Deep Learning Techniques for Dragonfly Action Recognition

Martina Monaci[1], Niccolò Pancino[1,2][a], Paolo Andreini[1][b], Simone Bonechi[1][c],
Pietro Bongini[1,2][d], Alberto Rossi[1,2][e], Giorgio Ciano[1,2][f], Giorgia Giacomini[3][g],
Franco Scarselli[1][h] and Monica Bianchini[1][i]

[1] *University of Siena, Department of Information Engineering and Mathematics, via Roma 56, 53100, Siena (SI), Italy*
[2] *University of Florence, Department of Information Engineering, via S. Marta 3, 50139, Florence (FI), Italy*
[3] *University of Siena, Department of Biochemistry and Molecular Biology, via Aldo Moro 2, 53100, Siena (SI), Italy*

Keywords: Dragonfly, Machine Learning, Action Recognition, Deep Learning.

Abstract: Anisoptera are a suborder of insects belonging to the order of Odonata, commonly identified with the generic term dragonflies. They are characterized by a long and thin abdomen, two large eyes, and two pairs of transparent wings. Their ability to move the four wings independently allows dragonflies to fly forwards, backwards, to stop suddenly and to hover in mid–air, as well as to achieve high flight performance, with speed up to 50 km per hour. Thanks to these particular skills, many studies have been conducted on dragonflies, also using machine learning techniques. Some analyze the muscular movements of the flight to simulate dragonflies as accurately as possible, while others try to reproduce the neuronal mechanisms of hunting dragonflies. The lack of a consistent database and the difficulties in creating valid tools for such complex tasks have significantly limited the progress in the study of dragonflies. We provide two valuable results in this context: first, a dataset of carefully selected, pre–processed and labeled images, extracted from videos, has been released; then some deep neural network models, namely CNNs and LSTMs, have been trained to accurately distinguish the different phases of dragonfly flight, with very promising results.

1 INTRODUCTION

Odonata are an order of medium/large hemimetabolous insects, composed of more than 5000 species which differ in color and size. Odonata are morphologically divided into two main infraorders: Zygoptera and Anisoptera. Although they are very similar in structure, they can be easily distinguished by the shape of the wings. In fact, Zygoptera, commonly called "damselflies", are characterized by two pairs of almost identical wings, kept folded at rest, along or above the abdomen. Unlike Zygoptera, forewings of Anisoptera are narrower than the hind wings; they are kept flat and far from the body at rest, and make Anisoptera much more skilled in flight.

Commonly, Anisoptera are also referred to as "Dragonflies". They live mainly in freshwater environments, such as ponds, rivers and lakes. They are characterized by a long and thin body, two large multifaceted eyes — made up of thousands of elementary eyes called ommatidia —, two pairs of transparent wings and six legs. They can move the four wings in a fully independent way. This feature, unique in the world of insects, allows them to reach speeds of up to 50 km/h and to obtain formidable performance in flight and hunting, where they can perform backward movements, very narrow turns of death and stops in mid–air.

Although the biology of dragonflies has been widely surveyed, there are still very few studies on the kinematic analysis of these insects. In 1975, the Swedish biologist Norberg was the first to study their flight by filming a dragonfly in the open field (Norberg, 1975). He measured parameters such as the width and frequency of the wing flapping, revealing that dragonflies keep their body in an almost horizon-

[a] https://orcid.org/0000-0003-2212-4728
[b] https://orcid.org/0000-0002-7790-6818
[c] https://orcid.org/0000-0002-5540-3742
[d] https://orcid.org/0000-0001-9074-0587
[e] https://orcid.org/0000-0003-1688-6961
[f] https://orcid.org/0000-0003-2863-4315
[g] https://orcid.org/0000-0002-2939-7203
[h] https://orcid.org/0000-0002-8206-8142
[i] https://orcid.org/0000-0003-1307-0772

Monaci, M., Pancino, N., Andreini, P., Bonechi, S., Bongini, P., Rossi, A., Ciano, G., Giacomini, G., Scarselli, F. and Bianchini, M.
Deep Learning Techniques for Dragonfly Action Recognition.
DOI: 10.5220/0009150105620569
In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2020)*, pages 562-569
ISBN: 978-989-758-397-1; ISSN: 2184-4313
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

tal position during flight. A decade later, Azuma *et al.*, using a more advanced video camera, showed that the flaps of the dragonfly's wings follow a trajectory which can be well represented by a sinusoidal function (Azuma et al., 1985), thus confirming the vortex theory, postulated since 1979. Moreover, by collecting both morphological and kinematic data, they were able to define the first mathematical expression of the wing speed. Since then, numerous experiments have been carried out in this particular research context. New technologies (Wang et al., 2003) were exploited, for instance, to analyze the muscle movements during flight (Faller and Luttges, 1991a; Faller and Luttges, 1991b), in order to produce prototypes of robotic drones capable of accurately simulating the flight of a dragonfly (Couceiro et al., 2010). These simulations, however, failed to fully reproduce the dexterity, capacity, flexibility and freedom of maneuver of dragonflies (Hu et al., 2009).

This paper aims at creating an *action recognition* model capable of distinguishing the different phases of the dragonfly flight, using deep learning techniques. Given the wide interest of the biological research community in the study of dragonflies[1], we assume that it is quite useful to develop a reliable system for recognizing dragonfly actions. Indeed, research in different fields could benefit from the recognition system to test hypotheses on dragonfly anatomy, flight dynamics and predatory behaviors.

The term Deep Learning (DL) refers to neural network models composed by many layers, arranged in a cascade, so that the information fed to the network is processed thoroughly, layer after layer. Despite their considerable computational complexity, these models have proved to be particularly flexible for different tasks, such as object recognition in images and video, speech recognition, up to the classification of phenomena of the animal world (Trnovszky et al., 2017), and in particular of insects (Stern et al., 2015). Deep learning, and specifically Convolutional Neural Networks (CNNs), is actually emerging as the leading machine learning tool in computer vision. Recently, deep learning has achieved impressive results in several visual recognition tasks, often outperforming classical image processing approaches. CNNs are currently broadly employed in different domains that range from object classification and detection (Krizhevsky et al., 2012; He et al., 2016; Szegedy et al., 2017), to semantic segmentation (Donahue et al., 2015; Zhao et al., 2017; Andreini et al., 2018) and medical image analysis (Spanhol et al., 2016; Pereira et al., 2016; Andreini et al., 2019; Anthimopoulos et al., 2016; Milletari et al., 2016). Despite huge efforts made by the research community in developing methods able to reduce the amount of data needed for training DL architectures (Khoreva et al., 2017; Bonechi et al., 2018; Shu et al., 2018; Bonechi et al., 2019a), the success of state–of–the–art CNNs is usually based on the availability of large sets of supervised images. For this reason, the release of public benchmarks to develop and evaluate new algorithms is becoming increasingly critical (Deng et al., 2009; Lin et al., 2014; Bonechi et al., 2019b; Bonechi et al., 2019c). Just as CNNs significantly increased the performance in computer vision tasks, a comparable success has been obtained by Recurrent Neural Networks (RNNs) in the analysis of sequential data. Long Short–Term Memory (LSTM) networks are one of the most successful RNN models, designed to reduce the problems caused by long–term dependencies and vanishing gradients. LSTMs are currently broadly employed in tasks involving the analysis of sequential data, which range from Natural Language Processing (NLP) (Graves and Jaitly, 2014; Sutskever et al., 2014; Kowsari et al., 2019) to human mobility (Rossi et al., 2019) and music composition (Sturm et al., 2016). The integration of CNNs and RNNs allows to take advantage from the characteristic of both the models, and has been successfully applied to tasks like video classification and image captioning (Donahue et al., 2015; Vinyals et al., 2015).

In this work, we propose a dragonfly action recognition system capable of classifying video frames in five classes: take–off, flight, landing, stationary and absent (frames in which the dragonfly is not present). Deep learning requires a huge set of fully annotated data, but, unfortunately, we are not aware of a publicly available labeled dataset of dragonfly images. To train a deep learning architecture, we first collected a suitable number of samples from online videos, which were appropriately preprocessed and labeled frame by frame. Then, different classifier networks for action recognition were compared. First of all, a standard convolutional neural network was tested: this model elaborates one frame at a time, discarding the information of previous frames. To correctly identify the action, the information contained in the previous frame could be fundamental. Therefore, we also trained an LSTM model, which is capable of elaborating frame sequences.

The paper is organized as follows. In Section 2, the dataset construction is described. Then, in Section 3, the main architectural components responsible

[1]Odonata are very ancient animals. Fossils dating back to the Permian period have been found, coming from approximately 285 million years ago; dragonfly–like fossils date back 300 million years (Carboniferous period), about 40 million years before the appearance of dinosaurs.

for processing (sequences of) images are briefly introduced, as well as the experimental settings and the obtained results, comparing the performance of different architectures and providing a detailed analysis of the problems encountered during their implementation. Finally, Section 4 draws some conclusions and proposes future works.

2 DRAGONFLY DATASET COLLECTION

Generally, datasets used to study insects are composed of pre–processed videotapes recorded by researchers or by specialized personnel with ad hoc equipments. Nevertheless, for this project, it was not possible to proceed in the usual way, mainly due to two reasons: first of all, we did not have access to adequate recording tools, since qualitatively valid videos require a suitable equipment, such as a stable tripod, high resolution video cameras, powerful zoom, which were not available. Last but not least, the speed and the "nomadic" nature of the dragonfly make the process of positioning the shooting instruments extremely complex. Under these circumstances, we decided to build a dataset using dragonfly videos readily available online. Although this kind of set–up may seem more superficial than direct recording, it offers the substantial advantage of the variability in setting and recorded specimens, thus being potentially useful to improve the generalization capabilities of machine learning models.

Therefore, about 40 videos have been downloaded and converted, through a frame–by–frame decomposition, into several image sequences. In particular, the sequence generation process was composed of three phases. First, it was verified that the number of frames per second was comparable for all the videos and, actually, we found that they were constituted by 25 to 30 fps, so no further data pre–processing was necessary. Then the speed of each video was standardized, i.e. the slow–motion videos were "speeded up". Due to the documentary nature of the videos — composed of several independent sequences related to different specimens of dragonflies, recorded in different phases of flight — in the last phase, several flight sequences were extrapolated.

The creation of classifiers, based on supervised learning, requires the availability of a consistent set of labeled data, that is, in which a specific target class is associated with each example. For this reason, a class label has been manually associated with each frame of the dataset. In particular, five classes have been identified: three of them describe the different phases of the dragonfly flight, namely *take–off*, *flight* and *landing*, together with two more classes which refer to the dragonfly states immediately preceding or following its movement, such as *stationary* and *absent*. From the original set of videos, about 400 distinct sequences were manually identified, corresponding to a hundred targeted sub–sequences, for a total of about 61K frames.

Most of the data — approximately 77% of the images — belongs to the stationary class, probably due to the aforementioned difficulty in capturing dragonflies during their flight, making the dataset strongly unbalanced. This issue could compromise the process of learning, because the model tends to follow the prior probability, assigning all the examples to the most common class and ignoring the others. Since the dataset is intended for the use in machine learning applications, we decided to pre–process the data, randomly removing about 80% of the frames labeled as *stationary* (also getting rid of many sequences entirely composed of such images). By doing so, a final dataset of about 300 sequences was obtained, for a total of about 23k images: more precisely, 41% of the images belongs to the stationary class, 12% to the absent class, 4% to the take off class, 8% to the landing class, and 35% to the flight class[2].

The DragonFly dataset is composed of strongly heterogeneous images, from the recorded setting to the aspect ratio or the resolution of the tapes. For this reason, the images have been further processed by removing noisy elements — such as logos or text — and by scaling them to a fixed square size of 224×224 pixels, a format often used in convolutional networks (Simonyan and Zisserman, 2014) running on a standard computer. Since the aspect ratio of all videos was not 1:1, images have been resized to have the longest horizontal side at a fixed length, maintaining unchanged the original aspect ratio. Given an image with width x and height y, the new dimensions x' and y' are computed according to:

$$\begin{aligned} x' &= 224 \\ y' &= x \cdot \frac{224}{y} \end{aligned} \tag{1}$$

which means that the longest dimension x has been scaled to 224 pixels, while y has been adapted accordingly. Therefore, in order to obtain square images, a padding operation was performed, inserting two bands of the same thickness, composed of black pixels, above and below the image, as shown in Fig.1. This technique is very common in image processing, since the insertion of rows and columns coded with constant values for all the images of the dataset is

[2]The Dragonfly dataset is available upon request.

easily learnable and, therefore, does not constitute a disturbing element for the training [3].

Images are represented in the RGB color space, where each color is coded by three values corresponding to the red, green and blue channels. In conclusion, the size of a generic input image is $224 \times 224 \times 3$, where the last dimension is given by the three RGB channels associated with each pixel.

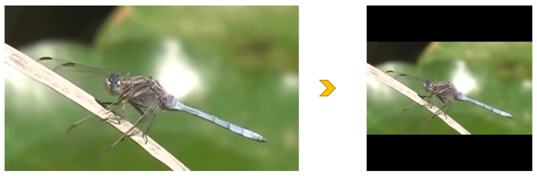
Figure 1: Example of the dataset transformation: On the left, the raw image, with size 640×360 pixels; on the right, the final image, with size 224×224, after resizing and padding.

3 EXPERIMENTAL SETUP AND RESULTS

To develop a dragonfly action recognition method, we used two types of neural networks: CNNs and LSTMs. Both models take color images as input data, with size $224 \times 224 \times 3$ as described in Section 2, with normalized values in the range $[-1, 1]$. The models were trained with Adam optimizer (Kingma and Ba, 2014), based on the cross–entropy loss function. In the following, the classes are referred to by their initials:

- A: Absent;
- S: Stationary;
- TO: Take–Off;
- L: Landing;
- F: Flight.

All the procedures described in this section have been entirely developed in Python, with the help of the Keras API (TensorFlow backend) for the realization of the neural architectures and the analysis of the obtained results (Rossum, 1995; Chollet, 2015). The experiments were carried out in a Linux environment on a single NVIDIA GeForce GTX 1080 with 8 GB GDDR5X.

Initial tests were performed using CNNs. This kind of network is typically used to process only one image at a time. In fact, it is not designed to deal with video analysis: CNNs only extract features from the individual frames, without capturing the temporal correlation between the elements of a sequence. Although this task is well suited for LSTM models — they can make predictions based on time series data — the use of CNNs is fundamental because it allows to extract relevant features from each of the frames. Two methods based on CNNs are proposed in the following, based respectively on architectures which were built from scratch and on pre–trained models.

[3]Actually, padding allows also to design deeper networks and improves performance by keeping information at the borders. The main drawback of padding is to increase computational time and memory consumption. Possible artifacts can be avoided by using mirror padding.

3.1 Classification via non Pre–trained CNNs

In this experiment, we tried to devise an ad hoc model for the dragonfly action recognition task. Several architectures, with different depths, were tested. All the networks share the functional and structural characteristics described below:

- Weights are randomly initialized using a Standard Normal Distribution;
- The convolutional layers have a kernel size of 3 and a stride equal to 1 pixel;
- The layers are grouped in blocks of two, with a number of filters which is doubled from a block to the next one;
- At the output of each convolutional layer, a batch–normalization function and a ReLu activation function are applied;
- After each block, a max–pooling procedure is applied on non–overlapping 2×2 regions of the feature maps;
- A flatten or global max 2D pooling function is applied at the end of the convolutional layer sequence, in order to transform a multidimensional structure into a vector;
- Two fully connected layers are used to process the results of the global max–pooling, the latter having a softmax activation function, obtaining the final output of the network.

A first set of experiments was designed with the aim of identifying a good combination of hyperparameters, such as the number of convolutional blocks or the number of feature maps. The best performing architecture is shown in Fig.2. It is based on 5 convolutional blocks associated with 16, 32, 64, 128 and 256 feature maps, respectively. Its predictions reached an accuracy of 77% on the validation set and 71% on the test set.

Table 1 shows the confusion matrix related to the test set, as well as the performance on each class. This

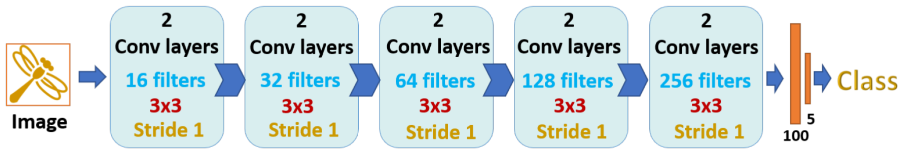

Figure 2: Best non pre–trained CNN architecture. The input image is processed by five convolutional blocks (in light blue) and then by two fully connected layers (in orange).

Table 1: Confusion Matrix and Accuracy of a non pre–trained CNN. The generic element m_{ij} represents the number of images belonging to the i–th class misclassified as belonging to the j–th class. The number of correct predictions associated with each class is then found on the main diagonal. The last column shows the accuracy on each class.

	A	S	TO	L	F	Accuracy
A	131	10	0	29	119	45,3%
S	36	1518	65	15	179	83,7%
TO	3	61	12	0	107	6,6%
L	6	67	15	37	73	18,7%
F	19	107	42	6	659	79,1%

network can classify the images belonging to classes S and F with high accuracy, while it has more difficulties in recognizing class A. Furthermore, as we can see from the results, this model is completely unable to correctly distinguish elements belonging to class TO and L. As expected, given the nature of this network, which processes one image at a time, it is challenging for it to distinguish *take–off* images from *landing* ones, and *flight* images from *stationary* ones.

3.2 Classification via CNNs and Transfer Learning

Transfer learning is a machine learning technique where a network is pre–trained on a task and then used as a starting point for a new model, which is trained to solve another task, not necessarily related to the first one. Some form of correlation between the two problems, though, guarantees better results. The main advantage of this technique is the reduction in training times, because the second model weights are initialized according to the knowledge acquired during the first training, rather than at random. For the same reason, the minimum number of examples required for training the second model is reduced. These two advantages are often fundamental for deep networks, which require vast amounts of data and long times to be properly trained.

Three CNN models, which are well known in literature, were used for transfer learning: the MobileNet (Sandler et al., 2018), the VGG16 (Simonyan and Zisserman, 2014) and the DenseNet121 (Huang et al., 2016). All of them are pre–trained on the ImageNet dataset (Deng et al., 2009). Two different transfer learning methods were adopted, which represent two different degrees of "preservation" of the knowledge acquired on ImageNet. The first method consists in re–training only the last (fully–connected) layers of the CNN, while the weights inherited from the original network are kept unchanged. In this experiment, the convolutional section of the network behaves like a feature extractor, while the fully connected layers allow the network to deal with the new task. The second strategy, called *fine tuning*, consists in adapting all the weights of the original CNN model by re–training the entire network on new data. The latter (see Fig.3) showed better performance, correctly classifying 69% of images belonging to the test set.

The performance on the classes S and F are satisfactory, while there is a certain difficulty in recognizing the other three classes (see Table 2). As mentioned in Section 3.1, this result was expected, considering the frame–by–frame classification logic of a CNN.

Table 2: Confusion Matrix and Accuracy of the transfer–learning–based CNN.

	A	S	TO	L	F	Accuracy
A	131	188	41	14	1	34,9%
S	110	1553	5	49	96	85,7%
TO	14	136	9	1	23	4,9%
L	27	77	6	21	67	10,6%
F	66	120	10	4	633	76,0%

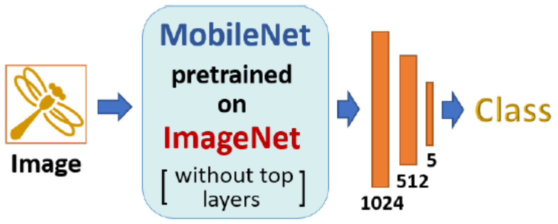

Figure 3: Fine–tuning–based CNN architecture. The input image is processed by the MobileNet (pre–trained on ImageNet, in light blue), and then by three fully connected layers (in orange) composed by 1024, 512 and 5 neurons, respectively.

3.3 Recurrent Classification with LSTMs

The basic idea is to process a video sequence as it is, rather than analyzing its frames separately. This aspect is fundamental since each frame is inevitably related to the others by a cause–effect relationship. Indeed, taking into account the information coming from adjacent images is useful to obtain a more adequate recognition of the different phases of flight.

Since the recurrent layers are not able to process multidimensional data, a feature extractor network — composed of replicas of the same MobileNet model showed in Fig.3 — has been inserted upstream, followed by a flattening–merge mechanism to produce one–dimensional data, compatible with the recurrent layers of the LSTM model. The resulting composite network, shown in Fig.4, is a hybrid "CNN+LSTM" model, and represents the architectural prototype of the second classification approach. The LSTM network is characterized by two dense layers composed of 100 and 5 neurons, respectively. To analyze the entire sequence, a sliding window with a size of 7 frames was used. A supersource transduction was carried out in the LSTM layers, meaning that the output propagated to the final dense layers comes from the last frame of the analyzed subsequence (i.e. the seventh frame).

Two different strategies for providing sequences to the network were adopted in the training process. The first consists in training on sub–sequences obtained by extracting the sliding windows in each sequence of the training set, so as to obtain more overlapping sub–sequences (*sequential* approach, Fig.5). The second strategy is based on a *random* selection of sub–sequences from randomly selected sequences. The results for both the approaches are shown in Tables 3 and 4, respectively. It is worth noticing that the sequential approach reaches an accuracy of 73,9%, while the random one got 68,8%.

As shown by these results, the random approach does not seem to bring advantages to classification, while the network trained with a sequential approach can appropriately manage the information coming from temporal coherence. Unfortunately, once again, the results improve only for classes A, S and F, while performance on classes S and TO are really low. These latter classes are inherently difficult to be distinguished and, in addition, sequences belonging to these two classes are only a small subset of the dataset, making the learning process even harder.

Table 3: Confusion Matrix and Accuracy of the LSTM–based architecture for sequential frames.

	A	**S**	**TO**	**L**	**F**	**Accuracy**
A	180	12	8	0	77	65,0%
S	70	1299	1	11	168	83,9%
TO	1	136	2	0	44	1,1%
L	10	64	0	0	112	0,0%
F	33	36	0	0	740	91,5%

Table 4: Confusion Matrix and Accuracy of the LSTM–based architecture for randomly selected frames.

	A	**S**	**TO**	**L**	**F**	**Accuracy**
A	211	12	17	2	35	76,2%
S	124	1202	113	19	91	77,6%
TO	21	107	23	9	23	12,6%
L	23	64	7	13	79	7,0%
F	48	94	46	2	619	76,5%

4 CONCLUSIONS

This paper proposes some deep learning approaches to dragonfly action recognition from images/videos (see Table 5, in which performance and time per epoch are summarized for all the analyzed models). In particular, apart from the collection of a large labeled dataset, some guidelines for the calibration, design and implementation of deep models to face this task have been provided.

Table 5: Performance for all the above analyzed deep architectures.

Tested Networks	**Accuracy**	**Time per epoch**
CNN 3.1	71%	30 sec
CNN 3.2	69%	30 sec
LSTM 3.3 Seq	73,9%	5 min
LSTM 3.3 Rand	68,8%	10 min

It will be a matter of future research to improve the classification performance of the proposed models, for instance by collecting a larger dataset — in particular providing more frames for the take–off/landing classes — or employing data augmentation techniques in order to extend the available data. A further improvement could be brought by the introduction of more pre–processing operations, compatible with the data type, in order to reduce the disturbing elements in the images and to facilitate the classification task.

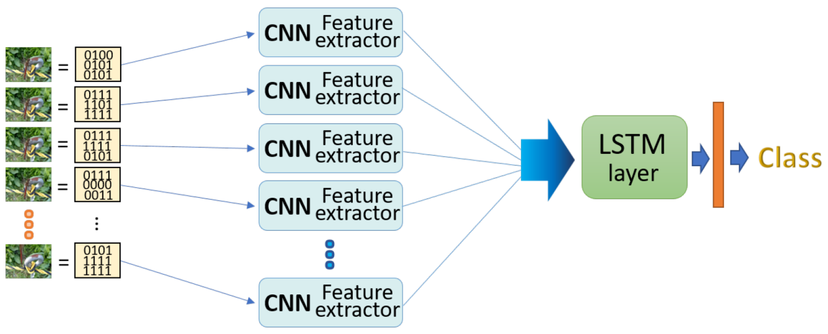

Figure 4: Architecture of the CNN + LSTM hybrid model. The images are converted into $224 \times 224 \times 3$ matrices and processed separately by replicas of the same CNN model (in blue). The results are then processed by the LSTM layer (in green). Finally, dense layers (in orange) re–elaborate the information and perform the classification.

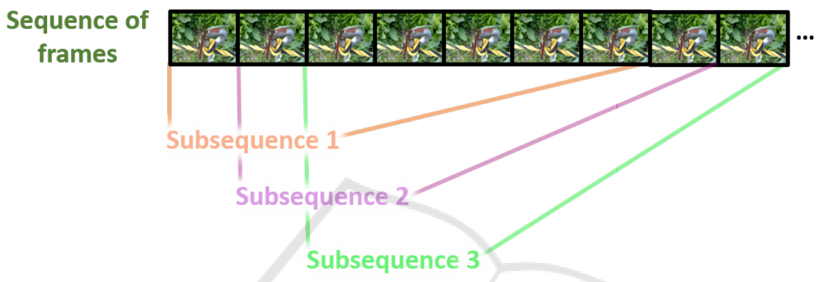

Figure 5: Sequential sub–sampling of frames for training the LSTM–based network.

REFERENCES

Andreini, P., Bonechi, S., Bianchini, M., Mecocci, A., and Scarselli, F. (2018). A deep learning approach to bacterial colony segmentation. In *International Conference on Artificial Neural Networks*, pages 522–533. Springer.

Andreini, P., Bonechi, S., Bianchini, M., Mecocci, A., Scarselli, F., and Sodi, A. (2019). A two stage GAN for high resolution retinal image generation and segmentation. *ArXiv*, abs/1907.12296.

Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., and Mougiakakou, S. (2016). Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE Transactions on Medical Imaging*, 35(5):1207–1216.

Azuma, A., Azuma, S., Watanabe, I., and Furuta, T. (1985). Flight mechanics of a dragonfly. *Journal of Experimental Biology*, 116(1):79–107.

Bonechi, S., Andreini, P., Bianchini, M., Pai, A., and Scarselli, F. (2019a). Confidence measures for deep learning in domain adaptation. *Applied Science*, 9(11):2192.

Bonechi, S., Andreini, P., Bianchini, M., and Scarselli, F. (2018). Generating bounding box supervision for semantic segmentation with deep learning. In *LNCS 11081*, pages 190–200. Springer.

Bonechi, S., Andreini, P., Bianchini, M., and Scarselli, F. (2019b). COCO_TS dataset: Pixel–level annotations based on weak supervision for scene text segmentation. In *LNCS 11729*, pages 238–250.

Bonechi, S., Andreini, P., Bianchini, M., and Scarselli, F. (2019c). Weak supervision for generating pixel-level annotations in scene text segmentation.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Couceiro, M., Ferreira, N., and Tenreiro Machado, J. (2010). Modeling and control of a dragonfly-like robot. *Journal of Control Science and Engineering*, 2010.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large–Scale Hierarchical Image Database. In *CVPR09*.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long–term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

Faller, W. E. and Luttges, M. W. (1991a). Flight control in the dragonfly: a neurobiological simulation. In *Advances in Neural Information Processing Systems*, pages 514–520.

Faller, W. E. and Luttges, M. W. (1991b). Recording of simultaneous single–unit activity in the dragonfly ganglia. *J. Neurosci. Methods*, 37(1):55–69.

Graves, A. and Jaitly, N. (2014). Towards end–to–end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of* *the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hu, Z., McCauley, R., Schaeffer, S., and Deng, X. (2009). Aerodynamics of dragonfly flight and robotic design. *2009 IEEE International Conference on Robotics and Automation*, pages 3061–3066.

Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Khoreva, A., Benenson, R., Hosang, J., Hein, M., and Schiele, B. (2017). Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 876–885.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing systems*, pages 1097–1105.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V–net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE.

Norberg, R. (1975). Hovering flight of the dragonfly Aeschna Juncea l., kinematics and aerodynamics. In *Swimming and Flying in Nature*, volume 2, pages 763–781.

Pereira, S., Pinto, A., Alves, V., and Silva, C. A. (2016). Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251.

Rossi, A., Barlacchi, G., Bianchini, M., and Lepri, B. (2019). Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Transactions on Intelligent Transportation Systems*.

Rossum, G. (1995). Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

Shu, R., Bui, H., Narui, H., and Ermon, S. (2018). A DIRT–T approach to unsupervised domain adaptation. In *International Conference on Learning Representations*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large–scale image recognition. *CoRR*, abs/1409.1556.

Spanhol, F. A., Oliveira, L. S., Petitjean, C., and Heutte, L. (2016). Breast cancer histopathological image classification using convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2560–2567. IEEE.

Stern, U., He, R., and Yang, C.-H. (2015). Analyzing animal behavior via classifying each video frame using convolutional neural networks. *Scientific Reports*, 5:14351.

Sturm, B., Santos, J., Ben-Tal, O., Korshunova, I., et al. (2016). Music transcription modelling and composition using deep learning. *arXiv:1604.08723*.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. *Advances in NIPS*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception–v4, inception–resnet and the impact of residual connections on learning. In *Thirty–First AAAI Conference on Artificial Intelligence*.

Trnovszky, T., Kamencay, P., Orjeek, R., Benco, M., and Sykora, P. (2017). Animal recognition system based on convolutional neural network. *Advances in Electrical and Electronic Engineering*, 15.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Wang, H., Zeng, L., Liu, H., and Yin, C. (2003). Measuring wing kinematics, flight trajectory and body attitude during forward flight and turning maneuvers in dragonflies. *Journal of Experimental Biology*, 206(4):745–757.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890.