Deep Rearing: Mice Behaviour Analysis in Open Field Test using CNN

Kostya Sebov
Stanford University

ksebov@stanford.edu

Abstract

Existing software solutions used to analyze overhead video recording of mice movement in an Open Field test rely on traditional computer vision-based image analysis. While quite successful in providing basic behavioral metrics, face significant difficulties in analyzing more complex patterns. In addition, they require elaborate setup and configuration to work properly. Machine-learning and convolutional neural networks (CNN) specifically promise to be less susceptible to these limitations and provide fully automated workflow, accommodating to wider margin of image resolution and quality. This project is an attempt to detect 6 types of events of mice behaviour in the context of an open field test, paying particular attention to detecting rearing, or standing on rear limbs using pre-trained generic CNN with minimal training.

1. Introduction

Drug development requires testing of novel compounds in animal models of human disease. The Open Field is one of the most known primary behavioral tests. It is a useful tool for assessing impairment in animal models of neuromuscular disease and efficacy of therapeutic drugs that may affect locomotion and/or muscle function. During the test, movements of a mouse is analyzed for basic metrics like distance traveled or interior/exterior preference as well as more complex patterns, like interacting with objects or standing on rear limbs (rearing).

Traditionally, such data collection was performed using expensive specialized hardware that use intersecting infrared beams to provide real-time 2D and 3D localization of animal and related software [1]. Much easier and more cost-effective ways to collect such data is to record overhead videos for further analysis. Video analysis software solutions on the market, like TopScan by CleverSys [2] rely on traditional computer vision-based image analysis and, hence require elaborate manual marking up of image areas before the analysis can be conducted. While quite successful in providing overall behavioral metrics, face significant difficulties in analyzing more complex patterns. In order to detect rearing events it requires the mouse to place 2 paws on the outside wall on the 2 outside walls of the enclosure. Besides being unable to detect rearing near inner walls it's also very demanding to the video quality and resolution.

Figure 1: Gathering video in Open Field test and classification events

This project is a proof of concept to assess feasibility of using machine learning algorithms in providing similar or better analysis features. It was implemented in Tensorflow 1.1 [3] and uses ImageNet-trained SqueezeNet network [4] with final classifier (1000-conv2d/relu/avg-pool) layers are replaced with similar 256-feature-computing layers, followed by 2 fully-connected layers for final event classification into 6 classes: empty field, walking, grooming, looking up, rearing on the wall and rearing freely (Figure 1).

The results demonstrate that generic pre-trained CNNs allow static classification without temporal features with minimal training and significantly high precision even without CNN layers fine tuning, while the latter noticeably improve such precision even further.

2. Related Work

The majority of existing video analysis research uses traditional computer vision techniques to extract features from video frames and only uses machine learning algorithms to analyze the latter.

In [5] authors present a complex integrated hardware and software system that combines synchronized top view video tracking with 3D depth sensing and traditional computer vision algorithms to extract 27 features then uses various machine learning approaches for automatic detection and quantification of social behaviors involving close and dynamic interactions between two mice of different coat colors in their home cage.

[6] groups together several open source projects into one toolbox and yet none of the tools use CNNs to perform low-level image analysis.

A software system presented in [7] uses both convolution and morphology methods to extract as many as 3720 features per frame in the context of mice behaviour analysis. The software also allows temporal factors to be used to classify complex behaviour patterns.

3. Dataset

For training and evaluation purposes open field video file (open_field_vid1.avi) available from [6] was used.

3.1. Labeling:

The video has been manually labeled at a frame level into 5 types:

- Empty
- Looking up
- Grooming
- Rearing on the wall
- Rearing freely

Unlabeled frames were assigned a default "Walking" class.

Although not required to differentiate between rearing on the wall—most common case then the animal is touching the wall with its front paws—and when it's simply standing up, we analyzed these as two classes.

Duration	10 min 55 sec
Resolution	848x480
Format	MPEG-4 (XVID)
FPS	29.97
Frames	19000+

Table 1: Dataset key characteristics

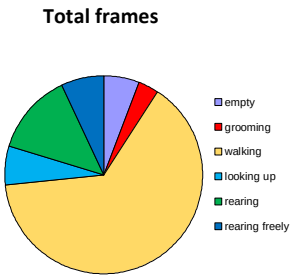


Figure 2: Total number of frames per event type

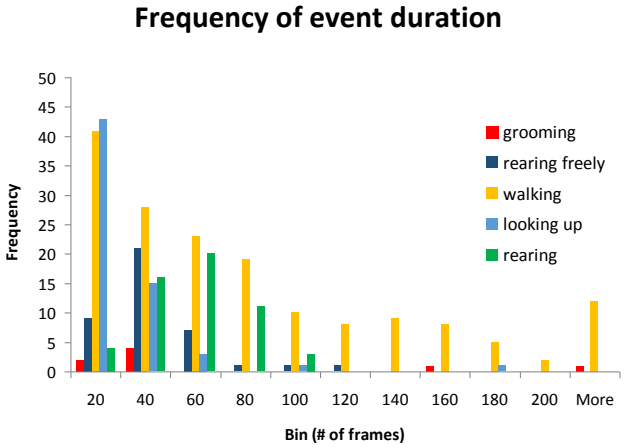


Figure 3: Frequency of event duration

Since some fames can be ambiguous to classify even for human we assumed some level of error and in order to make the data usable for future temporal analysis we limited the duration of each individual event to no less than 10 frames (1/3 of a second).

Using OpenCV [8], the video frames have been converted to grayscale, cropped to the relevant region and resized to 224x224 resolution, as expected by ImageNet-classifier [9] and have been saved along with event directory and index into HDF5 [10] database file.

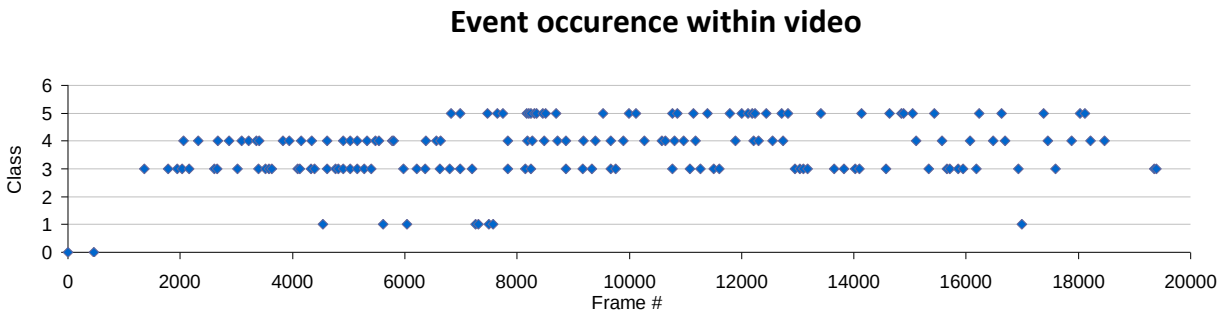


Figure 4: Event occurrence within video

Figure 5: Case of "mice-classification": Failure by pre-trained SqueezeNet to classify stock mice images

3.2. Dataset augmentation

In order to achieve higher variability in our training data the frames have been subjected to the following modifications:

- Rotation by 0, 90, 180 and 270 degrees
- Mirror Image
- Negative image

Therefore each frame has been used 16 times in total—either for training or testing/validation.

4. Architecture

4.1. Knowledge transfer with SqueezeNet

In order to avoid excessive need of training on our limited set of data it was decided to take advantage of an ImageNet-pre-trained CNN. Homework assignment #3 of Stanford cs231n course provided a variant of SqueezeNet-style network [11] implemented in Tensorflow along with the checkpoint containing weights already pre-trained on ImageNet dataset.

Interestingly enough, the network was not capable of correctly classifying frames from our data set as well as stock images of mice as shown in Figure 5.

4.2. Preprocessing layers

Besides the layers adopted from the cs231n/hw3 the network consists of several initial layer designed to:

- Convert the grayscale database images to the rgb expected by the CNN
- Implement data augmentation by rotating, flipping and inverting input images according to augmentation parameter
- Image normalization from SqueezeNet training set mean and standard deviation

4.3. SqueezeNet layer modifications

Final 2 layers have been modified to produce 256 image features instead of 1000 classification scores.

4.4. Simple classifier

On top a simple linear classifier has been added with 1 fully connected layer with relu activation and final 6-class fully-connected layer with no activation. The classifier also used 2 dropout layer providing regularization.

In the future work it is planned to replace this classifier with several RNN layers to account for temporal event characteristics.

Name	**Layer**	**Output size**
Input		224x224x1
Convert to rgb	stack	224x224x3
Augmentation	rotation+flip+negative	224x224x3
Normalization	-mean /std	224x224x3
Cs321n hw3	conv3x3/2x64+relu	111x111x64
	maxpool 3x3/2	55x55x64
	fire 16	55x55x128
	fire 16	55x55x128
	maxpool 3x3/2	27x27x128
	fire 32	27x27x265
	fire 32	27x27x256
	maxpool 3x3/2	13x13x256
	fire 48	13x13x384
	fire 48	13x13x384
	fire 64	13x13x512
	fire 64	13x13x512
Image features	conv1x1x256+relu	13x13x256
	global average	256
Simple classifier	dropout	256
	fc256x256+relu	256
	dropout	256
	fc256x6	6
	softmax/cross-entropy	6
Loss	weighted sum	1
Evaluation	confusion matrix	6x6
	accu/precision/recall/f1	1+1+1+1

Table 2: Neural network layers

4.5. Loss and evaluation layers

For each sample in the minibatch softmax with sparse cross-entropy has been used. In order to account for highly skewed data set the final batch-wide loss used sum of individual sample losses weighted by inverse of the probability of the ground truth sample according to the following formulas:

$$Loss_{batch} = \sum_{i \in batch} \frac{1}{p(y_i)} CE(\mathbb{I}\{i = y_i\}, pred(i))$$

$pred(i) \in \mathbb{R}^C$ is a network's softmax prediction for an ith sample in the batch

y_i is a ground truth class label of the ith sample in the batch

$\mathbb{I}\{i = k\}$ is a one-hot vector a the position k

$CE(X, Y) = -\sum_{c=0}^{C-1} X_c log(Y_c)$ is a cross entropy of the two vectors $X, Y \in \mathbb{R}^C$

$p(c) = \frac{N_c}{N}$ is probability of frame labeled with class c occuring in the data set

N_c and N is a total number of frames labeled with class c

$N = \sum_{c=0}^{C-1} N_c$ is a total number of frames in the data set

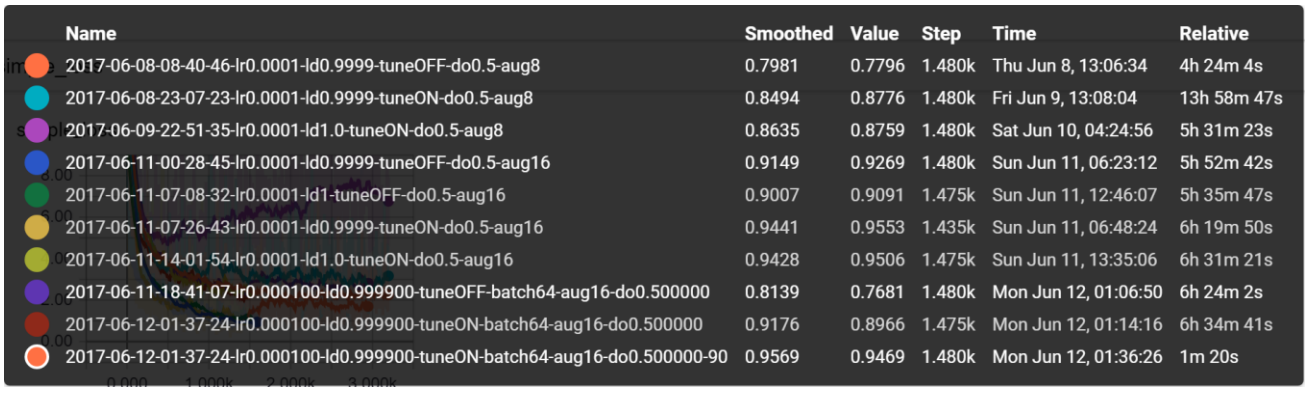
For immediate network performance during learning we used absolute batch loss value as well as exact match accuracy.

Periodically for validation set evaluation as well as for final test performance we used confusion matrix for all 6 classes as well as F1 score for 2 rearing classes 4 and 5, where we considered positive outcome when either or those classes were predicted.

4.6. Experiments

4.7. Dataset partitioning

The whole dataset has been randomly partitioned 80/10/10 between training validation and testing. Furthermore to avoid highly correlated successive video frames, 270 degree rotation has been specifically reserved for validation and testing subsets.



Name	Smoothed	Value	Step	Time	Relative
2017-06-08-08-40-46-lr0.0001-ld0.9999-tuneOFF-do0.5-aug8	0.7981	0.7796	1.480k	Thu Jun 8, 13:06:34	4h 24m 4s
2017-06-08-23-07-23-lr0.0001-ld0.9999-tuneON-do0.5-aug8	0.8494	0.8776	1.480k	Fri Jun 9, 13:08:04	13h 58m 47s
2017-06-09-22-51-35-lr0.0001-ld1.0-tuneON-do0.5-aug8	0.8635	0.8759	1.480k	Sat Jun 10, 04:24:56	5h 31m 23s
2017-06-11-00-28-45-lr0.0001-ld0.9999-tuneOFF-do0.5-aug16	0.9149	0.9269	1.480k	Sun Jun 11, 06:23:12	5h 52m 42s
2017-06-11-07-08-32-lr0.0001-ld1-tuneOFF-do0.5-aug16	0.9007	0.9091	1.475k	Sun Jun 11, 12:46:07	5h 35m 47s
2017-06-11-07-26-43-lr0.0001-ld0.9999-tuneON-do0.5-aug16	0.9441	0.9553	1.435k	Sun Jun 11, 06:48:24	6h 19m 50s
2017-06-11-14-01-54-lr0.0001-ld1.0-tuneON-do0.5-aug16	0.9428	0.9506	1.475k	Sun Jun 11, 13:35:06	6h 31m 21s
2017-06-11-18-41-07-lr0.000100-ld0.999900-tuneOFF-batch64-aug16-do0.500000	0.8139	0.7681	1.480k	Mon Jun 12, 01:06:50	6h 24m 2s
2017-06-12-01-37-24-lr0.000100-ld0.999900-tuneON-batch64-aug16-do0.500000	0.9176	0.8966	1.475k	Mon Jun 12, 01:14:16	6h 34m 41s
2017-06-12-01-37-24-lr0.000100-ld0.999900-tuneON-batch64-aug16-do0.500000-90	0.9569	0.9469	1.480k	Mon Jun 12, 01:36:26	1m 20s

Figure 6: Tensorboard visualization of various hyperparameter combination runs

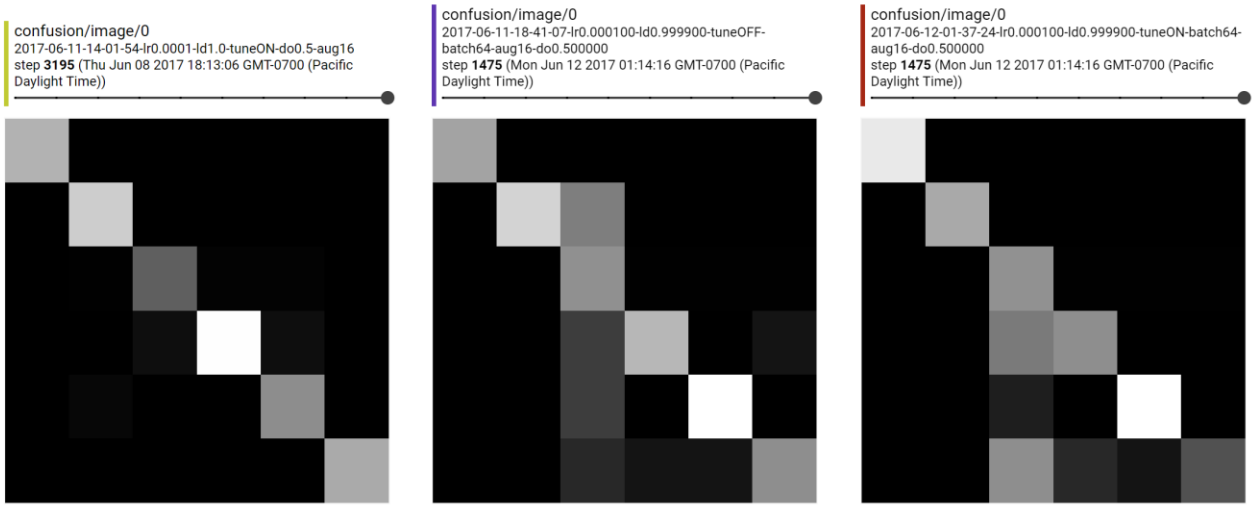


Figure 7: Tensorboard visualization final confusion matrices of best 3 models

All training examples were subjected to all 16 combinations of augmentation transformations.

During training minibatch samples were randomly selected from the whole range of data with the same minibatch cycled through all augmentation combinations (except the ones reserved for validation/testing)

After every 5 training/augmentation minibatch sets a cumulative F1 score and confusion matrix have been computed for the whole validation subset though batches of 256 samples. These metrics were recorded in Tensorboard [12] summaries for monitoring and analysis.

4.8. Hyperparameters

For training we used Adam optimizer with minibatch size of 64 and initial learning rate of 1e-4. The weights were initialized with default Xavier initializer, except for the SqueezeNet layer, which have been loaded from the downloaded checkpoint. 0.5 has been used for dropout regularization.

We evaluated the following combination of hyperparemeter settings:

- Fine-tune Squeezenet Layers: ON or OFF
- Number of epochs: 6 or 12
- Learning rate decay: 0.9999 or 1 (no decay)
- Number of agmentations: 8 (no negative) or 16

Figure 6 and 7 illustrate the progress of traing using various combination of the above hyperparameters and the final confusion matrices respectively.

4.9. Training runs

Training runs involving SqueezeNet layers fine-tuning have been performed on an instance of a Google Cloud with NVIDIA Tesla K80 GPU with 12GB of available RAM at a rate of roughly 1 16-aumented epoch per hour.

Non-fine-tuning runs have been performed on a Dell PC using NVIDIA GTX960 GPU with 4GB available RAM at about the same time per epoch.

4.10. Dealing with imbalanced dataset

The first challenge to overcome was the imbalance nature of the data set—smallest class (grooming) is represented by 20 as few frames as the largest (walking). While our most useful rearing classes account for 20% of data one needs to be careful to create the network that pays more attention to infrequent classes and makes sure to learn more from them when they are encountered.

The earliest iteration of the model used uniform batch loss and it gravitated heavily towards the largest (walking) class failing to classify most of the remaining ones. Once weighted loss has been implemented this problem disappeared.

4.11. Dealing with overfitting

While imbalanced data was relatively easy to handle the biggest challenge in achieving meaningful results proved to be overfitting. Limited data set with highly correlated samples—by the very nature of continuous stream of video frames—made training particularly susceptible to this problem.

As model evolved various iterations produced unrealistically good results and their analysis demonstrated that while the network was good at memorizing the data using limited set of bottle-neck image features, this was of limited use when applied to the new data, be it a variant of new augmentation transforms or totally new video.

The first break through in achieving useful results was to reserve one rotation setting exclusively for validation. Once this feature is implemented one could easily observe the divergence of training and validation metrics and presented the need for regularization tools. Dropout has been very successful in achieving generalized.

The interactive classification tool described in the next

Figure 8: Sample frame from the video classification tool

section has been helpful in explaining model's mistake and lead to some improvement ideas, e.g. data augmentation with negative images.

4.12. "Understanding" model decisions

In order to trouble shoot overfitting and other training problem a tool allowing for interactive video analysis and classification has been developed.

The tool accepts any video and in real time classifies each frame, providing confidence levels for classes with softmax scores over 1%. The tool super imposes saliency map over the image to verify the model's "decision process" and spot potential source of mistakes.

It is also possible to do some basic transformation on the source video, e.g. cropping, rotation, contrast, sharpening or negative to test possible solutions to the classification mistakes as well as save the result in another video. Examples of the latter can be seen on at

https://www.dropbox.com/sh/h4q5708r22x23tp/AAAo6LOE31O63Tgi7BSeHooNa?dl=0

5. Results

Table 3 summarizes final metrics of the best models.

Fine-tune CNN	Other Hyperparameters	Min Loss	Val. EM	Val. F1
No	augm 16, lr decay <1	1.95	95.4	86.1
Yes	augm 16, lr decay =1	1.16	95.9	96.5
Yes	augm 16, le decay <1	0.78	97.7	96.4

Table 3: Final result of the 3 best models

5.1. Gitlab

The project source code is available at https://gitlab.com/ksebov/alcamice

5.2. Analysis and conclusion

The results clearly show that static image analysis has clear potential in identifying basic events in individual frames of video with static analysis. While future models are likely to require RNN layers to handle temporal aspect of animal behaviour, for relatively simple events it doesn't seem to be necessary.

Another, somewhat expected result is that knowledge transfer from pre-trained general-purpose networks really helps to achieve useful results with a few hours of training on modest hardware. While fine-tuning of deeper general-purpose layers does noticeably improve performance it is not critical and can be optionally switched off in the environments with limited resources.

5.3. Future Work

Frankly, there is still suspicion that surprisingly good results indicate some level of overfitting to the particular conditions of the experiment. While I'm quite confident the model will perform well when analyzing new videos shot under the same conditions—same box, lighting, camera, color of mouse—still a valuable result, it struggled classifying videos shot under different circumstances.

Textured background, white mice (very common in lab setting), foreign objects and markings, shaky camera, non-centered view, occlusions—all these factors significantly confused the model. Unfortunately, labeling and training

on this new data was too time consuming for the purpose of this project it showed that there is still work necessary to achieve truly universal setup-free solution.

Particularly important problem to address is models being able to work with low quality images and infrared or near-infrared images as it is often impossible to obtain good lighting conditions for the camera since the animals may be sensitive to bright light thus changing the behaviour being analyzed.

Furthermore, the model can be extended with other classifiers, like animal localization, pose estimation multiple animal detection. Adding RNN layers may prove useful in removing data noise as well as detecting detect complex patterns like novel object exploration, inter-animal interactions or motion traits.

5.4. Acknowledgements

I would like to thank my wife Viktoria Kheifets for her handling the most mundane aspect of this project—the data labeling—as well as for her help, patience and support. I am also grateful to her colleagues at Alkahest, Inc. Arnaud Teichert and Ian Gallager for their encouragement and insight into the specifics of the field.

References

[1] Tatem K S, Quinn J L, Phadke A, Yu Q, Gordish-Dressman H, Nagaraju K, Behavioral and Locomotor Measurements Using an Open Field Activity Monitoring System for Skeletal Muscle Diseases., 2014, http://dx.doi.org/10.3791/51785

[2] TopScan Suite (software) by CleverSys, http://cleversysinc.com/CleverSysInc/csi_products/topscan-suite/

[3] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, http://tensorflow.org/

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, arXiv:1602.07360

[5] Hong, W., Kennedy, A., Burgos-Artizzu, X. P., Zelikowsky, M., Navonne, S. G., Perona, P., & Anderson, D. J. (2015). Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning. Proceedings of the National Academy of Sciences of the United States of America, 112(38), E5351–E5360. http://doi.org/10.1073/pnas.1515982112

[6] Patel, T. P., Gullotti, D. M., Hernandez, P., O'Brien, W. T., Capehart, B. P., Morrison, B., Meaney, D. F. (2014). An open-source toolbox for automated phenotyping of mice in behavioral tasks. Frontiers in Behavioral Neuroscience, 8, 349. http://doi.org/10.3389/fnbeh.2014.00349 http://www.seas.upenn.edu/~molneuro/autotyping.html

[7] Kabra M, Robie AA, Rivera-Alba M, Branson S, Branson K., JAABA: interactive machine learning for automatic annotation of animal behavior., Nat Methods. 2013 Jan;10(1):64-7. doi: 10.1038/nmeth.2281. Epub 2012 Dec 2, https://www.researchgate.net/publication/233828747_JAABA_Interactive_machine_learning_for_automatic_annotation_of_animal_behavior

[8] Rosebrock A, Basic motion detection and tracking with Python and OpenCV, 2015, http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/

[9] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L, ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015 arXiv:1409.0575

[10] The HDF Group, Hierarchical data format version 5, 2000-2010, http://www.hdfgroup.org/HDF5

[11] Li FF, Johnson J, Yeung S, Stanford CS231n: Convolutional Neural Networks for Visual Recognition, Assignment3 http://cs231n.github.io/assignments2017/assignment3/ http://cs231n.stanford.edu/assignments/2017/spring1617_assignment3_v3.zip http://cs231n.stanford.edu/squeezenet_tf.zip

[12] Mané D, Hands-on TensorBoard (TensorFlow Dev Summit 2017), https://www.youtube.com/watch?v=eBbEDRsCmv4