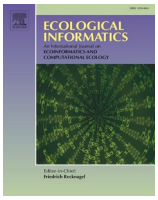
Ecological Informatics 61 (2021) 101215

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf



Identification of animals and recognition of their actions in wildlife videos using deep learning techniques

Frank Schindler *, Volker Steinhage

Department of Computer Science IV, University of Bonn, Endenicher Allee 19A, D-53115 Bonn, Germany

A B S T R A C T

Biodiversity crisis has continued to accelerate. Studying animal distribution, movement and behaviour is of critical importance to address environmental challenges such as spreading of diseases, invasive species, climate and land-use change. Camera traps are an appropriate technique for continuous animal monitoring in an automated 24/7/52 documentation. This study shows a proof-of-concept for an end-to-end pipeline to detect and classify animals and their behaviour in video clips. Video clips are captured with 8 frames per second by camera traps using infrared cameras and infrared flash-lights. The clips show deer, boars, foxes and hares - mostly at night time. Our approach shows an average precision of 63.8% for animal detection and identification. For action recognition the achieved accuracies range between 88.4% and 94.1%.

1. Introduction

Already 25 years ago, the large-scale destruction of habitats as well as biodiversity losses were alarming researchers and politicians. The Convention on Biological Diversity, (United Nations, 1992) is one of several responses. Biodiversity protection objectives were initially set for 2010 by the European Parliament, for example, and then extended to 2020.[1] The biodiversity crisis has continued to accelerate and the related political agreements have not yet achieved a reversal of the trend.[2] Therefore, the automatization of biodiversity monitoring is still and will be in great demand. Camera-traps show several advantages for animal monitoring: (1) Using invisible infrared flash-lights, camera traps work non-invasive by having no effect on animal behaviour. (2) Camera-traps work efficiently for several weeks. (3) Camera traps deliver permanent documentation records of date, location, and species. (4) Camera traps can record animal behaviour. (5) Images and video clips can be used for education and promotion. In Germany, some highways passing through forests, are bridged to give animals the opportunity to cross the highways safely. Some of these so-called green-bridges are equipped with camera traps to monitor animal abundance. In this study we use night shot video clips of mammals that were recorded by six camera traps using infrared cameras and an infrared flash-lights near a highway bridge in Bavaria between the end of 2013 and spring 2016. The data material was provided by the Bavarian Highway Directorate.[3] The video clips show mainly four kinds of animals: deer, boars, foxes and hares.

1.1. Objectives

This contribution shows a proof-of-concept study aiming at the coherent detection of animals and their actions in video clips. Conceptualization and implementation involves the evaluation of different architectural alternatives. Detected animals are depicted via bounding boxes and precise contour shapes attributed with the associated type of action.

Fig. 1 depicts the workflow of the resulting pipeline for animal detection and action recognition. The used data set comprises night shot video clips showing four classes of animals: deer, boar, fox and hare. Two networks are trained for animal detection and action recognition, respectively. Both trained networks are synchronized to predict the appearances of animals and their action.

1.2. Related work

Related work is reported using three perspectives. First, a geographical perspective lists visual detection approaches to terrestrial animals. Second, recent progress with respect to visual object detection in images and video clips is reported. Third, relevant approaches to action recognition are reported.

* Corresponding author.
E-mail address: schindl@cs.uni-bonn.de (F. Schindler).

[1] See the report of the European Commission COM (2010) 548 and http://www.eea.europa.eu/publications/the-european-grassland-butterfly-indicator-19902011.

[2] See "Living Planet Index 2014": http://www.wwf.de/living-planet-report/.

[3] With permission of the Autobahndirektion Nordbayern, Sachbereich Landschaftsplanung.

https://doi.org/10.1016/j.ecoinf.2021.101215
Received 15 July 2020; Accepted 2 December 2020
Available online 27 January 2021

1.2.1. Related work on visual animal detection

Recent approaches to visual animal detection in images and video clips from camera-traps show many applications in the African wilderness. For example, Brust et al. (2017) and Körschens et al. (2018) detect individual gorillas respectively elephants in images using face detection approaches to monitor the population. The work of Norouzzadeh et al. (2018), and Okafor et al. (2016) identify several African animal classes in images, but do not perform detection by drawing bounding boxes. Kellenberger et al. (2018) detect animals in UAV (Unmanned Aerial Vehicles) images and count the number of occurring animals. Zeppelzauer (2013) detects and tracks elephants in wildlife videos from Africa. Burghardt and Ćalić (2006) analyse the behaviour of lions in wildlife videos. The lions are tracked by face detection in the video. The locomotive behaviour of the animals is predicted automatically.

Willi et al. (2019) use different datasets mainly from Africa, but also a dataset containing images of animals from the North American Wildlife to perform image classification. Verma and Gupta (2018) classify images of Animals from North America. In the work of Wang (2019) location and movement data are analysed to describe animal behaviour. Movements of animals from North America are tracked by GPS data. The work of Chen et al. (2019a) deals with identifying one European animal class, badgers, in videos to monitor the spreading of diseases from badgers to farm animals.

The dataset used in this study, was also investigated by Follmann and Radig (2018) to detect animals with Faster R-CNN in the camera-trap images taken near a highway bridge in Bavaria, Germany.

Comprehensive surveys and discussions of the applications of camera trap methodologies and applications are given by O'Connell et al. (2011), Meek et al. (2014) and Falzon et al. (2020).

In this study, a combined approach to animal detection, identification and action recognition is applied to video clips of camera traps that are placed in Bavaria, Germany. In contrast to the aforementioned approaches, the observed animals, i.e., deer, boars, foxes and hares, show up mainly at dusk and night time resulting in gray-scale video clips. Tracking and locomotion behaviour cannot rely on face detection like the aforementioned studies on lions, gorillas, chimpanzees, etc. due to not observable face features.

1.2.2. Related work on visual object detection

As this study is aiming to detect individual animals in video clips, state-of-the art approaches to instance segmentation in video clips are the choice. Instance segmentation means to identify each instance of each object class. For example, if three boars are visible, the result of instance segmentation should be the identification and localization of each individual boar - and not just a classification of the scene as showing boars. Results of instance segmentation are bounding boxes as well as segmentation masks, i.e., exact shapes of the detected object appearances.

The survey of Liu et al. (2020) presents many different deep learning approaches for object detection. Mask R-CNN (He et al., 2017) is still the superior network for deriving segmentation masks. Mask R-CNN outperforms all previous winners of the COCO segmentation challenge and reaches an AP value for bounding boxes of 37.1 for the instance segmentation of the official COCO test dataset. A novel approach for instance segmentation is TensorMask Chen et al. (2019b) using sliding windows and a new architecture to tackle very dense segmentation tasks with many different objects. Additionally, the sliding window approach is more flexible with respect to the size of input images. But, the results of Mask R-CNN are slightly superior than the TensorMask results and Mask R-CNN can handle the size of the input images in this study well. Yang et al. (2019) also use a Mask R-CNN Network for instance segmentation in videos. An additional branch is added to the basic network structure for tracking individual frames. Hu et al. (2017) use recurrent neural networks to take the temporal coherence between frames for computing the segmentation masks and bounding boxes into consideration. Long-term temporal structures in the video are exploited to improve the segmentation results. In Vora and Raman (2018) the authors apply a flow-free approach for segmenting objects in videos. However, only a segmentation of foreground objects without a classification of the segmented object is performed. Flow-Guided Feature Aggregation (Zhu et al., 2017) improves detection networks to better deal with videos by including temporal information in terms of optical flow in their internal feature computation.

In this study, the combination of Mask R-CNN with Flow-Guided Feature Aggregation will be investigated to optimize instance segmentation in video clips.

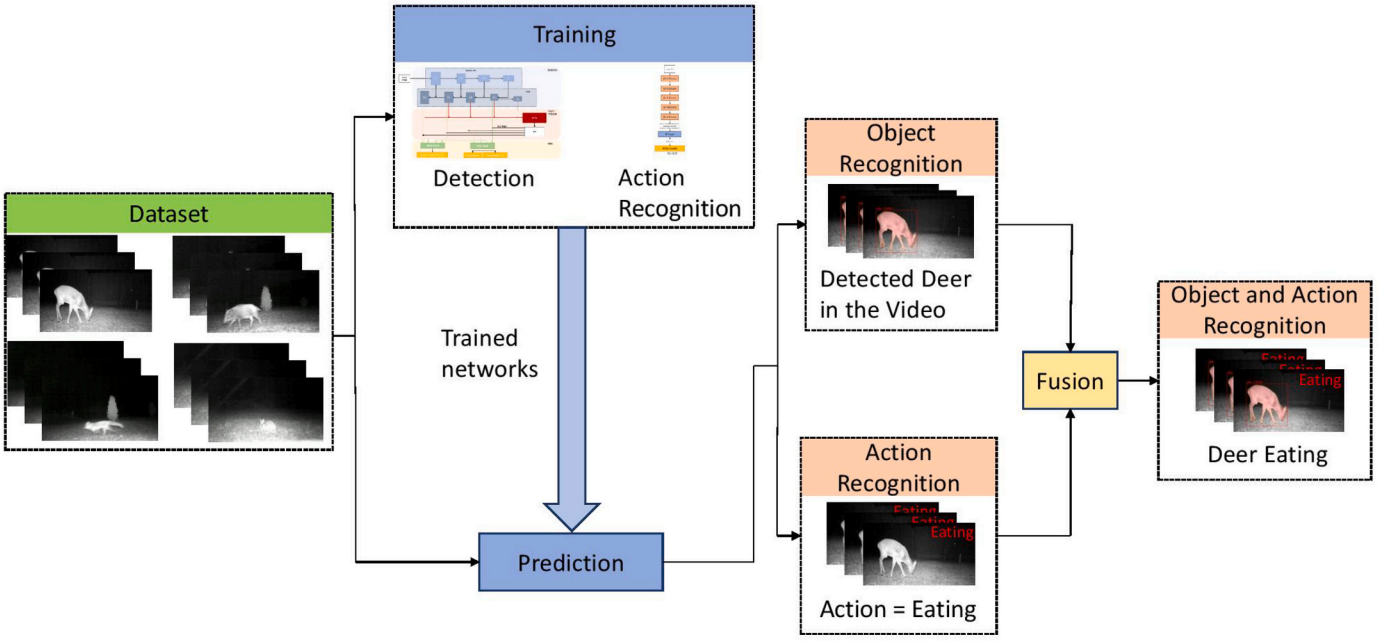


Fig. 1. The workflow in our developed detection and action recognition pipeline.

1.2.3. Related work on visual action recognition

Carreira and Zisserman (2017) compare different action recognition networks on the Kinetics dataset and also introduce a new Two-Stream Inflated 3D ConvNet for action recognition. In Sudhakaran et al. (2019) Gate-Shift networks for action recognition are introduced. Compared to many C3D approaches this architecture is more lightweight but delivers competitive results. In Lin et al. (2019) the authors present a new Temporal Shift Module (TSM), which is computationally efficient. This module is used for real-time action recognition. Moreover, TSM also detects objects in videos. Piergiovanni and Ryoo (2018) present a new TGM layer that can efficiently capture longer-term temporal information for action recognition.

Zhao et al. (2017) deal with action detection. A new type of network is proposed, the structured segment network, which describes the temporal structure of an action instance using a structured temporal pyramid. This network is used for detecting and describing the actions. The authors of Zeng et al. (2019) concentrate on the relations between action proposals to localize the corresponding actions. For this purpose a Graph Convolutional Network is designed to model the relationships between the proposals. Kalogeiton et al. (2017) present an Action-Tubelet detector that outputs sequences of bounding boxes for a couple of frames as input. This approach distinguishes better actions like sitting down or standing up, which cannot be classified by observing just single frames.

Tran et al. (2018) compare three different ResNet-18 variations: 3D ResNets (R3D), ResNets with Mixed Convolutions (MC ResNet) and R(2 + 1)D ResNets. All three variants use residual learning with spatio-temporal convolutions. Feichtenhofer et al. (2019) propose SlowFast showing a single-stream architecture that works on two different frame rates. This approach has also a biological counterpart in the primate visual system. In 2018, the R(2 + 1)D ResNet, outperformed all state-of-the-art approaches for action recognition on the Sports-1 M dataset. The SlowFast architecture has scored the highest accuracies in 2019 on the action recognition datasets Kinetics-400 (top-1 accuracy of 79.8), Kinetics-600 (top-1 accuracy of 81.8) and Charades (top-1 accuracy of 45.2). In a direct comparison, the R(2 + 1)D ResNet achieves a top-1 accuracy of 73.9 for the Kinetics-400, which is still competitive. Furthermore, SlowFast achieves the best accuracies on the AVA dataset for action detection.

Therefore, in this study all the three different ResNet-18 variations and the SlowFast architecure are evaluated to implement the action recognition task.

2. Dataset and annotation

The data material used in this study is provided from the Bavarian Highway Directorate, Germany. It shows video clips, each about 10 s with 8 fps (frames per second) and a resolution of 1280 × 720 pixel. The clips were taken by camera traps positioned at an "animals' bridge" (wildlife crossing) that is crossing the federal motorway 7 near the city Oberthulba. The camera traps are equipped with infrared camera and infrared flash lights. After sorting out unusable clips like those showing no animals, 528 clips taken at nighttime remain showing deer, boars, foxes and hares (cf. Fig. 2) with occurrence frequencies depicted in Table 1.

2.1. Annotation for object detection

For training and evaluation of the object detection, 40 clips containing 2434 usable frames in total are used yielding the subset $Annotations_{Detect}$. Table 2 shows the frequencies of the four occurring animal classes. Since the overall data is not balanced with respect to the occurrences of these four classes (e.g., there are only 8 clips showing foxes), the selection shows a compromise between the burden of annotation and almost balanced occurrence frequencies.

The VGG Image Annotator (VIA) Version 2.0.8 (Dutta and Zisserman (2019)) is used for the annotation of the segmentation dataset. For each occurring animal in a frame of the training set, a polygonal segmentation mask depicting the visible silhouette, a bounding box and a class label, i.

Table 1
Frequencies of occurrence of deer, boar, fox and hare videos in the dataset.

	Deer	Boar	Fox	Hare
Number of videos	477	20	9	23
Percentage of the dataset	90.2%	3.8%	1.7%	4.3%

Fig. 2. Exemplary frames of the dataset.

Table 2
Frequencies of four animal classes in the subset $Annotations_{Detect}$.

	Deer	Boar	Fox	Hare
Number of videos	12	10	8	11

e.: deer, boar, fox, hare. A polygon for a deer's silhouette shows typically around 70 vertices. The polygonal silhouette of a hare may show only around 35 vertices.

2.2. Annotation for action recognition

In this proof-of-concept study, three action classes are considered: eating, moving and watching. For training and evaluation, clips with lengths from 3 up to 9 s showing unique actions were annotated with corresponding action labels yielding the subset $Annotations_{Actions}$.

Table 3 depicts the frequencies of the action classes in the subset $Annotations_{Actions}$.

3. Methods

3.1. Detection networks

Visual detection and identification of observed animals in the video frames of the camera traps should result per frame in: (1) a bounding box for each detected animal depicting its location, (2) a segmentation mask per animal depicting the exact shape, and (3) a class label depicting the type of animal (here: deer, boar, fox, hare). Two approaches are evaluated: Mask R-CNN (He et al., 2017) and Flow-Guided Feature Aggregation (Zhu et al., 2017). Mask R-CNN is the approach of choice due to its superior recognition performance in images (cf. Section 1.2.2). Flow-Guided Feature Aggregation in turn, offers the opportunity to improve object detection in video clips by including temporal information in terms of the optical flow in their internal feature computation. Thereby, optical flow stands for the apparent motion of objects in consecutive frames.

3.1.1. Mask R-CNN

Mask R-CNN consists of three main parts: the backbone, the region proposal and the head. The network architecture is illustrated in Fig. 3.

The first stage of the backbone is typically a ResNet-50. After the ResNet-50 follows a Feature Pyramid Network (FPN). The outputs of the backbone are extracted features that are used to detect the objects in the following stages. The FPN takes the features from 4 layers of the ResNet and uses a top-down architecture, starting from the smallest feature map going up to the largest feature map as it is illustrated in Fig. 3. The FPN is important to detect objects at different scales. The FPN upsamples the feature maps and uses lateral connections from the ResNet to predict the spatial locations of the features more exact. All 5 blocks of the FPN have a 256-dimensional output of feature maps.

In the next stage, region proposals are generated. A Region Proposal Network (RPN) creates Regions of Interest (ROI) from the features of the FPN. A very important parameter in the RPN is the Non-Maximum Suppression (NMS). With this parameter it is controlled, when proposals are removed because of overlap. For creating masks and bounding boxes, the ROI have to be transferred to the original image positions. To fulfill this goal, a ROI Align layer calculates exact pixel positions of the ROIs with bilinear interpolation.

The last stage of the network is the so-called head generating three outputs: the bounding box, the class label and the segmentation mask. The bounding box and class label are predicted in one path using fully connected layers. The Mask Head produces the segmentation masks. Its path does not contain fully connected layers to not lose the spatial structure information that is necessary for a segmentation mask.

Table 3
Frequencies of action classes in the subset $Annotations_{Actions}$.

	Eating	Moving	Watching	Overall
Number of videos	257	39	60	356
Percentage of the dataset	72.2%	11.0%	16.8%	100%

3.1.2. Flow-guided feature aggregation

The two key ideas of Flow-Guided Feature Aggregation (FGFA) are motion-guided spatial warping and the feature aggregation. The aggregation parameter K defines a neighbourhood of frames. More precisely, the frames I_{i-K}, …, I_{i-1}, I_{i+1}, …, I_{i+K} are called the neighbouring frames of I_i. For example, $K = 1$ means that the frame before and the frame after the reference frame form the neighbourhood.

For the flow-guided warping, a flow-field $M_{i\to j} = F(I_i, I_j)$ is calculated with the help of a flow network. The optical flow is calculated between two frames. The calculated flow-field is used to warp feature maps of the neighbouring frames to the reference frame.

The feature aggregation step combines the information from the warped feature maps of the neighbouring frames and from the original feature maps of the reference frame. For the aggregation different weights for different spatial locations p on the feature maps are used. Each weight is the same for all feature channels. The new aggregated feature replaces the old feature in the detection process. The aggregated features are computed for each frame in the currently selected video.

For Mask R-CNN, the feature aggregation is performed for all 5 feature map groups from the FPN. A pretrained backbone is used in the training process. Only the RPN and the head of the Mask R-CNN are trained. Fig. 4 shows the overall architecture of FGFA for $K = 1$.

3.2. Action recognition networks

We compare three different ResNet-18 variations (3D ResNets (R3D), ResNets with Mixed Convolutions (MC ResNet) and R(2 + 1)D ResNets) (Tran et al., 2018) with the SlowFast architecture (Feichtenhofer et al., 2019). The action recognition networks output an action labels for video clips.

3.2.1. 3D ResNet, MC ResNet and (2 + 1)D ResNet

All three ResNet variants use residual learning with spatiotemporal convolutions. The input of the networks is in the form $3 \times T \times H \times W$. Here T denotes the number of frames in this clip. A typical value for T is 8 or 16. Further, H and W are the height and width of the image. The 3 represents the three color channels RGB. The output of the i-th convolutional block of the residual network is z_i. It is computed as

$$z_i = z_{i-1} + F(z_{i-1}, \theta_i) \tag{1}$$

One block consists of two consecutive convolutional layers and ReLU activation functions after each layer. The function F comprises these steps for an input with the weights θ_i. All ResNet-18 variations consist of 5 blocks following each other, where the first block is a special stem block containing only one convolutional layer instead of two and not using a residual connection. A stem and a convolutional residual block are depicted in Fig. 5. A global average pooling over the spatiotemporal volume follows after the last layer. After reshaping, a 512-dimensional feature vector remains. This vector is the input to a fully connected layer for the classification. An action proposal can be extracted with a following softmax function and taking the action with the highest value.

Using simple 2D convolution blocks does not incorporate temporal structure. For a 2D convolution the input is reshaped as $3T \times H \times W$. The frames are arranged correctly in time. However, this order will be lost during the following convolutions. To incorporate temporal information 3D convolution blocks are used. Here z_i is 4-dimensional in the form $N_i \times L \times H_i \times W_i$. The 3D convolution encompasses the temporal and spatial domain.

The R3D architecture consists only of 3D convolutional blocks. The Mixed Convolutions ResNet (MC ResNet) combines layers of 3D

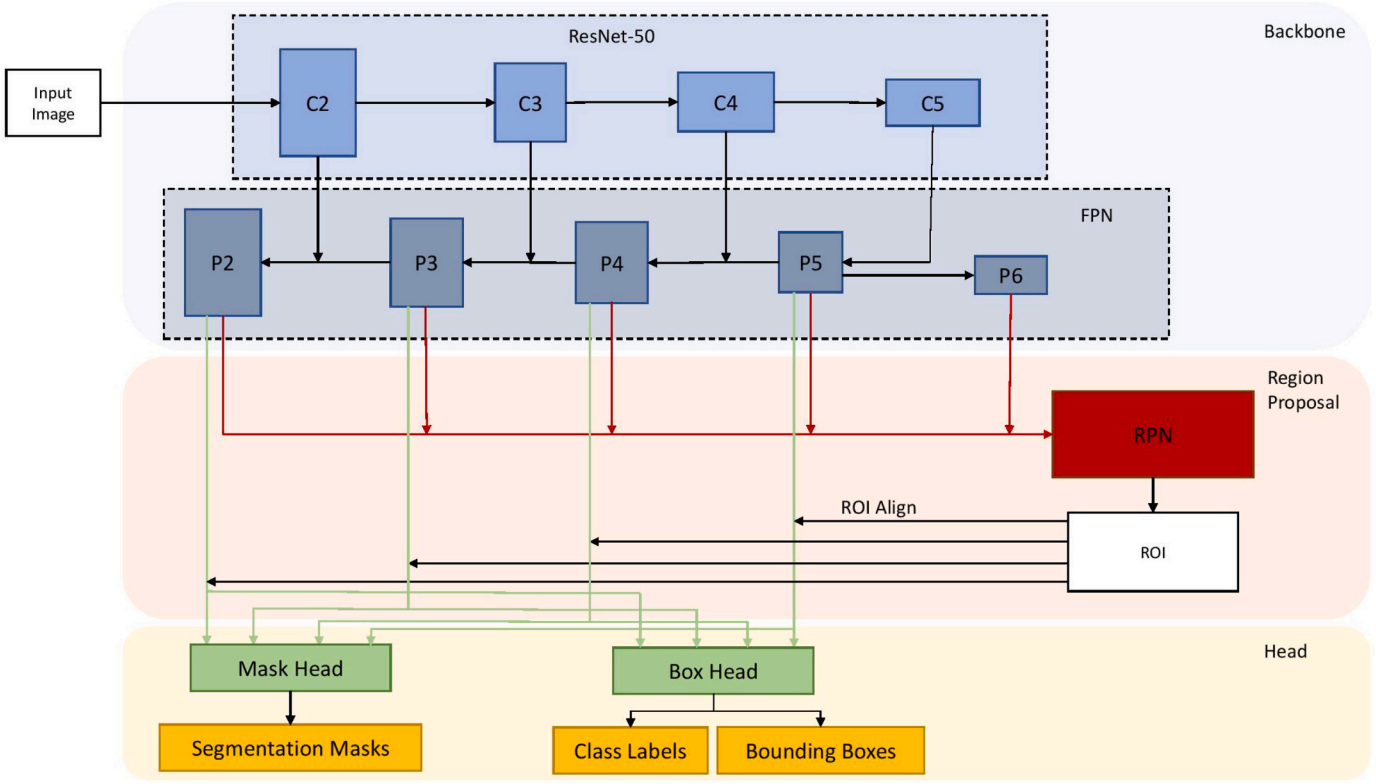


Fig. 3. The network architecture of Mask R-CNN: Mask R-CNN consists of three main stages, the backbone, the region proposal and the head.

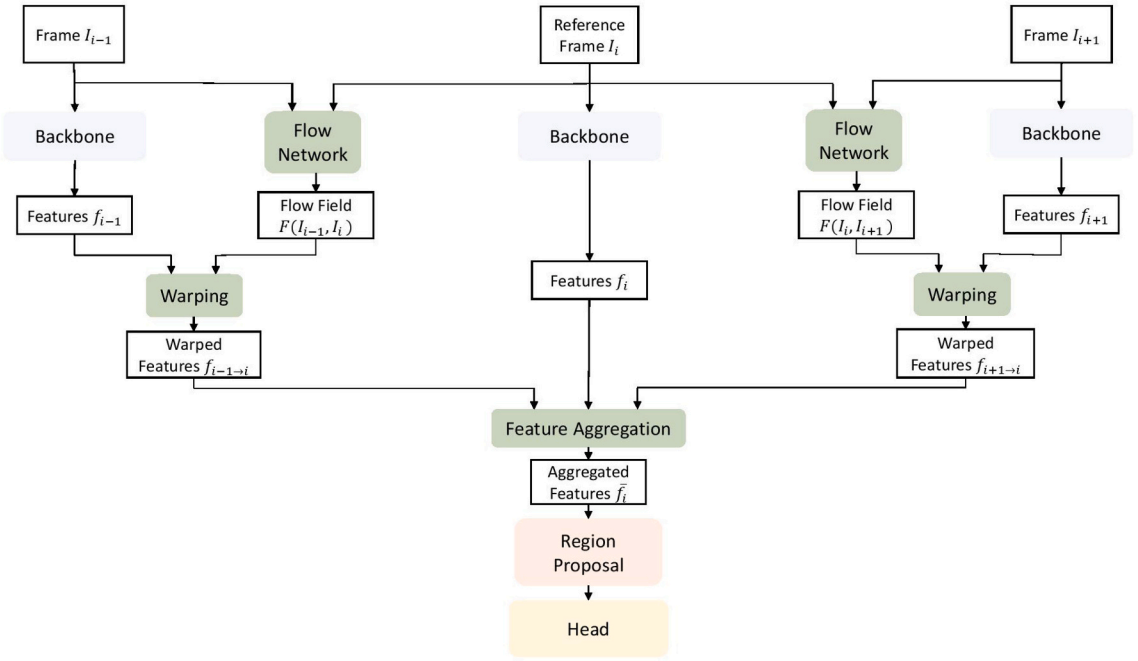


Fig. 4. The network architecture of FGFA: The FGFA architecture extends the parts of the Mask R-CNN. It is shown for an aggregation range of K = 1, considering one neighbour frame before and after the reference frame.

convolutions and 2D convolutions. The reason for this mixture is that motion modeling is only necessary in the first layers. In the last layers semantic abstraction is more important than temporal information. Tran et al. (2018) test different combinations of 2D and 3D layers, also including a reverse approach of switching the order beginning with 2D convolutions and ending with 3D convolutions. The best combination is the MC3 variation, starting with two blocks of 3D convolutions followed by three blocks of simple 2D convolutions.

Tran et al. (2018) also introduce a new design for convolutions, the (2 + 1)D convolution. They split the 3-dimensional convolution into a 2-dimensional spatial convolution followed by a 1-dimensional temporal convolution. The authors evaluate that using (2 + 1)D convolutions

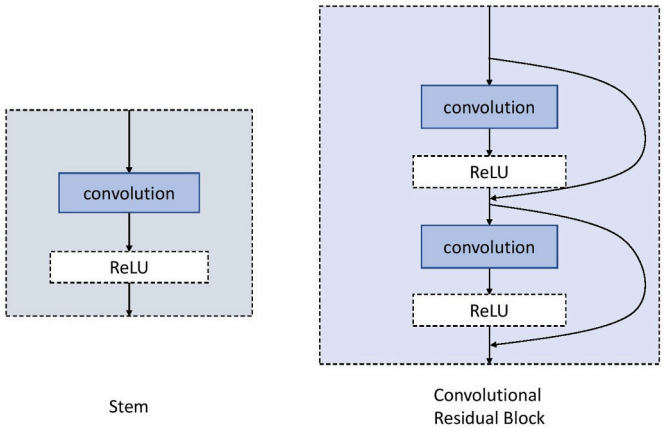


Fig. 5. A stem and a convolutional residual block of a ResNet are shown. The convolution layer can be implemented with a 2D, 3D or (2 + 1)D convolution.

leads to lower training and testing errors compared to 3D convolutions.

The three different ResNet variations are shown in Fig. 6.

3.2.2. SlowFast architecture

In this section we follow Feichtenhofer et al. (2019). The key idea of SlowFast networks is using two pathways for the video input, a *Slow pathway* and a *Fast pathway*. The Slow pathway works on a low frame rate and low temporal resolution for detecting spatial semantics. The Fast pathway uses a high frame rate to capture motion and the temporal aspects in the video. Simultaneously the Fast pathway reduces its channel capacity so it becomes lightweight, but is still able to learn the temporal information of the video. The backbone for the pathways can be implemented by different networks. The best results achieve ResNets. SlowFast is a single-stream architecture that works on two different frame rates.

The Slow pathway uses a large temporal stride τ on the input frames. The pathway considers one out of τ frames from the video.

The Fast pathway should model the temporal structure of the video. The temporal stride of this pathway is τ/α with $\alpha > 1$. The stride is smaller than the stride of the Slow pathway. The parameter α represents the frame ratio between the Fast and the Slow pathway. The Fast pathway samples αT frames from the input video. Hence, its input is a denser representation of the input clip. A typical value is $\alpha = 8$. In the Fast pathway in all layers the feature maps have the same high temporal resolution of αT. There is no temporal downsampling until the average pooling layer is reached. The channel capacity of the Fast pathway can be reduced to make it more lightweight and still deliver high accuracies. In the Fast pathway the number of channels equals β times the number of channels of the Slow pathway, where $\beta < 1$. The authors state that a typical value for β is 1/8. An interpretation of this reduced channel capacity is that the Fast pathway does not need to model spatial information with such a high detail as the Slow pathway.

To combine the features from the Fast pathway with the Slow pathway the authors use lateral connections. They are unidirectional and go from the Fast pathway to the Slow pathway. The lateral connections are positioned after each building block of the ResNet. The structure of a ResNet has already been described in Section 3.2.1.

The average pooling is done on both pathways. Then the results of both pathways are concatenated and build the input for the fully connected layer.

The described SlowFast architecture is shown in Fig. 7.

4. Results and evaluation

4.1. Evaluation metrics

We exemplify the evaluation metrics that we use in the following sections. We use the official COCO evaluation metrics and follow Lin et al. (2014) and Everingham et al. (2010).

Mask R-CNN and FGFA assigns to each bounding box or segmentation mask prediction a confidence score that indicates how reliable this prediction is. An important role for the metric calculation plays the Intersection over Union (IoU). IoU is defined by

$$IoU(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2}$$

where B_p and B_{gt} denote the area of the predicted bounding box and the area of the ground truth bounding box. For segmentation masks we use an analogous definition.

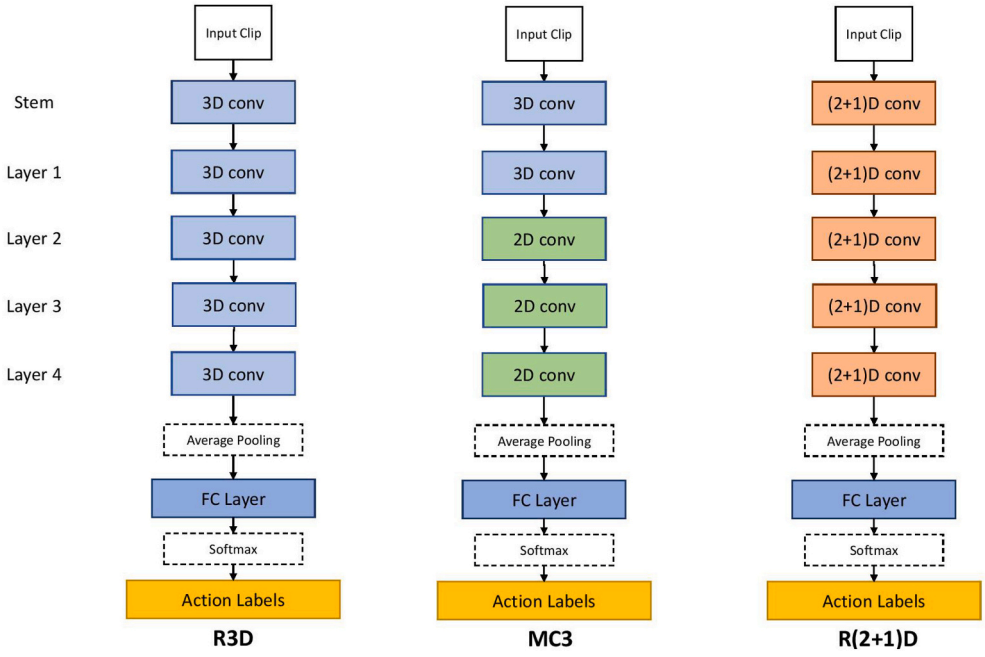


Fig. 6. The architectures of the three ResNet variations are depicted. They only differ in the way the convolutional residual layers are built.

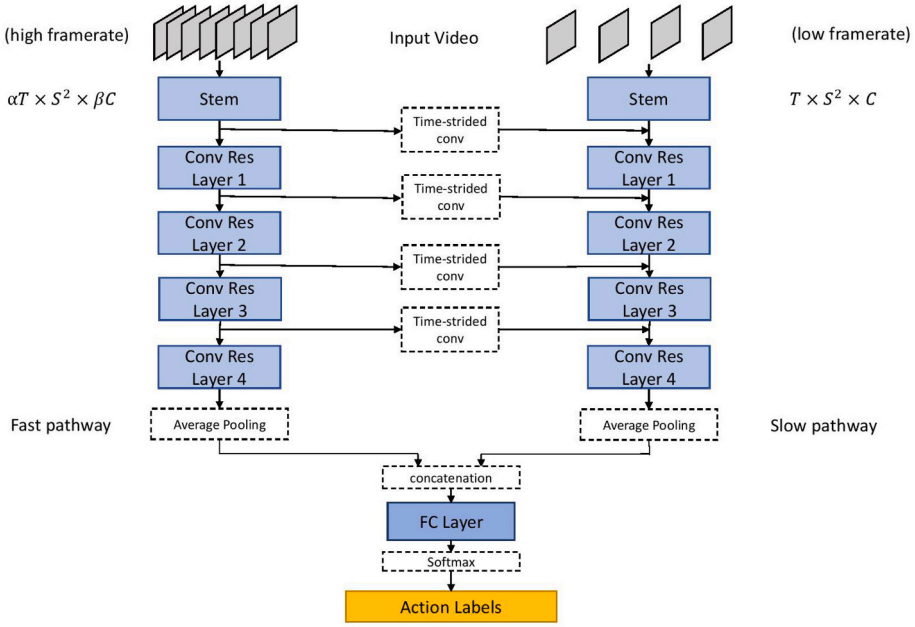


Fig. 7. The SlowFast architecture: The backbone for the Slow and Fast pathway is a ResNet. Above, for both pathways the feature map sizes are denoted before the first convolution in the Stem. The letter T represents the number of frames that are sampled by the Slow pathway. Further, C is the number of channels and S is the width and height of a square crop from a frame.

A true positive detection (TP) is given if

- the confidence score is greater than a threshold, e.g. 0.5 (Is the detection reliable?)
- the IoU is greater than a threshold, e.g. 0.5 (Is the detection in the right place?)
- the class prediction is correct

If the confidence score is greater than the confidence threshold, but the class prediction is wrong or the IoU is lower than the desired threshold (or both), then the detection is a false positive (FP). A false negative (FN) occurs, when the confidence score of a detection is lower than the threshold for a ground truth object that should be detected. Precision and Recall are defined in the usual way.

Different threshold values for the confidence score imply different precision and recall pairs. Plotting the recall on the x-axis and the precision on the y-axis results in a precision-recall curve. Decreasing the confidence threshold from 1 to 0 leads to a monotonically decreasing recall. The precision may go up and down, but essentially will shrink when the confidence threshold decreases.

4.1.1. Average precision

The average precision (AP) is a numeric metric to compare precision-recall curves from different detectors. AP computes the average precision for different recall levels of the same detector. The precision-recall curve is gained by the interpolation of several (recall, precision)-values. To be precise, the interpolated precision p_{interp} for the recall level r is defined by

$$p_{interp}(r) = \underset{r' \geq r}{maxp}(r') \tag{3}$$

So p_{interp} is the greatest precision value that is reached for a recall level r' greater or equal than r. The considered recall levels r are selected equidistant. For the COCO metrics neighbouring recall levels differ by 0.01. Therefore, 101 recall levels from 0.0 to 1.0 are considered for the computation. Hence, the AP for COCO datasets for a class i at the IoU threshold th is calculated as

$$AP_{i,th} = \frac{1}{101} \sum_{r \in \{0.0:0.01:1.0\}} p_{interp}(r) \tag{4}$$

The AP is averaged over all classes, which is often also called mean average precision (mAP). In the COCO evaluation metrics the mean average precision is denoted by AP, too. We follow the naming convention from COCO datasets. For K classes with the corresponding $AP_{i,\ th}$ the mean AP_{th} at IoU threshold th is calculated as

$$AP_{th} = \frac{\sum_{i=1}^{K} AP_{i,th}}{K} \tag{5}$$

The AP of the COCO evaluation metric (except $AP_{0.50}$ and $AP_{0.75}$) is computed as the averaged AP for different IoU thresholds. The averaging considers the IoU thresholds 0.50,0.55, … 0.95. So, the AP is the average over all classes and IoU thresholds. The AP is calculated as

$$AP = AP_{0.50:0.05:0.95} = \frac{AP_{0.50} + AP_{0.55} + \ldots + AP_{0.95}}{10} \tag{6}$$

We also consider $AP_{0.50}$ and $AP_{0.75}$, which are part of the above calculation as an evaluation metric. $AP_{0.50}$ is the AP for IoU threshold 0.50, also known as Pascal VOC metric and $AP_{0.75}$ is the AP for IoU threshold 0.75, also called strict metric in the sense of COCO.

4.1.2. Average recall

The average recall (AR) is averaged over different IoU thresholds. For COCO datasets we consider the 10 IoU thresholds 0.50, 0.55, … 0.95. They are identical to (6). Therefore, the AR for class i at IoU threshold th is computed as

$$AR_{i,th} = \frac{r_{0.50} + r_{0.55} + \ldots + r_{0.95}}{10} \tag{7}$$

The AR is also averaged over all K classes. This average can also be called the mean average recall (mAR), but in COCO the mAR is denoted as AR. The AR_{th} at IoU threshold th is calculated as

$$AR_{th} = \frac{\sum_{i=1}^{K} AR_{i,th}}{K} \tag{8}$$

We consider two different ARs, $AR_{max=1}$ and $AR_{max=10}$. They are

calculated for 1 detection respectively for a maximum of 10 detections in one image. Both ARs are averaged over the described 10 IoU thresholds

$$AR = \frac{AR_{0.50} + AR_{0.55} + \ldots + AR_{0.95}}{10} \quad (9)$$

4.2. Implementation details

The training and testing was performed with a GeForce RTX 2080 Ti GPU with 11 GB graphic memory, 16 GB RAM and an Intel Core i7-6700K 4.00 GHz CPU.

All programs use Python 3 and PyTorch (Paszke et al. (2019)) for building and training the networks. The Mask R-CNN model is built around the PyTorch detection model of Mask R-CNN. For the implementation of FGFA we use the PyTorch models for the backbone, the RPN and the head of the implementation of Mask R-CNN model. The three ResNet variations are also based on PyTorch detection models. We always use the pretrained versions of the detection models. The Mask R-CNN is pretrained on *COCO 2017train* dataset. The ResNets are pretrained on *Kinetics-400* dataset. We use these PyTorch models because they allow more flexibility than other implementations. They are better adjusted to the PyTorch training functions and are well-arranged. Moreover, the PyTorch detection models do not have other dependencies (e.g. special libraries) unlike other implementations.

4.3. Object detection

Two approaches to object detection via instance segmentation are evaluated. Mask R-CNN is trained on video clips, i.e., the video frames are presented in the temporal sequence given in each clip. This Mask R-CNN approach is then compared with the Flow-Guided Feature Aggregation (FGFA).

For training and evaluation of the object detection, 40 clips containing 2434 usable frames in total are used yielding the subset $Annotations_{Detect}$ (cf. Section 2.1). This subset is split again into a training subset and test subset. This split is the same for all experiments. The test set contains 8 clips, two of each animal class. We select such clips that show quite normal scenarios but also clips showing more challenging scenarios (e.g. with rain or fog). The training set contains accordingly 32 clips.

Stochastic Gradient Descent (SGD) is used as optimization technique. We start with a learning rate of 0.0005, a momentum of 0.9 and weight decay of 0.0005. Different settings of momentum and weight decay are checked but show no improvements. In the training process, we decrease the learning rate every 10 epochs by multiplying it with $\gamma = 0.1$. In the first epoch we use warmup iterations for the learning rate with a warmup factor of 1/1000. These parameters gave the best results in our evaluations. We train our models for 30 epochs.

We use a pretrained backbone for FGFA. The backbone is extracted from a Mask R-CNN network trained on the segmentation dataset. During the training process FGFA trains only the head and the RPN. We implement the flow network in the FGFA architecture with a SPyNet (Ranjan and Black (2017)). Other popular choices are FlowNet (Dosovitskiy et al. (2015)) and FlowNet 2.0 (Ilg et al. (2017)). SPyNet is more lightweight than the FlowNets. It uses 96% less parameters than FlowNet. Therefore it is more efficient in an embedded environment like our FGFA architecture. Moreover, the results of SPyNet are more accurate than the results of the FlowNets.

4.3.1. Data augmentation for segmentation and detection

We utilize different data augmentation techniques during the training process to prevent overfitting and to generalize better. We use the python library imgaug (Jung et al. (2020)) for the data augmentation of our images and videos. For Mask R-CNN and FGFA the same augmentation technique with identical parameters is applied to each frame of the video. We use the augmentation techniques *horizontal flip*, *add / subtract intensity*, *Guassian blur* and *additive Gaussian noise*. All the mentioned augmentation techniques are used simultaneously. A further special augmentation function from imgaug we use, is *Fog*. This complex technique simulates fog in the video. Since it is a very strong augmentation, we combine it only with the random horizontal flip. The *Fog* augmentation is performed with a probability of 10%. The other combined augmentation techniques described above are applied with probability of 90%.

4.3.2. Comparison between mask R-CNN and FGFA

Both networks are trained for 30 epochs. The results on the test set are evaluated using the COCO evaluation metrics. The evaluation results of object detection by bounding boxes and segmentation masks are shown in Table 4. In all categories Mask R-CNN shows the best results. All detection results via segmentation masks are equal or slightly worse than the detection results via bounding boxes. This is on the one hand obvious because the derivation of the segmentation masks is more challenging but on the other hand the differences are quite small, i.e. AP values show comparable qualities of both results. Fig. 8 shows examples of detection results of all observed animal classes in this study using Mask R-CNN.

For comparison, He et al. (2017) reach an AP value of 39.8 for bounding boxes and 36.7 for segmentation masks. These results are achieved on the COCO dataset with 80 classes. However, it should be noted that a fair comparison with our results is difficult to draw as the authors have to deal with significantly more classes.

This is surprising at first view, since FGFA should improve the detection results because it aggregates features from neighbouring frames and thereby includes temporal information into the detection process. But this inter-frame inspection might not be helpful when observing articulated target objects that may change their visual appearance more or less significantly by overall rotation relative to the camera trap or by movements of body parts like legs, neck, head, etc. The features in FGFA are aggregated with a warping function using optical flow. Own experiments show, that the warping can predict correct feature positions more robust when observing more compact and not strongly articulated target objects. Moving the legs during walking or turning around leads to false predictions in the optical flow. Optical flow cannot predict a leg that moves forward and crosses another leg in the video. The warping cannot distinguish between the legs, for example. We support this theoretical argumentation with experiments for all three networks. We test this hypothesis on video clips where animals show as much non-articulated behaviour as possible. For example, this is the case for deer that eat and hardly move.

The evaluation results for frame sequences showing animals with less articulated characteristics are depicted in the right columns of Table 4. Both networks show similar results. Especially FGFA achieves much higher precision and recall values than for the basic evaluation set. Also the values for Mask R-CNN increase on this test set. FGFA is still slightly worse than Mask R-CNN.

Last but not least, the video clips of the given material in this study show a frame rate of 8 frames per second (cf. Section 2). Higher frame rates like 25 frames per second will deliver more dense temporal information where differences in visual appearances of target objects between neighbouring frames will be less significant and will be utilized by FGFA. On the other hand the provision of energy and data transfer are critical issues for the deployment of camera traps for wildlife monitoring and both issues can benefit from lower frame rates used in the camera traps.

4.4. Action recognition

The three ResNet variants are trained for 40 epochs starting with an initial learning rate of 0.001. The learning rate is decreased every 10 epochs by multiplying it with a $\gamma = 0.1$. For optimization we utilize the Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a

Table 4
The evaluation results for bounding boxes and segmentation masks with the COCO metrics on the test set for our both network architectures, each trained 30 epochs. In comparison to the basic evaluation are the results for scenes with animals that show mainly non articulated behaviour. The best result of each row is marked in green. Articulated and non-articulated objects are compared separately.

Metric	Mask R-CNN	FGFA	Mask R-CNN (non articulated objects)	FGFA (non articulated objects)
bounding boxes				
AP	0.638	0.574	0.798	0.784
$AP_{0.50}$	0.934	0.894	0.990	0.990
$AP_{0.75}$	0.735	0.663	0.985	0.973
$AR_{max=1}$	0.610	0.566	0.803	0.801
$AR_{max=10}$	0.691	0.629	0.831	0.820
segmentation masks				
AP	0.590	0.535	0.808	0.797
$AP_{0.50}$	0.926	0.877	0.990	0.990
$AP_{0.75}$	0.676	0.587	0.985	0.973
$AR_{max=1}$	0.575	0.534	0.803	0.803
$AR_{max=10}$	0.648	0.590	0.831	0.820

(a) Example frame of detected deers

(b) Example frame of detected boars

(c) Example frame of a detected fox

(d) Example frame of detected hares

Fig. 8. Exemplary detection results for all four animal classes by Mask R-CNN, the best performing network. The ground truth segmentation mask is represented by the yellow line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

weight decay of $5 \cdot 10^{-4}$. As described in the dataset Section 2.2, especially in Table 3, the frequencies of action classes in the dataset are quite unbalanced. To account for this, we use balance weights as (1.0,4.5,4.5) for the classes eating, moving and watching. The loss function is the cross entropy loss. This parameter setting has shown the best results in our experiments. The input for the ResNets are 16 consecutive frames of the video. These frames correspond to 2 s of the video because our videos have 8 fps.

The SlowFast architecture is trained in 40 epochs with an initial base learning rate of 0.1. We use 5 warmup epochs with a warmup learning rate of 0.001. Momentum is set to 0.9 and the weight decay parameter is $1 \cdot 10^{-4}$. The learning rate is decreased with a cosine strategy. The optimization technique is SGD and the loss function is the cross entropy loss. In this parameter selection we follow Feichtenhofer et al. (2019). We set the temporal length of the input to 16 frames. We set the sampling rate to 4 and $\alpha = 2$. The parameter β is set to 1/8 as proposed by

Feichtenhofer et al. (2019). This parameter combination achieved the best results in our experiments. The Slow and Fast pathways are implemented by a ResNet-101.

We split the annotated dataset $Annotations_{Actions}$ (cf. Section 2.2) randomly (but fixed for all experiments and all networks) into a train set, validation set and test set. The test set includes 20% of the video clips of each class. Of the remaining clips, again 20% of the clips in each class form the validation set. The remaining clips are the training set.

4.4.1. Data augmentation for action recognition

For training, we use common data augmentation techniques for action recognition as explained in Tran et al. (2018) and Feichtenhofer et al. (2019) to augment our action recognition dataset. Three augmentation techniques are used simultaneously: temporal jittering, spatial crops and horizontal flips. For temporal jittering the starting point of the 16 frames in the input video is chosen randomly (with the constraint that there are 16 frames after the starting point in the video). Our videos in the action recognition dataset are between 3 and 9 s long, consisting of 24 to 72 frames.

Random spatial crops are performed on the selected 16 frames long clip. A square crop is randomly selected in the first frame. The same spatial region is also cropped in the remaining video clip. A horizontal flip of the complete clip is performed with a probability of 0.5. For the flip we use again the imgaug library.

For testing, several video clips are sampled from one video at a uniform interval of 1 s. As an example, a 3 s long video results in two sample clips. Sampling is necessary, because the input size of the network is still fixed to 16 frames. Each testing video is labeled with one action class, but one whole video is too large and therefore split in smaller clips for the prediction. Spatial crops are always center crops. The cropping adjusts the size of the video to the required input size of the network. Performing the temporal sampling in fixed intervals and applying center crops leads always to the same test set. This makes the test set uniform and comparable.

For the SlowFast networks the same data augmentation techniques are performed. They only differ in the size of the spatial crops due to the different architecture.

4.4.2. Comparison between different networks

The four different networks are compared with respect of the achieved accuracies and the training time. Table 5 shows the results on the test set for the networks.

The (2 + 1)D ResNet reaches the highest accuracy. (2 + 1)D ResNet also achieves the best results in the experiments of Tran et al. (2018). 3D ResNet and MC ResNet show similar accuracies. Both architectures are quite similar. MC ResNet differs only in the usage of 2D convolutions in the last layers.

tSNE stands for T-distributed Stochastic Neighbor Embedding (van der Maaten and Hinton, 2008) and is an approach to visualize high-dimensional data by giving each datapoint a location in a two or three-dimensional map. Fig. 9 shows a tSNE embedding of the 512-dimensional feature vectors of the R(2 + 1)D ResNet. This feature vector is the input for the last, fully connected layer that uses this information to classify the action. The embedding shows a 2D projection of the feature vectors of the clips from the test set. Each data point is marked in the color of its true class. The style of the points indicates whether the network made a true prediction (marked as a cross) or false prediction (marked as a dot). The true positive predictions of the three action classes form three indistinguishable clusters. The smaller action classes (containing fewer samples in the datasets) Moving and Watching are closely neighboured. The R(2 + 1)D network classifies all clips with the action Moving correctly. False predictions can be seen in the clusters' border area. Fig. 10 visualizes the action recognition of a deer. The action proposal of the (2 + 1)D ResNet is found in the upper-left corner of each frame. The proposal considers the last 16 frames, which correspond to the last 2 s of the video clip. This latency is the reason why some frames may show after a change of the action the former action label of their preceding frames (cf. Fig. 10 (d)).

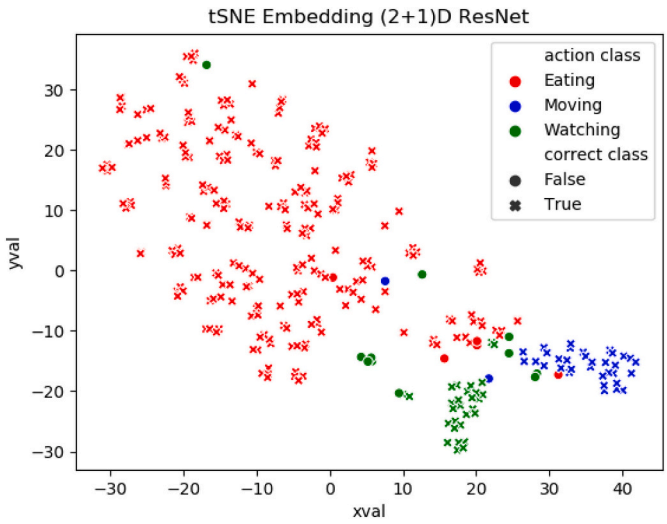


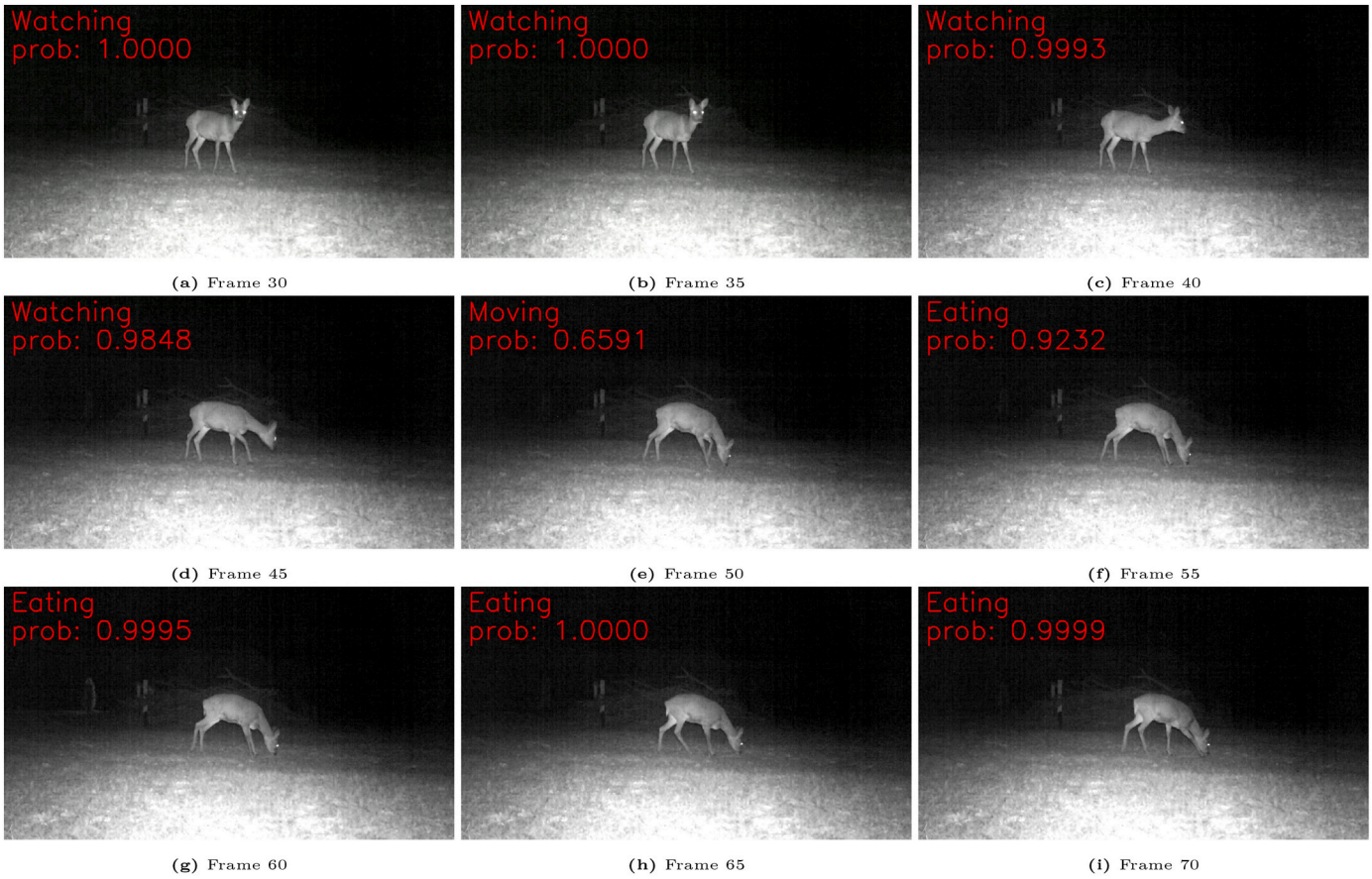
Fig. 9. tSNE embedding of the 512-dimensional feature vectors before the last, fully connected layer of the (2 + 1)D ResNet. Each data point is marked in the color of its true class. The style of the points indicates whether the network made a true prediction (marked as a cross) or false prediction (marked as a dot). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

SlowFast is the least accurate of all networks for all datasets. Feichtenhofer et al. (2019) use videos with 30 fps while our videos only have 8 fps. As a preprocessing step the official github implementation of SlowFast samples the videos so that they have 30 fps again. Applying this preprocessing step to our videos leads to many duplicate frames in a video. The videos from the *Kinetics-400* and *Charades* datasets used by Feichtenhofer et al. (2019) have 25 and 24 fps, respectively. The high number of duplicates in our case changes the appearance of the motion of the animals, which probably affects the accuracy. Moreover, we deal with grayscale videos instead of color videos. Feichtenhofer et al. (2019) also experiment with grayscale videos as input for the Fast pathway but the Slow pathway still gets color videos as input. It seems that for the extraction of the features in the Slow pathway color information is more important than in the Fast pathway. These are the reasons why the SlowFast approach does not perform as well as the simpler ResNet networks.

The ResNet variations show similar training times. SlowFast shows a drastically higher training time than the other networks due to the

Table 5
Evaluation of approaches to action recognition. Best accuracy values are marked in green.

Metric	3D ResNet	MC ResNet	(2+1)D ResNet	SlowFast
accuracy	0.9346	0.9283	0.9408	0.6634
train time [sec]	2562.1	2531.0	2820.2	17362.4



(a) Frame 30 (b) Frame 35 (c) Frame 40

(d) Frame 45 (e) Frame 50 (f) Frame 55

(g) Frame 60 (h) Frame 65 (i) Frame 70

Fig. 10. Action recognition for a deer video performed by (2 + 1)D ResNet. The frames are extracted in steps of 5 frames from the video. The whole video has 72 frames. The recognized action with its probability is shown in the left corner of the frame. The proposal considers the last 16 frames (2 s). In this figure, the action predictions in the frames are magnified for better readability.

complex architecture of SlowFast where two pathways have to be trained and then fused. The significantly shorter training time can also be seen as a secondary selection criterion for ResNets.

4.5. Fusion of object detection and action recognition

In this study, the evaluations in Sections 4.3 and 4.4 suggest the

(a) Frame 26 (b) Frame 27 (c) Frame 28

(d) Frame 29 (e) Frame 30 (f) Frame 31

Fig. 11. Fusion of detection and action recognition for six consecutive frames of a video showing a deer. The action class can be seen in the upper left corner. The animal class of the detection step is noted at the top of the bounding box.

fusion of Mask R-CNN Video for object detection and (2 + 1)D ResNet for action recognition.

Fusion of the results is implemented with a mask based blending. One mask is blended in the upper left corner of each frame to depict the action prediction in the frame. The mask for the object detection covers the remaining part of the frame accordingly depicting bounding boxes, segmentation masks and animal identities - in this study: deer, boar, fox, haze.

Fig. 11 presents the fused results from the detection network and the action recognition network. Six consecutive frames are shown where the predicted action is denoted in the upper left corner.

5. Conclusion and future work

This study presents a deep learning-based architecture combining (1) the detection of individual animals yielding bounding boxes as well as precise shapes of appearances with (2) the recognition of the actions of the observed animals. The resulting end-to-end pipeline is applicable in camera traps for tracing of animal populations and analysing their abundance and behaviour.

The study discusses and evaluates thoroughly various design alternatives of the architecture covering state-of-the-art methods like Mask R-CNN, Flow-Guided Feature Aggregation (FGFA), three different ResNet variations (3D ResNet, MC ResNet and (2 + 1)D ResNet), and the Slowfast architecture. The evaluation is based on video clips that are captured with 8 frames per second by camera traps using infrared cameras and infrared flash-lights. The clips show deer, boars, foxes and hares - mostly at night time. Given this data material, the combination of Mask R-CNN Video for object detection and (2 + 1)D ResNet for action recognition, yield the best results with an average precision of 63.8% for animal detection and identification. For comparison, Mask R-CNN achieves an average precision of 39.8% on the official COCO challenge dataset (He et al. (2017)). For action recognition the accuracies range between 88.4% and 94.1%.

Future work will include action detection instead of action recognition. This would enable to detect different actions performed simultaneously by different animals. In such a case, action recognition decides based on the number of animals and the position of the animals. Action recognition will reflect the action of the majority of the animals. At the same time, animals in the focus of the scene have a greater influence than animals in the background. The current data material contains only a few videos where animals perform simultaneously different actions in one video. An additional important extension will be the re-identification of recurring individuals for estimations of population sizes and occurrence and co-occurrence frequencies.

References

Brust, C.A., Burghardt, T., Groenenberg, M., Kading, C., Kuhl, H.S., Manguette, M.L., Denzler, J., 2017. Towards automated visual monitoring of individual gorillas in the wild. In: The IEEE International Conference on Computer Vision (ICCV) Workshops.

Burghardt, T., Ćalić, J., 2006. Analysing animal behaviour in wildlife videos using face detection and tracking. IEE Proc. Vision Image Signal Process. 153 (7), 305–312. https://digital-library.theiet.org/content/journals/10.1049/ip-vis_20050052.

Carreira, J., Zisserman, A., 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Chen, R., Little, R., Mihaylova, L., Delahay, R., Cox, R., 2019a. Wildlife surveillance using deep learning methods. Ecol. Evol. 9, 9453–9466. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ece3.5410, doi:https://doi.org/10.1002/ece3.5410, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ece3.5410.

Chen, X., Girshick, R., He, K., Dollar, P., 2019b. Tensormask: A foundation for dense object segmentation. In: The IEEE International Conference on Computer Vision (ICCV).

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T., 2015. Flownet: Learning Optical Flow with Convolutional Networks, in: The IEEE International Conference on Computer Vision (ICCV).

Dutta, A., Zisserman, A., 2019. The VIA annotation software for images, audio and video. In: Proceedings of the 27th ACM International Conference on Multimedia. ACM, New York, NY, USA. https://doi.org/10.1145/3343031.3350535.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. 303–338. https://doi.org/10.1007/s11263-009-0275-4.

Falzon, G., Lawson, C., Cheung, K.W., Vernes, K., Ballard, G.A., Fleming, P.J., Glen, A.S., Milne, H., Mather-Zardain, A., Meek, P.D., 2020. Classifyme: a field-scouting software for the identification of wildlife in camera trap images. Animals 10, 58.

Feichtenhofer, C., Fan, H., Malik, J., He, K., 2019. Slowfast networks for video recognition. In: The IEEE International Conference on Computer Vision (ICCV).

Follmann, P., Radig, B., 2018. Detecting animals in infrared images from camera-traps. In: Pattern Recognition and Image Analysis, pp. 605–611. https://doi.org/10.1134/S1054661818040107.

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. Mask r-cnn. In: The IEEE International Conference on Computer Vision (ICCV).

Hu, Y.T., Huang, J.B., Schwing, A., 2017. Maskrnn: Instance level video object segmentation. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, 30. Curran Associates, Inc., pp. 325–334. URL. http://papers.nips.cc/paper/6636-maskrnn-instance-level-video-object-segmentation.pdf

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T., 2017. Flownet 2 0: Evolution of optical flow estimation with deep networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Jung, A.B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.M., Weng, C.H., Ayala-Acevedo, A., Meudec, R., Laporte, M., et al., 2020. imgaug. https://github.com/aleju/imgaug. Online; accessed 01-Feb-2020.

Kalogeiton, V., Weinzaepfel, P., Ferrari, V., Schmid, C., 2017. Action tubelet detector for spatio-temporal action localization. In: The IEEE International Conference on Computer Vision (ICCV).

Kellenberger, B., Marcos, D., Tuia, D., 2018. Detecting mammals in uav images: best practices to address a substantially imbalanced dataset with deep learning. Remote Sens. Environ. 216, 139–153. http://www.sciencedirect.com/science/article/pii/S0034425718303067. https://doi.org/10.1016/j.rse.2018.06.028.

Körschens, M., Barz, B., Denzler, J., 2018. Towards automatic identification of elephants in the wild. arXiv:1812.04418.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L., 2014. Microsoft coco: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014. Springer International Publishing, Cham, pp. 740–755.

Lin, J., Gan, C., Han, S., 2019. Tsm: Temporal shift module for efficient video understanding. In: The IEEE International Conference on Computer Vision (ICCV).

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M., 2020. Deep learning for generic object detection: a survey. Int. J. Comput. Vis. 128, 261–318.

Meek, P.D., Ballard, G., Claridge, A., Kays, R., Moseby, K., O'Brien, T., O'Connell, A., Sanderson, J., Swann, D.E., Tobler, M., Townsend, S., 2014. Recommended guiding principles for reporting on camera trapping research. Biodivers. Conserv. 23.

Norouzzadeh, M.S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M.S., Packer, C., Clune, J., 2018. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proc. Natl. Acad. Sci. 115, E5716–E5725. URL: https://www.pnas.org/content/115/25/E5716, doi:https://doi.org/10.1073/pnas.1719367115, arXiv:https://www.pnas.org/content/115/25/E5716.full.pdf.

O'Connell, A.F., Nichols, J.D., Karanth, K.U. (Eds.), 2011. Camera Traps in Animal Ecology Methods and Analyses. Springer.

Okafor, E., Pawara, P., Karaaba, F., Surinta, O., Codreanu, V., Schomaker, L., Wiering, M., 2016. Comparative study between deep learning and bag of visual words for wild-animal recognition. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. https://doi.org/10.1109/SSCI.2016.7850111.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, vol. 32. Curran Associates Inc., pp. 8024–8035

Piergiovanni, A., Ryoo, M.S., 2018. Temporal gaussian mixture layer for videos. arXiv:1803.06316.

Ranjan, A., Black, M.J., 2017. Optical flow estimation using a spatial pyramid network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Sudhakaran, S., Escalera, S., Lanz, O., 2019. Gate-shift networks for video action recognition. arXiv:1912.00381.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M., 2018. A closer look at spatiotemporal convolutions for action recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

United Nations, 1992. Convention on Biological Diversity (text with annexes). United Nations, New York.

van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9, 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Verma, G.K., Gupta, P., 2018. Wild animal detection using deep convolutional neural network. In: Chaudhuri, B.B., Kankanhalli, M.S., Raman, B. (Eds.), Proceedings of 2nd International Conference on Computer Vision & Image Processing. Springer Singapore, Singapore, pp. 327–338.

Vora, A., Raman, S., 2018. Flow-free video object segmentation. In: Rameshan, R., Arora, C., Dutta Roy, S. (Eds.), Computer Vision, Pattern Recognition, Image Processing, and Graphics. Springer Singapore, Singapore, pp. 34–44.

Wang, G., 2019. Machine learning for inferring animal behavior from location and movement data. Ecol. Inform. 49, 69–76. http://www.sciencedirect.com/science/article/pii/S1574954118302036. https://doi.org/10.1016/j.ecoinf.2018.12.002.

Willi, M., Pitman, R.T., Cardoso, A.W., Locke, C., Swanson, A., Boyer, A., Veldthuis, M., Fortson, L., 2019. Identifying animal species in camera trap images using deep learning and citizen science. Meth. Ecol. Evol. 10, 80–91. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13099, doi:https://doi.org/10.1111/2041-210X.13099, arXiv:https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13099.

Yang, L., Fan, Y., Xu, N., 2019. Video instance segmentation. In: The IEEE International Conference on Computer Vision (ICCV).

Zeng, R., Huang, W., Tan, M., Rong, Y., Zhao, P., Huang, J., Gan, C., 2019. Graph convolutional networks for temporal action localization. In: The IEEE International Conference on Computer Vision (ICCV).

Zeppelzauer, M., 2013. Automated detection of elephants in wildlife video. EURASIP J. Image Video Process. https://doi.org/10.1186/1687-5281-2013-46.

Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D., 2017. Temporal action detection with structured segment networks. In: The IEEE International Conference on Computer Vision (ICCV).

Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y., 2017. Flow-guided feature aggregation for video object detection. In: The IEEE International Conference on Computer Vision (ICCV).