep Learning ...................................... 18

4 Related Work ............................................. 20
   4.1 Supervised learning techniques for animal tracking ...... 21
   4.2 Supervised learning techniques for behavior classification 22
   4.3 Unsupervised Learning Techniques for behavior classification 22
   4.4 Manual descriptions of zebrafish larvae behavior ...... 23
   4.5 Supervised classification of zebrafish larvae behavior .. 24
   4.6 Unsupervised classification of zebrafish larvae behavior 24

5 The Dataset .............................................. 26
   5.1 Data acquisition ................................... 27
      5.1.1 Experimental setup to record behavior ......... 27
9 Future Work

9.1 Zebrafish behavior analysis toolbox using neural networks

9.2 Improving general classification accuracy via Data Augmentation

9.3 Understanding the structure of the zebrafish’s locomotor repertoire using Autoencoders

9.4 Extending the behavior analysis pipelines to low-resolution setups

9.5 Creating classification networks based on tail dynamics over time
List of Figures

2.1 Zebrafish ................................................................. 7
2.2 Zebrafish phylogenetic tree ......................................... 7
2.3 Beat and glide motion .................................................... 9
2.4 Sterotypical zebrafish movements ..................................... 10

3.1 Perceptron mode .......................................................... 13
3.2 Multi-layer perceptron. Green: Inputs; Blue: Hidden Layer; Red: Outputs .................................................. 14
3.3 Sigmoid activation function ........................................... 15
3.4 \( \tanh \) activation function ........................................... 16
3.5 ReLU activation function ............................................... 16

4.1 State of the art behavior classification pipeline .................. 24
4.2 Movement centroids ..................................................... 25

5.1 Simplified schematic of the setup’s components. Not to scale. ............... 28
5.2 Example image of tracking conditions. Zoomed in version of the larvae inside the red square. ................................................. 29
5.3 Steps of the tail tracking algorithm ................................... 30
5.4 Number of samples per class .......................................... 31
5.5 Vector representation of a bout ....................................... 33
5.6 \( \text{t-distributed Stochastic Neighbor Embedding (t-SNE)} \) embedding of 500 random points per class. Each color represents a class, according the previous nomenclature. .................. 34
5.7 Mean trajectory per class. .............................................. 35
5.8 100 random trajectories per class. The jumps in the traces occurs when the tracking algorithm failed to detect the fish as it was moving. ................ 36
5.9 Noisy bout. ............................................................... 37
5.10 Noise identified as a bout. ............................................. 38
5.11 Low quality bout classifier ........................................... 40
## List of Tables

5.1  Number of samples per bout type in entire dataset. .......................... 31
5.2  Number of samples per bout type on clean dataset. .......................... 47
6.1  Animal trajectory network dimensions. ........................................... 54
6.2  Animal posture network dimensions. ............................................. 55
6.3  Animal posture and trajectory network dimensions. ........................... 55
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>AL</td>
<td>Active Learning</td>
</tr>
<tr>
<td>AS</td>
<td>Approach Swim</td>
</tr>
<tr>
<td>BS</td>
<td>Burst Swim</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>DPF</td>
<td>Days Post Fertilization</td>
</tr>
<tr>
<td>GD</td>
<td>Gradient Descent</td>
</tr>
<tr>
<td>HAT</td>
<td>High Angle Turn</td>
</tr>
<tr>
<td>JAABA</td>
<td>Janelia Automatic Animal Behavior Annotator</td>
</tr>
<tr>
<td>KNN</td>
<td>K-Nearest Neighbor</td>
</tr>
<tr>
<td>LED</td>
<td>Light-Emitting Diode</td>
</tr>
<tr>
<td>LCS</td>
<td>Long Capture Swim</td>
</tr>
<tr>
<td>LLC</td>
<td>Long Latency C-Start</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi Layer Perceptron</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>ResNet</td>
<td>Residual Neural Network</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>RT</td>
<td>Routine Turn</td>
</tr>
<tr>
<td>SAT</td>
<td>Shadow Avoidance Turn</td>
</tr>
<tr>
<td>SCS</td>
<td>Short Capture Swim</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SLC</td>
<td>Short Latency C-Start</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>t-SNE</td>
<td>t-distributed Stochastic Neighbor Embedding</td>
</tr>
</tbody>
</table>
1 Introduction

Contents

1.1 Motivation .................................................. 3
1.2 Goals of the Project ........................................ 4
1.3 Contributions ............................................... 4
1.4 Document Structure ....................................... 5
The brain is the most complex organ in the human body and despite many efforts we still do not fully understand how it works. Unlike current artificial intelligence algorithms, which can only handle a single task, the brain handles several complex functions such as sensory perception, planning, motion execution and, learning. Despite its complexity, the nervous system has one main purpose, to produce behaviors. As behavior is fundamental to understand the function of the brain, it is instructive to look toward what has been studied in regard to animal behavior. Therefore I turn to ethology, the study of animal behavior under natural conditions; a prime tenet of ethology states that the brain produces meaningful behaviors through the organization of simple and stereotyped movement primitives into complex sequences [1]. Literature suggests that these movement primitives can be produced by dedicated neuronal circuits and thus, the resulting behavior could be explained by the transitions between the activation of said circuits [2]. As such, studying the brain’s final output, the behavior that an animal produces, and identifying and characterizing the simple motor primitives can help understanding how the brain and its underlying mechanisms work.

In order to fully test this idea, the field needs animal models that have complex behaviors that can be studied in the lab, and tools that provide good access to the nervous system of the animal. Zebrafish larvae are small transparent animals that can perform complex innate behaviors. Because these larvae are small, and the skull and brain are transparent to light, it is possible to image the neuronal activity across the entire brain with single cell resolution. In addition, the larvae can be easily tracked in a lab environment and their movements have natural separations, making zebrafish larvae an ideal animal model to study the relationship between behavior output and its underlying brain activity.

Larval zebrafish behavior has been studied in the lab for decades, and recently with the improvements in digital recording and storage of video data, many datasets pertaining to their behavior have become available. The work described by Marques et al. [3] in the Vision to Action Lab created what is currently the most complete description of the zebrafish’s locomotor repertoire, opening the door to a refined study of the neural control of the zebrafish larvae motion and its underlying motor circuits.

1.1 Motivation

While the approach of Marques et al. has produced a powerful description of larval zebrafish behavior, it has a major drawback for further use: the complexity of the behavior-classification pipeline. In order to classify a movement into one of the possible types, Marques converts the tail movements into 73 kinematic parameters that are projected to a 4D space via Principal Component Analysis (PCA) where the classification label is attributed via K-Nearest Neighbor (KNN). The KNN classifier segments the data into 13 unique movements motifs. This description of motion was created in a fully unsupervised way, nonetheless, the clusters detected by the clustering pipeline identified movements which the fish
have been confirmed to perform.

This complex pipeline is not ideal for a fast real-time movement identification, which is key for mapping the relationship between the current movement of the larvae and the neuronal activity in the brain via closed loop stimulation. In addition, sometimes is not possible to obtain the tail angle and position data needed to be able to accurately classify the data, leading to a lack of utility of this approach for the field.

1.2 Goals of the Project

I will address these shortcomings by taking advantage of the Marques dataset; this project will focus on studying how deep learning techniques can be used to improve current state of the art zebrafish behavior analysis methods. The aim of this project is to create a set of classifiers that achieve robust classification of the zebrafish locomotor repertoire in accordance to the work of [3], using only some of the tracking data used by Marques and colleagues, but without the need for any kinematic feature calculation and without using any dimensionality reduction techniques. This will open the door to real-time classification of bout identity across different experimental conditions.

1.3 Contributions

Key contributions of this thesis are presented below.

1. Three different Deep Artificial Neural Network (ANN)s that achieve high accuracy featureless classification of the zebrafish larva locomotor repertoire in accordance to the work of Marques and colleagues [3], the most complete description of the larva’s behavior to date.

   (a) A network that classifies the behavior based on the movement of the larva’s tail;

   (b) A second network that classifies the behavior using the trajectory of the larva as it moves;

   (c) A third network that combines the two previous inputs to increase classification performance when both information sources are available.

2. A deep binary classification network that was created to automatically detect the low-quality samples over the entire dataset. A tool that can be deployed again in the future as a data cleaning step in more complex analysis pipelines.

3. A curated dataset containing several million zebrafish larva movements that can be explored in future projects.
1.4 Document Structure

This document organized as follows. Chapter 2 introduces zebrafish as an animal model both in genetics and neuroscience, and how its behavior is measured. Chapter 3 explains the basic concepts related to artificial neural networks used in this work. A review of the current state of the art techniques used in neuroscience for animal tracking and behavior classification can be found in Chapter 4. In this work we use a big dataset containing millions of zebrafish movements. Data acquisition, organization, and cleaning are addressed in Chapter 5. Chapter 6 details the network architectures used in this work and describes the methodology used to train such networks. The results of the optimization can be found in Chapter 7. Concluding remarks are discussed in Chapter 8. Finally, in Chapter 9 future work is addressed where some possible ideas that build on this work are explored.
Zebrafish, an animal model for genetics and neuroscience

Contents

2.1 Zebrafish as an animal model ........................................ 7
2.2 Study of zebrafish larvae locomotion ................................ 8


2.1 Zebrafish as an animal model

Zebrafish (see Figure 2.1), *Danio rerio*, is a small fresh water fish native to southern Asia that can be found in regions such as India, Pakistan, Bangladesh, or Nepal. Because this small teleost fish has a fast generation time, large clutches with over 100 embryos per mating pair, small size, external development, and larvae transparency, it was initially used as an animal model in the 1980s for developmental and genetic studies in a cost-efficient manner.

![Figure 2.1: Left: Adult zebrafish, Right: zebrafish larvae.](image)

Despite zebrafish being phylogenetically distant from humans (see Figure Figure 2.2), they have a nervous system organized similarly to other vertebrate. Additionally, zebrafish have 10,660 genes in common with the human, mouse and chicken genomes and 70% homology to humans [4–6].

![Figure 2.2: Phylogenetic tree for comparative analyses of zebrafish brain versus other model organisms and humans. Zebrafish and mammals including humans have generally similar brain characteristics. Adapted from [7]](image)

Nowadays, a large library of transgenic and mutant fish is available, enabling researchers to target specific cell types or provide vertebrate models of human neurodevelopmental, neurological and neurodegenerative diseases such as Alzheimer, epilepsy or autism [6–10]. The development of state of the art optical imaging methods and optogenetics has placed zebrafish as an interesting vertebrate model for systems neuroscience as well. Because zebrafish larvae are small and transparent, they provide an
attractive model for optical imaging of their brain development and function. Some of the techniques that have been recently used include Two-Photon Scanning Microscopy, Selective Plane Illumination Microscopy, and Light-Field Microscopy. These approaches can simultaneously monitor the activity dynamics over large brain regions, potentially even the whole brain.

The extensive and growing toolbox of genetic modifications possible in the fish complement the available microscopy techniques by providing ways of reporting neuronal activity or directly controlling that activity with light. Optogenetic reporters such as GCaMP [11], a genetically encoded calcium indicator that changes its fluorescence properties in response to the binding of $Ca^{2+}$, can be expressed in selected populations of neurons in the fish’s brain. Calcium normally enters neurons when they are active, allowing one to visually capture neuronal activity through the fluorescence changes of the GCaMP indicator. Optogenetic actuators such as channelrhodopsin [12] induce or suppress neuronal activity in response to light, and can be expressed in zebrafish neurons, allowing researchers to probe the causal role of neuronal activity in selected populations of neurons.

Unlike in other animal models such as mice, these manipulations are performed without the need for surgery or anesthesia, requiring only the larva to be head-restrained in agarose, leaving the eyes and the tail free to move [13]. This ability to simultaneously monitor sensory and motor areas in a behaving animal – due to the small size of the brain in comparison to the field of view of available microscopes – make zebrafish an ideal model for the holistic approach on how the brain generates behavior [14]. From a pure ethological point of view, the zebrafish has even more advantages as an animal model to study questions about behavior.

### 2.2 Study of zebrafish larvae locomotion

Like most fish, freely swimming zebrafish navigate in 3D environments, performing different behaviors at different depths in the water column. However, zebrafish larvae can be experimentally confined to shallow waters, reducing their behavior space into a 2D plane that can be easily recorded with a single camera without any stereoscopic vision tools.

The zebrafish larvae propels itself through patterns of body undulations consisting largely of curvature of the tail in the horizontal plane. In addition, the larva aids its swimming using rotations of the eyes and movements of the two pectoral fins. All of these movements can be easily tracked and broken down into more elementary units. In fact, zebrafish locomotor patterns are characterized by swimming episodes intermingled with non-swimming episodes called "beat and glide" (see Figure 2.3). The discrete segments formed by the beat and glide swimming pattern in larvae are called tail bouts and they last between 80 to 400 milliseconds, with beat frequencies between 30 and 100 Hz [15].

Zebrafish larvae as young as five days post-fertilization already exhibit a rich repertoire of innate be-
behaviors that enable them to explore the world [16,17]. Observe the examples in Figure 2.4: Figure 2.4(A), fish can maintain their position in a changing environment with slow swims [18,19]; Figure 2.4(B), they can hunt [20,21]; Figure 2.4(D), they escape from threatening stimuli [22–25]; Figure 2.4(C), and they avoid other larvae [26]. This repertoire of behavior makes zebrafish larva an attractive model to study in the lab.
Figure 2.4: Stereotypical tail movements. Super imposed images of the larva’s body as it moves. The trajectory of the head is shown by the white line and the black arrows represent the head orientation at the beginning and end of the bout. Adapted from [27].

(A) Small tail bend and low beat frequency. 
(B) Fine orientation turn associated with prey capture. This is achieved by a pronounced bend at the tip of the tail. 
(C) Common turn used as a reorientation mechanism. This movement is described by its slow speed and large bend angle. 
(D) High velocity short duration turn associated with predator avoidance behaviors. Named after the C shape the larva’s body generates at the beginning of the movement.
3.1 Training a Neural Network .................................................. 13
3.2 Activation Functions ....................................................... 15
3.3 Splitting the dataset ......................................................... 17
3.4 Regularization ............................................................... 17
3.5 Deep Learning .............................................................. 18
ANNs are systems inspired in some respects by the biological neural networks that constitute the animal brain.

Following the work of McCulloch and Pitts [28] demonstrating the potential for even a simple neural network to perform any arithmetic or logical function, in 1958 Frank Rosenblatt created the perceptron [29], the simplest kind of feed forward neural network.

![Figure 3.1: The Perceptron model. Connection weights $W$, that model the relative influence of each input; Neuron bias $b$, a constant responsible for shifting the function so that it is not constrained to the origin; Activation function $g$ that adds non-linearity to the perceptron;](image)

In this context, a neuron is merely a function $h(x) = g(b + \sum_i w_i x_i)$ that creates a non-linear mapping between the inputs and the output. The full modeling potential of the perceptron arises when several neurons are layered together feeding the output of one as the input of the next, thereby creating a multi-layer perceptron, as shown in Figure 3.2.

The Multi Layer Perceptron (MLP) is used in supervised learning tasks where there is a known label (or class) for each input given to the network.

### 3.1 Training a Neural Network

Training a neural network involves finding the set of weights and biases that minimizes a given loss function. The loss function is associated with the difference between the known output and the output the network predicts.

Though unpopular in the early days of machine learning, one of the most common optimization algorithms used today to minimize the error function is Gradient Descent (GD). In its simplest form, GD consists of iteratively taking steps in the weights space, traveling in the opposite direction of the gradient...
A step of the GD algorithm, can be defined as:

\[ W^{n+1} = W^n + \alpha \nabla E, \]

where \( \nabla E \) represents the gradient of the error function \( E \) relative to the weights \( W \), \( n \) is the training iteration and, \( \alpha \) the learning rate. Several variations of the basic GD algorithm try to improve its robustness and convergence time. A review of such variations can be found in [30].

Current machine learning problems involve large training sets, and in these scenarios even the most optimized versions of GD are computationally prohibitive in real world problems. This is due to the need for calculating the real gradient of the entire training set in order to know in which direction to travel in the weight space. Therefore, it is a common practice to use Stochastic Gradient Descent (SGD) instead. Instead of sweeping through the whole training set for each iteration, SGD allows for steps to be taken using only a reduced set of data called a batch.

The weight space is often non-convex. Since this process can run forever, a manual stopping criteria must be used to terminate the training. Common stopping criteria include running the algorithm a fixed number of iterations or having it stop when the error is lower than a given threshold value.
3.2 Activation Functions

The activation function plays a critical role determining the output of the neuron in response to a set of inputs as it imposes a non-linear response to the neuron’s inputs. As such, changing this function affects the behavior of the neuron. There are several activation functions designed with different purposes. In the following subsections, we describe three of the most commonly used activation functions: the sigmoid, the hyperbolic tangent, and the rectified linear unit.

3.2.1 Sigmoid

The sigmoid non-linearity is defined as:

\[
1 \\
1 + \exp\{- (b + \sum_i w_i x_i)\}
\]

(3.2)

Where \(w\) is the weight for the input coming from neuron \(x\) and \(b\) the neuron bias. It takes as input a real-valued number and then confines it between 0 and 1 as Figure 3.3 shows.

![Sigmoid activation function](image)

**Figure 3.3**: Sigmoid activation function

Despite being historically one of the most used, the sigmoid has the major drawback of saturating the gradient; if the neuron’s activation function happens to saturate, the local gradient will often tend to zero and effectively shutdown the backpropagation update process during training. Therefore, it has recently been dropped in favor of other activation functions.

3.2.2 Hyperbolic Tangent

The hyperbolic tangent (tanh) non-linearity is defined as:

\[
\frac{\exp\{2(b + \sum_i w_i x_i)\} - 1}{\exp\{2(b + \sum_i w_i x_i)\} + 1}
\]

(3.3)
Where $w$ is the weight for the input coming from neuron $x$ and $b$ the neuron bias. It takes as input a real-valued number and then confines it between -1 and 1 as Figure 3.4 shows.

![Tanh Function](image)

**Figure 3.4: tanh activation function**

Similarly to the sigmoid, this activation function saturates. However, in this case the output is zero-centered which helps since the neurons in the later layers of the network would be receiving inputs that are zero-centered. For this reason the tanh non-linearity is usually preferred to the sigmoid.

### 3.2.3 Rectified Linear Unit

The Rectified Linear Unit (ReLU) non-linearity is defined as:

$$\max(0, b + \sum_i w_i x_i).$$

(3.4)

Where $w$ is the weight for the input coming from neuron $x$ and $b$ the neuron bias. Unlike the other activation functions, the ReLU only thresholds the signal below zero, as displayed in Figure 3.5.

![Rectified Linear Unit (ReLU)](image)

**Figure 3.5: ReLU activation function**

ReLU considerably accelerates the convergence of stochastic gradient descent when compared to
the sigmoid or tanh functions as this function’s gradient can only be 0 or 1. Since the neuron’s output is not bounded, the weights are free to update in such a way that the neuron will never activate again if a large gradient flows through the neuron and thus stalling the training process.

3.3 Splitting the dataset

If the entire dataset is used to train the network, there is no way to methodically test its ability to generalize and have good performance in samples it has never seen before which can lead to overfitting and other problems with testing the actual usefulness of the network.

During the process of training a neural network the data is usually split in three sets:

- **Training set**: as the name implies, it is used to train the network. These are the samples that the algorithm will use to adjust its weights and biases.

- **Validation set**: this set is continuously being evaluated during training though the samples in this set do not contribute directly to weights and biases adjustment. This set is used to monitor the ability to generalize to new samples and to improve the network hyperparameters (e.g. learning rate, momentum, number of epochs, batch size).

- **Test set**: This set is only evaluated after all hyperparameters of the network have been optimized with the two previous sets. Performance in this set is the only measure of the network’s performance that can be used to compare across networks, since its performance was not influenced by the tuning of the hyperparameters.

3.4 Regularization

Most networks used to solve complex tasks have a big number of adjustable parameters. The abundance of parameters to adjust can lead to considerable overfitting of the network. One could combine the predictions of many different models as this technique has been shown to be quite good at reducing test errors [31, 32]. However, this will greatly increase the complexity and training time of the model.

Dropout [33] is a fairly recent, yet very common, regularization technique designed to reduce overfitting. It involves setting the output of each hidden neuron to zero with a certain probability. This way, selected neurons do not contribute to the forward pass and do not participate in backpropagation. Disabling the output of random neurons every time an input is presented is effectively forcing the neural network to sample different architectures, except all these architectures share weights. Because each architecture shares weights with others, this reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of other particular neurons. Therefore, the network is forced to learn more
robust features that are useful in conjunction with many different random subsets of the other neurons. Dropout is only used during training, so during the evaluation all neurons contribute to the final input.

Regularization techniques like Dropout help control the training of neural networks and reduce overfitting. The usage of these techniques is key for successfully training deeper networks with more stacked layers of perceptrons as they tend to overfit.

3.5 Deep Learning

Complex artificial neural network models that are based on the organization of the brain have long been theorized, but lack of computational machinery to drive such models prevented practical implementations of these complex models. In recent years, the increase in computational power and the development of tools like TensorFlow [34] has facilitated the implementation of more complex artificial neural networks.

The work of Hubel [35] in the 1950s describing visual receptive fields in the visual cortex of cats led Fukushima [36] in the 1980s to develop the concept of convolutional and downsampling layers [36]. These layers are the two core building blocks of Convolutional Neural Network (CNN)s, used today in multiple contexts including image recognition [37], natural language processing [38] and are present in game playing algorithms such as AlphaGo [39].

Network architectures such as Residual Neural Network (ResNet) [40] modeled after the pyramidal cells found in the cortex, have been demonstrated to improve the performance of standard deep convolutional networks, as measured by achieving higher scores on image recognition tasks [40].

In the brain, neurons perform highly nonlinear operations to transform their inputs into spiking outputs and communicate via connections that are rarely unidirectional but more often bidirectional, potentially via more than one synapse [41]. Inspired by cortical neurons, ResNet inputs are not limited to the output of the layer directly before them, and can sample the outputs of different layers. This ability to sample inputs from multiple layers, allows ResNets to overcome the problem of vanishing gradients, common in very deep artificial networks, where as more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train. This happens because certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1 and therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small and, as the small derivatives over the network are multiplied together, the gradient decreases exponentially as we propagate down to the initial layers.

In a Recurrent Neural Network (RNN), each unit performs a simple operation: at each time instant, it applies a nonlinear function to the sum of its inputs and passes this to other neurons. This simple mechanism provides rich patterns of activity in each neuron, even with a network of small size. RNNs
are a key component of Long Short-Term Memory (LSTM) networks [42], used today to classifying, processing and making predictions based on complex time dependent data.

Before it was possible to leverage the power of deep artificial neural networks, the focus of researchers when creating machine learning based solutions was designing and perfecting the features that would be used to represent the data. Deep artificial neural networks take advantage of their high number of degrees of freedom to learn their own internal representation of the data, creating models that are free from the human bias associated with feature selection and crafting. Due to this autonomy from handcrafted features, it makes sense to revisit problems that have been previously solved using human made features and explore how deep learning based models can improve or provide a different perspective such problems.
4 Related Work

Contents

4.1 Supervised learning techniques for animal tracking . . . . . . . . . . . . . . . . . . . . 21
4.2 Supervised learning techniques for behavior classification . . . . . . . . . . . . . 22
4.3 Unsupervised Learning Techniques for behavior classification . . . . . . . . . . . . 22
4.4 Manual descriptions of zebrafish larvae behavior . . . . . . . . . . . . . . . . . . . . 23
4.5 Supervised classification of zebrafish larvae behavior . . . . . . . . . . . . . . . . . . 24
4.6 Unsupervised classification of zebrafish larvae behavior . . . . . . . . . . . . . . . . . 24
The quick spread of powerful personal computers enabled the research community to develop programs that allow the annotation of video recordings of animals performing different behaviors over time. Here a human would annotate a video recording by attributing one or more behaviors to each video frame according to some generally accepted criteria, while classifying the behavior into generally accepted types of movements. Although enabling correction and classification by multiple subjects, frame by frame video annotation techniques are time consuming and their performance are subject to large inter-observer variability.

This chapter explores cases where supervised machine learning techniques have been used to improve the study of animal behavior by automating animal tracking and behavior annotation. It also elaborates on previous work defining animal behavior without relying on handcrafted features and how descriptions of the zebrafish larva behavior have improved over time as more complex machine learning algorithms have been applied to this domain.

### 4.1 Supervised learning techniques for animal tracking

Over the decades, several technological advancements have made capturing behavior data the increasingly easier. However, the task of extracting particular aspects of a behavior for further analysis is still time consuming due to the time it takes to assign behaviors to frames. One of the most common practices to aid the task of tracking specific body parts is to add reflective markers to the areas of interest. These markers are often intrusive, especially in small animals such as mice. As laboratory behavior is already likely to examine only a subset of true animal behaviors, one should aim to keep intrusive techniques to a minimum. Using state-of-the-art deep learning methods for human limb configuration detection, Mathis and colleagues created the DeepLabCut [43] toolbox that allows for a marker-free tracking of several body parts across different animals and objects. DeepLabCut requires only a minimal set of manually labeled frames to train the network, achieving performances comparable to humans.

DeepLabCut's network architecture revolves around strong body part detectors from an algorithm for human pose estimation called DeepCut [44]. DeepCut is built using extremely deep Residual Networks [40] with 50 layers in an architecture known for achieving outstanding performance in object recognition competitions. Taking advantage of transfer learning, the deep feature detectors are previously trained on the well-known image classification dataset ImageNet [45]. Transfer learning thus enables learning a new task through the transfer of knowledge from a related task that has already been learned, leading to a smaller amount of training on the new task to achieve high performances.

Technological advancements such as high-speed video cameras, infrared illumination and ultrasound microphones have not only made it easy to capture behavior, but have also enabled capturing behavior in ways that were never possible before; offering access to new spatial and temporal scales of behavior,
and thus have become vital tools in some domains of behavioral neuroscience [46]. Curiously, advances in the production of smartphones led to the creation of cheap miniaturized sensors that produce robust measures of behavior. These sensors can be unintrusively fixed to an animal and can continuously transmit information over long periods [46].

Despite all these new data sources, the most common way to analyze behavior data is still to use manually annotated ethograms. Nonetheless, these technological advancements in data acquisition have led to the creation of copious amounts of data, introducing the field of ethology to the realm of big data and its struggles [46].

4.2 Supervised learning techniques for behavior classification

With the introduction of big data levels of information to parse and understand, machine learning based solutions such as Janelia Automatic Animal Behavior Annotator (JAABA) [47] have been created to enable researchers to automatically compute statistics describing video of behaving animals. Users encode their intuition about the structure of behavior by labeling the behavior of the animal in a small set of video frames. The system then converts these manual labels into behavior detectors that can then be used to automatically classify the behaviors of animals in the remainder of the dataset with high throughput.

Because these approaches are supervised, they must be trained with manually labeled data. This has the drawback of being dependent on the quality of the data provided by the manual labels, making the system prone to human bias. Furthermore, supervised systems like this will never create new labels for new data nor will they be able accurately predict a new movement primitive to which they have not been trained. With the mindset that ideally the behavior description should manifest itself directly from the data, some unsupervised solutions for behavior data classification have been developed in recent years.

4.3 Unsupervised Learning Techniques for behavior classification

The work of Stephens and colleagues [48] showed that it is possible to describe motor behavior as a projection of the body shape onto a set of low dimensional templates whose strength varies continuously. In this work, using only a 4 dimensional space, the team was successfully able to capture 95% of the variance of all the space of shapes adopted by the nematode Caenorhabditis elegans. In addition, within this framework, known discrete behaviors still exist but they emerge as attractors of the underlying dynamics.

Berman and colleagues [49] demonstrated that it is possible to construct behavioral space for freely
moving Drosophila melanogaster starting with just high-quality videos of moving animals. Here images are aligned to a reference and then decomposed via PCA into a relatively low-dimensional set of time series directly from the pixel intensity values of the images. These times series are then used to create a spectrogram for each postural mode separately and each time point is mapped into a two-dimensional plane via t-SNE [50]. In this space, sequences of movements that pause near repeatable positions represent stereotyped behaviors. This space allowed the team to observe that the flies exhibit approximately 100 stereotyped recognizably distinct behaviors such as walking, running, head grooming, and wing grooming that are interspersed with frequent bouts of non stereotyped behaviors.

Combining depth sensing cameras and machine learning techniques, the work developed by Wiltschko and colleagues [51] showed that the 3D pose dynamics of mice are structured at the sub-second timescale and that these fast dynamics effectively describe mouse behavior as a series of reused and stereotyped modules with defined transition probabilities, creating a mouse body language that is built from identifiable components and is organized in a predictable fashion.

These automated unsupervised learning techniques are better equipped to deal with large and often high-dimensional data and have the advantage of being free from the common human biases of supervised machine learning techniques. However, the solutions to which unsupervised methods usually converge are quite different from the previous manual observations of ethologists which can complicate the interpretation of the algorithm’s results.

4.4 Manual descriptions of zebrafish larvae behavior

As the main focus of this work is the behavior of larval zebrafish, it is important to review how machine learning tools have been used in the past to improve the description of such behaviors. Like Sections 4.2 and 4.3, the following Sections explore the usage of supervised and unsupervised classification algorithms to describe the zebrafish larvae’s locomotor repertoire.

As described in Section 2.2, zebrafish larvae move using a beat and glide motion, this way of locomotion provides a natural segmentation to the zebrafish larva movements as the animal moves in a burst of activity and then stops for some amount of time. As such, most of the literature describing the zebrafish’s locomotor repertoire revolves around identifying kinematically distinct categories of swim bouts. One of the first descriptions of zebrafish’s locomotor repertoire was created by Budick and O’Malley in [16]. The work consisted on using high-speed cameras to film larvae responding to several stimuli such as light flashes or feeding. The footage was then manually annotated and different features of motion such as the “Head–tail angle” or the “Bend mid-point” were measured for each frame during the movements. The team showed larvae use different kinematic parameters to respond to different stimuli and the features defined in this work, became the standard kinematic features of the zebrafish’s locomotor repertoire.
By analyzing the evolution of the kinematic parameters of the zebrafish larva motion over feeding and slow movement behaviors, [20] describes how the larva is capable of fine axial motor control revealing an unforeseen complexity of the descending motor control pathways in zebrafish.

4.5 Supervised classification of zebrafish larvae behavior

Manual labeling techniques are both laborious and limited in scope for high-throughput behavioral screens. In an effort to speed up the data labeling process Mirat et al. [52] created ZebraZoom, a toolbox that uses known kinematic parameters to classify the zebrafish larvae behavior into forward swims, routine turns and escape responses. ZebraZoom’s classification architecture consisted of two consecutive Support Vector Machine (SVM)s. They first classified the data as either a forward swim or any other movement type. The data that was not classified as being a forward swim would be sorted as either a routine turn or an escape response by the second SVM. While this two step classification can be efficiently applied to large datasets, it lacks flexibility when confronted with behaviors that it has not been trained on.

4.6 Unsupervised classification of zebrafish larvae behavior

ZebraZoom, while being a good way of classifying a limited set of behaviors, falls short when other behaviors not trained are being performed. Using an unsupervised approach might help to correct this. In fact, the work of Marques et al [3] showed how unsupervised clustering methods can be used to reveal new structures in the data without any prior knowledge how the data should be separated. In this work, the group gathered several million samples of zebrafish movements across a wide range of contexts like spontaneous swimming, prey capture, social interactions and, predator avoidance.

This collection of samples combined with a robust clustering algorithm exposed the existence of thirteen bout types. In these 13 bout types, the algorithm indicated the existence of seven previously described bout types and revealed six new movement types.

Figure 4.1: Analysis pipeline of the two-stage clustering approach used to identify and classify bout types. The first clustering step is marked in green, and the second clustering step is highlighted in red. Adapted from [3].
As described in Figure 4.1, the algorithm started from the evolution of the tail data over time during the larva's movement, and an array of 73 kinematic parameters was calculated for each bout. These arrays of kinematic parameters were then projected via PCA to a 4D space where it was clustered using a new density-based algorithm called *clusterdv* [53]. Because this clustering algorithm can make reliable unsupervised estimations of the number of the density-based clusters across several datasets, the team was able to reliably pool the data representing the same behavior over different contexts.

The similarity between the set of all clusters generated by the first classification was used to embed the clusters into a lower dimensional space using t-SNE [50] where they were clustered once again using the same algorithm, repeatedly producing a space with 13 density peaks that define the center of the cluster for each one of the bout types.

The results of these processes create the most complex description of the zebrafish larva locomotor repertoire and defined for the first time the movements of the zebrafish larvae using only unsupervised learning techniques. An example movement for each bout type can be found in Figure 4.2 and this is what will be considered as the ground truth movement in this project.

![Figure 4.2: Sequence of tail postures associated with the bout nearest to the centroid of each cluster. Adapted from [3]](image-url)
5

The Dataset

Contents

5.1 Data acquisition ..................................................... 27
5.2 Understanding the original dataset ................................. 30
5.3 Data cleaning .......................................................... 36
5.4 Final dataset description ............................................ 45
5.1 Data acquisition

In this work I will use the data from Marques et al. [3] as the ground truth label for the tail bout classes. To fully take advantage of this dataset, it is important to understand the origin and structure of the data before proceeding.

5.1.1 Experimental setup to record behavior

This data was acquired in two different types of experimental setups: 1) high resolution imaging of the eyes and tail of a larva, or 2) a low resolution imaging of one or more larvae.

For some of the stereotyped movements that larval zebrafish perform, it is known that the larva’s eyes also move with the tail. This is readily apparent in the case of the prey capture behavior where the larvae converge their eyes to enhance stereoscopic vision. [16]. To observe the dynamics of such coordinated eye and tail movements, one setup was designed to capture high resolution data of the larva’s eye and tail at the same time. This setup’s arena consists of a (2.5 × 2.5) cm square.

For other behaviors, the movement of the eyes is not known to be important. Therefore, higher resolution can be traded for high throughput of data acquisition. A second setup was designed to make it possible to record up to 9 larvae in their own individual arenas, or to have a bigger arena with multiple larvae in it, as in the case of the experiments concerning social interactions. To capture a wider area, the data coming from these experiments has a lower spatial resolution, making eye tracking unreliable. As such, no eye tracking data is available in these datasets. Figure 5.1 shows the basic schematic for all the setups used in the experiments.

The data for both setups was acquired using an infrared sensitive high-speed camera filming at 700 fps pointing downwards to the arenas. These arenas were illuminated from below by a custom built 850 nm infrared Light-Emitting Diode (LED) array. Visual stimulation was provided by a projector displaying images on the bottom of the arenas. In order to prevent the camera from filming the projected images, an 850 nm optical low-pass filter was coupled to the camera’s objective. For acoustic stimulation protocols, a set of small speakers was added to the setups.

5.1.2 Tracking software

The size of the arenas can constrain the animal’s behavior, so the arenas had to be large enough to avoid this. Since zebrafish larvae are so small, measuring around 5mm at 6 Days Post Fertilization (DPF), the cameras need to have a high spatial resolution to be able the film the animal in detail; the cameras produce (950 × 950) pixel images. In addition, in order to capture even the fastest movements with high image quality, the cameras used in this project acquire data at 700 fps to ensure high temporal resolution. These high spatial and temporal resolutions lead to a data acquisition source that produces
a notably large amount of data, around 631.75 MB/s. Additionally, for some experiments, it is required to know the larva’s position and orientation in real time in order to reliably trigger the stimuli in the same way relative the larva. It is infeasible to store such data volumes from both a technological and data managing perspectives.

Taking into account that the raw image data cannot be stored and that the larva’s position must be known, a program was created to track the larva in real time and to control the stimuli, recording only enough metadata – position, orientation and tail conformation, and eye angle – that would allow a reconstruction of the larva’s behavior during the experiment. Figure 5.2 shows the an example of the tracking conditions. Inside the zoomed in square, it is possible to see the black dots that represent the tracked points along the larva’s tail.

5.1.2.A Tracking Algorithm

The tracking algorithm starts by isolating the larva from the rest of the arena by subtracting the current image with a model for the background. The background model was obtained by calculating the mode for each pixel across several frames spread over a small-time interval to ensure that the larva is moving between the frames, avoiding its incorporation in the background model.

In the high-resolution arenas, designed to track both eyes and tail, the fish was found by locating one of the eyes through the identification of the darkest pixel in the image, as the eyes are reliably the darkest part of the fish. After one eye was found the other eye was located by finding the darkest pixels on a radial search along a circumference centered on the first eye with a radius around the average value for the inter-eyes distance.

The shape of the eyes is calculated by performing a modified version of flood fill algorithm [54]
seeded on the two previously found points to return the first and second order image moments for each eye. Using these moments, the center of mass of each eye is obtained and the middle point between the centers of mass of both eyes defines the larva’s head position (Figure 5.3 3).

The direction of the tail was obtained by performing a radial search on a circumference centered on the point defining the head position and with a radius of 450 µm, which is the usual distance to a recognizable body landmark (the swim bladder). The curvature of tail is calculated by traveling a series of up to ten consecutive tail segments 310 µm apart (Figure 5.3 4). Starting at the swim bladder, the points along a 120 degree arc around the previous tail segment are used to calculate the rotation of the next tail segment. The direction of the rotation is calculated through the center of mass of the intensity of the pixels around the arc (Figure 5.3 5).

In the low resolution arenas, the fish was tracked using a method that has been used in previous studies [18, 22]. In this method, after the background is subtracted, the image is blurred using a boxcar filter of 9 pixels, transforming the fish into a blob that is the fish’s body with single peak. The same flood
Figure 5.3: Steps of the tail tracking algorithm. 1) Background subtracted image. 2) Determining the position of the eyes. 3) Calculating midpoint between the eyes. 4) Determining tail direction by searching for a recognizable body landmark. 5) Consecutive arc search to find segments along the tail. 6) Tracked larva

fill algorithm defined above is used on this blob to estimate its center of mass and orientation. The tail curvature is then calculated using the same algorithm as in the high resolution arenas starting at the blob’s center of mass.

Despite each arena having its own method for defining the starting point for the tail search, all tail data is interpolated to a set of reference points separated by 310 µm starting at a point 1345 µm behind the eyes.

5.2 Understanding the original dataset

The dataset produced by Marques et al. is a collection of 3,640 experiments containing a library of 3,233,374 bouts, distributed across 1,151 files for a total of 480 GB of data. This data consists of the information produced by the tracking software, all the kinematic parameters of movement used by the classifier described in [3], the classification results, information about the stimulation protocols used and all metadata indicating when each movement occurred during the experiment.

In this work I will focus on being able to classify the data with high accuracy while removing the dependency on kinematic parameters that need to be calculated *a posteriori*.

In order to achieve real time classification, the network must only rely on information that is present during the experiment such as the larva’s position and orientation in the arena.
Table 5.1: Number of samples per bout type in entire dataset.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Approach Swim (AS)</th>
<th>Slow1</th>
<th>Slow2</th>
<th>Short CS</th>
<th>Long CS</th>
<th>Burst Swim (BS)</th>
<th>J-Turn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>59,436</td>
<td>229,427</td>
<td>446,971</td>
<td>25,112</td>
<td>11,727</td>
<td>9,900</td>
<td>70,506</td>
</tr>
<tr>
<td>Right</td>
<td>56,338</td>
<td>212,670</td>
<td>428,737</td>
<td>19,051</td>
<td>9,647</td>
<td>9,093</td>
<td>68,125</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Direction</th>
<th>High Angle Turn (HAT)</th>
<th>RT</th>
<th>Shadow Avoidance Turn (SAT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>178,925</td>
<td>465,112</td>
<td>67,658</td>
</tr>
<tr>
<td>Right</td>
<td>163,793</td>
<td>459,154</td>
<td>66,245</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Direction</th>
<th>O-Bend</th>
<th>Long Latency C-Start (LLC)</th>
<th>Short Latency C-Start (SLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>13,000</td>
<td>40,536</td>
<td>14,365</td>
</tr>
<tr>
<td>Right</td>
<td>12,134</td>
<td>40,300</td>
<td>13,059</td>
</tr>
</tbody>
</table>

5.2.1 Class imbalances in the dataset

It is know that, depending how the larva initiates the movement, the wave front that describes the movement of the tail can start with a negative or positive curvature propelling the animal to the left or to the right. The original work of Marques et. al [3] did not try to discriminate between left and right movements. There are cases when it would be useful to know the direction of the movement, so I decided to expand the 13 original classes (see Figure 4.2) into 26, by separating the data according to the signal of the amplitude of the first bout.

Figure 5.4: Number of samples per class. For each color, the left and right bars represent movements to the left and to right, respectively, for the same bout type.

Figure 5.4 shows the number of samples for each bout type and Table 5.1 displays the number of movements per class discriminated by direction. Behaviors that the larva uses to reorient itself, such
as RT, are orders of magnitude more common than a movement only used to capture prey such as the Short Capture Swim (SCS). This discrepancy in the probability of the fish performing each particular bout led to increased complexity of the clustering and classification pipeline of [3] and is something that must be taken into consideration as well in this work.

5.2.2 Animal Posture

One of the biggest advantages of the zebrafish larva as an animal model in neuroscience is its transparent body and small size that allow imaging the whole brain at single cell resolution at the same time. In these preparations, the larvae are restrained with a small drop of agarose while the tail and eyes are free to respond to the stimuli. Restraining the larva will render any trajectory-based classification networks useless since the position of the animal in the arena will always be the same.

Also, the source of information for most features defined by Marques et al. are the tail angles and how they change over time. In this dataset, for each movement the larva makes, the position and angle of seven segments along its tail were recorded.

This dynamic system with seven inputs evolving in time with a visible internal structure presents itself as a good case for studying deep learning models such as the ones described in Section 3.5. Since tail bouts have a stereotypical duration, I chose to focus first on feedforward networks, although recurrent architectures could be considered in the future. As such, instead of presenting the data as seven concurrent signals evolving in time, the input of the network is a vectorized snapshot of the movement as can be seen in Figure 5.5.

Each movement is composed of a multidimensional time series representing the angle between seven points along the larva's tail over time. In this work, this multidimensional time series is flattened into a unidimensional series by concatenating the data across consecutive points.

This vector definition of a bout enforces a fixed size on the input data, thus not supporting arbitrarily long movements. The vast majority of movements in the dataset last up to 400 milliseconds which is consistent with previous studies. In addition, the work of Marques et al. denoted that the initiation of movement is more important than the end to achieve good performance in discriminating between movement types. With this in mind, I decided to keep all movements lasting up to 265 frames (378 ms).

The number of tracked tail segments varies between 5 and 10 across the experiments in the dataset. Nonetheless, the data for all tail points has been previously interpolated so that all the same point represents the same location across experiments and arenas. Since the original classification from Marques et al. was created by analyzing only data from the first seven tail segments, I decided just to discard all bouts coming from experiments where less than seven segments were tracked. For consistency, from the experiments where more than seven segments were tracked I took the first seven and discarded the remaining segments.
Using t-SNE to visualize this high-dimensional data, in Figure 5.6 it is possible to see that there is an evident organization in the data that is consistent with the classes described by Marques et al., even without using the kinematic features of the movement.

Movements that have similar dynamics, that evolve with comparable amplitude and shape are close to each other creating groups of bout types such as forward slow-moving movements where AS (cyan), Slow 1 (light blue) and Slow 2 (dark blue) cluster together.

Another group of bout types that it is very easy to identify is the one where movements have a first beat with bigger amplitude than all the consecutive beats where we can find O-bends (purple), SAT (lavender), RT (dark green) and HAT (light green).

Despite the lack of physical meaning for the axis in this embedding, the spatial distribution of the points represents how similar they are between each other. The clusters of points formed by elements of different classes, as the one found around \((-15, 40)\), indicate that there are points from several that share a similar shape. In addition, almost every class has a considerable amount of points randomly spread across the space. These inconsistencies in the embedding reveal that the dataset contains notably large amounts of poor quality data that, if used as training data in this project, can decrease the classifier's accuracy.
Figure 5.6: t-SNE embedding of 500 random points per class. Each color represents a class, according the previous nomenclature.

5.2.3 Animal Trajectory

The data acquired by Marques et al. used to create this complex description of the zebrafish larvae’s locomotor repertoire was collected using a custom tracking system that is not widely available to the research community.

Nonetheless, any tracking system must output the position of the target over time. If one can create a network that is able to classify the movements based only on the trajectory of the animal over time, the work of Marques et al. [3] could possibly be used in data created elsewhere. Owing to the stereotypy of tail movements that propel the animal forward, it is reasonable to assume that the animal’s trajectories during movement would be stereotyped as well. By subtracting the initial positions of all \((x, y)\) coordinates of the larva’s center of mass and rotating them according to the initial orientation relative to the arena, it is possible to visualize that such stereotyped trajectories do indeed exist.

By plotting the mean trajectory for a bout type, in Figure 5.7, it can be seen that known escape
movements, such as the O-Bends (purple) propel the larva backwards, while in precision movements, such as the AS (cyan) the larva moves slightly forward. As expected, in between these extreme cases we find orientation-related turns, such as the RT (dark green) and HAT (light green).

5.2.4 Errors in dataset

A close analysis of the dataset reveals that there are many unacceptable samples in the dataset. In some cases, these corrupted samples show the fish jumping across the arena, in other cases, they do not display any movement at all but random noise instead. These corrupted samples exist mostly due to limitations of the tracking software and the bout detection algorithms. Since these low quality samples will degrade the overall performance of our algorithms a data cleaning step is required.

5.2.4.A Tracking program errors

Due to the small size of the larva and the simplicity of the tracking algorithm, there were occasions where tracking software failed to track the animal and tracked something else instead, such as the edges of the arena, creating fast changes in the tail angles and quick jumps in the larva’s position in the arena as Figure 5.8 shows.
5.2.4.B Noisy samples

This large collection of movements was acquired over the course of several years on evolving setups. Changes in illumination conditions, proximity of the camera to the arena, and different materials for the arenas all contribute to the general quality of the dataset. It is possible to see that the quality of the samples varies greatly across the bout collection, without any apparent link between the type of experiment and experimental setup to the quality of the bout. In Figure 5.9, despite the degraded quality of the signal, it is still possible to see that a movement was clearly detected.

5.2.4.C Bout segmentation errors

The beginning and end of the movements in the dataset were identified by tracking sudden changes in the angle values. Because of this, spiky or noisy data originated from tracking errors can sometimes be identified as a bout. Figure 5.10 shows an example of random noise that was identified as a bout.

5.3 Data cleaning

Any classifier can only be as accurate as the training data allows it to be. In the best case, high numbers of low-quality samples will just make the training process harder as the bad samples will interfere with the correct calculation of the gradient. In the worst case, the network will learn patterns in the low quality
samples that it will subsequently use for classifying the data. Due to the existence of a considerable amount of bad samples originating in different points along the data analysis pipeline, it is imperative to clean the data in order to remove samples that could degrade the performance of the classifiers.

The original work developed using this dataset uses manually selected threshold values based on valid ranges for some kinematic features to sort good and bad samples. Since a key component of this work depends on not relying on the kinematic features of motion, I had to find strategies to identify the bad samples based only on the tail angles and trajectories. An added benefit of using a different approach is that by relying only on information contained in the recording, I will further decrease the the degree of human bias feeding into the classifier.

Some of the errors presented in Section 5.2.4 such as large displacements in the larva’s trajectory can be easily removed by a manually cleaning of the dataset.

However, there exist situations such as noisy signals or incorrectly segmented bouts generate samples with structured properties that can be picked up by the bout classifiers and influence the classification results.

Just as an example, let us imagine that for some reason, most bouts for an underrepresented class had a common type of error such failures in tracking caused by poor illumination.

Its not hard to see this as an useful, yet wrong, feature that the classifier could learn to represent that particular movement class.

In addition, it is difficult to create a mathematical definition that represents these types of errors,
complicating the creation of a simple check on the data to evaluate its quality.

Nonetheless it is possible to leverage the fact that this type of errors have a structure, and develop a classifier that was purposely trained to catch the bad samples, thereby creating a more general purpose tool that could be used in other scenarios to clean new data in an automated fashion.

5.3.1 Manual data cleaning

In trajectories that suffered from tracking errors, the larva’s position quickly changes from one place to the next. It is known that even in most extreme movements, the larva moves with a instantaneous speed around 5 cms/s. It is possible that several movements exceed that speed in the dataset. The bouts with instantaneous speeds greater than the larva is capable of performing should be the points where the tracking failed and the tracked object was found on a different place in the arena. Therefore, I simply removed the bouts where for any given point the speed was over 7 cms/s.

5.3.2 Improving the dataset by automatically detecting noisy tail movements

Unlike the errors in the larva’s trajectory that can be easily picked out by a simple analysis of the instantaneous speed, errors caused by failures in either the tracking or the bout detection algorithms produce changes in the tail angles that share some structural similarities with valid larva movements. These common properties complicate the task of setting threshold values for manually selected features.
Nonetheless, there are patterns and structures in the properties between valid and invalid movements that machine learning tools can learn and use to discriminate between high and low quality tail movements.

To take advantage of this, I decided to develop a classifier for the binary task of differentiating between high and low quality tail movements, described in the upcoming sections.

5.3.2.A Low quality bout classifier inputs

For a movement to be consistently good and clean, it must propagate smoothly along the tail across time. If we accumulate the tail angles over time and look only at this signal some properties emerge in the data:

- Errors caused by 1) the tracking of random points or 2) the bout detection algorithm identifying structured noise when the fish is stationary as a bout, should produce a very noisy signal

- Failures to track the tail during the bout should produce discontinuities in the smoothness of the signal,

- Failures to detect the beginning of the bout should still be visible on the data, and

- Finally, we can expect the general quality of the bout not to change much during the movement, so we can only analyze part of the movement instead of the whole bout.

With these assumptions in mind, I can simplify the problem of correctly labeling a bout as valid or invalid. Instead of looking at the data across the seven tail segments, I will look only at the cumulative sum of all the tail angles over time, and I will only analyze the movement on the first 100 frames, approximately 143 milliseconds, after the movement started as identified by the bout detection algorithm.

5.3.2.B Low quality bout classifier’s architecture

As depicted in Figure 5.11, our classifier is a feedforward network composed of four fully connected layers whose weights can be adjusted. Following the rational of Section 5.3.2, network input is the cumulative sum of the seven first tail angles over hundred frames, the angle for each time point is fed into one of the network’s 100 input neurons.

Since I am looking for a binary classification regarding the quality of our sample, where 1 indicates a bad sample and 0 a good sample, the network’s final layer has just a single neuron. The remaining three fully connected layers have 50 neurons each. The ReLU non-linearity is applied to the output of every fully-connected layer with the exception of the final one where the data is normalized through the usage of a sigmoidal activation function.
The network was trained using the Adam [55] optimization algorithm since it is well suited for problems that are large in terms of data and/or parameters. Training was performed in batches of 500 examples, with a learning rate of 0.001, $\beta_1$ 0.9 and $\beta_2$ 0.999. The loss function that the algorithm minimized was the binary cross-entropy between the real class and the predicted classes. I initialized the weights in each layer according to the Glorot [56] initializer and initialized the neuron biases with the constant 0. I applied the dropout regularization technique in the first three hidden layers with a dropout rate of 0.5. The network trained for 300 epochs, which took around 20 minutes on one NVIDIA GTX 1080 GPU and Intel i7-3770k @ 4.0GHz CPU.

The network was implemented in the Keras 2.2.4 [57] neural network framework using TensorFlow 1.10 [34] as backend.

5.3.2.C Using Active Learning to boost classification performance

Given the complexity, variability and substantial imbalances in class representation, it would require thousands or even tens of thousands of manually labeled data points in order to train a classifier that achieves good classification performance without discarding rarer bout types by default.

This would not be a problem if there was a simple mathematically defined way to automatically label and evaluate the data, as there would be no need to build such a complex classifier. But the reality is that the only way to be highly confident if a movement was good or bad is to look at it and compare it to the abstract notion of a clean, realistic and biologically feasible signal that one can only obtain by examining countless other movements.
This imposes a time cost on labeling each one of the 3,191,020 tail movements; nonetheless I have set out to train a well performing classifier. Naturally, the following question arises: Which samples must be manually labeled to maximize the performance of the model while constraining the task of labeling new instances to the minimum amount possible? One possible answer to this question comes from the field of Active Learning (AL).

Within the AL framework, the algorithm is given the possibility to choose the data from which it learns, maximizing, in theory, the information it extracts from each sample and performing better with less training. AL is a useful tool in more sophisticated supervised learning tasks where acquiring new labeled instances is a very difficult and time-consuming tasks. One example of such a task can be found in the field of speech recognition where it can take expert linguists up to ten times the duration of an audio sample to annotate all the words contained in it [58].

The way AL systems overcome the bottleneck of having a limited number of labeled samples is by carefully selecting data from the large pool of unlabeled samples and querying the oracle (e.g. a human annotator) for the correct label. By carefully selecting the correct unlabeled samples the algorithm achieves high performance in its task with few labeled samples as possible, keeping the cost number of manually labeled points to a minimum.

Figure 5.12: Basic pool-based sampling active learning cycle

A – Pool-based sampling  Within the AL framework, there are several scenarios in which the active learner may pose queries (ask a human to label a sample), as well as several different strategies used to decide which instances are most informative. In this work, I used the Pool-Based Sampling strategy [59] that has been applied to many real-world machine learning problems ranging from text classification [60]
to speech recognition [61] and cancer diagnosis [62].

This strategy assumes that there is a small set of labeled data \( \mathcal{L} \) and a large pool of unlabeled data \( \mathcal{U} \), which is a common scenario in today’s big data problems where large collections of unlabeled data are gathered faster than curators can verify and label the data. In pool-based sampling, queries are selectively drawn from the unlabeled data pool \( \mathcal{U} \) following some greedy policy related to an informativeness measure that is used to evaluate all samples in \( \mathcal{U} \). As described in Figure 5.12, the usual pool-based sampling AL cycle starts by creating a small set of labeled data \( \mathcal{L} \) out of the large pool on unlabeled data \( \mathcal{U} \). A machine learning model is trained on the task at hand and it is used to evaluate all samples in \( \mathcal{U} \).

The result of the evaluation is then used to calculate some measure of the usefulness of each unlabeled sample that the learning algorithm uses to select which samples are going to be queried to the oracle. After the oracle labels the selected data, the newly labeled data is simply added to the label set \( \mathcal{L} \) that the learner will use to improve its model, leveraging its new knowledge.

**B – Query strategy - Uncertainty Sampling** All AL scenarios involve evaluating the informativeness of unlabeled instances and there have been many proposed ways of formulating such query strategies in the literature. Uncertainty sampling [59] is one of the simplest and most commonly used query frameworks. Here, an active learner queries the instances about which it is least certain how to label.

When using a probabilistic model for binary classification, uncertainty sampling simply queries the instance whose posterior probability of being positive is nearest 0.5 [59, 63].

For problems with three or more class labels, a more general uncertainty sampling variant might query the instance whose prediction is the least confident:

\[
x^{*}_{LC} = \arg \max_{x} (1 - P_{\theta}(\hat{y}|x)),
\]

where \( \hat{y} = \arg \max_{y} P_{\theta}(y|x) \) is the class label with the highest posterior probability under the model \( \theta \), \( x \) the input parameters, \( y \) the predicted class and \( x^{*}_{LC} \) the sample to which the algorithm is the least confident about its prediction.

Following the uncertainty sampling principle, at every iteration of the AL process, I randomly sample a thousand instances in the prediction range 0.4 to 0.5 and thousand more from the 0.5 to 0.6 range. Given the variability of the data across the 26 classes and the fact that there are several types of noisy signals I need to be sure that, for instance, the algorithm is not just assuming all samples from rarer classes are considered bad movements, I manually corrected 500 misclassified instances in areas where the algorithm should be fairly confident of its prediction, but it is wrong. I corrected 500 clearly good bouts that have been classified as bad in the 0.95 to 1.0 range and did the same to undoubtedly bad samples labeled as good in the 0.0 to 0.05 confidence range.
C – Stopping criteria  Because AL is concerned with improving accuracy of the model while remaining sensitive to labeling costs, a crucial element of AL is the concept of knowing when to stop the learning process. Since there is a cost associated to keep labeling more samples, the algorithm will eventually reach a plateau in its performance from which acquiring more samples becomes a waste of resources. Even if such plateau is not reached, there is always a point at which the cost of acquiring and labeling new training data will be greater than the improvement it will give to the current model.

In this work, there is not a cost related to acquiring the data, there is only the time cost of manually going through the data and labeling the samples. As such, I defined the stopping criteria to be based of the effort to find high confidence misclassified instances and the difficulty to discern good and bad samples around the 0.5 confidence range.

5.3.2.D Active Learning Performance

From all the 3,191,020 samples, three sets were created to train and evaluate the classifier. In the first iteration of the algorithm, the labeled set $L$, which is the training set of the classifier is composed of 7,491 randomly sampled bouts. To adjust hyperparameters of the classifier and to ensure that it is not overfitting to the training data, there is also a test set of 1,873 randomly sampled bouts not belonging to the training set.

Finally, to evaluate the classifier’s performance discriminating smooth continuous tail movements from noisy data across all 26 classes and for good and bad bouts, there is also a validation set composed of 50 samples per class for both good and bad bouts not belonging to either of the two previously mention sets. At every iteration of the active learner, the classifier was trained using the data from the labeled set $L$, and the all samples in the unlabeled set $U$ were classified.

Based on these classifications and following the querying strategy defined in Section 5.3.2 B –, 3,000 points from the unlabeled set $U$ were selected to manually labeled by a human. Once the data was manually labeled, it was added the training set $L$ and the process repeated following the algorithm in Section 5.3.2.A –.

The active learner cycle was stopped after four iterations after the fourth iteration the performance on correctly classifying the validation set consistently kept degrading. This was determined as the samples around the 0.5 confidence range getting extremely difficult to label in a consistent way between good and bad samples and, the classifier making so few mistakes in the high-confidence regions (0.0 to 0.05 and 0.95 to 1.0) that it was requiring looking through several hundreds of samples to find a misclassified example.

In Figure 5.13 we can see how the classifier’s performance in predicting the validation set progressed as the active learner evolved. Here it is possible to visualize how the performance of the classifier started around 84.6%, peaking at 93.5% by the end of the second iteration, just to steadily drop to 89.1% on
the fourth iteration. One of the reasons I believe caused the performance to decrease was the fact that
the difficulties in reliably labeling samples in the 0.5 range was causing similar samples to be labeled
to different classes adding more noise to the algorithm than helpful information. Nonetheless, even with
the classification accuracy decreasing, the classifier was still improving in another metrics.

Figure 5.13 shows the evolution of the classifier's performance over the 4 iterations the active learner
ran for. Here we can track the dynamics of the model's accuracy, specificity, sensitivity and, precision.
It is possible to see that despite the model's sensitivity continuously increasing, the overall performance
of the classifier starts degrading after the second iteration. With this in mind I decided to freeze the
classifier at the second iteration of the AL algorithm as this seems to be a good compromise between
all metrics and maximizes accuracy.

By the end of the AL process, low quality bout classifier accuracy increased by 10.5%, the specificity
10.8% , the sensitivity 9.7% and the precision 13%.

As important as the classifier's accuracy, is its capacity to have a good classification performance
separating good instances from bad ones across all the different possible movement shapes without
being biased for some specific movement type or direction.

Because the shape of the movements varies between classes and there exist several different types
of bad samples, being able to accuracy predict the correct label for each individual class is as important
as the overall classification performance of the classifier.

In Figure 5.14 we can see how the classifier's accuracy evolved during the AL process discriminated
by movement type.
For each bout class, the bar plot on the left represents the movements for the left and the right bar plot movements to the right. The outlined plot is the classifier’s accuracy before the AL and the plot without the lines is the performance after the second iteration of the AL where accuracy was maximum. Overall, the classifier’s performance improved for all classes never having an accuracy lower than 83% and improving over 25% for some bout types. The only bout type whose performance did not improve was the LLC bout type where the accuracy on detecting low quality samples in the data dropped by 8.7%.

The low quality bout classifier successfully manages to discriminate between high quality and low quality samples across a wide range of possible sources of data contamination. In Figure 5.15 we can see that regardless of the bout type there are almost no classification errors and that when errors occur, it is usually a high quality movement that is identify as a low quality one and thus being discarded from the rest of the analysis. Considering the size of the dataset discarding accidentally some samples is not a reason of concern, unlike adding bad samples to the dataset by mistake.

5.4 Final dataset description

After using the instantaneous speed to manually clean the trajectories and the classifier to automatically detect low quality tail traces, the final dataset was created by the intersection of samples having positive results in both tests.
Figure 5.15: Low quality bout classifier results. (A),(B) represent movements to the left, (C),(D) represent movements to the right. (A) One random bout per bout type class identified by the classifier as a high quality bout. (B) One random bout per bout type class identified by the classifier as a low quality bout. (C) One random bout per bout type class identified by the classifier as a high quality bout. (D) One random bout per bout type class identified by the classifier as a low quality bout.
Table 5.2: Number of samples per bout type on clean dataset.

<table>
<thead>
<tr>
<th>Direction</th>
<th>AS</th>
<th>Slow1</th>
<th>Slow2</th>
<th>Short CS</th>
<th>Long CS</th>
<th>BS</th>
<th>J-turn</th>
<th>HAT</th>
<th>RT</th>
<th>SAT</th>
<th>O-Bend</th>
<th>LLC</th>
<th>SLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>41611</td>
<td>146852</td>
<td>506112</td>
<td>8842</td>
<td>1698</td>
<td>6861</td>
<td>48677</td>
<td>137021</td>
<td>398089</td>
<td>50480</td>
<td>6989</td>
<td>28951</td>
<td>803</td>
</tr>
<tr>
<td>Right</td>
<td>39386</td>
<td>139958</td>
<td>347491</td>
<td>7811</td>
<td>1423</td>
<td>6366</td>
<td>46482</td>
<td>117964</td>
<td>379261</td>
<td>48955</td>
<td>6784</td>
<td>30136</td>
<td>888</td>
</tr>
</tbody>
</table>

Out of the original dataset of 3,233,374 bouts, only 2,408,635 bouts were selected as having enough quality to be part of the final dataset.

As Figure 5.16 shows, some of the classes where more affected by the data cleaning than others. Classes such as the SLC have almost no representation in the final dataset increasing the unbalance between classes even more.

Regardless of the amount of points removed per class, the methods applied before do not seem to have a preference for a particular direction of motion as samples from movements for left and right seem to have been removed somewhat evenly across all the bout types.

![Number of samples per bout type. Clean dataset](image)

**Figure 5.16**: New class ratios

The final dataset is composed of 2,408,635 movements across 26 classes. Table 5.2 shows the number of samples per bout type.

### 5.4.1 Animal Posture Dataset

By comparing the two-dimensional embedding of the vectored tail angles over time on the clean dataset in Figure 5.17 right panel, and the same embedding for the original dataset, Figure 5.17 left panel, there is a significant visible difference on the quality of the data that originated this new space. The new
embedding shows almost no points randomly spread across the space, and the identifiable families of bout types that appeared before have become even more clearly isolated and organized. Movements such as the BS (orange), LLC (yellow), SLC (red) have now been organized into two groups one for each direction of motion which did not happen in the original data.

Figure 5.17: Comparison between the t-SNE embeddings of 500 random points per class. Left: Before data cleaning. Right: After data cleaning.

5.4.2 Animal Trajectory Dataset

The quality of the trajectory dataset has improved as well. Where in Figure 5.8 we can see every class containing trajectories with tracking errors that caused high discontinuities in the trajectory, the new dataset has almost no issues of this kind, as Figure 5.18 shows.

Though there are some erroneous trajectories in the new dataset, the trajectories seem to generally follow stereotyped paths in space, creating a much cleaner mean trajectory per class, as can be seen in Figure 5.19.

In this Chapter, using t-SNE, it was possible to visualize that there is already some structure to the Posture dataset and that this space can be organized according to the results of Marques et al. without calculating any features of motion. The same t-SNE embedding also showed that there were points in the dataset that had to be removed as they often represented tracking or movement segmentation mistakes. Given the structured nature of this data, a simple deep binary classifier was created to automatically classify movements as good or bad. Using Active Learning, the performance of the classifier was boosted from 83% to 93.5%. This classifier is a powerful tool that can be used in the future to automate part of the data analysis pipelines.
Figure 5.18: 100 random trajectories per class on the clean dataset.

Figure 5.19: Mean trajectory per class on the clean dataset.
6 Behavioral Classifier Optimization

Contents

6.1 Network Architectures ............................................. 51
6.2 Network Inputs .................................................. 53
6.3 Training Methodology ............................................ 55
In this project we intend to study how different parameters affect the networks ability to accurately predict the movement type. In this Chapter we will go over all the different parameters that will be optimized to achieve the best classification accuracy possible. These parameters include the network’s inputs, its activation function and architectures and, how the training process was conducted.

6.1 Network Architectures

Increasing the complexity of a network’s architecture tends to increase its performance, as adding more neurons or layers increments the degrees of freedom the network has, allowing it to learn more complex expressions in the data.

However, increasing the network’s complexity also increases training time, as more computations need to performed every training iteration. Other issues such as overfitting arise, since these new degrees of freedom allow the network to increase its performance by “memorizing” the input data. Working with even deeper networks also introduces other problems to training, requiring the implementation of techniques to fight vanishing gradients for the network to be able to improve with training. Finally, there is also the problem of having the hardware necessary to train such complex networks. As the complexity of the architecture increases so do the resources required to store and compute all operations of training and prediction steps.

In this Section, we test and compare the performance of three different neural network architectures with increasing complexity on the same classification task.

6.1.1 Simple Architecture

The simplest network architecture possible is a single layer of fully connected neurons linking the input to the output layers, as shown in Figure 6.1. The simplicity of this network provides a baseline performance that we can use to compare to other, more complex, network architectures.

6.1.2 Ramp Architecture

Since the inputs used in this project are so high-dimensional (e.g. See Table 6.2), one could imagine that the network will not be able to learn all interesting structures in the data in just a single layer of operations.

Based on the encoding section of autoencoder networks, the design of the second network is intended to enable the network to learn more complex patterns by adding more layers. At the same time, the ever-shrinking layer size should mitigate overfitting problems that arise with the added depth by forcing the network to discard information between layers.
As Figure 6.2 depicts, at each layer, this architecture decreases the number of neurons by half until it reaches the size of the output layer.

6.1.3 Deep Architecture

The third and final network we will test in this project is the most complex of all.

Inspired by the convolution followed by pooling pattern of convolutional neural networks, the design of this network extends the previously described Ramp network architecture by adding more fully connected layers of the same size, before reducing the number of neurons in the layer again.

Let $s$ be the input size. After the input, the network will have another layer of size $s$, followed by 3 layers of size $s/2$, finishing with 4 layers of size $s/3$, before the final output layer, as shown in Figure 6.3.

The goal of this architecture is to test if having more layers of the same size before forcing the network to shrink and discard information can improve its performance, perhaps by enabling the network to learn advantageous patterns at different scales.

Given the variability of the network architectures and input sizes, we decided to test the \texttt{tanh} and \texttt{ReLU} activation functions for all networks and see how they influence the final performance.

Due to the high number of tunable parameters, the Dropout regularization technique was used to mitigate overfitting in the networks. Dropout was applied with the same rate to every layer of the networks with the exception of the input layer.
6.2 Network Inputs

One of the main goals of this project is to expand the range of conditions in which the classification defined by Marques and colleagues [3] can be used. In this work I will try to develop a classifier that can be used to accurately label the movement using information from the larva’s trajectory and tail movements.

6.2.1 Animal Trajectory

The trajectory dataset is defined as the concatenation of all positions along the horizontal axis \(x\) (mm), all positions along the vertical axis \(y\) (mm) and the orientation in respect to horizontal axis \(\theta\) (rad), for the first 265 frames after the movement was initialized plus 20 frames before the movement started to compensate for possible failures to detect the exact moment the fish started moving.

These trajectories have been translated so that the movement always starts at \((0, 0)\) and rotated according to \(\theta_0\) in order to have the larva in the same orientation regardless of where it was in the arena at that time.

Considering the input size of 885 produced by the combining \(x, y\) and \(\theta\), the networks used in the trajectory dataset has the dimensions presented in Table 6.1.
Table 6.1: Animal trajectory network dimensions.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>855-26</td>
</tr>
<tr>
<td>Ramp</td>
<td>855-428-214-107-53-26</td>
</tr>
</tbody>
</table>

### 6.2.2 Animal Posture

One of biggest advantages of the zebrafish larvae as an animal model in neuroscience is its transparent body and small size that allow imaging the whole brain at single cell resolution at the same time.

In these preparations, the larvae are restrained with a small drop of agarose while the tail and eyes are free to respond to the stimuli. Restraining the larva will render any trajectory-based classification networks useless as the position of the animal on the arena is always the same.

The most common source of information describing the larvae’s behavior is the evolution of the tail posture as it moves. Since in microscope preparations the tail is free to move, it is worth investigating the ability to describe the animal's behavior just by looking at tail oscillations. Also, the source of information for most features defined by Marques et al. is tail angle and its temporal evolution.

The second input we will test in this project is a concatenation of the tail angles for the first seven segments of the zebrafish’s larva as it moves. Just like in the trajectory dataset, input here contains information for the first 265 frames after the movement starts plus 20 frames of the activity before the movement was initiated.

Considering the input size of 1995 produced by the combining all 285 timesteps for the seven tail
segments, the networks used in the tail dataset have the dimensions displayed in Table 6.2.

**Table 6.2: Animal posture network dimensions.**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>1995-26</td>
</tr>
</tbody>
</table>

6.2.3 Tail Angles and Trajectory

Lastly, it is also interesting to explore if, in the cases where both tail and trajectory data can be acquired simultaneously, there is any advantage to combine both sources of information. Perhaps having information from tail and trajectory will increase the robustness of the network when the quality of one source is degraded, or perhaps there might be some structure in the data that can only be uncovered when presenting both information at the same time.

To test this hypothesis, we create a third dataset, which is just a concatenation of the previous two datasets.

Considering the input size of 1995 produced by combining both tail and trajectory inputs, the networks used in the tail and trajectory dataset have the dimensions shown in Table 6.3.

**Table 6.3: Animal posture and trajectory network dimensions.**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>2850-26</td>
</tr>
<tr>
<td>Ramp</td>
<td>2850-1425-713-356-178-89-45-26</td>
</tr>
</tbody>
</table>

6.3 Training Methodology

In order to have a more methodical evaluation of the classification performance across the different networks, every network was trained using the same conditions. All networks were trained for 150 epochs, using the Adam [55] optimizer.

The training conditions where set as follows:

- **Loss Function**: The networks were trained to minimize the cross-entropy loss. This loss function measures the performance of a classification model whose output is a probability value between 0 and 1. As the predicted probability diverges from the actual label the loss increases. A perfect model would have a logloss function of 0.
In a multiclass classification scenario, we calculate a separate loss for each class label per observation and sum the result,

$$- \sum_{c=1}^{M} y_{o,c} \log (p_{o,c}),$$  \hspace{1cm} (6.1)  

where $M$ is the number of classes, $y$ a binary indicator if class label $c$ is the correct classification for observation $o$ and $p$ predicted probability observation.

- **Dataset Splitting:** As described in Chapter 5, the dataset is highly unbalanced, the most common class being several orders of magnitude more frequent than the most underrepresented class.

  The validation and test sets both contain 200 randomly selected samples of each class and the training set contains all the remaining samples.

- **Batch Generation:** To prevent any possible bias related to unbalance between classes, every epoch a new batch of 400 randomly selected samples per class was generated.
Results
The work of Marques and colleagues relied on simultaneously tracking both the tail movement and position of the larva in order to classify behavioral data. As described in Chapter 5, there are several applications where this approach is not feasible, either because the fish is restrained in agarose and thus not moving around in the arena, or because the behavior recording setup may not have enough spatial resolution to resolve the tail with sufficient detail to track its different segments. The focus of this work revolves around the capability of different network architectures to accurately classify the same dataset, given different input sources. Hence, factors, such as training time or number of epochs used, should not be considered in this Chapter.

The choice of activation functions affect the performance of the network. Before conducting an extensive optimization of all hyperparameters, the performance of all network architectures was tested with a small subset of the parameters to assess whether there were any significant differences between using either ReLU or tanh activation functions.

The choice of activation functions affect the performance of the network. Before conducting an extensive optimization of all hyperparameters, the performance of all network architectures was tested with a small subset of the parameters to assess whether there were any significant differences between using either ReLU or tanh activation functions.

![Figure 7.1](image)

**Figure 7.1:** Relative performance when using tanh vs ReLU. By plotting the performance using the hyperbolic tangent in one axis and using the ReLU in the other, networks that perform with equal accuracy in both cases should be on the diagonal line and the farther they move away from the diagonal, the bigger the difference between the results of the activation functions is.
As depicted in Figure 7.1, the results of this test reveal that, for this particular task, the hyperbolic tangent outperforms the ReLU in most networks. When this is not the case, both activation functions seem to produce networks with approximately the same accuracy. Having chosen the hyperbolic tangent as the better-suited activation function, the remaining network optimizations were conducted.

Figure 7.2 describes how the accuracy in the classification task evolved with different dropout levels for the trajectory only dataset. In the horizontal axis, we can find the different levels of dropout tested and in the vertical axis the network’s accuracy on classifying a particular dataset. Each color represents a different network architecture: the Simple network being blue, the Ramp orange, and the Deep in yellow. For each one of the networks, the dashed lines indicate the training set and the solid line the test set.

Figure 7.3 describes how the accuracy in the classification task evolved with different dropout levels for the posture only dataset. Figure 7.4 describes how the accuracy in the classification task evolved with different dropout levels for the combined network with both posture and trajectory dataset.

First, similar classification accuracy can be achieved regardless of the chosen input source, with the best result for the ‘trajectory dataset’ being 86.06% and the ‘tail angles only dataset’ 84.41%. Using both input sources, the best performance achieved was around 88.17%. Having such similar results across the different inputs demonstrates that these networks can be deployed in a wide range of applications studying larval zebrafish behavior, despite embedding of the larva or a lack of information about tail angles.

Figure 7.2: Evolution of classification accuracy with dropout on trajectory dataset.

Considering most of the features defined by Marques and colleagues were calculated using infor-
mation from the tail angles, it was surprising to see that the networks based solely on tail data were outperformed by the trajectory-based one. This difference in performance could be due to the tendency of overfitting that the trajectory based networks clearly demonstrate, causing a failure to generalize in the test set.

The use of the regularization technique Dropout improved the accuracy of all networks independently of the input source and reduced overfitting. This validates the need for using such techniques in networks with a large number of tunable parameters. The tests confirmed our expectation that for high Dropout values the performance of all networks would be degraded as more neurons in the network are disabled. Nonetheless, across the different input sources and Dropout values tested, the Simple network, designed to set a baseline accuracy, was mostly able to be in par with the other more complex architectures. Furthermore, it outperformed them for high Dropout values, showing an increased robustness when compared to the others. The Deep network architecture, on the other hand, which should have been able to model more complex structures in the data, was highly susceptible to Dropout. It rapidly changed from overfitting to underfitting and failed to accurately predict the classes. The Ramp network proved to be the most stable and consistent network across the different inputs.

Looking at the performance across networks and input sources, it appears that the performances of the networks were all capped around 90%. The fact that a combination of both input sources produced a higher accuracy than each input source individually suggests that these basic features may not be sufficient to describe the full known locomotor repertoire of the larval zebrafish. Alternatively, this ceiling

![Figure 7.3: Evolution of classification accuracy with dropout on posture dataset.](image-url)
may suggest that there are inconsistently labeled samples in the dataset, which are preventing further improvement of the networks.

Nonetheless, considering the substantial differences in the quality of data coming from particular classes (Chapter 5), there may be one or more classes that the networks are repeatedly failing to classify, due to the inclusion of unrepresentative samples. If the performance of the network is indeed impacted by particular classes, that should become apparent when looking at the confusion matrix, that summarizes with count values the correct and incorrect predictions broken down by each class. Figure 7.7 shows the result of this test for the best network for each input type and here is possible to see that even though the performance varied across the input sources, there were no cases where a given class could not be reliably classified.

Since all networks are correctly classified the data across all classes, another possible justification for this ceiling effect is, that when the dataset labels were split into left and right, the data was not separated properly. If the metric used to discriminate between left and right was wrong, it would be expected that for a given bout, the second most probable class would be the same type of bout in the opposite direction. Looking at the confusion matrices reveals that, despite existing some cases where movements were classified as being from the correct type but different direction, there are not enough movements bring down the classification accuracy that much.

However, one pattern of sample misclassification was evident. Slow1 (light blue) both left and right versions, when misclassified tend to be labeled as AS (cyan). Similarly, J-Turns (coral), HAT (light
green), RT (dark green), SAT (lavender) and O-bends (purple) are recurrently being misclassified and attributed to classes within this group. The same occurs between BS (orange), LLC (yellow) and SLC (red). In reference to Figure 5.17 in Chapter 5, these groups are the main clusters that showed up in the t-SNE embedding, signifying that points in these classes are, in fact, similar to each other. One possible explanation for the underlying structure in this label misclassification is that these are points at the borders of the regions defining the classes in the work of Marques and colleagues [3]. This pattern of misclassification was somewhat excepted.

A final look at the confusion matrices reveals that SCS(gray) and Long Capture Swim (LCS)(black), are consistently the classes with the worst accuracy. These types of movements are extreme rare as they only occur when the animal is feeding. This behavior can only be present in a very small subset of the experiments but there is no context information preventing the classifier from attributing this labels to any bout. One possible justification for this error is then that a significant portion of the movements labeled as SCS(gray) or LCS(black) are in fact originated from low-quality samples that could not fit in any other classes and thus the original classification algorithm that set the ground truth decided to put such samples in one of these to classes.
Figure 7.5: Confusion matrix for the trajectory dataset
**Figure 7.6: Confusion matrix for the posture dataset**

```
<table>
<thead>
<tr>
<th>True class</th>
<th>AS L</th>
<th>SLOW 1 L</th>
<th>SLOW 2 L</th>
<th>SCS L</th>
<th>LCS L</th>
<th>BS L</th>
<th>J-TURN L</th>
<th>HAT L</th>
<th>RT L</th>
<th>SAT L</th>
<th>O-BEND L</th>
<th>LLC L</th>
<th>SLC L</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS R</td>
<td>187</td>
<td>28</td>
<td>16 183</td>
<td>3 12</td>
<td>3 18</td>
<td>11</td>
<td>1 136</td>
<td>110</td>
<td>2 146</td>
<td>4 18</td>
<td>5 186</td>
<td>5</td>
<td>18 21</td>
</tr>
<tr>
<td>SLOW 1 R</td>
<td>7</td>
<td>1</td>
<td>14 1 103</td>
<td>3 5</td>
<td>2</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>SLOW 2 R</td>
<td>1</td>
<td>1</td>
<td>11 100</td>
<td>1 4</td>
<td>1 10</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>SCS R</td>
<td>7</td>
<td>14 100</td>
<td>11 166 9</td>
<td>4</td>
<td>166</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>LCS R</td>
<td>3</td>
<td>1 4</td>
<td>11 146</td>
<td>1 4 1 11</td>
<td>4 4 1 4 130</td>
<td>5 178</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS R</td>
<td>1</td>
<td>1 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>J-TURN R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>HAT R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>RT R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>SAT R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>O-BEND R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>LLC R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>SLC R</td>
<td>1</td>
<td>3 1 1 2</td>
<td>166 5</td>
<td>1 2</td>
<td>1</td>
<td>1</td>
<td>1 109</td>
<td>5</td>
<td>2 189</td>
<td>1 5 1 4</td>
<td>1 177</td>
<td>2</td>
<td>187 3</td>
</tr>
<tr>
<td>Predicted class</td>
<td>AS L</td>
<td>SLOW 1 L</td>
<td>SLOW 2 L</td>
<td>SCS L</td>
<td>LCS L</td>
<td>BS L</td>
<td>J-TURN L</td>
<td>HAT L</td>
<td>RT L</td>
<td>SAT L</td>
<td>O-BEND L</td>
<td>LLC L</td>
<td>SLC L</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>----------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>----------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
</tr>
</tbody>
</table>
```
### Figure 7.7: Confusion matrix for the posture plus trajectory dataset

<table>
<thead>
<tr>
<th>True class</th>
<th>AS L</th>
<th>SLOW 1 L</th>
<th>SLOW 2 L</th>
<th>LCS L</th>
<th>BS L</th>
<th>J-TURN L</th>
<th>HAT L</th>
<th>RT L</th>
<th>SAT L</th>
<th>O-BEND L</th>
<th>LLC L</th>
<th>SLC L</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS R</td>
<td>1</td>
<td>14</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SLOW 1 R</td>
<td>2</td>
<td>24</td>
<td>167</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SLOW 2 R</td>
<td>1</td>
<td>13</td>
<td>176</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SCS R</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>176</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>148</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>LCS R</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>148</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>BS R</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>177</td>
<td>1</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>190</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>J-TURN R</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>12</td>
<td>180</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>192</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>HAT R</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>172</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>192</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RT R</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>172</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>192</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SAT R</td>
<td>5</td>
<td>13</td>
<td>15</td>
<td>3</td>
<td>7</td>
<td>163</td>
<td>9</td>
<td>5</td>
<td>13</td>
<td>15</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>O-BEND R</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>LLC R</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>SLC R</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

*Predicted class*

The confusion matrix above shows the performance of a posture plus trajectory dataset model. Each cell represents the number of predictions made for a specific class, with the diagonal indicating correct predictions.
Conclusion
Taking advantage of the collection of several million samples of zebrafish larva movements across different contexts, in this work, we used deep artificial neural networks to achieve a featureless classification of the zebrafish larva locomotor repertoire in accordance to the work of Marques and colleagues, which is the most complete description of the larva’s behavior to date.

From this work originate three different ANNs that can accurately classify the zebrafish larvae movement using only information from the tail movements, the trajectory of the animal as it moves or both at the same time. This is a big advancement for the field since current classification techniques require complex computations to extract the features that enable the classification prohibiting its use in fast real-time movement identification applications, which are key for mapping the relationship between the current movement of the larvae and the neuronal activity in the brain via closed loop stimulation.

In addition, since the networks created in this work have distinct input modalities, they can be used to classify the data in cases where it was not possible to gather all the information to compute the features of motion, such as when the larva is restrained or when the video acquisition tools do not offer enough spatial resolution to accurately track all points along the larva’s tail.

In order to train the behavior classification networks, the original dataset where all the movements came from had to be cleaned since it contained substantial amounts of low-quality samples. In order to clean the dataset, a deep binary classifier was created to automatically detect the low-quality samples over the entire dataset. This tool can be deployed again in the future as a preprocessing step in more complex analysis pipelines, to help the research community identify meaningful zebrafish larvae movements in their datasets.

Finally, this work also created a curated dataset containing several million zebrafish larva movements that can be explored in future projects as those defined in Chapter 9 to explore other aspects of zebrafish larval behavior and its relationship to the underlying neural circuit activity.
Future Work

Contents

9.1 Zebrafish behavior analysis toolbox using neural networks . . . . . . . . . . . . . . . . 71
9.2 Improving general classification accuracy via Data Augmentation . . . . . . . . . . 71
9.3 Understanding the structure of the zebrafish’s locomotor repertoire using Autoen-
  coders . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
9.4 Extending the behavior analysis pipelines to low-resolution setups . . . . . . . . . 72
9.5 Creating classification networks based on tail dynamics over time . . . . . . . . . 74
Marques et al. gathered one of the biggest collections of larval zebrafish movements and as a result created one of the most comprehensive descriptions of the locomotor repertoire of the zebrafish larva. The biggest hurdle preventing this description from being used over a wider range of scenarios, such as low spatial resolution setups, experiments where the larva is restrained from moving or when there is a need to use real-time classification, is the dependency on kinematic features of motion. In other cases, such kinematic features of motion sometimes cannot be calculated due to a lack of information, or may take too long to calculate. To overcome these shortcomings, several feed-forward neural networks were developed in this work. The networks can be used in multiple contexts to achieve a high classification performance without depending on the kinematic features of motion.

9.1 Zebrafish behavior analysis toolbox using neural networks

In the process of establishing these behavior classification networks, another network was created that automatically assesses a particular movement’s data quality, as a pre-processing step to clean future datasets. This collection of networks for data cleaning and behavior classification could potentially be the foundation of a prospective toolbox that uses neural networks to facilitate the study of the larval zebrafish locomotor repertoire.

9.2 Improving general classification accuracy via Data Augmentation

Despite achieving a relatively high performance in the classification task, the network is reliant on the particular segmentation algorithm that had been previously applied to the data used in this study. As a result, applying a different technique that defines new start and end points will temporally shift the data and cause the network to fail at predicting the correct class. One possible way to circumvent this problem is, for instance, the use of data augmentation techniques like those utilized in the field of computer vision and create new variants of the same movement with different temporal shifts.

9.3 Understanding the structure of the zebrafish’s locomotor repertoire using Autoencoders

The space generated by the t-SNE projection of the dataset containing the larva’s posture, indicated that there clearly are interesting structures in this dataset.
Using autoencoder networks, as shown in Figure 9.1, one could try to compress this space to learn which are the relevant features originating from the larva’s posture. In addition, autoencoders could be used as a data compression tool that would reduce the dimensionality of the data, facilitating the data analysis, or just as a denoising tool to create cleaner data samples.

9.4 Extending the behavior analysis pipelines to low-resolution setups

The robustness of some of these networks to data loss, caused by the Dropout during training, poses the question whether there is a necessity of having such a high temporal resolution in the recording setups. The hardware required for achieving such high temporal resolution is orders of magnitude more expensive than the one required to acquire data at lower frame rates, enabling the development of more efficient high-throughput setups.

When analyzing how the trajectory and tail movements degrade as temporal resolution is downsampled, it is evident that the tail data quickly lose their internal structure with decreasing frame rate. This implies that tail data are much more dependent on the temporal resolution than on the trajectory data, preserving most of their characteristics even as low as 100 fps, as can be seen in Figure 9.2.

Although being more dependent on the acquisition speed, Figure 9.2 shows that in lower resolution scenarios the tail information provides higher accuracy than the trajectory. This can be perhaps due to the network being able to pull data coming from the different tail regions to improve the results of its predictions.
Since the networks developed in this work seem to be able to handle such a wide range of frame rates, it might also be interesting to consider creating super-resolution networks that would upsample the data coming from a lower fps camera and boost the classification performance of the classification network.

Evaluating the classification accuracy of the networks as the framerate decreases (see Figure 9.3), it is possible to see that using just the tail data will continuously result in higher performances when compared to using just the trajectory information. However, for slow acquisition speeds 100fps where the signals are heavily degraded, both inputs modalities start producing low accuracy networks, but the trajectory based networks are less affected by this, which makes sense considering that the trajectory traces are much more independent from the temporal resolution when compared to the tail traces.

Figure 9.2: Evolution of signal quality with the acquisition framerate.

Figure 9.3: Classification accuracy at different framerates.
9.5 Creating classification networks based on tail dynamics over time

Finally, it would also be interesting to see if the internal structure of tail motion could be employed in other types of networks known to capture patterns at different scales, such as RNNs or CNNs, in order to create better representations of larval zebrafish motion.
Bibliography


76


