Analysis of Feature Learning in CNNs on the Example of Video Recordings of Zebrafish

Bennet Breier

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2019

Abstract

Deep Neural Networks have reached superior performance in various image recognition tasks making them an increasingly popular tool in applied domains. However, these powerful networks remain a black box. Reaching better transparency helps to build trust in their classifications and makes learned features interpretable to experts. We train Convolutional Neural Networks (CNNs) to distinguish prey and spontaneous swim bouts in video recordings of larval zebrafish. Using a recently developed technique called Deep Taylor Decomposition, we generate heatmaps to highlight input regions of high relevance for predictions. We find that our best CNN achieves a classification accuracy of 96.69% by focusing on the steadiness of the tail's trunk. Interestingly, this markedly differs from the features used for classification in [62]. We further uncover that the network pays attention to experimental artifacts. Removing these artifacts would likely improve performance and ensure the validity of predictions. Our work thus demonstrates the utility of AI explainability for CNNs.

Acknowledgements

First and foremost, I would like to thank my supervisor Arno Onken for our insightful discussions and his knowledgeable and helpful answers, which made these fruitful results possible.

Furthermore, many thanks go to the University of Edinburgh and the operators of the Teaching Cluster and the James and Charles Cluster for providing the required computing resources.

Finally, I would like to thank the Max Planck Institute of Neurobiology under Herwig Baier, which kindly made all of the raw video data available.

Table of Contents

Chapter 1

Introduction

In recent years, Convolutional Neural Networks (CNNs) have proven to reach high accuracies in classification tasks on images and videos. After Lecun and Bengio [47] introduced them in the 90s, CNNs had their break-through in 2012. The CNN by Krizhevsky et al. [44] managed to outperform the other competitors in the ImageNet Large Scale Visual Recognition Competition (ILSVRC2012) by about 10% points [32]. Since then, more and more sophisticated architectures have been designed enabling them to identify increasingly abstract features. This development has become possible due to the availability of larger training sets, computing resources, GPU training implementations, and better regularization techniques, such as Dropout [27, 75]. While these more complex deep neural network architectures achieved better results, they also remained a black box This caused CNNs to come with significant drawbacks: a lack of trust in their classifications, missing interpretability of learned features in the application domain, and the absence of hints as to what data could enhance performance [52].

In order to overcome these drawbacks, subsequent research has developed approaches to shed light on the inner workings of CNNs. Such approaches typically fall into one of four modes of explanation: making the CNN describe the image with analytic statements, showing several high-scoring examples of one class, showing several low-scoring examples of one class (i.e. counter-examples), or visualizing learned features in the input domain [25]. In this dissertation, we will focus on the latter. In specific, we will create heatmaps over examples highlighting the areas which the CNN puts attention on when making a prediction. This approach has been successfully used for uncovering how CNNs might learn spurious correlations, termed "Clever Hans" predictions [46], or in colloquial terms: cheating. This can become harmful if the

predictions entail decisions with severe consequences [48]. AI explainability might even become a legal requirement in certain applications [2]. Also, since deep neural networks have become a popular machine learning technique in applied domains, spurious correlations would undermine scientific discoveries.

An example for the utility of AI explainability can be found in skin research [70]. The authors are able to train a CNN to successfully classify skin lesions into malignant or benign based on appearance. Furthermore, by laying heatmaps over the sample images, they can visualize what their trained CNN focuses on. The authors find that the focus mostly falls on regions which exhibit features similar to the ones used by dermatologists. Also, they discover that the classifier reacts to various image artifacts, such as specular reflections and gel application. These insights direct future data collection, build trust in the classifier among experts, and prove its validity.

This dissertation focuses on zebrafish research as an applied domain of AI explainability, considering that the research community around this organism has grown immensely. The zebrafish is an excellent model organism for vertebrates, including humans, due to the following four reasons: first and foremost, the genetic codes of humans and zebrafish are about 70% inherited from a common ancestor (i.e. orthologue) [28]. Organs such as the brain therefore exhibit considerable commonalities. Secondly, the fish are translucent which allows non-invasive observation of changes in the organism [8]. Furthermore, zebrafish are relatively cheap to maintain, produce plenty of offspring, and develop rapidly. Finally, they are capable of recovering their brain structures within days after a brain injury [40, 41].

Future research in this field is likely to make use of deep learning as a tool for new discoveries, which for example might ultimately lead to better treatments of brain injuries in humans [40]. For this reason, we deem it important to show the utility of AI explainability techniques on the basis of a real application. The authors of [62] have created a set of highly controlled video recordings which they used to classify zebrafish swim bouts. Their analysis elicits a number of discriminating features making their videos the perfect dataset for comparing these extracted features to the learned features of our CNN. Moreover, by training on video data the technique demonstrated in this dissertation constitutes an advancement to the visualization of neural networks. While existing research has focused on features learned from static images, we visualize features mainly learned from motion, in particular optical flow.

This dissertation is structured as follows: In Chapter 2 we outline existing research around motion-oriented classification, AI explainability techniques, and the analysis

of zebrafish movements. We state our goals in Chapter 3. In the subsequent chapter, we explain the data used for training the baseline and CNN, detail its architecture and our training procedure, and explain how we analyzed feature learning in CNNs, which we call relevance analysis. We present the achieved performance of our CNN and the results from our relevance analysis in Chapter 5. In Chapter 6, we discuss an interpretation of the learned features and compare them to the extracted features used in [62]. After detailing caveats to the superior performance of our CNN, we finish this thesis with a conclusion and outline of future work in Chapter 7.

Chapter 2

Background

2.1 Human Action Recognition

One popular task of video classification is human action recognition. Such recognition software can become crucial for the automatic organization and analysis of large amounts of video data. Examples could be surveillance monitoring, billions of YouTube videos, or the creation of trailers. Available datasets used to contain rather few and simple categories of motion with a relatively small number of samples. One example is the IXMAS dataset with 11 action classes and 33 clips in total [71]. After the performance of classification algorithms had saturated due to the lack of complexity of available datasets, new sets were created which have become the current standard [45]. Common datasets are HMDB [45] and UCF101 [66]. HMDB for example contains almost one billion videos with 51 action categories, such as laugh, eat, or handstand. General challenges stem from camera motion, video quality, occlusion, and the sheer amount of data. Yet, more complex datasets have been created, such as the Kinetics Human Action Video Dataset [36] to account for the rapidly increasing classification performance.

While architectures for CNNs have converged to a number of established designs, such as ResNet [26] or GoogLeNet [67], this process has yet to be done for video classifiers. Carreira and Zisserman [13] identified a non-exhaustive list of five types of video architectures: CNN + LSTM (Long-Short-Term Memory), 3D-CNN [34], Two-Stream [64], 3D-Fused Two-Stream, and Two-Stream 3D-CNN. They differ in whether the convolutions are based on 2D or 3D kernels, whether optical flow is added, and how consecutive frames exchange information. For some architectures presented here, computing optical flow plays a crucial role. It can be described as the horizontal and

vertical displacement of a pixel from one frame to the next [22]. Several algorithms for flow calculation exist, such as TV-L1 [74], Brox [12], and Farneback [22], although the minutia of differences between them plays a minor role here. Even novel deep learning approaches have been developed, e.g. FlowNet [20, 31]. They deal with difficulties such as large displacements, fine details, and runtime.

Based on their observations, the authors proposes another architecture called Two-Stream Inflated 3D-CNN (I3D). They uses these architectures to investigate the benefits of transfer learning when the pre-trained weights are derived from video datasets as opposed to image datasets, because transfer learning had been successfully tested on images in previous studies [63]. In specific, they pre-trained on the Kinetics dataset and fine-tuned and tested on HMDB and UCF101. They found that pre-training improved the performances of all architectures, but most strongly of Two-Stream 3D-CNN and I3D. Furthermore, it became evident that the optical flow stream of the two-stream models always lead to a significant boost in performance, indicating the large information content of optical flow. Regarding limitations of their work, it should be noted that the tasks and videos of the considered datasets were very similar making the transfer of weights very effective [73]. Further investigation would need to address other tasks, such as semantic video segmentation or video object detection, as well as other kinds of video datasets, like the one applied in this dissertation.

The above mentioned two-stream architecture originated in the work by Simonyan and Zisserman [64]. This seminal paper arose from preceding work on video classification which had focused on shallow representations of motion. Common features included Histograms of Oriented Gradients [18], Histograms of Optical Flow, Motion Boundary Histograms [19], and general kinematic features (divergence, curl, shear) [33]. Such features would then be bundled into a Bag of Features and used as input to a Support Vector Machine (SVM). The authors tackled the question of whether such local features can be generalized into a CNN and whether this improves performance. Their architecture was partially inspired by the two-streams-hypothesis [24], which postulated that the human visual cortex processes object recognition and motion information in two distinct streams, called ventral/spatial and dorsal/temporal streams. The authors found that the temporal stream outperformed the spatial stream by 8.4% points in accuracy, despite the spatial stream using pre-trained weights. Notwithstanding, the spatial stream achieved unexpected high accuracies, too. This was probably due to the fact that appearance alone can be highly informative, e.g. in the UCF101 dataset the presence of a flute strongly favors the class “Playing Flute”. Furthermore,

they highlighted that optical flow is more informative than multiple RGB-frames, regarding state-of-the-art CNN architectures. Although not explicitly mentioned in this paper, Carreira and Zisserman [13] further discovered that two-stream networks are surprisingly robust against overfitting.

2.2 AI Explainability Techniques

Current AI explainability techniques on images can be largely categorized into two types: attribution and feature visualization [56]. Attribution relates regions in a sample image to activations of units in the CNN, while feature visualization uncovers what kinds of inputs strongly activate a particular unit. With unit we refer to hidden and output units. In the following, we will focus on attribution techniques, because they seem more interpretable on the optical flow data we analyzed in this dissertation. We made use of a tool called "iNNvestigate" [2], which provides implementations of many of the AI explainability techniques mentioned here. We finish this section by briefly mentioning popular feature visualization approaches.

Attribution tries to find those input pixels which are most relevant for a given activation or prediction. One of the earlier and quite successful attribution approaches is called sensitivity analysis [65]. It is based on the idea that important pixels have the strongest influence on a prediction. Therefore, if a single pixel were marginally changed, the prediction would worsen significantly for an important pixel and only slightly for less important ones. This measure equals the gradient with respect to the input pixels, which is convenient because the gradient is already available due to backpropagation. The saliency maps which Simonyan et al. [65] generated with this technique largely focused on the objects which had to be classified. Although they are not particularly clear to human eyes, they allowed for excellent results in object segmentation. For segmentation, they assumed color continuity of objects such that the segmentation included not only the pixels of the saliency map but the entire object.

Furthermore, Zeiler and Fergus [75] showed a simple way of producing approximate heatmaps. By occluding parts of an image with a gray square one by one and observing the change in activation, they could measure the influence of different image regions. This allowed them to check whether the CNN focused on important objects or only performed its classification based on contextual information. Similar in principle but slightly more sophisticated is the approach taken in [60]. While their tool called Local Interpretable Model-agnostic Explanations (LIME) can handle tabular data and

text too, we are interested in image classification. The tool generates a set of "superpixels" consisting of areas of similar color and texture. A selection of these regions is then grayed out and iteratively classified by the model. Although this can be a rather slow process [52], the tool can then show the combination of patches leading to a certain prediction. The goal of their study was to help practitioners gain trust into machine learning models and identify potential issues. Therefore, they had human subjects test their technique and found that they can successfully spot spurious correlations. It should be noted that the subjects were graduate students and familiar with the material. Actual practitioners might have less experience and thus be less likely to find issues in classifiers.

It is a non-trivial task to find those areas in a sample image which are most salient for making a correct classification. If we assume that the CNN can identify such highly discriminative areas after training, we can try to visualize its attention by overlaying a heatmap over the sample image. We assume that the saliency of a pixel is determined by how much the classification accuracy would deteriorate if we altered this pixel. Bach et al. [5] presented a heuristic for distributing the output value of a classifier on the input pixels according to their relevance to the output. Since in neural networks this distribution happens layer-by-layer, their intuition is called Layer-wise Relevance Propagation (LRP), that is:

$$\sum_p R_p(\boldsymbol{x}) = \ldots = \sum_{i\in(l-1)} R_i^{(l-1)}(\boldsymbol{x}) = \sum_{j\in(l)} R_j^{(l)}(\boldsymbol{x}) = \ldots = f(\boldsymbol{x}) \tag{2.1}$$

where $\boldsymbol{x}$ represents the input image, p the input layer index (pixel), $(l-1)$ and (l) hidden layers, and $f(\boldsymbol{x})$ the real-valued network output. Positive relevance $R > 0$ contributes evidence for the presence of a structure and vice-versa. In general, a neuron i from the preceding layer $(l-1)$ collects the relevances from a layer (l) according to its relative influence on each of the neurons in that layer:

$$R_i^{(l-1)} = \sum_j R_j^{(l)} \frac{\zeta_{ij}}{\sum_h \zeta_{hj}} \tag{2.2}$$

where ζ_{ij} is the influence of neuron i on neuron j computed after some propagation rule.

Two things should be noted about this approach: Firstly, there is no single correct heatmap. Heatmaps only have to be positive and conservative:

$$\forall \boldsymbol{x}, p : R_p(\boldsymbol{x}) \geq 0 \qquad \text{and} \qquad \forall \boldsymbol{x} : f(\boldsymbol{x}) = \sum_p R_p(\boldsymbol{x}) \tag{2.3}$$

Therefore, the quality and interpretability of the produced heatmap depends entirely on the chosen layer-wise propagation rule.

Secondly, their choice of propagation rules is heuristic and lacks a strong theoretical justification. For example, we could define the influence of a neuron by its activation:

$$R_i^{(l-1)} = \sum_j R_j^{(l)} \frac{g(a_i w_{ij})}{\sum_h g(a_h w_{hj} + b_j)} \tag{2.4}$$

where a_i denotes the activation of neuron i, $g(x)$ a non-linearity, e.g. ReLU-function, and b_j a bias-term. The bias-term absorbs some of the relevance. The authors improved their heuristic further and provided empirical evidence that their propagation rules yield interesting heatmaps. However, there are layers which standard LRP cannot handle, such as local renormalization layers [9].

Deep Taylor Decomposition (DTD) [53] builds on the idea of LRP but provides a solid mathematical justification for its propagation rules. Taylor decomposition is presented in contrast to sensitivity analysis [64], because both methods rely on the first derivative of a point in input space. The crucial difference between these two lies in the point chosen for evaluating the gradient: Taylor decomposition uses a root point $\tilde{\boldsymbol{x}}$ in proximity of $\boldsymbol{x}$, such that $f(\tilde{\boldsymbol{x}}) = 0$ (i.e. a point on the decision boundary), while sensitivity analysis uses $\boldsymbol{x}$ directly. Bach et al. [5] argued that the gradient at $\boldsymbol{x}$ does not necessarily point into the direction of a misclassification and does therefore not measure the actual relevance of each pixel x_p. The gradient at a root point, however, naturally points into the direction of misclassification, since it lies on the decision boundary between two classes. This gradient describes the relevance of the original point much better. In consequence, we use first-order Taylor decomposition, with Taylor residual ε:

$$f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \left(\frac{\partial f}{\partial \boldsymbol{x}} |_{\boldsymbol{x}=\tilde{\boldsymbol{x}}} \right)^T (\boldsymbol{x} - \tilde{\boldsymbol{x}}) + \varepsilon \tag{2.5}$$

$$= 0 + \sum_p \frac{\partial f}{\partial x_p} |_{\boldsymbol{x}=\tilde{\boldsymbol{x}}} (x_p - \tilde{x}_p) + \varepsilon \tag{2.6}$$

$$= \sum_p R_p(\boldsymbol{x}) + \varepsilon \tag{2.7}$$

In step 2.6 we used $f(\tilde{\boldsymbol{x}}) = 0$. Step 2.7 is true by derivation of relevance from the gradient of the root point (c.f. [53] for detailed derivations). From the final equation we can directly see that Taylor decomposition is an approximation of the LRP-problem. For Deep Taylor Decomposition we simply apply the same approach layer-wise, meaning

that we distribute $f(\boldsymbol{x})$ via hidden layers R_j onto the input layer, as shown in Equation 2.1. Yet, the question arises how to find such a root point. This depends on the type of layer in question. We briefly describe the approach at the example of a linear layer with ReLU-activation function:

$$R_j = \max\left(0, \sum_i x_i w_{ij} + b_j\right) \tag{2.8}$$

Here, the nearest root in terms of Euclidean distances must lie on the plane $\sum_i \tilde{x}_i^{(j)} w_{ij} + b_j = 0$ and on the line of maximum descent on the ReLU function $\{\tilde{x}_i\}^{(j)} = \{x_i\} + t\ \boldsymbol{w}_j$, where $t \in \mathbb{R}$. The upper index (j) reflects the fact that each neuron j can have a different root point. The root point is then given by the intersection of these two subspaces and yields the following propagation rule:

$$R_i = R_j \sum_j \frac{w_{ij}^2}{\sum_{i\prime} w_{i\prime j}^2} \tag{2.9}$$

As we would expect, the rule propagates relevance according to the learned weights between the two layers and takes the general form as defined in Equation 2.2. The so-called relevance model then performs the layer-by-layer propagation ending at the input layer. It can be shown that DTD produces heatmaps which fulfill the requirement of being positive and conservative. Empirical analysis on MNIST and ImageNet [54] showed that DTD is immediately dependent on the quality of the network architecture. For example, the deeper GoogLeNet can produce sharper heatmaps than a more shallow CaffeNet. Also, the heatmaps exhibit different checkerboard artifacts due to different stride sizes, which look similar as what has been observed in deconvolutions [55].

It should be noted that LRP and DTD can not only be applied to images and neural networks, but in principle any type of data used in neural networks. It has been put into use with text [4] and speech [7] as well. However, the only paper we are aware of which employs DTD for video data is [3]: A common practice when training CNNs on video data, such as the Sports1M dataset [35], is cutting the original videos into small snippets. This approach is based on assuming locality of label information meaning that discriminating information can be found locally within each snippet. Nevertheless, many labels might rather require the knowledge of global context. Anders et al. [3] were able to quantify the distribution of relevant information on the sequence of frames using DTD. In particular, they observed that the network typically pays rather

little attention to the first 16 frames, which they called the lookahead-effect. They ascribed this effect to the fact that many videos include an introductory sequence showing the title of the video, authors etc. Furthermore, they found that the CNN puts significantly more relevance on the first and last few frames and interpreted this as the classifier trying to looking for context beyond the small snippet. They called this the border-effect. They tried to prove their intuition by testing the network with varying subsampling rates, thus trading temporal resolution for covered duration. Classification accuracy peaked at temporal resolution of $1/2$ and diminished for smaller resolutions. Yet, the border-effect kept getting smaller, as well. In other words, the CNN was still looking for information beyond the given context even though its accuracy had dropped significantly precisely because it was provided with more context. It is therefore questionable whether their intuitive explanation that the network is in fact looking for more context is fully justified. Nevertheless, relevance analysis directed them toward improving their CNN's performance by about 1% points just by lowering the temporal frequency during evaluation.

Apart from attribution techniques, deep neural networks can be explained using direct feature visualization approaches. Feature visualization tries to find intuitive representations of the patterns which a given unit responds to particularly strongly or weakly. Deconvolution networks (deconvnets) [75] are closely related to sensitivity analysis. They try to reverse the entire CNN architecture by computing convolutions, ReLUs, pooling layers etc. backwards. Further, by optimizing the input toward maximally activating a particular unit in the network, it is possible to visualize what concepts this single unit has learned [21, 57]. Instead of creating an abstract representation of features, one can select samples which highly activate or suppress a particular unit. Bau et al. [6] went even a step further by hiding irrelevant parts of the input image, similar to what LIME does [60].

2.3 Behavioral Analysis of Zebrafish

As we argue in Chapter 1, zebrafish are a highly suitable model organism for the human body. They serve as the object of study in many fields, such as wound repair [40, 41], visual processing in the brain [61, 23, 62, 68], cancer research [72], and genetic modifications [30]. Especially in neuroscientific research, understanding behavior and behavioral changes in response to cerebral interventions is of high importance [43]. Previous studies therefore closely investigated the motion patterns of zebrafish [10,

51]. Borla et al. [10] described characteristic motion patterns during prey bouts, such as the precise bending of the very tip of the tail to bring the head into position and the subsequent strong arching of the tail's center. They highlighted that prey bouts did not exhibit so-called C-turns, but instead characteristic J-turns. Such J-turns are deflections of the tail for several hundreds of milli-seconds and are supposed to improve stability during the approach of prey. Based on their observations, they concluded that zebrafish must possess fine axial motor control of all parts of their tail.

Given that prey movements can be clearly distinguished from other types of movements, Semmelhack et al. [62] hypothesized that there is a dedicated circuitry in the brains of zebrafish. They proved this by first identifying the respective regions using two-photon imaging during prey bouts. In order to verify that specific bouts were indeed prey bouts and not spontaneous ones, they trained an SVM to distinguish these two. As prey stimulus they used small moving dots on an LED-screen as well as the natural prey of zebrafish, so-called paramecia. They found a dedicated pathway from retinal ganglion cells to an area called AF7 projecting to the optic tectum, the nucleus of the medial longitudinal fasciculus, and the hindbrain, which in turn produces the characteristic motor output. They verified their finding by ablating the AF7 neuropil and observing that lesioned fish did not respond to prey stimuli with a movement that the SVM would classify as prey.

Although not utterly important for the purpose of their study, we point out that their reported accuracy of 96% is the 5-fold cross-validated accuracy and not the accuracy on a held-out test set. They optimized the hyperparameters of their SVM on a cross-validation set. This means that they explicitly fitted this set. As a consequence, it can be expected that on a held-out test set the generalization error will be larger.

Chapter 3

Goals

The major goal of this dissertation is to make the learned features of a CNN trained on optical flow more transparent. First of all, we aim to find out whether our deep learning algorithm learns meaningful features which we can relate to extracted features identified and successfully used for classification in a Support Vector Machine (SVM) in [62]. We expect a well-trained CNN to utilize similar features, because the SVM reaches a cross-validated accuracy of 96%. It will be interesting to see if our CNN can match or even surpass their performance. Since high-performing classifiers can act as an important tool in neuroscientific research on zebrafish, it is crucial to build trust in them and verify their decisions. Moreover, visualizing learned features might yield much more valuable insights to experts than plain performance measures. This dissertation establishes a precedent for transparent deep neural networks in applied motion-based classification tasks.

Secondly, while current literature about AI explainability has focused on tasks which rely exclusively on static appearance, our classification task is supposed to depend mainly on motion cues. This dissertation expands explainability to flow-based learning. Insights can be helpful in many motion-based classification tasks, such as human action recognition.

Our third goal will be to check whether the CNN can make correct distinctions solely relying on static appearance, without further motion cues. Considering that the task asks to distinguish swim movements, we would expect that a high-performing classifier needs motion information.

Chapter 4

Methods

The following sections describe the pre-processing and augmentation of data acquired in [62], as well as the model fitting procedure of the SVM-baseline. Then, we explain the data augmentation algorithm and present the details of CNN architecture and training. The subsequent section describes how we analyze the learned weights of the CNN using relevance analysis with heatmaps. We end this chapter with a few technical challenges we had to overcome. Figure 4.1 gives an overview over all steps detailed in this chapter. In general, we seeded all scripts with a seed of 462019. We used openly available distributions of NumPy [69], Matplotlib [29], tqdm [17], OpenCV [11], scikit-learn [59], PyTorch [58], h5py [16], TensorFlow [1], Keras [15], and iNNvestigate [2].

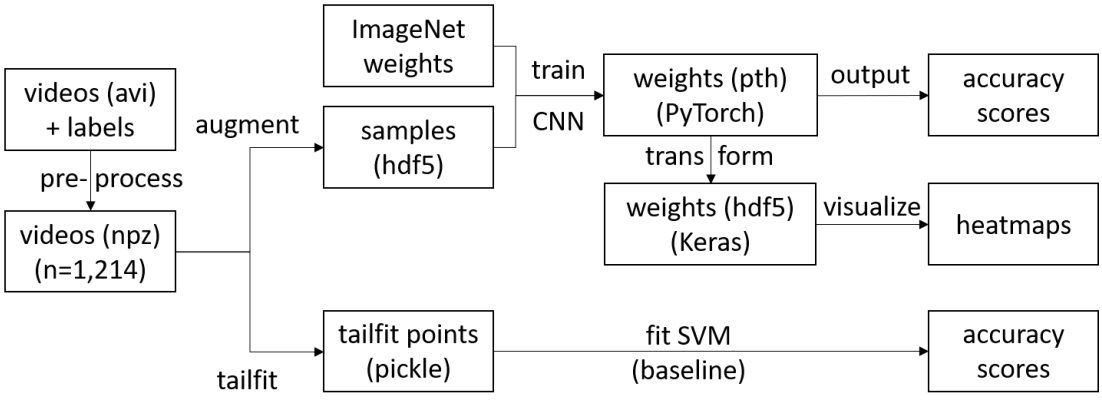

Figure 4.1: Overview over the whole project pipeline.

4.1 Data Pre-Processing and Augmentation

We have gained access to the raw video files used in [62] labeled as either spontaneous (0, negative) or prey (1, positive) bout. Videos were recorded with a high-speed camera at 300 frames per second. Our pre-processing script turned these avi-files into one npz-file containing 1,214 video snippets of 150 frames, whose size was 2.1 GB in compressed format. We chose to extract 150 frames per bout, because one bout usually takes roughly 110 frames. The goal was to keep a crop of size 256x256 pixels with the dark bladder at the left and including preferably the entire tail. We did not include the head of the fish in the crop, because the eye movements would give away information about the type of bout [8].

After turning the video into grayscale and normalizing by linearly rescaling into range 0 to 255, we therefore performed a bladder detection and placed the bladder central at the left edge. The bladder is the darkest part of the whole fish. In order to distinguish the bladder from the rest of the fish, we implemented a gamma correction with gamma set depending on the skewness of the histogram over all pixels: $\gamma = \exp\left(-\text{skewness}/\text{param}\right)$, where param was tweaked to maximize detection performance (here 4.3). This ensured that negative skewness set gamma in $[0,1)$ and positive skewness in $(1,\infty)$. Dependence on skewness improved generality of our procedure and worked well for all processed videos, recorded in varying settings. After that we applied a binary threshold at value 3, because after normalization and gamma correction the pixels of the bladder have been separated mostly below this value. While the fish is quite light, the eyes as the second darkest part of the fish might still fall under this threshold. Therefore, we first detected all contours to discard tiny contours ($< 0.01\%$ of the whole frame) and then kept only the right-most contour. Since this had to be the bladder now, we could get the crop dimensions from the right-most pixel of that contour. The contours and the result are shown in Figure 4.2.

Cropping the videos saved a lot of disk space. It only remained to extract single bout events. Each raw video mainly consisted of a still fish interspersed with a few short bouts. These were the events we extracted into consecutive 150 frames each. The idea was to detect motion by checking the percentage of pixel value changes from one frame to the next, considering only the tail. We omitted pixels other than the tail with a simple binary threshold at value 200. The pixels had to change in our case at least 0.38% of the entire pixel range (height $\times$ width $\times$ 255) in order for motion to be detected. If the algorithm detected motion for a certain number of consecutive

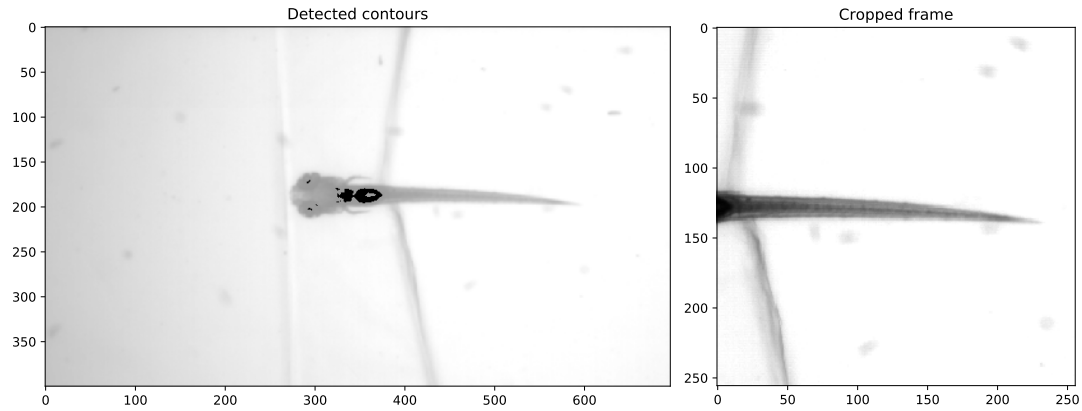

Figure 4.2: Detected contours (black) and the resulting cropped frame after normalization.

frames, it set this as the start of an event. Also, it added a preceding buffer of 15 frames. The end was set 150 frames from the start. If no motion was detected, we automatically took the first frame as a start. We had to take care that extracted videos did not overlap, i.e. in part contained identical frames, even if there were two or more distinct movements contained within 150 frames. This might have otherwise lead to train/test-contamination. Therefore, we discarded any detected motions which fell in the range of a previous video. One special case we had not yet considered was when the start was less than 150 frames from the end of the file. We shifted the start back by just enough frames to fit in, but this might have made it overlap with a previous video. Since this case was probably very rare and not detrimental, we have kept the code as it is. We manually have checked the above explained cropping and motion detection in 5% of all videos to verify correct performance.

In the following we describe our procedure for batch-wise data augmentation, which is depicted in Figure 4.3. We deemed this step highly important due to the small number of available videos. While data augmentation could be done during training on-the-fly, it would certainly deteriorate training speed because of the time consuming optical flow calculation, which took about 14 seconds per video. We used the algorithm by Farnebäck [22], because it is conveniently provided with the standard Python OpenCV distribution and yielded good results. We chose the following parameters which seemed to detect flow even when the tail moved quite fast: pyramid scale = 0.8, number of pyramid levels = 10, averaging window size = 10, number of iterations per pyramid level = 10, size of the pixel neighborhood for polynomial expansion = 13, and

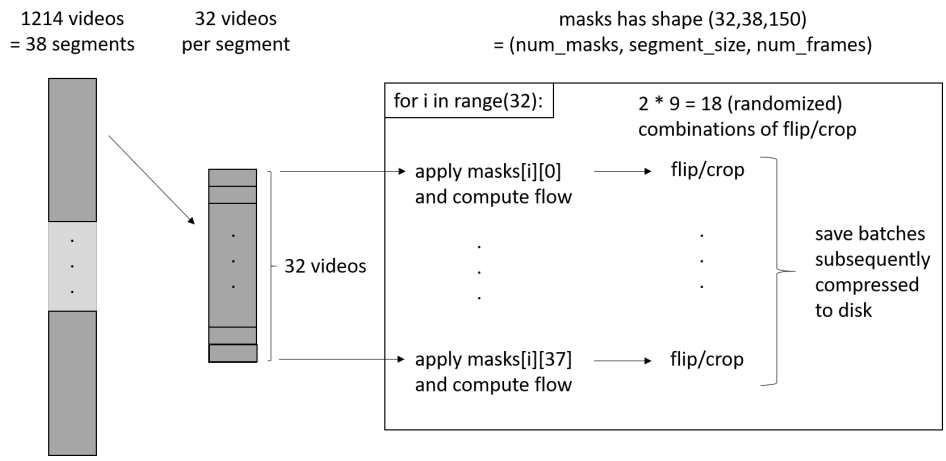

Figure 4.3: Data augmentation procedure.

standard deviation of the Gaussian used in polynomial expansion = 1.8.

Before presenting the algorithm, it should be noted that it was conceived with the thought in mind that the training procedure would do batch-wise data loading. This is because we initially had single-threading for data loading in mind. For this reason, we wanted to store single compressed files to make batch-wise data loading as efficient as possible. A difficulty of this approach was to create batches which contained samples stemming from different original videos. If a batch contained only samples derived from the same original video, the gradient would be a bad approximation of the whole dataset. This additional constraint for data augmentation made us conceive the algorithm presented here.

Only after the first training run we realized that concurrent data loading would be required to achieve acceptable batch times during training. We therefore implemented the PyTorch `DataLoader` and `Dataset` modules. They are designed for loading single samples instead of entire batches. For this reason, we needed a data structure for accessing our data sample-wise instead of batch-wise and decided to use hdf5. Nevertheless, a possible advantage of our algorithm is that it requires no shuffling and thus allows sequential disk reads.

The procedure was fully parallelizable and thus extremely scalable without any inter-process dependencies. That is because each worker created their own hdf5-file which in the end were all linked together in a final hdf5-file for lookup. This was necessary for the PyTorch `DataLoader` module to seamlessly parallelize batch-loading during training. Therefore, data augmentation could be upscaled to an arbitrary number of threads depending on the number of cores and the amount of memory available.

We assume in the following that each worker augmented a different segment, but for upscaling it would be possible to have several workers augment one segment. Furthermore, we assume the original data, consisting of 1,214 videos, to be randomly shuffled.

The algorithm performs three augmentation steps: flipping, cropping, and subsampling, which achieve augmentation factors of 2, 9, and 8 respectively. It would be reasonable to start with the smallest factor to keep computational effort as low as possible. However, subsampling requires by far the most computation time, because of optical flow calculation. It is desirable to compute optical flow after subsampling, not before, in order to increase the difference between augmented samples. Therefore, the algorithm starts with subsampling, followed by flipping and cropping. In the subsampling step, it randomly selects 86 out of 150 frames under the constraint that between two frames there may be omitted 2 frames at max. This is to ensure meaningful flow calculation afterwards, because the tail can move quite fast. Although it is theoretically possible to make the same selection twice, this is neither likely nor detrimental for training. Therefore, this case can be ignored.

After subsampling, for each video the procedure selects one of the 86 frames to be the input for the spatial network. Then it computes the optical flow resulting in 85 flow frames with one x and y component each. Since the flow is calculated in floats, the algorithm minimizes storage space by rescaling each frame to the range 0 - 255, compressing them with lossy JPEG-compression at level 40, and turns them into unsigned integers. It stores the minimum and maximum values for being able to rescale them into the original scale later.

From one mask the script can then generate 18 batches by flipping and cropping the frames of this segment. Since each loop generates one batch, it uses a randomly shuffled index in the range 0-17 for each sample in the segment, indicating which flip and which crop to add to the batch. The script then performs vertical flipping and takes 9 crops of size 224x224 with the upper left corner at (8,8), (8,16), (8,24), (16,8), (16,16), (16,24), (24,8), (24,16), and (24,24). Since the CNN needs the flow frames as channel inputs (c.f. Figure 4.5), the algorithm stacks the resulting 170 frames by alternating x and y frames, before appending this batch to the hdf5-file. Each worker finally creates a file of 32 * 2 * 9 * 8 = 4,608 augmented samples.

We generated three datasets of different sizes on a node in the James and Charles Cluster, which was a Dell PowerEdge R815 with four 16 core Opteron CPUs, 256 GB memory, and 4 TB of disk space (HDD). Originally, we wanted the largest dataset to

ID	MASKS PER VIDEO	SAMPLES	COMPRESSED SIZE	EPOCH TIME
S	1	21,852	25 GB	0:51H
M	4	87,408	108 GB	7:29H / 9:49H
L	8	174,816	216 GB	14:20H / 22:18H

Table 4.1: Summary of the utilized datasets.

be generated with 32 masks to amount to approx. 17.1% of what would be used in ImageNet if the task was binary. We assumed this to be feasible and appropriate for the relatively small network. However, training showed epoch times of over 14 hours when the dataset was generated with only 8 masks. Given the short time frame of the project and observing fruitful results on smaller datasets, we decided to stick to the datasets summarized in Table 4.1. As for the varying epoch times, it became apparent that some nodes in The Teaching Cluster must have had differing configurations with less memory, slower GPUs, and different numbers of CPU cores. This led to confusion about the maximum possible batch size and varying epoch times during training.

Since we wanted the segment size to equal the batch size (= 32) we used $\text{ceil}(1214/32) = 38$ workers, which resulted in a peak RAM usage of 125 GB. We used Python's built-in `multiprocessing` module. The files were compressed with gzip-compression at maximum compression level. In contrast to that, lzf-compression would yield extremely fast computation but at a low compression level. We realized that maximum gzip-compression outperformed lzf-copmression by a factor of 1.76 when comparing training times (batch times of 2.66 seconds vs. 4.68 seconds). This advantage can be ascribed to heavy parallelization of uncompression (16-32 workers, depending on the available node) and relatively low transfer speeds between hard drive and memory.

4.2 Fitting the Baseline SVM

For each frame in the 1,214 videos, we applied the same tail-fitting code used in [62] to compute points along the tail. Each point was set at 5 pixels distance to the previous one. Typically this resulted in 30-45 points per frame, as shown in Figure 4.4. Furthermore, we initialized the procedure central and 8 pixels from the left edge, because after pre-processing we assumed this to be just next to the right end of the bladder. Yet, some of the videos contained frames which the tail-fitting code had problems process-

ing, so that such frames had no or very few distinct fitted points. We accepted frames only if they had at least 13 unique points. Potential issues might have been caused by cropping the videos, because this might have cut off tails that were strongly curved. After substantial debugging, we were able to maximize the number of correctly processed videos to 953 out of 1,214 videos and use this as the SVM dataset, including 482 (50.6%) spontaneous and 471 (49.4%) prey bouts.

We fed this dataset to the feature extraction and model fitting algorithm which split the set into 85% training and 15% held-out test set. Semmelhack et al. [62] have identified 5 key features which allowed their SVM to achieve a cross-validation accuracy of 96%. These features were "maximum tail curvature", "number of peaks in tail angle", "mean tip angle", "maximum tail angle", and "mean tip position", sorted by decreasing importance. We used their provided code to extract these features. Then we performed a grid search to tune SVM-kernel, gamma, and parameter C. (RBF-kernel with gamma in [1e-1, 1e-2, 1e-3, 1e-4] and C in [0.01, 0.1, 1, 10], linear-kernel with C in [0.01, 0.1, 1, 10]). For validation we used stratified 5-fold cross-validation [42]. The fitting procedure took only a few seconds. It should be pointed out that we did not use augmented data here, because the SVM would not have gained from flipping, cropping, or subsampling the recordings.

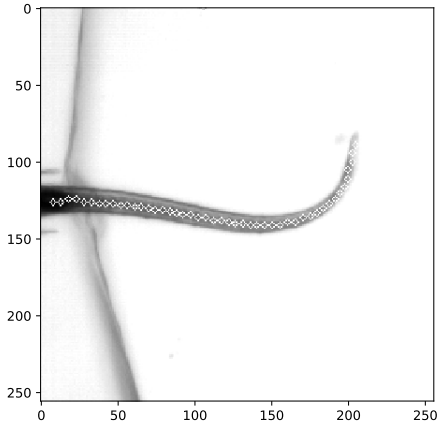

Figure 4.4: Distribution of points along the tail.

4.3 CNN Architecture and Training Procedure

Just like Simonyan and Zisserman [64], we used a two-stream network with an adapted CNN-M-2048 network [14] for each stream. This is because we needed a model that could deal with a small sample size and could be trained quickly. The results of the two-stream network in [13] looked promising. As depicted in Figure 4.5, the full network consists of 5 convolutional layers and 3 fully connected layers, interspersed with max-pool, local response normalization, and dropout layers. Its number of parameters amounts to 135,488,740 (spatial 67,346,882, temporal 68,141,858), including biases.

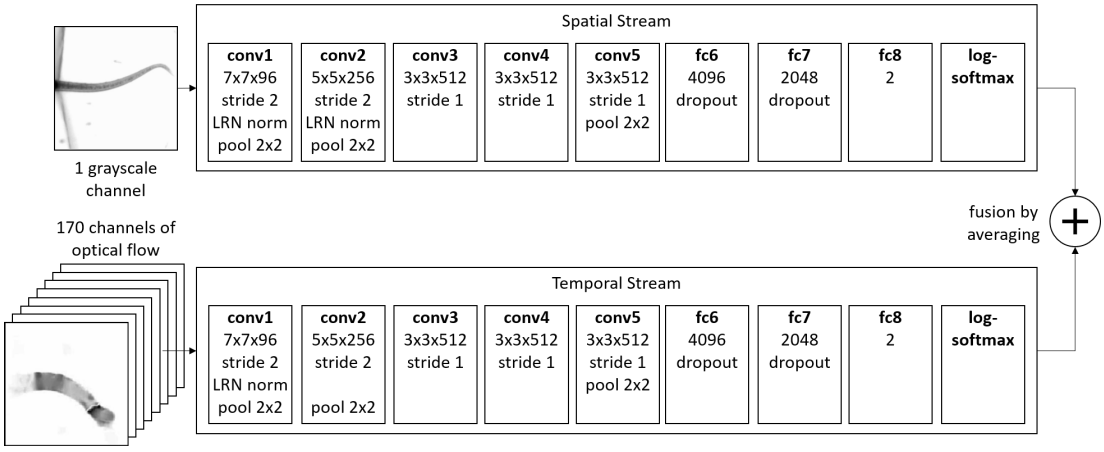

Figure 4.5: Full CNN architecture.

We set as activation function the ReLU-function. The spatial stream takes one grayscale image as input (shape = (b, 1, 224, 224)), whereas the temporal stream receives 170 flow frames of datatype float as channels (shape = (b, 170, 224, 224)). After obtaining the predicted probabilities of each stream by calculating the log-softmax of the individual two outputs, they are fused by averaging. We compute the negative log-likelihood loss of the individual streams and their average. The average can be interpreted as the joint log-probability of both streams, assuming their statistical independence, which therefore sums to 1.

We implemented PyTorch's `Dataset` module to make use of the multiprocessing capabilities of the `DataLoader` class. We only had to take care of converting the running sample index into a file index, considering that samples were stored in one of 38 hdf5-files. Importantly, we split the data into train, validation, and test datasets by file (i.e. segment) and not by samples in order to prevent train/test-contamination. That was possible because each file had been derived from 32 unique original videos. In consequence, the train dataset was made up of 28 files, validation of 4, and test of 6, while the last file contained slightly fewer samples. They were processed without shuffling, because we had taken care of randomness before and might gain speed through sequential reads. We performed our experiments on the Teaching Cluster which provided about 50 nodes with 4 GPUs (NVIDIA GTX1060 6 GB) each. Nodes contained between 16 to 32 CPU cores allowing for heavily multi-processed data. This was necessary because the bottleneck of training was loading from disk and un-compression of consecutive batches. We were able to optimize the sending of optical flow frames to CUDA by performing the required re-scaling and conversion back to floats only after

sending them as unsigned integers. Moreover, since re-scaling was a linear element-wise operation, it was extremely fast on GPUs.

We chose a batch size of 32, because batch sizes larger than 50 exceeded memory. Also, we believe that it allowed a decent approximation of the gradient with some inherent noise, which is desirable because it avoids falling into sharp minima [37]. We made use of the Adam optimizer with standard settings and tuned its learning rate and weight decay coefficient – the neural network equivalent of L2-regularization [49]. Furthermore, we added a gamma learning rate scheduler which updated the learning rate by a multiplicative factor of $\gamma = 1/\sqrt{epoch}$ [39] every epoch. Our training framework computed accuracy on the validation set after each epoch to measure generalization performance. It was based on code from the Machine Learning Practical (MLP) course but strongly adapted. In addition to that, we outputted accuracy, F1-score, precision, and recall of the full network and of each stream individually.

Moreover, we initialized both streams with weights pre-trained on ImageNet, which were available as a Matlab-file from `http://www.vlfeat.org/matconvnet/models/imagenet-vgg-m-2048.mat`. While Simonyan and Zisserman [64] did not pre-train flow, Carreira and Zisserman [13] found a pre-trained temporal stream to reach superior performance. Moreover, given the short time frame of the project we needed to keep training time to a minimum. We averaged 500 units on the output layer for each output unit, because we dealt with only 2 classes instead of 1,000 as in ImageNet. Regarding inputs, for the spatial stream we took the average of the available 3 RGB-channels to get the weights for 1 grayscale-channel. For the temporal stream we copied the RGB-channels $56\frac{2}{3}$ times to get 170 channels, and added uniform random noise to all of them. This was to ensure that the channels evolve differently during training and should have aided learning. With initialization we hoped that training would require only fine-tuning and therefore fewer epochs.

As outlined in Section 4.3, by and large we drew on generally accepted hyperparameters for the Adam-optimizer and learning-rate scheduler. However, since we were initializing our CNN with weights learned from quite a different domain, we might need more specific tuning here. In particular, we focused on learning rate and weight decay. In the following, we describe the 5 most insightful training experiment runs. Each experiment was run over 3-8 epochs, which proved to be sufficient. Furthermore, we do not report standard errors on the statistics because of time constraints and because an extremely accurate hyperparameter search was not required for the purpose of this dissertation.

First of all, after some issues with a bug in the code preventing learning completely, we wanted to make sure the training loss was indeed affected by training. We ran experiments with very large learning rates ([0.1, 0.01]) and weak regularization on the smallest dataset. We did not include any statistics for these experiments, because they were of little interest. The results of the experiments described in the following can be found in Tables 5.1 and 5.2. After verifying that the loss became less and less erratic the smaller the learning rate, we decreased the learning rate one step further and turned up regularization (weight decay at 0.01 and dropout rate at 0.9). Carreira and Zisserman [13] successfully used this dropout rate with CNNs which had been initialized with ImageNet weights. This experiment (ID 0) yielded the first fruitful results (validation accuracy of 0.8854 in the first epoch). Observing that the loss was still not declining smoothly, we concluded that we could increase both learning rate and weight decay further. We completed a mini grid-search as our third run of experiments over learning rate ([1E-3, 1E-4]) and weight decay ([1E-2, 1E-3]) (IDs 1, 2, 3). The model with learning rate at 1E-4 and weight decay at 1E-3 achieved the best validation accuracy until then of 0.9591.

Confident that these hyperparameters achieve high performance, we ran our fourth experiment on the medium-sized dataset, even reaching slightly higher performance (ID 4). We set dropout to 0.95 here to increase regularization. Training progress (Figure 5.1) showed that the network still seemed to be overfitting and made large jumps in validation loss. Therefore, we concluded that even smaller learning rates and stronger weight decay might be beneficial. Considering that the previously achieved accuracy was highly sufficient for the purpose of analyzing learned features, we decided to stick with this model. Out of curiosity, we ran the remaining set of experiments with the most promising hyperparameters on the largest dataset for 3 epochs (dropout = 0.9 here).

4.4 Relevance Analysis with Heatmaps

Making our CNN more transparent required an AI explainability technique which would be well interpretable in the optical flow domain of our temporal stream. We expected feature visualization approaches [57, 6, 21] and deconvolutions [75] to yield less interpretable outputs than attribution techniques. In particular, we chose to analyze our CNN with Deep Taylor Decomposition (DTD), because it had been used successfully before [46, 70, 3] and was conveniently accessible in the iNNvestigate

toolbox [2]. The produced relevance heatmaps could be expected to give clues about what specific regions of optical flow, within a frame and across frames, the network is putting attention on. The toolbox only supports Keras with the TensorFlow-backend. Therefore, we re-implemented the exact structure of our CNN and initialized it with the extracted weights from PyTorch. The conversion could be done with tools like ONNX, but after a few unsuccessful attempts we transported the weights with a custom Python script. Also, we simplified the analysis by splitting the network into its individual streams. This was possible because no weights are learned after the final layer of each stream. Once the Keras network was initialized correctly, iNNvestigate made the generation of heatmaps surprisingly simple. Also, with about 50 minutes for the whole analysis it was quite fast even on CPU, because it effectively only needed one forward and backward pass per sample. Considering the time frame of the project, we decided to use the small dataset of 3,420 samples for analysis in order to simplify and accelerate the process.

A caveat to the iNNvestigate toolbox emerged after heatmap generation: it had problems analyzing 1,578 of the 3,420 samples, which produced an empty output. We made sure the problematic samples did not follow any pattern by checking the indices of correctly analyzed samples, the ratio of true positives and negatives and false positives and negatives, as well as the confidence distribution (as in Figure 5.12) after the analysis. Since all numbers were the same as before the analysis, we continued with 1,842 samples for further investigation.

For the final analysis of all relevance heatmaps, we expected samples with high confidence of prediction especially insightful, because we assumed the network to have found a discriminating feature here. The distribution of confidence can be viewed in Figure 5.12. Therefore, we selected the most confident true positives/negatives and false positives/negatives for our individual case studies, overlaying flow frames with heatmaps. We visualized optical flow in simple grayscale because visualizing the exact direction of flow with colors would be distracting. Furthermore, we analyzed heatmaps averaged over specific types of samples to better understand the characteristics of correct and false responses.

Chapter 5

Results

5.1 Classifier Training and Test Statistics

We performed several training runs to reach high accuracy in distinguishing prey bouts of larval zebrafish from spontaneous swims. The results refer to three datasets of different sizes which were derived from the original videos by applying three different augmentation factors. All models were initialized with pre-trained weights from ImageNet. We present the results of the seven most relevant CNN training experiments on training, validation, and test sets in Tables 5.1 and 5.2 in chronological order. For our baseline SVM we report the 5-fold cross-validated accuracy and the final accuracy on the held-out test set. The hyperparamters agree with the ones found in [62] (RBF-kernel with gamma = 0.001 and C = 1). In order to obtain fast results, we started by training our CNN on the small dataset, which allowed 8 epochs in 8 hours. In contrast, the medium sized dataset would have required about 60 hours for the same number of epochs. The outcome of each experiment informed which experiment to run next.

In particular, experiment 0 showed clearly that a learning rate of 1E-3 was too large, because training loss was lowest in the first epoch, even though still quite large, and diverged subsequently. Stronger weight decay alone as in experiment 1 could not change this either. However, experiment 2 with smaller learning rate showed clear improvements, reaching a validation accuracy of 0.9089. Yet, considering its training accuracy of 0.9890 and the diverging training and validation losses, the model was certainly overfitting. Presumably, this model was reaching such a low validation loss by giving not too overly confident predictions, thus avoiding high penalties in the negative log-likelihood loss for confident false predictions.

Finally, experiments 3 and 4 proved the quality of our elicited hyperparameters

with validation accuracies of 0.9592 and 0.9489. The lack of standard errors on these numbers makes further claims rather speculative, but we can probably assume that the larger dataset had a regularizing effect. This showed especially in comparison to experiment 5, where training and validation loss did not diverge as much as in 3 and 4. Comparing precision and recall of the two best models (4 and 5), we can report a rather interesting result. It seems as if model 5 was trading recall for precision. We discuss this further in 6.3. The final experiment showed that the additional regularization due to stronger weight decay drove the weights too small for effective learning. Due to the long training times of models 5 and 6, we decided to stick with model 4 for the subsequent relevance analysis in sections 5.2 and 5.3. Nevertheless, we can report a final test accuracy of 96.69% for our winning model, beating the baseline by 6.49%.

In addition, we compared the results of the two individual streams in Table 5.2. Across all experiments, loss and accuracy were dominated by the temporal stream. The advantage of model 5 over 4 originates in the significantly improved temporal stream, which could lower its test loss to 0.4127 and raise its accuracy to 0.9960.

We further take a closer look at training and validation statistics during training of model 4 over 8 epochs shown in Figure 5.1. First of all, training loss of the temporal stream always stayed far below that of the spatial stream. This is not the case for validation loss. Potential overfitting seems more pronounced in the temporal stream, driving up its validation loss thus increasing generalization error in the third epoch. Furthermore, temporal training accuracy started extremely high whereas the spatial stream took longer to reach comparable accuracy. Validation accuracy dropped for both streams already after the third epoch. This is another indicator of bad generalization ability. In fact, experiment 5 shows that a smaller learning rate is indeed beneficial. Coupling this with even stronger regularization might yield even better results. Furthermore, we highlight that the combined training accuracy remains always above that of the temporal stream. This suggests that the best validation accuracy benefits from both streams, which can be observed in the tables presented above as well.

5.2 Relevance Case Studies

Obtaining a CNN with classification accuracy of 95.96%, we assumed this model (ID 4 in Table 5.1) to have learned discriminating features. We tried to get an understanding of these features by visualizing its weights with the help of relevance heatmaps. Each relevance heatmap is specific for one sample. It visualizes all regions the CNN pays

ID	LR	WD	SET	EP	TRAIN L	TRAIN A	VALID L	VALID A	TEST L	TEST A
B	–	–	–	–	–	0.946	–	0.946	–	0.902
0	1E-3	1E-2	S	0	.5807	.8702	.7753	.8854	–	–
1	1E-3	1E-3	S	3	.7081	.9014	.9773	.8845	1.7813	.8435
2	1E-4	1E-2	S	3	.1777	.9890	.5564	.9089	.5164	.9417
3	1E-4	1E-3	S	4	.0933	.9976	.7592	.9592	.7815	.9591
4	**1E-4**	**1E-3**	**M**	**1**	**.0897**	**.9962**	**.6679**	**.9489**	**.7747**	**.9596**
5	**1E-5**	**1E-3**	**L**	**0**	**.1361**	**.9869**	**.6311**	**.9372**	**.7057**	**.9669**
6	1E-4	1E-4	L	1	.0594	.9979	1.2989	.9094	–	–

Table 5.1: Summary of the most relevant experiments showing learning rate (LR), weight decay coefficient (WD), dataset used for training (SET), best epoch (EP), losses (L) and accuracies (A). First row: SVM baseline.

ID	PREC	REC	SPATIAL L	SPATIAL A	TEMPORAL L	TEMPORAL A
1	.9403	.9227	.6798	.7702	.3530	.9349
2	.9205	.7100	.6853	.5632	2.8773	.8448
3	.9564	.9401	.9939	.7475	.4119	.9611
4	**.9451**	**.9654**	**.9263**	**.8216**	**.6230**	**.9441**
5	**.9911**	**.9334**	**.9986**	**.8138**	**.4127**	**.9660**

Table 5.2: Summary of evaluation on the test set showing precision and recall of the full network, and loss and accuracy of the individual streams.

most attention to in dark red color, while light red stands for low relevance.

First of all, we present the ten most interesting consecutive flow frames of the single most confident true positive sample (Figure 5.2), true negative (Figure 5.3), false positive (Figure 5.4), and false negative (Figure 5.5). Moreover, we gather five particularly interesting flow frames in Figure 5.6. We define prey bouts as positive and spontaneous swims as negative. With trunk we refer to the rostral part of the fish's tail, which is closer to the head. With end we refer to the caudal part of the tail, which is closer to the tip. Overall, flow frames show a surprising diversity of relevance patterns across samples. As expected, they also exhibit the checkerboard artifacts typical of kernels with stride 2 in the first convolutional layer [55, 53].

As for the true positive, we observe a very sharp relevance pattern along the edges

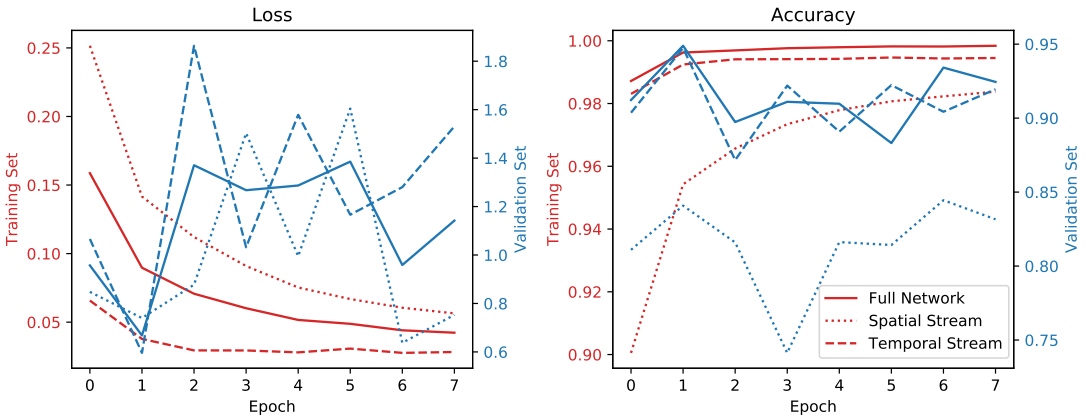

Figure 5.1: Loss and accuracy during training of the best architecture (ID 4) on training and validation sets.

of the tail. Relevance is highly concentrated at the trunk. Although the end of the tail is visible in many frames, the CNN pays little to no attention to it. The flow frames do not appear to depict a proper J-turn [62], yet the CNN confidently reached a correct classification. The example of a true negative seems to be a representative specimen of spontaneous movement. The relevance pattern reflects the high frequency of tail deflections. We can further observe that the trunk even at the very start of the tail is set in motion, although much less pronounced than at the tip. It also becomes apparent that the trunk moves more strongly up and down compared to prey bouts. This results in an arching of the entire tail which resembles the bending of a bow. The heatmap represents this fluctuation with a vertically more spread out relevance pattern.

The CNN made false predictions with much less confidence than true predictions. Figure 5.8 shows the sorted confidence of each sample. False predictions lie only between the red lines. Confidence for each sample was calculated as $-\log\left(\log(\mathrm{P}(1))/\log(\mathrm{P}(0))\right)$, such that positive values show high confidence for prey and vice versa. Values around zero show indifference. Furthermore, the mean is slightly shifted toward spontaneous bouts, indicating that the CNN could label spontaneous bouts with higher certainty. This is in line with the ratio of false positive to false negative samples, which is 1.72 regarding the entire test set.

With this knowledge in mind, the presented false predictions make more sense. The false positive sample in Figure 5.4 does neither include a prey nor a spontaneous bout. Yet, the pre-processing script has detected this movement of relaxation of the fish. Interestingly, it seems that the CNN was still looking for clues from the agarose

and impurities in the video. In the false negative example we can see that the CNN put attention on frames even if there was little flow.

We find the frames in Figure 5.6 insightful because they show five interesting things: Firstly, in some samples the heatmap shows relevance even for vertical edges within the tail. Secondly, even if the tail was quite certainly performing a J-turn, the CNN did not pay it much attention. Rather it looked at the agarose in the top left corner. Thirdly, occasionally optical flow shows for all objects in the frame at once. This might be induced by a flaw in the pre-processing script, because it re-centers the fish in every frame individually instead of centering all frames of a video the same way. The fourth subfigure shows that optical flow can become very washed out due to subsampling. We kept only 85 out of 150 frames with the constraint that frames could only be up to 3 frames apart. Yet, for fast movements this might have resulted in a large pixel difference between two frames. Flow was only detected because we were using a sufficiently large Gaussian kernel to still detect that movement. Finally and most strikingly, the CNN could make a correct classification even without considering the tail much. In fact, in all flow frames of this sample the relevance pattern is entirely concentrated on the agarose in the top left corner. We will analyze this closely in Chapter 6.

In the spatial heatmaps in Figure 5.7, we noticed that relevance follows the curvature of the tail. Yet, they do not show as characteristic discriminative patterns as the temporal ones. We observed that typically more attention was put on the trunk than on the end here, as well. The CNN seems to be extremely good at detecting edges, both tail edges and the center line in the tail. Relevance was often still uniformly distributed across the frame, indicating a not fully finished learning process.

5.3 Relevance Averages

For more comprehensive insights in the features learned by our CNN, we computed relevance across samples and frames, as well as split by label. This produced the averaged heatmaps presented in Figure 5.9. Our most important observation is that negative classifications in the temporal stream are based to a considerable extent on motion in the top left corner. This resembles very much a ”Clever Hans” [46] type of correlation. On the other hand, this spurious correlation does not seem to play a role in positive classifications. Moreover, we noted that the spatial stream, not relying on motion information, did not find this correlation. Furthermore, both temporal and spatial stream generally pay most attention to the trunk of the tail and practically none

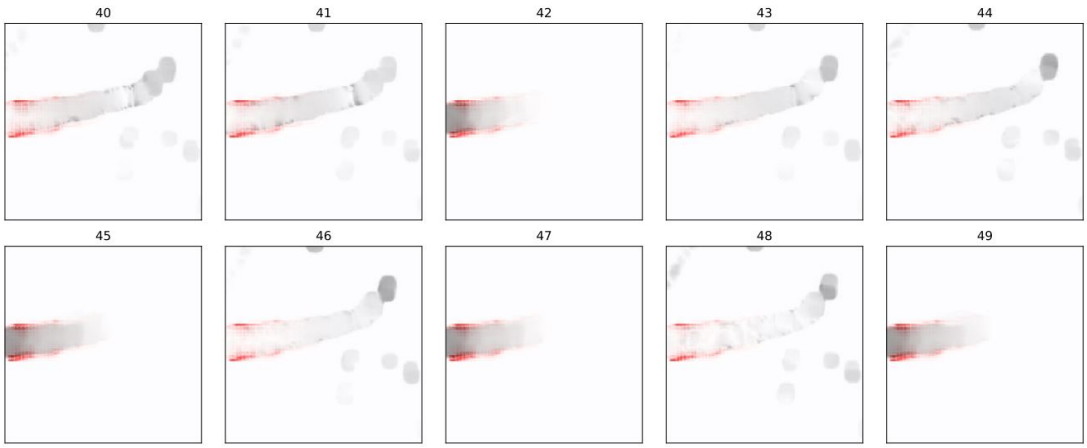

Figure 5.2: Flow frames of the most confident true positive sample. Optical flow in grayscale, relevance heatmaps overlayed in red.

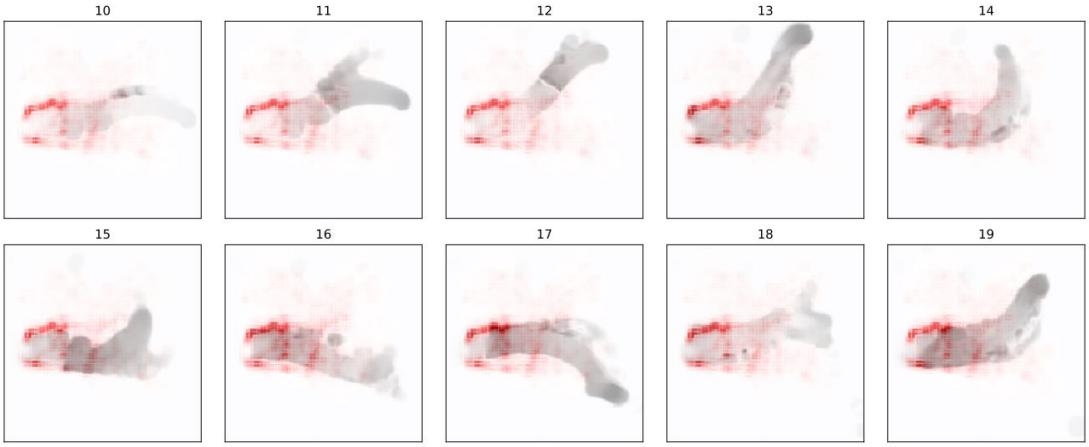

Figure 5.3: Flow frames of the most confident true negative sample.

to its tip. Also, they are vertically symmetric as we would expect since we feed both original and vertically flipped samples. Surprisingly however, this is not true for the spurious correlation discovered in the temporal stream. Furthermore, although we can observe in individual samples that the CNN did pay some attention to the blobs and impurities of the background, this seems to have happened relatively little, because it does not become visible in averaged heatmaps. Finally, we observed that the spatial stream showed some unexpected pattern at the right and bottom edges. We further split the considered subsets into true positives/negatives and false positives/negatives to get a more differentiated view, and considered only the 37 most confident samples in each class (20 in the case of false negatives, because no more samples were available), which

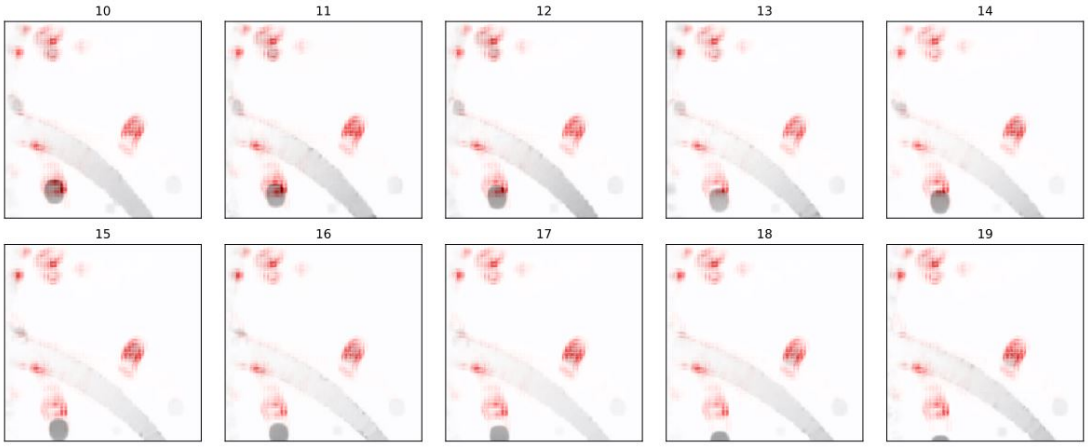

Figure 5.4: Flow frames of the most confident false positive sample.

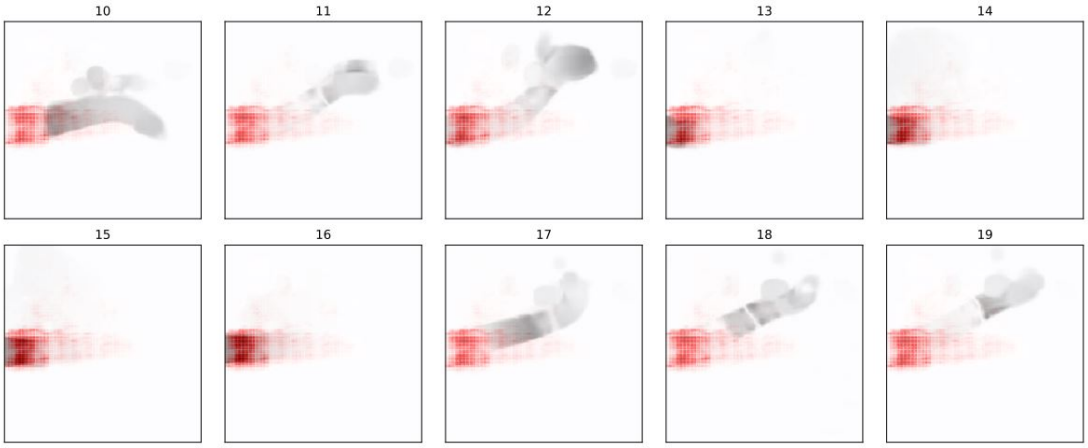

Figure 5.5: Flow frames of the most confident false negative sample.

equals 2% of the whole analyzed test set. The results are depicted in Figure 5.10. First of all, the spurious correlation was only observable in false positives. This indicates that the CNN could potentially exploit this spurious feature even more and improve its classification of negative samples. Most interestingly however, even when considering 184 samples (10% of the whole set), confident true negatives relied almost entirely on actual tail movements. Thus, for its most confident correct classifications the CNN must have used tail features and not some spurious correlation.

Since the most confident 10% of true negatives showed no relevance in the top left corner, we became interested in the remaining true negatives. We split them up into windows of 76 samples and present their averages in Figure 5.11. We observed that tail features got more and more relevance, the more confident the classification. Yet,

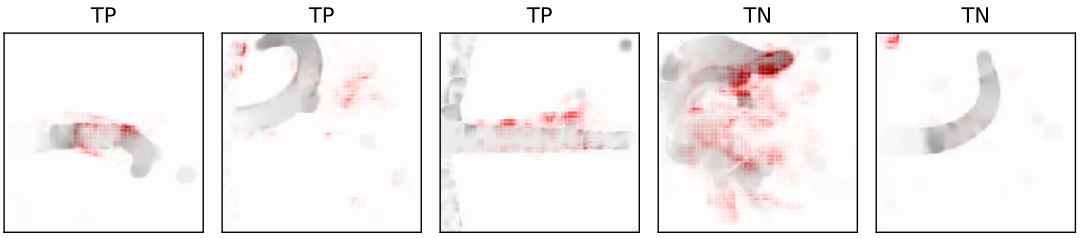

Figure 5.6: Flow frames of a selection of true positive (TP) and true negative samples (TN).

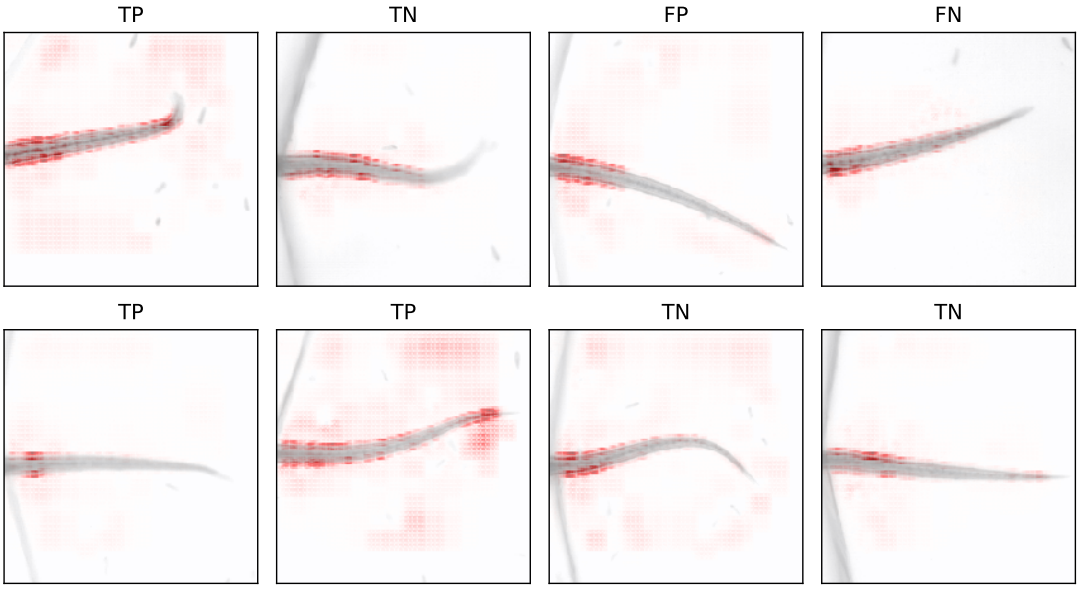

Figure 5.7: Upper row: spatial input of the most confident samples of each category. Lower row: spatial input of a selection of samples.

many of the less confident true negatives relied on agarose motion as a feature.

We further plotted the distribution of relevance over the sequence of frames in Figure 5.12. Most of the relevance is concentrated over the frames in range 7-46. The first seven frames are of least importance. Furthermore, we cannot identify a small subset of extremely important frames and the Pareto principle does not hold here either: 80% of the total relevance comes from 65% of all frames. In a boxplot, we could not identify any outliers either.

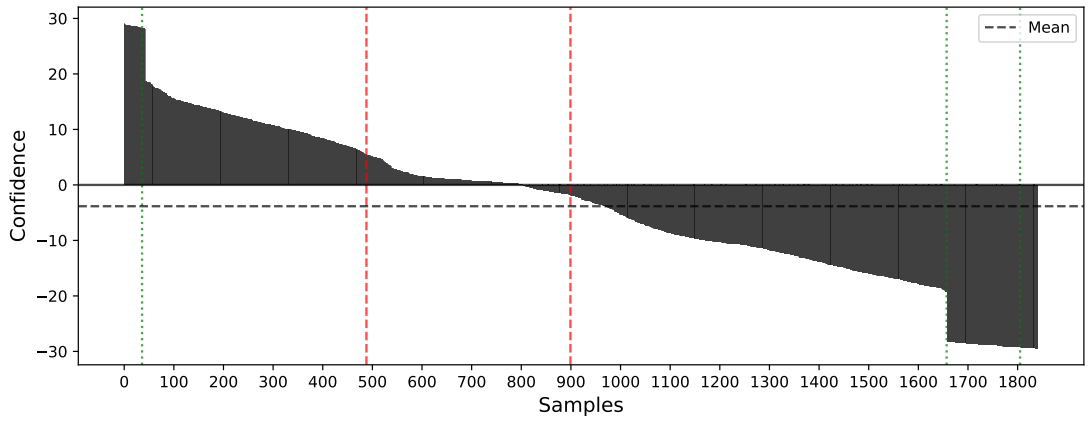

Figure 5.8: Sorted distribution of classification confidence over samples in the analyzed test set of 1,842 samples. Positive values stand for positive responses and vice versa. Red lines: most confident false positive (left) and negative (right). Green lines: 2%, 10%, and 98% quantiles.

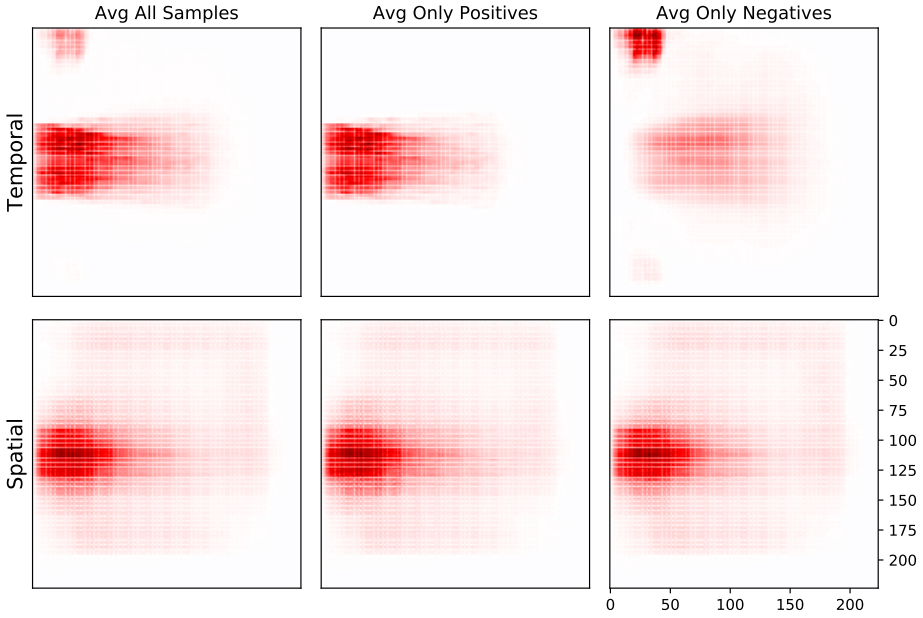

Figure 5.9: Averages of temporal and spatial heatmaps either over all samples, only positives, or only negatives.

Figure 5.10: Average heatmaps over 2% (= 37) of samples which are true positives/negatives (TP/TN) and false positives/negatives (FP/FN) respectively, and over 10% (= 184) which are true negatives (TN (10%)).

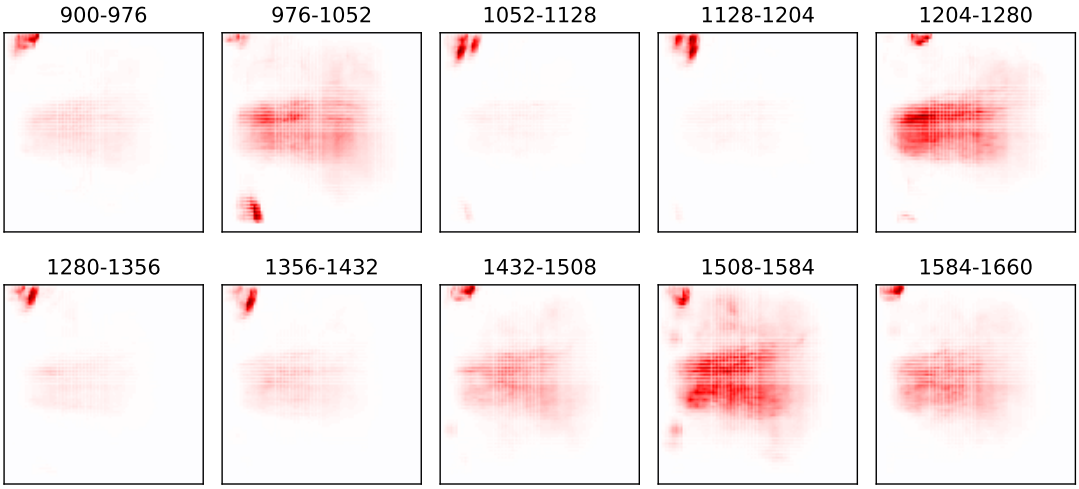

Figure 5.11: Average heatmaps of 76 samples per subfigure, sorted by increasing confidence for responding negative. The indices correspond to Figure 5.8.

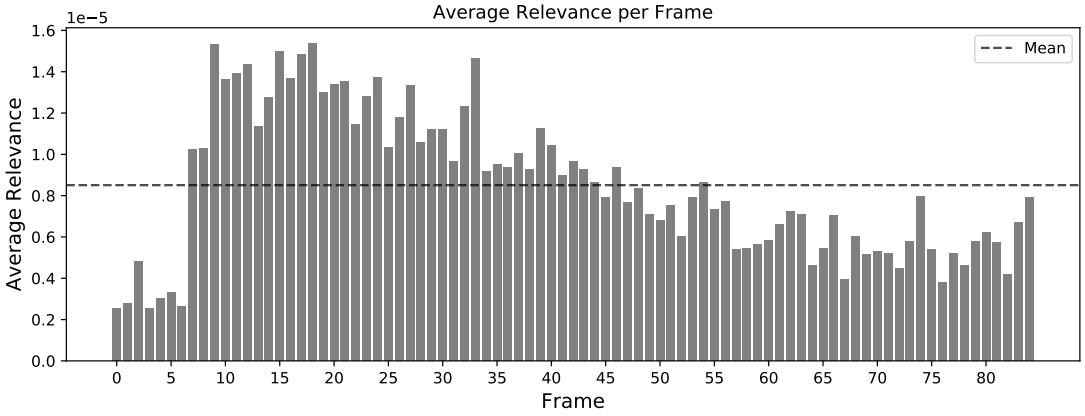

Figure 5.12: Distribution of relevance on the flow frames of the average sample.

Chapter 6

Discussion

6.1 Feature Analysis

A high classification accuracy and the produced relevance heatmaps of the temporal stream make clear that the network was able to differentiate the movements of zebrafish based on their characteristic motion. One piece of evidence supporting this conclusion is the concentration of relevance on frames 7-46 depicted in Figure 5.12. Due to our pre-processing procedure, which added a buffer of 15 frames before each detected motion, these frames were the ones to contain the actual bout movements for the vast majority of samples. The CNN was therefore focusing on those frames which are in fact the most relevant ones. Since the CNN has apparently not received much information from frames 0-7, we could conclude that a buffer of eight frames would have been sufficient. As expected, the border or lookahead effect as described in [3] could not be observed here.

Analyzing heatmaps of correctly classified samples, we observe that the network achieved correct classifications by looking for salient features in the upper half of the tail while largely disregarding the tip. Furthermore, it seems that pre-trained weights lead both streams to detect edges, which was evidently highly aiding correct predictions. This was crucial to our project, given the little training time available. The nature of edges especially along the trunk seems to be a differentiating learned feature in temporal heatmaps. The heatmaps of the most confident true positives depicted in Figure 5.10 show that a sharp and clear relevance profile confined to the edges of the trunk give a clear sign of a prey bout. The opposite speaks for a spontaneous bout, as the true negatives show in Figure 5.10. Here, attention spreads out to capture the strong vertical oscillation of the trunk. For this reason we conclude that the CNN makes its

predictions based on the steadiness of the trunk. We believe this interpretation to be in line with existing research on the kinematics of prey bouts. As shown in [10, 51], prey bouts require fine control of the tail's axial kinematics to perform precise swim movements. Zebrafish noticeably reduce their yaw rotation and stabilize the positioning of their head to make a targeted move at the prey. The heatmaps indicate that the network might have found clear evidence for this in the trunk of the tail.

However, we highlight that our optical flow calculation often disregarded the tip. This was possibly due to suboptimal parameter settings which could not handle the long distances which the tip travels between frames. The CNN had therefore more information about the trunk than the tip which might induce it to not consider tip features. Another point of criticism is that our subsampling procedure initialized each of the 38 workers with the same seed. As a consequence, all videos were subsampled with the same set of randomly generated masks. While we do not believe this to have introduced a systematic bias, it would have been better to use a different seed for each worker.

As for the role of the spatial stream, we can confirm the findings in [64], namely that the spatial stream by itself reaches a fairly competitive accuracy. This might come as a surprise, given that the task asks as to distinguish two movements, while the spatial stream receives no direct motion information. Yet, apparently there are insightful features to be learned from still frames. The spatial stream focused largely on the upper half of the tail just like the temporal stream. This suggests it was looking for very similar features. If that is the case, we should see improved performance when giving the spatial stream a sequence of frames. It should be interesting to probe whether the spatial stream could then match or even surpass the performance on optical flow. In its current form, the spatial stream provided only very minor improvement to the overall network.

An alternative explanation for the good performance of the spatial stream could be that it was learning utterly different features than the temporal stream. For example, it might have been looking for specific characteristics of each fish, thus being able to classify individual fish. This strategy would work if we assume that each fish has a rather lopsided ratio of spontaneous and prey bouts. Since there is no apparent reason for this assumption, we do not believe this explanation to be sufficient for explaining its quite notable classification accuracy of 82.16%.

6.2 Comparison to SVM Features

In the following, we analyze how the generated heatmaps might relate to the extracted features used in SVM training. Semmelhack et al. [62] have identified the following five features as the most impactful ones, ordered by descending importance:

1. Maximum tail curvature (maximum over the bout)
2. Number of peaks in tail angle
3. Mean tip angle (absolute value of tip angle in each frame, average over the bout)
4. Maximum tail angle (maximum over the bout)
5. Mean tip position (average position of last eight points in tail, with horizontal deflection as a fraction of the tail length)

The first feature computes the mean over all angles between three consecutive points on the whole tail and then keeps only the maximum. As a result, this feature establishes its discriminative power mainly in the frames of maximum deflection and relies on points over the entire tail, including the tip. These two deciding factors are not given in the optical flow frames used for CNN training. Firstly, the frames of maximum deflection might well be excluded from the actual sample, because subsamples include only 85 of 150 frames from the original video. Secondly, many frames do not even depict the tip. The optical flow algorithm often did not detect motion in the tip, thus showing only the trunk. Finally and most importantly, the relevance heatmaps show that the CNN paid most attention to the trunk and hardly any to the end or to the tip. Most certainly, the CNN was therefore not even trying to find the maximum tail curvature over the bout.

The same arguments hold true for features 3, 4, and 5. Feature 3 calculates the angle between tail trunk and tip for each frame, and keeps the mean. If the CNN had no access to the tip and did not put relevance on it, most likely it was not using this angle as a feature. Feature 4 finds the maximum angle between three consecutive points anywhere on the tail for each frame, and keeps the maximum. The chosen three points will most probably not lie on the trunk but on the tail, because here the strongest deflection takes place. Finally, feature 5 is based on tip position as well.

For these reasons, we argue that the CNN cannot possibly have learned features 1, 3, 4, and 5, but must have relied on features not considered in [62]. Due to its higher performance, we conclude that these features must bear higher discriminative power. On the other hand, feature 2 might coincide with features used by the CNN because technically the CNN could have counted the number of peaks in curvature. It might have done that not by looking at the tip but at the trunk.

Looking at it from another perspective, measuring the steadiness of the trunk by using points in a pixel distance of 5, as done for the SVM, might yield rather inaccurate results. It would be an interesting question to ask how we could measure steadiness along the upper half of the tail accurately. While we have discussed possible CNN features on the tail in Section 6.1, there might be some influence stemming from spurious correlations, which we discuss in Section 6.3.

6.3 "Clever Hans" Predictions

CNNs are incredibly powerful at finding any kinds of correlations in the input data even if they are not related to the object of interest. Lapuschkin et al. [46] have termed such spurious correlations "Clever Hans" predictions, because the model bases its prediction not on what we want it to focus on, but some unintended artifacts in the data. Figure 5.9 shows clearly that our CNN based a significant number of its negative responses mainly on motion in the top left corner. The motion stemmed from a substance called agarose, which the fish's head was embedded in to keep it steady. It is quite curious that, while not visible to human eyes, the agarose seems to be moving each time the fish performed a spontaneous swim bout, but not so for a prey bout.

Without further investigation we can only speculate about the origin of such a strong correlation between agarose motion and spontaneous movement. In general, videos were recorded without specific differences between spontaneous or prey bouts. In fact, fish randomly perform spontaneous bouts in between doing prey movements, since they are spontaneous. A specific distinction by fish or setup to such high accuracy is therefore very unlikely. Instead, we believe the most natural explanation to be the experimenters interfering with the setup during recording. After all, if the fish does not move frequently because its head is stuck in agarose, how could an experimenter still evoke movement? Quite likely the experimenter would tap the petri dish slightly, inducing a slight shift of agarose in the frames just before the bout. Another explanation could be that the fish was moving the agarose stronger when doing a spontaneous movement. We know from [10, 51] that they are doing more precise and cautious movements when homing in on prey, thus possibly moving the agarose slightly less. We find this explanation little convincing, because the agarose itself is quite a stiff material and the fish can probably not bring up enough force to deform it noticeably.

Despite evidence in the heatmaps of false negatives, we cannot say for certain that the CNN viewed motion of agarose in the top left corner as an unequivocal feature. The

false positives depicted in Figure 5.10 show that the CNN sometimes replied positive even though it perceived the characteristic agarose motion. This underlines that its high accuracy was not entirely derived from this "Clever Hans" type of feature. However, the figure also shows that the classifier could exploit this correlation further.

In fact, evidence in Table 5.2 suggests that the better trained model (ID 5) might be doing just that. Comparing precision and recall of models 4 and 5, the latter seems to be trading recall for precision. Considering that precision $= {}^{\mathrm{TP}}/_{(\mathrm{TP}+\mathrm{FP})}$ and recall $= {}^{\mathrm{TP}}/_{(\mathrm{TP}+\mathrm{FN})}$, model 5 has apparently learned to lean toward negative responses, thus decreasing FP. It therefore avoided situations where it responded positive even though it saw agarose motion, as Figure 5.10 shows for model 4. On the flip side, it commits more false negative responses. This in turn suggests that better learning of tail features could be possible, because the false negatives committed by model 4 originated in misinterpretation of tail motion, as Figure 5.10 shows. Also, observing that more training data and smaller learning rate exhibit more stable learning, we can expect improvement to still be possible. We would expect a learning rate of 1E-5 and weight decay of 1E-4 to be the most interesting next settings to check.

Chapter 7

Conclusion and Future Work

We trained a convolutional neural network (CNN) on 1,214 recordings of larval zebrafish to classify either prey or spontaneous swim bouts. We then visualized the learned weights by generating relevance heatmaps showing which regions of the input the network focuses on when performing its classifications. We applied a two-stream network made up of a spatial stream and a temporal stream as previously developed for human action recognition. The spatial stream received one randomly selected frame from the recording, while the temporal stream received the optical flow among a subset of frames. Since we initialized the network with pre-trained weights, data augmentation played a less crucial role than initially expected.

We find that our CNN is capable of learning highly discriminating tail features. The produced relevance heatmaps show that our CNN was strongly focusing on the upper half of the tail which is closer to the head and shakes only slightly during swim bouts. They further suggest that the CNN distinguished prey and spontaneous bouts by measuring the steadiness of the trunk during the recording. The steadier the trunk, the more confidently it could classify the movement as a prey bout. Vice versa, the wobblier the trunk, the more likely it faced a spontaneous movement. These features seem to be quite different from the ones used in the SVM classification. The SVM consulted features which were partially or fully derived from the tip of the tail, for example the maximum of the tail curvature over the bout.

Judging from the test accuracy, our CNN has learned better discriminating features than those used for the SVM, and has thus beaten manual feature engineering. The network reached a test accuracy of 96.69%, which is 6.49% points better than the baseline. However, this result comes with a big caveat. We have analyzed which input regions the CNN focuses on when performing its classification. The analysis shows

clearly that the network achieved a correct classification of a substantial number of spontaneous bouts solely based on a motion outside the tail. In particular, the model detects motion of the agarose which the fish's head was embedded in. We speculate that this artifact was introduced by experimenters to evoke spontaneous swim movements in the fish. Nevertheless, we highlight that there is a positive correlation between classification confidence and relevance on the fish's trunk. For this reason, we are quite certain that by digitally removing agarose from the recordings we could help the network focus on the tail and thus make more confident and potentially more accurate distinctions. We leave this to future work.

Contrary to prior beliefs, the spatial stream by itself was able to reach an accuracy of 82.16% showing that it learned competitive features on information from static appearance alone. Although the temporal stream was in fact the dominating one, the full network could improve its accuracy with support of the spatial stream by up to 1.55%. We find this surprising because we assumed the task of differentiating swim bouts to be solely driven by motion cues.

CNNs such as the one used in this dissertation could be used to investigate brain recovery in larval zebrafish. It has been shown on a cellular level that zebrafish can heal their brain within days after a lesion. However, this needs to be proven on a behavioral level [43]. Future work could perform a lesion study on the optic tectum in zebrafish [50, 61], a brain region responsible for translating visual input into motor output. CNNs could then assess swim bouts of recovered fish and give a measure for potential behavioral changes. Insights from relevance heatmaps would be required if the CNN could not distinguish recovered fish from healthy ones.

Another interesting project would be comparing a partially restrained experimental setup, where the fish is embedded in agarose, with the natural setup of freely moving fish. Fixated fish are easier to observe, for example for live brain imaging [38], and existing research is based on the assumption that they move the same as in a free setup. CNNs can be used to challenge this assumption by comparing swim bouts of each setup and even give hints as to what might be discriminating features.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate neural networks! aug 2018. URL `http://arxiv.org/abs/1808.04260`.

[3] Christopher Anders, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Understanding Patch-Based Learning by Explaining Predictions. jun 2018. URL `http://arxiv.org/abs/1806.06926`.

[4] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus Robert Müller, and Wojciech Samek. "What is relevant in a text document?": An interpretable machine learning approach. *PLoS ONE*, 12(8):e0181142, aug 2017. ISSN 19326203. doi: 10.1371/journal.pone.0181142. URL `https://dx.plos.org/10.1371/journal.pone.0181142`.

[5] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10

(7):e0130140, jul 2015. ISSN 19326203. doi: 10.1371/journal.pone.0130140. URL https://dx.plos.org/10.1371/journal.pone.0130140.

[6] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network Dissection: Quantifying Interpretability of Deep Visual Representations, 2017. URL http://openaccess.thecvf.com/content_cvpr_2017/html/Bau_Network_Dissection_Quantifying_CVPR_2017_paper.html.

[7] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. jul 2018. URL http://arxiv.org/abs/1807.03418.

[8] Isaac H. Bianco, Adam R. Kampff, and Florian Engert. Prey Capture Behavior Evoked by Simple Visual Stimuli in Larval Zebrafish. *Frontiers in Systems Neuroscience*, 5:101, dec 2011. ISSN 1662-5137. doi: 10.3389/fnsys.2011.00101. URL http://journal.frontiersin.org/article/10.3389/fnsys.2011.00101/abstract.

[9] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers. pages 63–71. Springer, Cham, 2016. doi: 10.1007/978-3-319-44781-0_8. URL http://link.springer.com/10.1007/978-3-319-44781-0_8.

[10] Melissa A Borla, Betsy Palecek, Seth Budick, and Donald M. O'Malley. Prey capture by larval zebrafish: Evidence for fine axial motor control. *Brain, Behavior and Evolution*, 60(4):207–229, 2002. ISSN 00068977. doi: 10.1159/000066699. URL www.karger.comwww.karger.com/bbe.

[11] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High Accuracy Optical Flow Estimation Based on a Theory for Warping. pages 25–36. Springer, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-540-24673-2_3. URL http://link.springer.com/10.1007/978-3-540-24673-2_3.

[13] João Carreira and Andrew Zisserman. Quo Vadis, action recognition? A new model and the kinetics dataset. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages

4724–4733, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.502. URL https://arxiv.org/pdf/1705.07750.pdf.

[14] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. 2014. URL http://www.robots.ox.ac.uk/http://arxiv.org/abs/1405.3531.

[15] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[16] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[17] Casper O. da Costa-Luis. tqdm: A Fast, Extensible Progress Meter for Python and CLI. *Journal of Open Source Software*, 4(37):1277, 2019. doi: 10.21105/joss.01277. URL https://hub.docker.com/.

[18] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, volume I, pages 886–893. IEEE, 2005. ISBN 0769523722. doi: 10.1109/CVPR.2005.177. URL http://ieeexplore.ieee.org/document/1467360/.

[19] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3952 LNCS, pages 428–441. Springer, Berlin, Heidelberg, 2006. ISBN 3540338349. doi: 10.1007/11744047_33. URL http://link.springer.com/10.1007/11744047_33.

[20] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow With Convolutional Networks, 2015. URL http://openaccess.thecvf.com/content_iccv_2015/html/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.html.

[21] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Visualizing Higher-Layer Features of a Deep Network Oracle Performance for Visual Captioning View project Semantic View project Visualizing Higher-Layer Features of a Deep Network Département d'Informatique et Recherche

Opérationnelle. (August 2014), 2009. URL https://www.researchgate.net/publication/265022827.

[22] Gunnar Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. pages 363–370. 2003. doi: 10.1007/3-540-45103-x_50. URL http://www.isy.liu.se/cvl/.

[23] Ethan Gahtan. Visual Prey Capture in Larval Zebrafish Is Controlled by Identified Reticulospinal Neurons Downstream of the Tectum. *Journal of Neuroscience*, 25 (40):9294–9303, 2005. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.2678-05.2005. URL http://rsb.info.nih.gov/ij/http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2678-05.2005.

[24] Melvyn A. Goodale and A.David Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, jan 1992. ISSN 0166-2236. doi: 10.1016/0166-2236(92)90344-8. URL https://www.sciencedirect.com/science/article/pii/0166223692903448.

[25] David (DARPA) Gunning. Explainable Artificial Intelligence (XAI), 2017. URL https://www.darpa.mil/attachments/XAIProgramUpdate.pdf.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. dec 2015. URL http://arxiv.org/abs/1512.03385.

[27] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. jul 2012. URL http://arxiv.org/abs/1207.0580.

[28] Kerstin Howe and Et Al. The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446):498–503, apr 2013. ISSN 00280836. doi: 10.1038/nature12111. URL http://www.nature.com/doifinder/10.1038/nature12111.

[29] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.55. URL http://ieeexplore.ieee.org/document/4160265/.

[30] Woong Y Hwang, Yanfang Fu, Deepak Reyon, Morgan L Maeder, Shengdar Q Tsai, Jeffry D Sander, Randall T Peterson, J-R Joanna Yeh, and J Keith Joung. Efficient genome editing in zebrafish using a CRISPR-Cas system. *Nature biotechnology*, 31(3):227–9, 2013. ISSN 1546-1696. doi: 10.1038/nbt.2501. URL http://www.nature.com/doifinder/10.1038/nbt.2501.

[31] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks, 2017. URL http://openaccess.thecvf.com/content_cvpr_2017/html/Ilg_FlowNet_2.html.

[32] ILSVRC. Large Scale Visual Recognition Challenge 2012, 2012. URL http://image-net.org/challenges/LSVRC/2012/results.html.

[33] Mihir Jain, Herve Jegou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2555–2562, 2013. doi: 10.1109/CVPR.2013.330. URL https://www.cv-foundation.org/openaccess/content_cvpr_2013/html/Jain_Better_Exploiting_Motion_2013_CVPR_paper.html.

[34] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.59.

[35] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei Fei Li. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.223. URL http://cs.stanford.edu/people/karpathy/deepvideo.

[36] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics Human Action Video Dataset. 2017. URL http://deepmind.http://arxiv.org/abs/1705.06950.

[37] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. sep 2016. URL http://arxiv.org/abs/1609.04836.

[38] Dal Hyung Kim, Jungsoo Kim, João C Marques, Abhinav Grama, David G.C. Hildebrand, Wenchao Gu, Jennifer M Li, and Drew N Robson. Pan-neuronal calcium imaging with cellular resolution in freely swimming zebrafish. *Nature Methods*, 14(11):1107–1114, nov 2017. ISSN 15487105. doi: 10.1038/nmeth.4429. URL http://www.nature.com/articles/nmeth.4429.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. dec 2014. URL http://arxiv.org/abs/1412.6980.

[40] N. Kishimoto, K. Shimizu, and K. Sawamoto. Neuronal regeneration in a zebrafish model of adult brain injury. *Disease Models & Mechanisms*, 5(2):200–209, 2011. ISSN 1754-8403. doi: 10.1242/dmm.007336. URL http://dmm.biologists.org/content/dmm/5/2/200.full.pdf.

[41] Caghan Kizil, Jan Kaslin, Volker Kroehne, and Michael Brand. Adult neurogenesis and brain regeneration in zebrafish. *Developmental Neurobiology*, 72(3):429–461, 2012. ISSN 19328451. doi: 10.1002/dneu.20918. URL https://onlinelibrary.wiley.com/doi/pdf/10.1002/dneu.20918.

[42] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference of Artificial Intelligence*, 1995. URL http://robotics.stanford.edu/ronnyk.

[43] John W. Krakauer, Asif A. Ghazanfar, Alex Gomez-Marin, Malcolm A. MacIver, and David Poeppel. Neuroscience Needs Behavior: Correcting a Reductionist Bias, feb 2017. ISSN 10974199. URL https://www.sciencedirect.com/science/article/pii/S0896627316310406.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[45] Hildegard Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. *HMDB: a large video database for human motion recognition*. 2011. ISBN 9781457711022.

[46] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10 (1):1096, dec 2019. ISSN 20411723. doi: 10.1038/s41467-019-08987-4. URL http://www.nature.com/articles/s41467-019-08987-4.

[47] Y Lecun and Y Bengio. Convolutional networks for images, speech, and time-series. Technical report, 1995. URL https://www.researchgate.net/publication/2453996.

[48] David Leslie. *Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector*. 2019. doi: 10.5281/zenodo.3240529. URL https://doi.org/10.5281/zenodo.3240529.

[49] Ilya Loshchilov and Frank Hutter. Fixing Weight Decay Regularization in Adam. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR'19)*, 2019. ISBN 1711.05101v2. doi: 10.1111/cbdd.12322. URL http://arxiv.org/abs/1711.05101.

[50] Angela L. McDowell, Lee J Dixon, Jennifer D Houchins, and Joseph Bilotta. Visual processing of the zebrafish optic tectum before and after optic nerve damage. *Visual Neuroscience*, 21(2):97–106, 2004. ISSN 09525238. doi: 10.1017/S0952523804043019. URL https://doi.org/10.1017/S0952523804043019.

[51] Melissa B. McElligott and Donald M. O'Malley. Prey tracking by larval zebrafish: Axial kinematics and visual control. *Brain, Behavior and Evolution*, 66(3):177–196, 2005. ISSN 00068977. doi: 10.1159/000087158. URL www.karger.com.

[52] Christoph Molnar. Interpretable Machine Learning, 2019. URL https://christophm.github.io/interpretable-ml-book/.

[53] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65:211–222, may 2017. ISSN 00313203. doi: 10.1016/j.patcog.2016.11.008. URL https://www.sciencedirect.com/science/article/pii/S0031320316303582.

[54] Grégoire Montavon, Wojciech Samek, and Klaus Robert Müller. Methods for interpreting and understanding deep neural networks, feb 2018. ISSN 10512004. URL https://www.sciencedirect.com/science/article/pii/S1051200417302385.

[55] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts. *Distill*, 1(10):e3, oct 2016. ISSN 2476-0757. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

[56] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2(11):e7, nov 2017. ISSN 2476-0757. doi: 10.23915/distill.00007. URL https://distill.pub/2017/feature-visualization.

[57] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The Building Blocks of Interpretability. *Distill*, 3(3):e10, mar 2018. ISSN 2476-0757. doi: 10.23915/distill.00010. URL https://distill.pub/2018/building-blocks.

[58] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[59] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v12/pedregosa11a.

[60] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?". In *Proceedings of the 22nd ACM SIGKDD International Confer-*

ence on Knowledge Discovery and Data Mining - KDD '16, pages 1135–1144, New York, New York, USA, 2016. ACM Press. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL http://dl.acm.org/citation.cfm?doid=2939672.2939778.

[61] Tobias Roeser and Herwig Baier. Visuomotor Behaviors in Larval Zebrafish after GFP-Guided Laser Ablation of the Optic Tectum. 2003. URL http://www.jneurosci.org/content/jneuro/23/9/3726.full.pdf.

[62] Julia L Semmelhack, Joseph C Donovan, Tod R Thiele, Enrico Kuehn, Eva Laurell, and Herwig Baier. A dedicated visual pathway for prey detection in larval zebrafish. 3:4878, 2014. doi: 10.7554/eLife.04878. URL https://cdn.elifesciences.org/articles/04878/elife-04878-v3.pdf.

[63] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, may 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2528162. URL http://ieeexplore.ieee.org/document/7404017/.

[64] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. sep 2015. URL http://arxiv.org/abs/1409.1556.

[65] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. 2014. URL http://arxiv.org/abs/1312.6034.

[66] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. 2012. URL http://arxiv.org/abs/1212.0402.

[67] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. sep 2014. URL http://arxiv.org/abs/1409.4842.

[68] Incinur Temizer, Joseph C. Donovan, Herwig Baier, and Julia L. Semmelhack. A Visual Pathway for Looming-Evoked Escape in Larval Zebrafish. *Current Biology*, 25(14):1823–1834, jul 2015. ISSN 09609822. doi: 10.1016/j.cub.2015.06.002. URL https://www.sciencedirect.com/science/article/pii/S0960982215006673.

[69] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2):22–30, mar 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37. URL http://ieeexplore.ieee.org/document/5725236/.

[70] Pieter Van Molle, Miguel De Strooper, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. Visualizing convolutional neural networks to improve decision support for skin lesion classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11038 LNCS, pages 115–123, 2018. ISBN 9783030026271. doi: 10.1007/978-3-030-02628-8_13. URL https://doi.org/10.1007/978-3-030-02628-8_13.

[71] Daniel Weinland, Edmond Boyer, and Remi Ronfard. Action Recognition from Arbitrary Views using 3D Exemplars. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007. ISBN 978-1-4244-1630-1. doi: 10.1109/ICCV.2007.4408849. URL http://ieeexplore.ieee.org/document/4408849/.

[72] Richard White, Kristin Rose, and Leonard Zon. Zebrafish cancer: The state of the art and the path forward, 2013. ISSN 1474175X. URL www.nature.com/reviews/cancer.

[73] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? pages 3320–3328, 2014. URL http://arxiv.org/abs/1411.1792.

[74] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L 1 Optical Flow. In *Pattern Recognition*, pages 214–223. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-74936-3_22. URL http://link.springer.com/10.1007/978-3-540-74936-3_22.

[75] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. 2014. URL https://cs.nyu.edu/fergus/papers/zeilerECCV2014.pdf.