ANALYSIS OF VIDEO FEATURE LEARNING IN TWO-STREAM CNNS ON THE EXAMPLE OF ZEBRAFISH SWIM BOUT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Semmelhack et al. (2014) have achieved high classification accuracy in distinguishing swim bouts of zebrafish using a Support Vector Machine (SVM). Convolutional Neural Networks (CNNs) have reached superior performance in various image recognition tasks over SVMs, but their learnt features are not immediately visible. Reaching better transparency helps to build trust in their classifications and makes learned features interpretable to experts. Using a recently developed technique called Deep Taylor Decomposition, we generated heatmaps to highlight input regions of high relevance for predictions. We find that our CNN makes predictions by analyzing the steadiness of the tail's trunk, which markedly differs from the manually extracted features used by Semmelhack et al. (2014). We further uncovered that the network paid attention to experimental artifacts. Removing these artifacts ensured the validity of predictions. After correction, our best CNN beats the SVM by 6.12%, achieving a classification accuracy of 96.32%. Our work thus demonstrates the utility of AI explainability for CNNs.

1 INTRODUCTION

In the study by Semmelhack et al. (2014), a well-performing classifier allowed to correlate neural interventions with behavioral changes. Support Vector Machines (SVMs) were commonly applied to such classification tasks, relying on feature engineering by domain experts. In recent years, Convolutional Neural Networks (CNNs) have proven to reach high accuracies in classification tasks on images and videos reducing the need for manual feature engineering. After Lecun & Bengio (1995) introduced them in the 90s, CNNs had their break-through in the competition ILSVRC2012 with the architecture of Krizhevsky et al. (2012). 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 (Hinton et al. (2012); Zeiler & Fergus (2014)).

While these more complex deep neural network architectures achieved better results, they also kept their learnt features hidden if not further analyzed. 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 (Molnar (2019)). Explaining the decisions made by CNNs might even become a legal requirement in certain applications (Alber et al. (2018)).

In order to overcome these drawbacks, subsequent research has developed approaches to shed light on the inner workings of CNNs. These approaches have been successfully used for uncovering how CNNs might learn unintended spurious correlations, termed "Clever Hans" predictions (Lapuschkin et al. (2019)). Such predictions could even become harmful if the predictions entailed decisions with severe consequences (Leslie (2019)). Also, since deep neural networks have become a popular machine learning technique in applied domains, spurious correlations would undermine scientific discoveries.

This paper 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 ex-

cellent model organism for vertebrates, including humans, due to the following four reasons: The genetic codes of humans and zebrafish are about 70% orthologue (Howe et al. (2013)). The fish are translucent which allows non-invasive observation of changes in the organism (Bianco et al. (2011)). 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 brain injury (Kishimoto et al. (2011); Kizil et al. (2012)).

In this paper, we adapt CNNs to work on highly controlled zebrafish video recordings and show the utility of a recently developed AI explainability technique on this task. We train the network on optical flow for binary classifying swim bouts and achieve superior performance when compared to the current state-of-the-art in bout classification (Semmelhack et al. (2014)). We then create heatmaps over the videos with the "iNNvestigate" toolbox (Alber et al. (2018)) which highlight the areas that our CNN pays attention to when making a prediction. The resulting heatmaps show that our CNN learns reasonable features which are very different from those manually composed by Semmelhack et al. (2014).

2 Related Work

In the following, we will give an overview over relevant CNN architectures and approaches. Then, we will summarize existing AI explainability approaches focusing on attribution techniques. Finally, we highlight important studies of behavioral zebrafish research and give details of the study by Semmelhack et al. (2014).

CNN architectures. Carreira & Zisserman (2017) identified five relevant types of video architectures: CNN + LSTM (Long-Short-Term Memory), 3D-CNN (Ji et al. (2013)), Two-Stream (Simonyan & Zisserman (2014)), 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. Optical flow can be described as the horizontal and vertical displacement of a pixel from one frame to the next (Farnebäck (2003)). Several algorithms for flow calculation exist, such as TV-L1 (Zach et al. (2007)), Brox (Brox et al. (2004)), and Farneback (Farnebäck (2003)). Even novel deep learning approaches have been developed, e.g. FlowNet (Dosovitskiy et al. (2015); Ilg et al. (2017)).

Carreira & Zisserman (2017) initialized several CNN architectures with pre-trained weights and trained them on human action recognition datasets to show the benefits of transfer learning in video classifiers. Previous studies had shown this for images (Shin et al. (2016)). Also, they found that adding a temporal stream based on optical flow always improved performance. This idea of a network with a spatial stream for appearance and a temporal stream for motion had been first developed by Simonyan & Zisserman (2014).

AI explainability techniques. Current AI explainability techniques on images can be largely categorized into two types: attribution and feature visualization (Olah et al. (2017)). Attribution relates regions of a sample image to activations of units in the CNN, while feature visualization uncovers what kinds of inputs strongly activate a particular unit.

One of the earlier and quite successful attribution approaches is called Sensitivity Analysis (Simonyan et al. (2014)). It is based on the idea that if a single pixel were marginally changed, the prediction would worsen significantly for an important pixel, but only slightly for less important ones. Furthermore, Zeiler & Fergus (2014) and Ribeiro et al. (2016) showed simple ways of producing approximate relevance heatmaps. By occluding parts of an image 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 the important objects of the image or performed its classification only based on contextual information.

The technique applied in this paper is called Deep Taylor Decomposition (Montavon et al. (2017)), which arose from Layer-Wise Relevance Propagation (Bach et al. (2015)). It has been put into use with text (Arras et al. (2017)), speech (Becker et al. (2018)), and only once with video data (Anders et al. (2018)). It highlights the areas in a sample image which the CNN deems most relevant for making a correct classification. We assume that the relevance of a pixel is determined by how much the classification accuracy would deteriorate if we altered this pixel. DTD distributes relevance from the output layer to the input layer by applying specific propagation rules for each layer. This

approach equals a first-order Taylor decomposition of the output function. The authors argue that it yields a better approximation of relevance than Sensitivity Analysis. Alber et al. (2018) included this technique in their "iNNvestigate" toolbox, which we used in our work.

Apart from attribution techniques, CNNs can be explained using direct feature visualization approaches. They try to find intuitive representations of the patterns a given unit responds to particularly strongly or weakly. Among these are deconvolution networks (Zeiler & Fergus (2014)), which are closely related to Sensitivity Analysis, and optimization approaches in the input domain (Erhan et al. (2009); Olah et al. (2018)). Also, instead of creating an abstract representation of features, one can select specific samples which highly activate or suppress a particular unit. Bau et al. (2017) even went a step further by hiding irrelevant parts of the input image, similar to LIME (Ribeiro et al. (2016)).

Behavioral zebrafish research. As we argue in Section 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 (Kishimoto et al. (2011); Kizil et al. (2012)), visual processing in the brain (Roeser & Baier (2003); Gahtan (2005); Semmelhack et al. (2014); Temizer et al. (2015)), cancer research (White et al. (2013)), and genetic modifications (Hwang et al. (2013)). Especially in neuroscientific research, understanding behavior and behavioral changes in response to cerebral interventions is of high importance (Krakauer et al. (2017)).

Previous studies therefore closely investigated the motion patterns of zebrafish (Borla et al. (2002); McElligott & O'Malley (2005)). Borla et al. (2002) 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. Based on their observations, they concluded that zebrafish must possess fine axial motor control in all parts of their tail.

Given that prey movements can be clearly distinguished from other types of movements, Semmelhack et al. (2014) hypothesized that there must be a dedicated circuitry in the zebrafish brain. They found a 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 findings by ablating the AF7 neuropil and observing that lesioned fish failed to respond to prey stimuli with a movement that a trained SVM would classify as prey. They have identified the following five features as the most discriminating 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)

In our work, we make use of the dataset they gathered for training their SVM and compare our CNN to the features above.

3 METHODS

We trained a Two-Stream CNN to distinguish prey and spontaneous swim bouts of larval zebrafish with short video samples. For data preparation, we first extracted standardized snippets from raw videos and then performed augmentation by subsampling, flipping, and cropping. After training, we computed heatmaps showing the regions of high relevance within frames[1].

Data pre-processing. We used the raw video files recorded by Semmelhack et al. (2014) with a high-speed camera at 300 frames per second and labeled as either spontaneous (0/negative, 56.1%) or prey (1/positive, 43.9%) bout. The heads of the fish were embedded in a substance called agarose to keep them steady. We turned the videos into grayscale, normalized them, and kept a crop of size 256x256 pixels from each frame, such that the right-most part of the bladder was central on the left,

[1]The code can be found at *omitted for anonymity*.

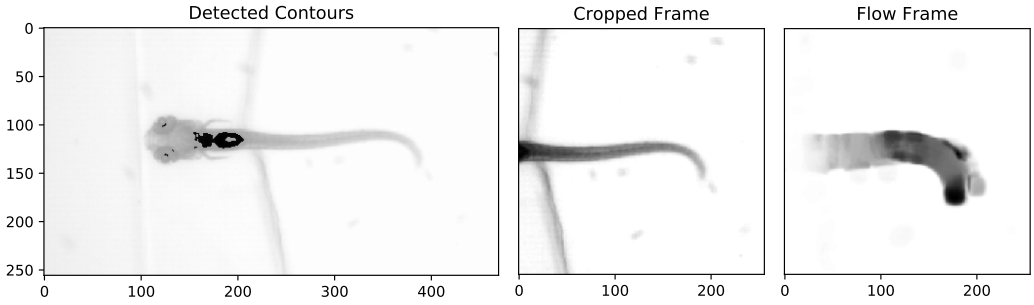

Figure 1: Detected contours (black) in the original frame, resulting cropped frame after normalization, and corresponding optical flow frame.

as shown in Figure 1. We did not include the head of the fish in the crop, because the eyes would give away information about the type of bout (Bianco et al. (2011)). More details in Appendix B.

For correct centering we implemented a gamma correction with $\gamma = \exp\left(-\text{skewness}/\text{param}\right)$, (where param was tweaked to maximize detection performance (here param $= 4.3$)) and applied a binary threshold at value 3 to separate the bladder from the rest of the fish. Since the eyes might also fall under this threshold, we declared the right-most contour the bladder. Each raw video contained several bout events which we extracted as sequences of 150 frames resulting in 1,214 video snippets. Each sequence started 15 frames before the actual bout motion was detected.

Extension to the pre-processing procedure. After training and heatmap analysis, we found that the trained CNN had found a shortcut for classification by using agarose motion instead of tail features. We therefore extended our pre-processing procedure by setting squares of 85x85 pixels in the upper and lower left-hand corners of all frames plain white. While this cut away some of the tail tip in rare occasions, it ensured that bouts could not be classified based on the agarose motion anymore.

Data augmentation. Our data augmentation procedure worked with batches of 32 videos with the original data randomly shuffled, allowing a decent approximation of the gradient during training with some inherent noise, which is generally desirable to avoid falling into sharp minima (Keskar et al. (2016)). The procedure performed three augmentation steps: first subsampling, then flipping, and cropping, which achieved augmentation factors of 8, 2, and 9 respectively, totaling 174,816 augmented samples. We decided to compute optical flow after subsampling, not before, in order to create differing augmented samples. All samples of an augmented batch were derived from different original videos to ensure a good gradient approximation during training.

In the subsampling step, our algorithm randomly selected 86 out of 150 frames under the constraint that between two frames, no more than 2 frames may be omitted. This was to ensure meaningful flow calculation in the subsequent step, because the tail could move quite fast. After subsampling, for each video our procedure selected one of the 86 frames to be the input for the spatial network. Then we computed the optical flow resulting in 85 flow frames with one x and y component each. They were stacked to 170 channels, alternating x and y. We used the optical flow algorithm by Farnebäck (2003) with parameters[2] detecting flow even when the tail moved fast. The procedure then generated 18 augmented batches from each subsampled batch by randomly flipping vertically and cropping into crops of size 224x224 with the upper left corner at coordinates (8,8), (8,16), (8,24), (16,8), (16,16), (16,24), (24,8), (24,16), and (24,24).

Since flow was calculated as floats, we minimized storage space significantly by rescaling each frame to range 0–255, compressing them with lossy JPEG-compression at level 40, and turning

[2]pyramid scale = 0.8, pyramid levels = 10, window size = 10, iterations per pyramid level = 10, pixel neighborhood for polynomial expansion = 13, std dev of the Gaussian for polynomial expansion = 1.8.

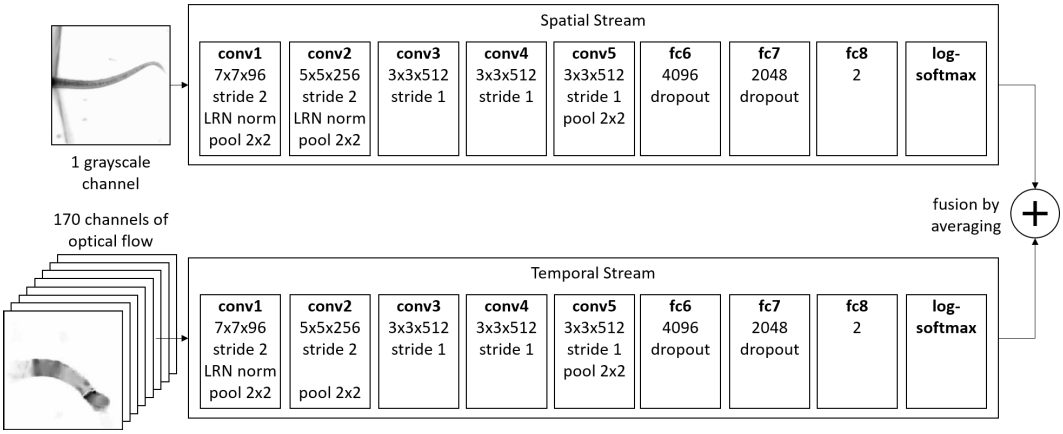

Figure 2: Full Two-Stream CNN architecture.

them into unsigned integers, similar to what Simonyan & Zisserman (2014) had done. Frames were rescaled later for training.

CNN architecture and framework. Just like Simonyan & Zisserman (2014), we used a two-stream network with an adapted CNN-M-2048 network (Chatfield et al. (2014)) for each stream, as depicted in Figure 2. As shown by Carreira & Zisserman (2017), this network can deal with a small sample size and learns quickly. The spatial stream had one gray-scale channel and the temporal stream 170 flow channels in the first layer. After obtaining the predicted probabilities of each stream by calculating the log-softmax of the individual two outputs, they were fused by averaging. We computed the negative log-likelihood loss of this average, which can be interpreted as the joint log-probability of both streams, assuming their statistical independence.

Our dataset was made up of 38 files, with 28, 4, and 6 files for training, validation, and test sets respectively. The individual sets were not shuffled during training in order to allow sequential reads, which might have decreased training time. Notwithstanding, batches consisted of random samples due to our augmentation procedure explained above.

Initialization of weights. We initialized both streams with weights pre-trained on ImageNet[3]. This has become a common initialization strategy in various classification tasks (Shin et al. (2016); Lee et al. (2016); Van Horn & Perona (2017)), due to the utility of general features learnt from natural images, such as edge detection. While Simonyan & Zisserman (2014) did not pre-train flow, Carreira & Zisserman (2017) found a pre-trained temporal stream to reach superior performance. With this initialization we hoped that training would require only fine-tuning and therefore less data and fewer epochs.

Specifically, for the weights of the input layer of the spatial stream we took the average of the pre-trained weights of 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 evolved differently during training and should aid learning. Regarding outputs, we averaged the pre-trained weights of 500 units on the output layer to obtain the weights of two output neurons, because we dealt with 2 classes instead of 1,000 as in ImageNet.

Training procedure. 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 (Loshchilov & Hutter (2019)). Furthermore, we added a gamma learning rate scheduler which updated the learning rate by a multiplicative factor of $\gamma = {}^{1}/\sqrt{epoch}$ (Kingma & Ba (2014)) every epoch. Our training framework computed accuracy on the validation set after each epoch to estimate generalization performance.

[3]http://www.vlfeat.org/matconvnet/models/imagenet-vgg-m-2048.mat

Table 1: Results on training, validation, and test (split by stream) sets for baseline (B), analyzed CNN (0), best CNN before (1) and after (2) correction.

ID	LR	WD	TRAIN	VALID	SPATIAL	TEMPORAL	FULL TEST
B	–	–	.946	.946	–	–	.902
0	1e-4	1e-3	.9962	.9489	.8216	.9441	.9596
1	1e-5	1e-3	.9869	.9372	.8138	.9660	.9669
2	1e-5	1e-3	.9997	.9597	.8213	.9581	.9632

Since we were initializing our CNN with weights learned from quite a different domain, fine-tuning was crucial. In particular, we performed a hyperparameter search over learning rate ({1e-3, 1e-4, 1e-5}) and weight decay ({1e-2, 1e-3, 1e-4}) using a smaller dataset and training for about 8 epochs. With the best hyperparameters we trained the CNN on the full dataset for 5 epochs.

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 (Olah et al. (2018); Bau et al. (2017); Erhan et al. (2009)) and deconvolutions (Zeiler & Fergus (2014)) 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 applied successfully before (Lapuschkin et al. (2019); Van Molle et al. (2018); Anders et al. (2018)) and was conveniently accessible in the "iNNvestigate" toolbox (Alber et al. (2018)). For validation purposes, we additionally generated saliency maps (Simonyan et al. (2014)) and heatmaps from Guided BackProp (Springenberg et al. (2015)). 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 was paying attention to.

Also, we simplified the analysis by splitting the network into its individual streams. This was possible because no weights were learned after the final layer of each stream. Once the 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. We used a small dataset of 3,420 samples for analysis by setting the subsampling factor from 8 to 1, in order to simplify the process.

4 RESULTS

We fine-tuned the training of our CNN to reach high accuracies in distinguishing prey bouts of larval zebrafish from spontaneous swims. Subsequently, we analyzed the learned weights by computing and averaging the relevance heatmaps of all samples grouped by class. We further define prey bouts as positive and spontaneous swims as negative.

CNN test results. We performed a small hyperparameter search over learning rate and weight decay, which proved sufficient because all models were initialized with pre-trained weights from ImageNet. For our baseline SVM – detailed in Appendix A – we report the 5-fold cross-validated accuracy and the final accuracy on the held-out test set. The hyperparameters[4] agree with the ones found by Semmelhack et al. (2014). We further present the accuracies of the CNN used for heatmap analysis, as well as the best CNNs before and after removal of experimental artifacts on training, validation, and test sets in Table 1. For the test set we additionally report individual accuracies of the spatial and temporal streams. We highlight that the final CNN attains a test accuracy of 96.32%, which is 6.12% points better than the baseline.

Relevance heatmaps. We used relevance heatmaps to visualize the regions the CNN pays most attention to when classifying a specific sample. We computed relevance averages across samples and frames, as well as split by class in Figure 3 for more comprehensive insights into the features learned by our CNN. Similar results from other explainability techniques can be found in Appendix C. As expected, the heatmaps exhibit the checkerboard artifacts typical of kernels with stride 2 in the first convolutional layer (Odena et al. (2016); Montavon et al. (2017)).

[4]RBF-kernel with $\gamma = 0.001$ and C = 1

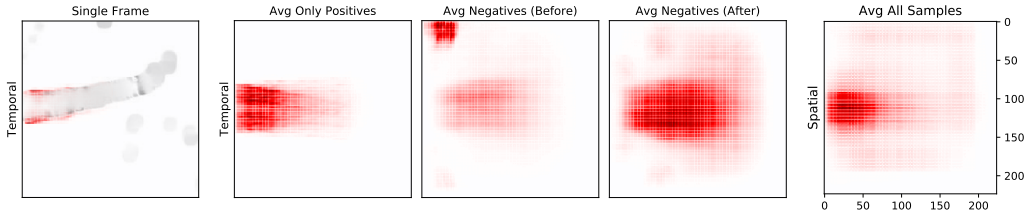

Figure 3: Relevance heatmaps of the CNN. High relevance is shown in dark red color, while light red stands for low relevance. Left: example of a temporal heatmap over a single frame (true positive, optical flow in grayscale, relevance heatmap overlaid in red). Middle: averages of temporal heatmaps split by class; negatives before and after correction. Right: average of spatial heatmaps over all samples.

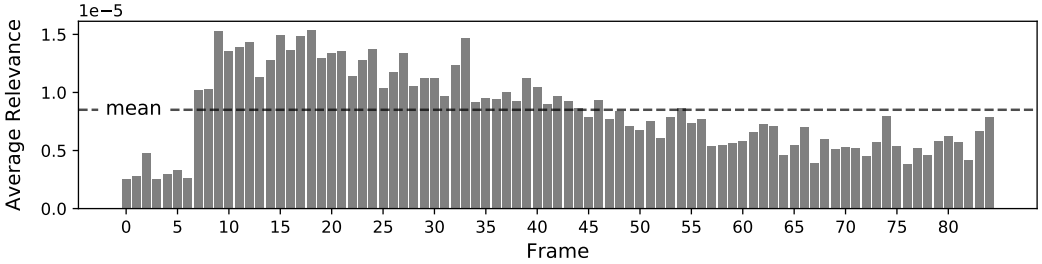

Figure 4: Distribution of relevance on the flow frames of the averaged sample.

Steadiness of the fish's trunk as differentiating feature. First and foremost, the heatmaps show that our CNN is able to differentiate the movements of zebrafish based on their characteristic motion. Relevance is highly concentrated at the trunk, i.e. the rostral part of the tail, for both temporal and spatial stream. We observe a very sharp relevance pattern along the edges of the tail. This indicates that the pre-trained weights helped the network to look for edges. The CNN pays little to no attention to the end of the tail even though it is visible in most frames. Instead, it makes positive classifications by looking at the very start of the trunk. The heatmaps suggest that a calm and steady trunk indicates a prey bout. As for the negatives, the spread out relevance pattern reflects the high frequency of tail deflections typical of spontaneous bouts, which had been identified by Semmelhack et al. (2014) before. This makes clear that the network is able to differentiate the movements of zebrafish to high accuracy based on their characteristic motion.

"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. (2019) 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 3 shows clearly that our CNN bases a significant number of its negative responses mainly on motion in the top left corner and focuses little on tail appearance and motion. This plays a role only in negative classifications and only in the temporal stream. While the heatmaps are vertically symmetric, as we would expect due to vertical flipping during augmentation, this is not true for the peculiar region in the top left corner. Figure 3 depicts the averaged heatmap after removing the artifacts in the top and bottom left hand corners and retraining our CNN. Relevance is now entirely focused on the tail.

Relevance distribution across frames. Most relevance is concentrated on the frames in the range 7–46, as depicted in Figure 4. The first seven frames are of least importance. This is very likely because our pre-processing procedure added a buffer of 15 frames before each bout, suggesting that the network focuses on the range of frames which are in fact the most relevant ones. This further supports the hypothesis that our CNN is able to differentiate zebrafish movements based on their characteristic motion patterns.

5 CONCLUSION AND FUTURE WORK

We trained a two-stream Convolutional Neural Network (CNN) on recordings of larval zebrafish to classify prey and spontaneous swim bouts. We then visualized the learned weights by generating relevance heatmaps showing which regions of the input the network focuses on while performing its classifications. We find that our CNN is capable of learning highly discriminating tail features. These features seem to be quite different from the ones used in the SVM classification by Semmelhack et al. (2014) - the previous state-of-the-art in bout classification. The heatmaps further uncovered a "Clever Hans" type of correlation. After removing this spurious correlation and retraining the network, the network reached a test accuracy of 96.32%, which is 6.12% points better than the accuracy achieved by Semmelhack et al. (2014). Judging from the test accuracy, our CNN has learned better discriminating features than those used for the SVM by Semmelhack et al. (2014), and has thus beaten manual feature engineering in this application domain.

Steadiness of the fish's trunk as differentiating feature. The relevance heatmaps and high accuracy show that the network achieves correct classifications by looking for salient features in the trunk of the tail while largely disregarding the tip. A sharp and clear relevance profile confined to the edges of the trunk gives a clear sign of a prey bout. The opposite speaks for a spontaneous bout. 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 our interpretation of learned features to be in line with existing research on the kinematics of prey bouts. As shown by Borla et al. (2002) and McElligott & O'Malley (2005), 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 their prey. Such precise movements are not required in spontaneous swim bouts. The heatmaps indicate that the network has found clear evidence for these kinds of motion in the trunk of the tail.

Furthermore, we argue that the CNN has learned features which are very different from the ones identified by Semmelhack et al. (2014). All of their features – as outlined in Section 2 –, except the second one, rely on information from the tip of the tail and a complete sequence of frames. However, many optical flow frames do not depict the tip of the tail because of its small size and high speed. This might have happened due to suboptimal parameter settings which could not handle the sometimes long distances which the tip traveled between frames. Also, subsamples include only 85 of the original 150 frames for each video. Due to its higher performance, we conclude not only that the CNN has learned a different set of features, but also that these features must bear higher discriminative power.

Origin of the "Clever Hans" correlation. The telltale motion in the top left corner stems 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. We speculate that this correlation was unintentionally introduced by the experimenters who might have tapped the petri dish to induce the fish to perform a spontaneous swim bout.

Future work. Calculating and storing optical flow is expensive. If we attained similar performance on original frames, training would be considerably cheaper. While we can confirm the findings by Simonyan & Zisserman (2014) that the spatial stream by itself reaches a fairly competitive accuracy, it provides only very minor improvement to the overall network. Yet, this stream is probably looking for very similar features as the temporal stream, because it focuses largely on the upper half of the tail, just like the temporal stream. 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 of the temporal stream.

Furthermore, CNNs such as the one used in this paper 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 (Krakauer et al. (2017)). Future work could perform a lesion study on the optic tectum in zebrafish (McDowell et al. (2004); Roeser & Baier (2003)), 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 were not able not distinguish recovered fish from healthy ones.

REFERENCES

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, and Others. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

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`.

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`.

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. URL `https://dx.plos.org/10.1371/journal.pone.0181142`.

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. URL `https://dx.plos.org/10.1371/journal.pone.0130140`.

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`.

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`.

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. URL `http://journal.frontiersin.org/article/10.3389/fnsys.2011.00101/abstract`.

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. URL `www.karger.comwww.karger.com/bbe`.

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

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

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, pp. 4724–4733, 2017. URL `https://arxiv.org/pdf/1705.07750.pdf`.

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`.

François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

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

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/.

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.

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.

Gunnar Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. pp. 363–370. 2003. URL http://www.isy.liu.se/cvl/.

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. URL http://rsb.info.nih.gov/ij/http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.2678-05.2005.

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.

Kerstin Howe, Matthew D Clark, Carlos F Torroja, James Torrance, Camille Berthelot, Matthieu Muffato, John E Collins, Sean Humphray, Karen McLaren, Lucy Matthews, and And Others. The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446):498–503, apr 2013. URL http://www.nature.com/doifinder/10.1038/nature12111.

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/.

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. URL http://www.nature.com/doifinder/10.1038/nbt.2501.

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.

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.

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.

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

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. URL http://dmm.biologists.org/content/dmm/5/2/200.full.pdf.

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

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.

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

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

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. URL http://www.nature.com/articles/s41467-019-08987-4.

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

Sue Han Lee, Yang Loong Chang, Chee Seng Chan, and Paolo Remagnino. Plant identification system based on a convolutional neural network for the lifeclef 2016 plant classification task. In *CEUR Workshop Proceedings*, volume 1609, pp. 502–510, 2016.

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

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

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. URL https://doi.org/10.1017/S0952523804043019.

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. URL www.karger.com.

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

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. URL https://www.sciencedirect.com/science/article/pii/S0031320316303582.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts. *Distill*, 1(10):e3, oct 2016. URL http://distill.pub/2016/deconv-checkerboard.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2(11):e7, nov 2017. URL https://distill.pub/2017/feature-visualization.

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. URL https://distill.pub/2018/building-blocks.

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.

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. URL http://www.jmlr.org/papers/v12/pedregosa11a.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?". In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 1135–1144, New York, New York, USA, 2016. ACM Press. URL http://dl.acm.org/citation.cfm?doid=2939672.2939778.

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.

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. URL https://cdn.elifesciences.org/articles/04878/elife-04878-v3.pdf.

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. URL http://ieeexplore.ieee.org/document/7404017/.

Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos, 2014. URL http://papers.nips.cc/paper/5353-two-stream-convolutional.

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.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. 2015. URL http://arxiv.org/abs/1412.6806.

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. URL https://www.sciencedirect.com/science/article/pii/S0960982215006673.

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/.

Grant Van Horn and Pietro Perona. The Devil is in the Tails: Fine-grained Classification in the Wild. 2017. URL www.inaturalist.orghttp://arxiv.org/abs/1709.01450.

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, pp. 115–123, 2018. URL https://doi.org/10.1007/978-3-030-02628-8_13.

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

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

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

A FITTING THE BASELINE SVM

For each frame in the 1,214 videos, we applied the tail-fitting code developed by Semmelhack et al. (2014) to compute points along the tail, as depicted in Figure S5. We initialized their procedure central and 8 pixels from the left edge, because after pre-processing we could assume this to be just next to the right end of the bladder. Some of the videos contained frames which the tail-fitting code had problems processing, possibly because the pre-processing procedure cut off the tip of the tail in these instances. This resulted in 953 correctly processed videos, including 482 (50.6%) spontaneous and 471 (49.4%) prey bouts. We performed no augmentation here because this would not have benefited the SVM.

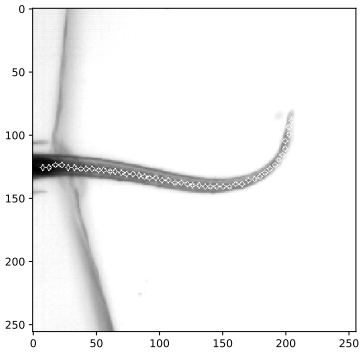

Figure S5: Distribution of points along the tail after running the tail-fitting procedure.

The feature extraction and model fitting algorithm then split the set into 85% training and 15% held-out test sets. Semmelhack et al. (2014) have identified 5 key features which allowed their SVM to achieve a cross-validation accuracy of 96%. They did not report results on a held-out test set. We used their provided code to extract these features. Then we performed a grid-search to tune SVM-kernel, γ, and C[5]. Just like Semmelhack et al. (2014) we used stratified 5-fold cross-validation (Kohavi (1995)).

B PROJECT PIPELINE

Figure S6 gives an overview over the whole project pipeline. The figure includes a depiction of the raw data input, pre-processing, augmentation, CNN and SVM training, and heatmap analysis. Figure S7 summarizes the data augmentation procedure. All scripts worked with a seed of 462019. We used the openly available distributions of NumPy 1.16.4 (van der Walt et al. (2011)), Matplotlib 3.1.1 (Hunter (2007)), tqdm 4.32.2 (da Costa-Luis (2019)), OpenCV 4.1.0.25 (Bradski (2000)), scikit-learn 0.21.2 (Pedregosa et al. (2011)), PyTorch 1.1.0 (Paszke et al. (2017)), h5py 2.9.0 (Collette (2013)), TensorFlow 1.14.0 (Abadi et al. (2015)), Keras 2.2.4 (Chollet et al. (2015)), and iNNvestigate 1.0.8 (Alber et al. (2018)).

Pre-processing. We aimed to center the fish's bladder on the left of each cropped frame. To achieve this, we applied a binary threshold at value 3, because after normalization and gamma correction the pixels of the bladder were separated mostly below this value. While the fish was quite light, the eyes as the second darkest part of the fish might still have fallen under this threshold. Therefore, we first

[5]RBF-kernel, $\gamma \in \{$1e-1, 1e-2, 1e-3, 1e-4$\}$, C $\in \{0.01, 0.1, 1, 10\}$; linear-kernel, C $\in \{0.01, 0.1, 1, 10\}$.

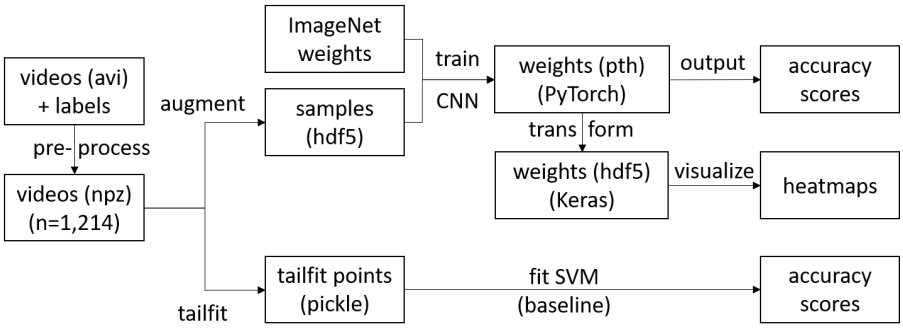

Figure S6: Overview over the whole project pipeline.

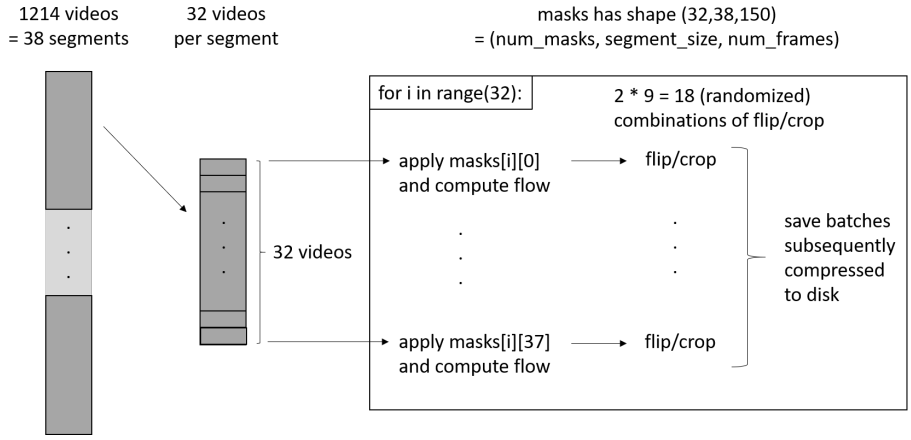

Figure S7: Data augmentation procedure.

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 using the right-most pixel of that contour.

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 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 did not take into account 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 it 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 was.

Data augmentation. We parallelized data augmentation with 38 workers, because optical flow calculation took roughly 14 seconds per subsample. We used a Dell PowerEdge R815 with four 16 core Opteron CPUs, 256 GB memory, and 4 TB of disk space (HDD). Furthermore, the resulting hdf5-files were compressed with gzip-compression at maximum compression level. This outperformed lzf-copmression by a factor of 1.76 when comparing training times. This advantage can be ascribed

to heavy parallelization of decompression and relatively low transfer speeds between hard drive and memory.

Training procedure. We implemented PyTorch's `Dataset` module to make use of the multi-processing capabilities of the `DataLoader` class on servers with 16–32 CPU cores and 4 GPUs (NVIDIA GTX1060 6 GB) each. This was necessary to achieve manageable epoch times.

Relevance analysis with heatmaps. The toolbox "iNNvestigate" (Montavon et al. (2017)) for analyzing the trained weights only supported Keras with the TensorFlow-backend. Therefore, we re-implemented the exact structure of our CNN and initialized it with the extracted weights from PyTorch. While the conversion could have been done with tools like ONNX, after a few unsuccessful attempts we transported the weights with a custom Python script.

A caveat to the "iNNvestigate" toolbox emerged after heatmap generation: it had problems analyzing 1,578 of the 3,420 samples. These produced an empty output. We made sure the problematic samples did not follow any systematic 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 distribution of classification confidence after the analysis. Since all numbers were the same as before the analysis, we continued with 1,842 samples for further investigation.

C Other Explainability Techniques

We generated saliency maps (Simonyan et al. (2014)) and Guided BackProp heatmaps (Springenberg et al. (2015)) analogous to the relevance heatmaps in Figures S8 and S9 for comparison with the more recent technique of DTD. It becomes apparent that these other two techniques allow similar insights, although slightly fuzzier. Importantly, they also uncover the "Clever Hans" prediction.

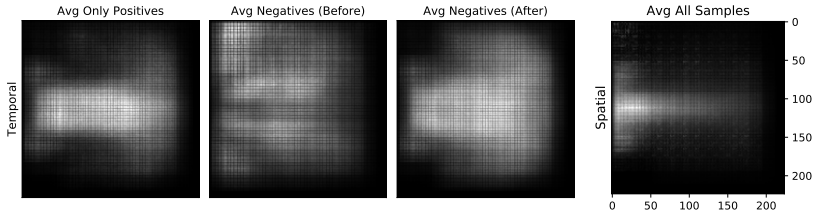

Figure S8: Averaged saliency maps, analogous to Figure 3.

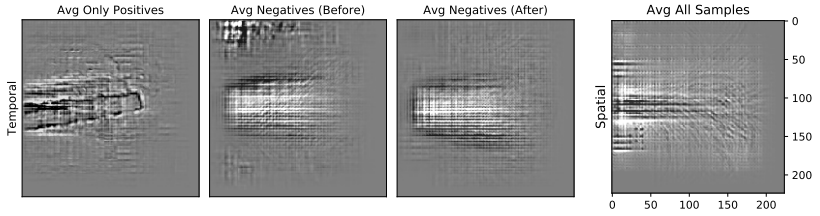

Figure S9: Averaged heatmaps from Guided BackProp, analogous to Figure 3.

D Selection of Additional Heatmaps

Here, we depict the ten most informative consecutive flow frames of the single most confident true positive (Figure S10), true negative (Figure S11), false positive (Figure S12), and false negative (Figure S13) sample. Figure S15 summarizes the spatial heatmaps of the same samples. Moreover, we gather five particularly informative flow frames in Figure S14 and four spatial heatmaps in Figure S15.

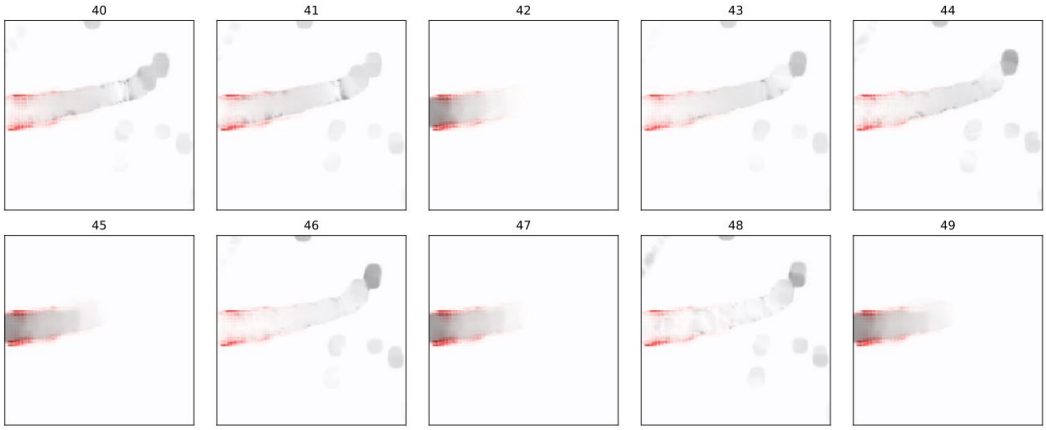

Figure S10: Selection of flow frames of the most confident true positive sample.

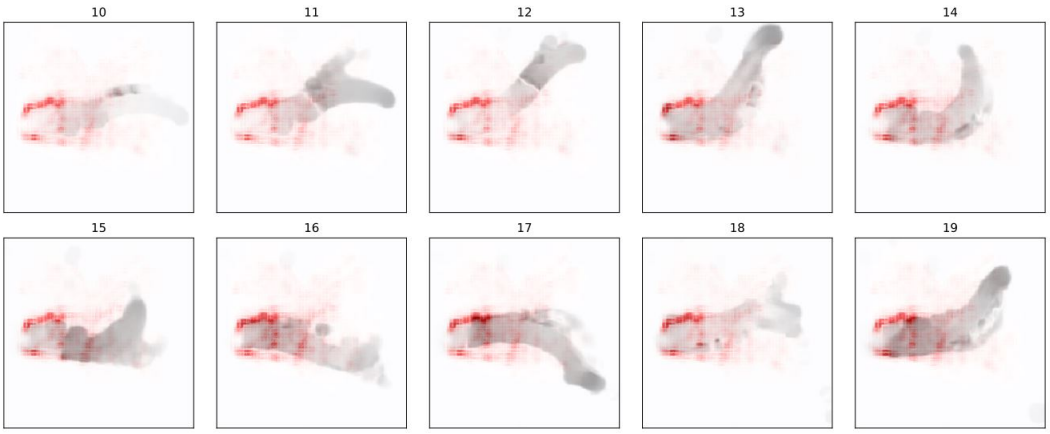

Figure S11: Selection of flow frames of the most confident true negative sample.

E Confidence Analysis on Heatmaps

We performed a confidence analysis on the heatmaps which depicted the "Clever Hans" feature of agarose motion in the top left corner. We sorted all negative classifications by increasing confidence calculated as $-\log\left(\log(\mathrm{P}(1))/\log(\mathrm{P}(0))\right)$ for each sample. We then grouped and averaged the heatmaps over windows of 104 samples, as shown in Figure S16. The analysis uncovered that the more confident a negative classification, the more the CNN relied on tail features. This in turn indicated that the CNN was able to learn actual features on the tail and did not entirely rely on agarose motion. Also, it suggested that tail features were a better predictor than agarose motion.

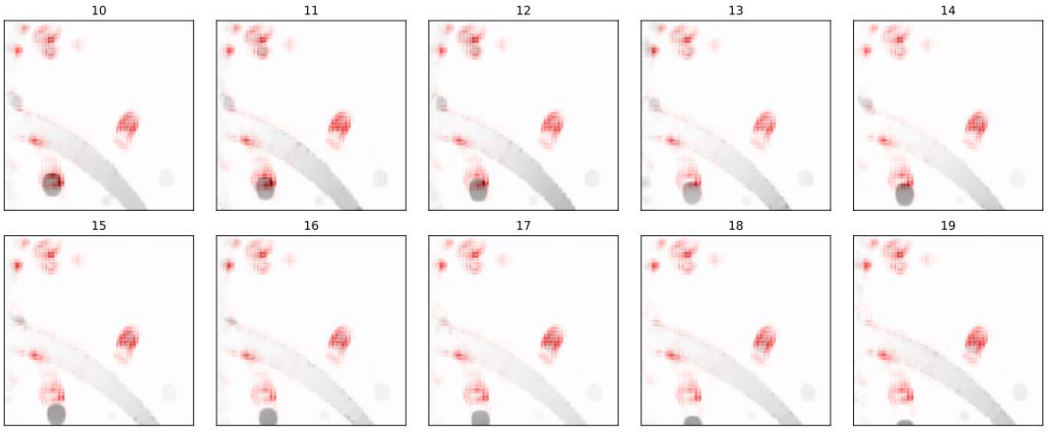

Figure S12: Selection of flow frames of the most confident false positive sample.

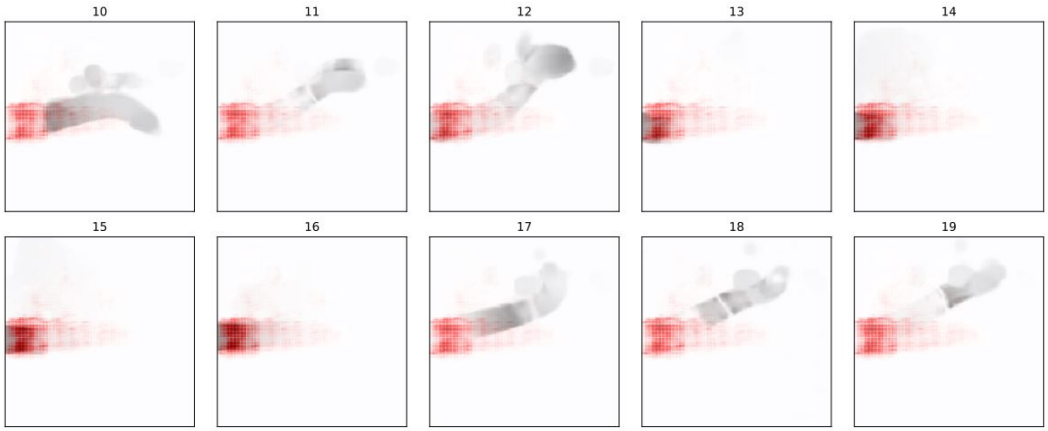

Figure S13: Selection of flow frames of the most confident false negative sample.

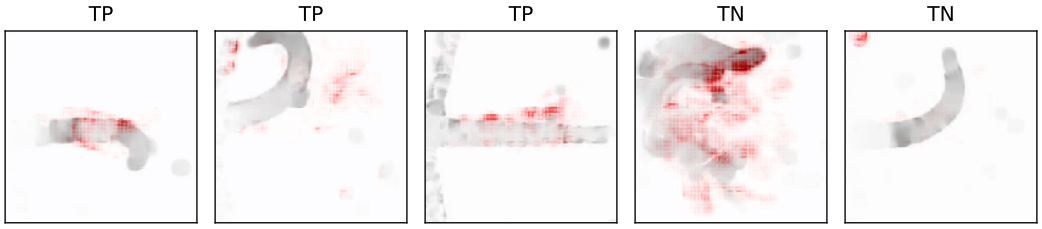

Figure S14: Selection of flow frames of a selection of true positive (TP) and true negative samples (TN).

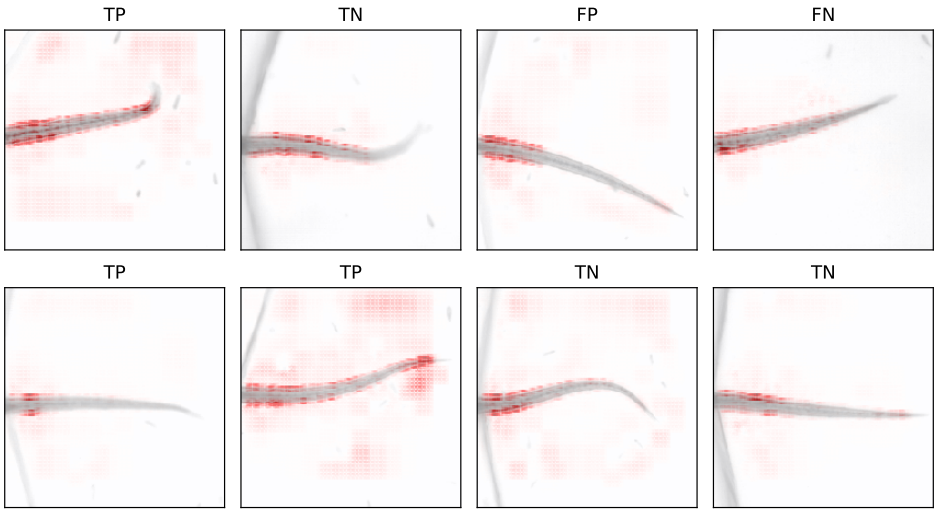

Figure S15: Upper row: spatial input of the most confident samples of each category. Lower row: spatial input of a selection of samples.

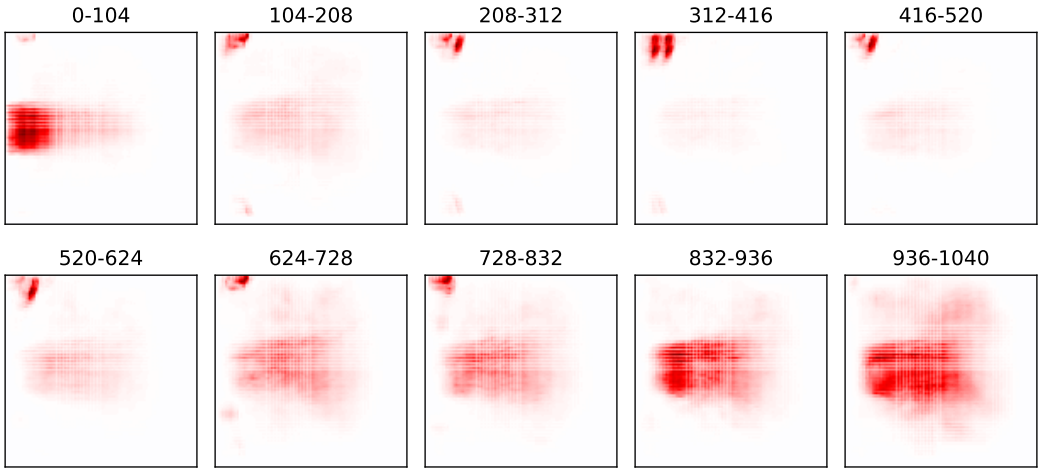

Figure S16: Averaged heatmaps of 104 samples per subfigure, sorted by increasing confidence for responding negative.