Mice Tracking Using The YOLO Algorithm

Richardson Santiago Teles de Menezes[1], **John Victor Alves Luiz**[2], **Aron Miranda Henriques-Alves**[3], **Rossana Moreno Santa Cruz**[4], **Helton Maia**[2]

[1] Department of Computer Engineering and Automation
UFRN, Natal-RN, Brazil

[2]School of Science and Technology
UFRN, Natal-RN, Brazil

[3]Brain Institute
UFRN, Natal-RN, Brazil

[4]Electrical Engineering Department
IFPB, Joao Pessoa-PB, Brazil

helton.maia@ect.ufrn.br

Abstract. *The computational tool developed in this study is based on convolutional neural networks, and the You Only Look Once (YOLO) algorithm for detecting and tracking mice in videos recorded during behavioral neuroscience experiments. We analyzed a set of data composed of 13622 images, made up of behavioral videos of three important researches in this area. The training set used 50% of the images, 25% for validation, and 25% for the tests. The results show that the mean Average Precision (mAP) reached by the developed system was 90.79% and 90.75% for the Full and Tiny versions of YOLO, respectively. Considering the high accuracy of the results, the developed work allows the experimentalists to perform mice tracking in a reliable and non-evasive way.*

1. Introduction

In recent years, researchers in computer vision have developed a large number of algorithms and techniques for intelligent image processing. Not long ago, some methods for identifying the objects in images overcame the human-level performance to categorize images. The method developed by Microsoft in the ImageNet (Large-scale visual recognition challenge, 2015) [Russakovsky et al. 2015], achieved this goal using Deep Neural Networks.

Convolutional Neural Networks (CNNs or ConvNets) are a type of neural network widely used in computer vision problems. Inspired by biological processes, CNNs are designed to obtain a large number of connections, which in combination can similarly respond to stimuli, such as the visual cortex [Cichy et al. 2016]. Among the applications currently are the recognition of patterns in video and images [Feichtenhofer et al. 2016], diagnosis and analysis of medical images [Rajpurkar et al. 2017, Esteva et al. 2017], object detection in images [Liu et al. 2016, Redmon et al. 2016] and surveillance monitoring systems [Rasti et al. 2016].

Multidisciplinary research areas such as Neuroscience can benefit from the improvement of laboratory experiments to complex data analysis using modern machine

learning techniques [Vu et al. 2018]. In this way, it is possible to perform the detection and classification of brain patterns, correlations between behavioral tasks and electrophysiological recordings, social stress tests, and animal interaction, among other experimental designs with rodents [Henriques-Alves and Queiroz 2016, Frasch et al. 2017].

Neuroscientists conduct a series of animal behavior experiments to validate their research and video tracking of animals is essential to achieve their goals [Menezes et al. 2018, Unger et al. 2017, Mathis et al. 2018]. In the case of tracking video objects, animal tracking can be done in real-time, or even, in an offline analysis after the experiments; it will depend on the type of research being carried out.

It is common for software that tracks animals on video requiring specific conditions for its proper functioning. For example, the use of different colors for the animal being tracked and the background facilitates the use of solutions based on the separation of objects using image threshold. Another possibility is the previous selection of a region of interest (ROI) to perform the tracking only in that area. This can help to remove extra information that confuses the algorithms for object detection. A well-known approach is to select a template image of the environment in which the experiment will be carried out, allowing simple mathematical operations of comparison between the acquired images and this reference template to identify the desired target [Aitken et al. 2017, Romero-Ferrero et al. 2018].

This work aims to develop techniques for tracking mice during behavioral neuroscience experiments, using convolutional neural networks and the You Only Look Once (YOLO) algorithm. Three different experimental datasets on mouse behavioral tasks were used to show the ability of CNNs to track animals in adverse conditions, which is difficult with previous techniques. The results obtained provide a fast and accurate alternative for mice tracking according to state of the art in this area of knowledge. The presented technique can be useful to detect the movement of animals in complex scenarios, which include reflections, multiple objects, or animals in the scene.

2. Methods

This work used the C++ programming language and the YOLOv3 [Redmon and Farhadi 2018] algorithm to detect mice in different scenarios provided by three datasets [Henriques-Alves and Queiroz 2016, Jhuang et al. 2010, Burgos-Artizzu et al. 2012]. The computational development was performed on a computer with CPU AMD Athlon II X2 B22 (2) @ 2.800GHz, RAM 4GB, GPU NVIDIA GeForce GTX 1060 6GB, OS Ubuntu 18.04.1 LTS, CUDA 9.1, CuDNN 7.1.

2.1. Computational Development

All steps required in this work are described in the UML activity diagram shown in Figure 1.

- Step 1: Creation of a dataset from images used in mice behavioral experiments. Selection of the ground truth boxes for future comparison with the box predictor and obtaining the confidence score using the intersection over union (IOU).
- Step 2: Selection of the Yolov3 algorithm in the Full or Tiny version for mice tracking.

- Step 3: Training, validation, and testing for mice tracking in the dataset images performed by the Convolutional Neural Network.

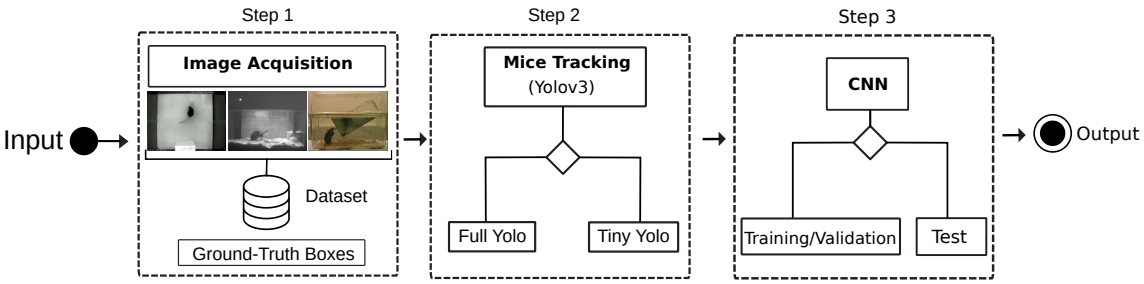

Figure 1. UML activity diagram of the developed system for video tracking and classification.

2.2. Image Dataset

To validate the computational development proposed in this work, a dataset was built with images from three researches that involve behavioral experiments with mice:

- **Ethological Evaluation** [Henriques-Alves and Queiroz 2016]: This research presents new metrics for chronic stress models of social defeat in mice. The authors built their image dataset and kindly provided a sample for our research.
- **Automated home-cage** [Jhuang et al. 2010]: In this study, a trainable computer vision system was introduced that allows the automated analysis of complex mouse behaviors, they are: eat, drink, groom, hang, micromovement, rear, rest, and walk. This dataset has two parts - Full: with more than 10.6 hours of continuous video of various behaviors, and Clipped: with 4200 small videos with specific behaviors.
- **Caltech Resident-Intruder Mouse dataset (CRIM13)** [Burgos-Artizzu et al. 2012]: It has videos recorded with superior and synchronized lateral visualization of pairs of mice involved in social behavior in thirteen different actions. Each video has a duration of ~ 10 min, totaling 88 hours of video.

Table 1 describes the sample size selected from each of the datasets used in this paper. For the Ethological Evaluation [Henriques-Alves and Queiroz 2016], 3707 frames were used, captured in a top view of the arena of social interaction experiments among mice. For the Automated home-cage [Jhuang et al. 2010], a sample of 3073 frames was selected from a side view of behavioral experiments: eat, drink, groom, hang, micromovement, rear, rest, and walk. For the Crim [Burgos-Artizzu et al. 2012], a sample of 6842 frames was selected, 3492 of a side view and 3350 of a top view for the following experiments: approach, circle, sniff, walk away. The sampling rate for selection of the frames in all cases was $\sim$ 1.5 s, avoiding similar images in the composition of the dataset.

Table 1. Information about datasets used

Dataset	**Images**	**Resolution**
Ethological Evaluation [Henriques-Alves and Queiroz 2016]	3707	640×480
Automated home-cage [Jhuang et al. 2010]	3073	320×240
Crim [Burgos-Artizzu et al. 2012]	6842	656×490
Total	**13622**	

The images were randomly selected into separate sets with the following distribution: 50% for the training set amounting to 6811 images, 25% for validation amounting to 3405 images, and finally, the remaining 3406 images were allocated in the test set, therefore, composing the dataset used in this work.

2.3. Mice tracking using CNN

In Neuroscience, mice models of social stress [Henriques-Alves and Queiroz 2016] are often used and require fast and accurate decision-making from the experimentalist to identify the correct response depending on the stimulus or the situation in which the animal is subjected. The essential element of such models has been a robust tracking algorithm. However, these algorithms are mainly limited by the computational cost, which hinders their usage in real-time and therefore limits the applicability for laboratory tests.

Commercial systems for video tracking of laboratory animals are often restricted to track the body or nose position, leading to inaccurate estimates of the head orientation or requiring manual artifices [Menezes et al. 2018, Kretschmer et al. 2012]. Thus, large-scale studies emphasize the need for automated high-throughput systems providing a reproducible behavioral assessment of freely-moving mice with only a minimum level of manual intervention [Unger et al. 2017].

In this context, this work focuses on the development of a computational tool designed with deep convolutional networks, the *Darknet* framework [Redmon and Farhadi 2018], the Open Source Computer Vision Library (OpenCV) and the C/C++ programming language for mice tracking during behavioral neuroscience experiments.

2.3.1. Convolutional Neural Networks

The idea behind CNNs is to use a cascade of convolutional and pooling layers to reduce the spatial dimension of an input image and combine local patterns to generate features that get more abstract as the data progress into the network. This cascade is called an encoder as raw input pixels are encoded into more abstract features [Lecun et al. 1998].

In this paper, f_i^{in} denotes the i-th input feature map, f_j^{out} denotes the j-th output feature map and b_j denotes the bias term for the j-th output feature map. For the convolution layer, n_{in} and n_{out} represent the number of input and output feature, respectively. For the Fully connected layer, n_{in} and n_{out} are the length of the input and output feature vector.

The convolutional layer takes a series of feature maps as input and convolves with convolutional kernels to obtain the output feature maps. A nonlinear layer, which applies nonlinear activation function to each element in the output feature maps, is often attached to convolutional layers. The convolutional layer can be expressed using (1):

$$f_i^{out} = \sum_{j=1}^{n_{in}} f_j^{in} \otimes g_{i,j} + b_j \qquad (1 \leq i \leq n_{out}), \tag{1}$$

where, $g_{i,j}$ is the convolutional kernel applied to j-th input feature map and i-th output feature map and b_j is the bias term for the j-th input feature map.

The pooling layer also referred as a down-sampling layer, which takes an $n \times n$ filter and a stride of length n, is then applied to the input vector and outputs the maximum or average values of each subarea. The reasoning behind this layer is that once we know that a specific feature is an original input, its exact location is not as crucial as its relative location to other features, such invariance can be provided by this layer as seen in [Scherer et al. 2010, Zhou et al. 2016]. Max-pooling can be expressed as (2):

$$f_{i,j}^{out} = \underset{p \times p}{max} \begin{pmatrix} f_{m,n}^{in} & \cdots & f_{m,n+p-1}^{in} \\ \vdots & & \vdots \\ f_{m+p-1,n}^{in} & \cdots & f_{m+p-1,n+p-1}^{in} \end{pmatrix}, \tag{2}$$

where, p is the pooling kernel size. This non-linear "downsampling" not only reduces the feature map size and the computation for next layers but also provides a form of translation invariance.

In a CNN, the encoder is often followed by some fully-connected layers as in a classical multi-layer perceptron setup. These layers apply a linear transformation to the input feature vector (3):

$$f^{out} = W \cdot f^{in} + b, \tag{3}$$

where, f^{in} is the input feature vector resulted from previous convolutions, W is an $n_{out} \times n_{in}$ transformation matrix, b is the bias term, and f^{out} is the output with the classes probabilities.

These layers take the highly abstracted encoded features as input, forwards then throughout the network using the feed forward algorithm and output global statistical predictions about the presence or absence of objects of interest in the input image [Ciresan et al. 2011].

2.3.2. YOLO Object Detector

You Only Look Once (YOLO) [Redmon et al. 2016, Redmon and Farhadi 2016, Redmon and Farhadi 2018] is an object detection algorithm targeted for real-time processing. It differs from other object detectors by using a single CNN for both classification and localization of the objects.

YOLO uses logistic regression for calculating the confidence score of an object in each bounding box. Another feature is to use a variant of Darknet, which has a 53-layer network trained on Imagenet dataset [Deng et al. 2009]. However, 53 more layers are stacked on it to form a 106-layer fully convolutional architecture for detection tasks. The algorithm also includes many important elements like residual blocks, skip connections, and upsampling in its architecture.

The analysis of a frame in the YOLO framework consists of three steps. First, the input image is resized, then a single CNN is run on it, and in the last step, thresholds using non-max suppression are applied in the resulting detections as described in [Redmon et al.

2016]. Figure 2 describes how an image of the dataset of this work is processed initially in the input, passing through the CNN, and finally in the output with mouse tracking.

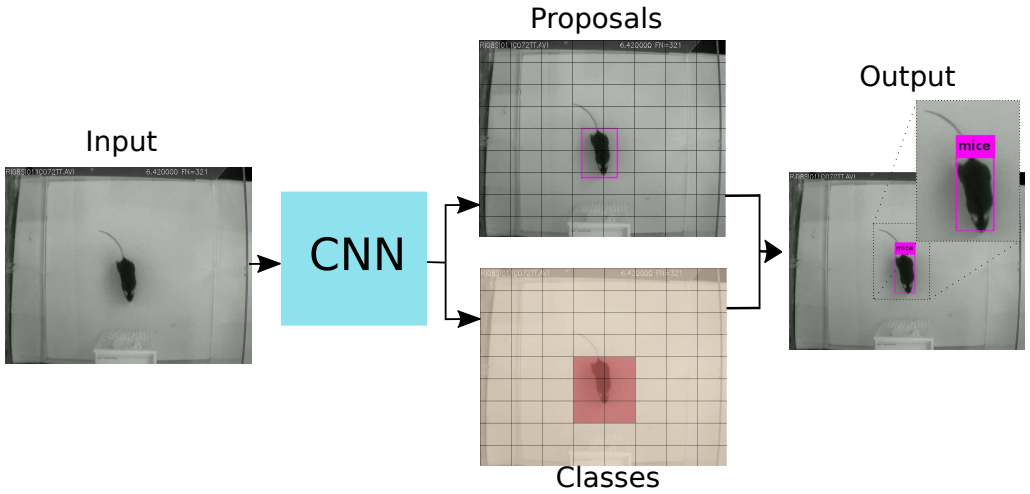

Figure 2. YOLO pipeline.

3. Results

Our approach used two versions of the YOLO network to detect mice within three different experimental setups. The results obtained in this work were based on the analysis of 13622 images, organized according to the dataset described in Section 2.2.

The first version of YOLO trained was the YOLO Full network which uses the *Darknet-53* [Redmon and Farhadi 2018] convolutional architecture that comprises 53 convolutional layers. Such a model was trained using transfer learning as described in [Redmon et al. 2016] starting from an Imagenet [Deng et al. 2009] pre-trained model. The model comprises a segment with convolutional layers and residual connections in a total of $65.29 \cdot 10^9$ floating point operations. Each model requires 235 MB of storage size. We used a batch of 64 images, a momentum of 0.9, and weight decay of $5 \cdot 10^{-4}$. The model took 139 hours to be trained.

We also trained a smaller and faster YOLO alternative, namely YOLO Tiny [Redmon and Farhadi 2018]. To speed up the process this "tiny" version comprises only a portion of the *Darknet-53* resources: 23 convolutional layers, resulting in $9.67 \cdot 10^9$ floating point operations, almost seven times fewer operations than the bigger counterpart. Each model requires only 34 MB of storage size. The network was trained as described in [Redmon et al. 2016], fine-tuning an Imagenet [Deng et al. 2009] pre-trained model. We used a batch of 64 images, a momentum of 0.9, and weight decay of $5 \cdot 10^{-4}$. The model took 19 hours to be trained. Figure 3 shows some examples, resulting from mice tracking performed on the three different datasets used.

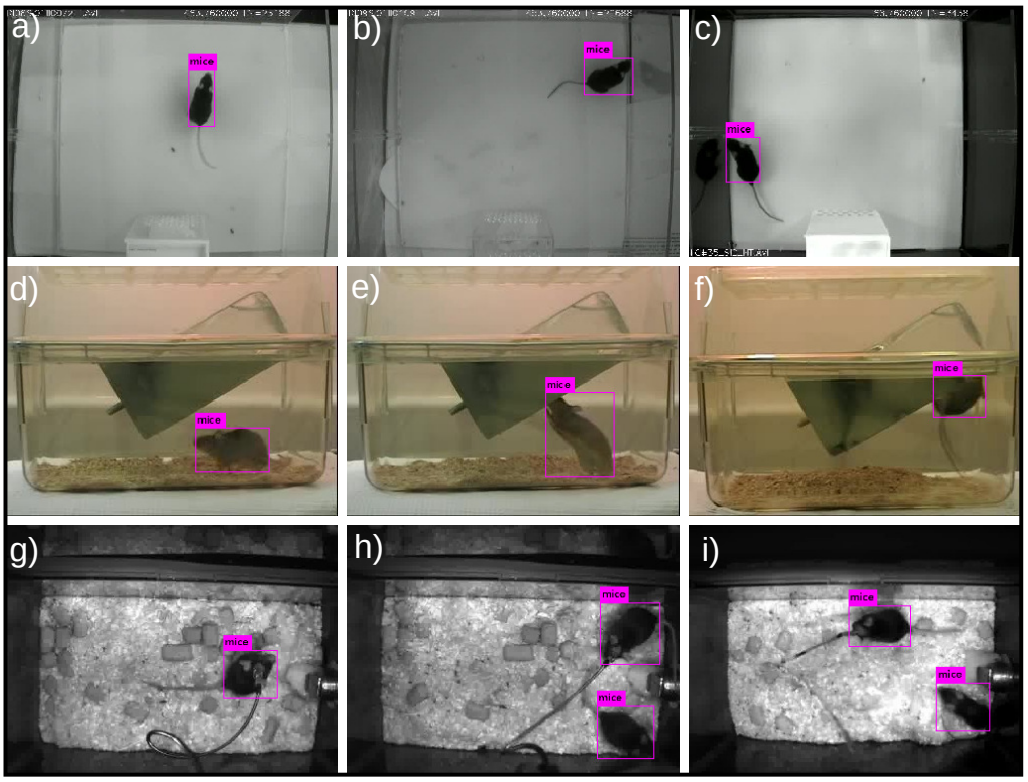

Figure 3. Output examples of the YOLO network. (a)-(c) refer to Ethological Evaluation [Henriques-Alves and Queiroz 2016], (d)-(f) refer to Automated home-cage [Jhuang et al. 2010] and (g)-(i) to Crim [Burgos-Artizzu et al. 2012].

Table 2 shows the classification metrics resulting from the classification step, using both, YOLO Full and Tiny. These metrics are used to evaluate the quality of the classifier output. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to all observations. F1-score is the weighted average of precision and recall, therefore, the maximum value assigned to it is 1, which means a perfect precision and recall.

Table 2. YOLO performance after 40000 epochs

Dataset size	Precision		Recall		F1-score	
	Full	Tiny	Full	Tiny	Full	Tiny
13622	0.99	0.98	0.99	0.99	0.99	0.99

Still on Table 2, for the Tiny network, the average values of Precision, Recall, and F1-score were all equal to 0.987, which indicates a good relation between Precision and Recall, whereas for the Full network, the results were slightly better, with the average values of Precision, Recall, and F1-score equal to 0.99.

More results related to mice tracking performance are shown in Table 3. First, it is shown the mean average precision (mAP), which is the mean value of the average precisions for each class, where average precision is the average value of 11 points on

the Precision-Recall curve for each possible threshold, that is all the probability of detection for the same class (Precision-Recall are evaluated according to the terms described in the PascalVOC [Everingham et al. 2015]). Finally, the average intersect over union (IoU), which is the average of the intersection between the predicted bounding box and the ground truth box divided by the the total area of both boxes.

Table 3. Results for the epoch 40000, mAP and IoU

Dataset size	Mean Average Precision (%)		Average IoU (%)	
	Full	Tiny	Full	Tiny
13622	90.79	90.75	84.67	82.81

The Full network shows slightly better statistics for both the mean average precision and average IoU, differing from the Tiny counterpart by 0.04% and 1.86%, respectively, as shown in Table 3.

Figure 4 shows the comparison of the mAP between the two models used, YOLO Full and Tiny. It was presenting the same sample of 100 images from the set of tests, for both models. Figure 4(a) shows high accuracy of the Full architecture with small oscillations of the accuracy curve during the training. In Figure 4(b), the high accuracy is maintained from the earliest times, and remains practically unchanged up to the limit number of epochs. Both architectures reached high mean average precision values while successfully minimizing the values of the loss function. The Tiny version of the YOLO network presented better stability in precision, which can be seen by the smoothness in its curve.

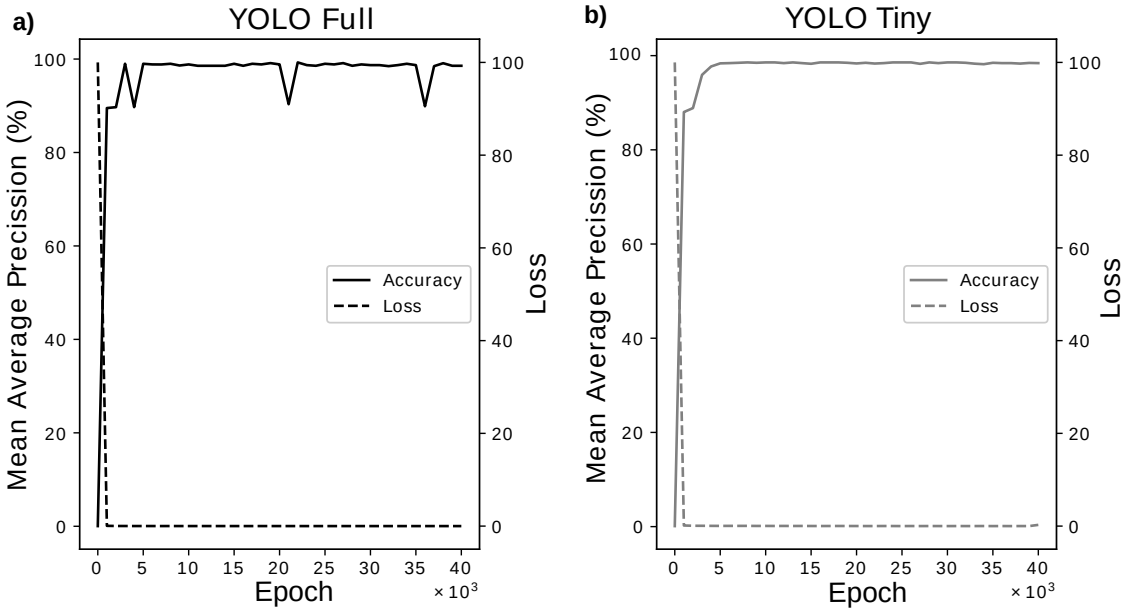

Figure 4. Comparison of the model's mAP every 1000 epochs classifying 100 images from the test set.

In terms of training and testing time for the YOLO network, Table 4 shows the efficiency of using the Full architecture in relation to the Tiny one. The training using the

Tiny architecture was performed in 19 hours, that is about seven times faster than the Full version. For the tests the use of the Tiny version was about four times faster than the Full counterpart.

Table 4. GPU time necessary to perform the training and test of the model

Train Time(hours)		**Test Time(seconds)**	
Full	Tiny	Full	Tiny
139	19	1188	261

Figure 5 is a bar graph showing the mean time spent on the classification of a single image in both architectures. The smaller size of the Tiny version gets a direct translation in execution time, having 0.08 ± 0.06 as the mean and standard deviation values, whereas the Full version has 0.36 ± 0.16 as the mean and standard deviation values, respectively.

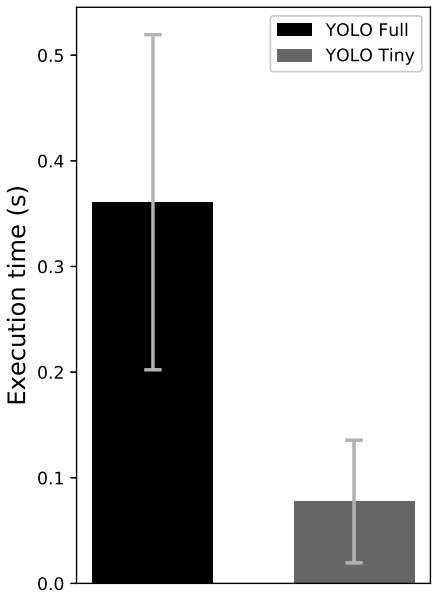

Figure 5. Bar graph of the GPU time necessary to perform the classification of a single image.

4. Conclusion

The results presented in Tables 2 and 3 suggest a higher classification accuracy for the YOLO network when using the Full architecture, although the difference obtained with the Tiny version was not substantial.

Given the aforementioned small difference between the two versions of the network, the possibility of a robust real-time system for mice tracking is made a reality with the Tiny version of the YOLO network. Due to the smaller demand of computing power, this work opens the possibility of real-time tracking systems where actions can be taken during the experiment in an automated and intelligent way without the need for human intervention.

The computational development presented in this paper performed mice tracking in behavioral experiments setups using the deep neural network model called You Only Look Once (YOLO). The accuracy of the YOLO model and its performance showed better results using the Tiny architecture in comparison with the Full architecture. The results obtained in this work, which include high rates of accuracy and fast computation, encourage experimentalists to conduct new experimental designs where real-time decisions can be made.

All computational tools develop during this work can be found in a repository available at `https://github.com/heltonmaia/ECT-proj-cnn-mice`, which, includes a tutorial for the usage of the proposed tools.

Acknowledgment

This work was supported by the School of Sciences and Technology at the Federal University of Rio Grande do Norte (ECT-UFRN).

References

Aitken, P., Zheng, Y., and Smith, P. F. (2017). Ethovision™ analysis of open field behaviour in rats following bilateral vestibular loss. *Journal of Vestibular Research*, 27(2-3):89–101.

Burgos-Artizzu, X. P., Dollár, P., Lin, D., Anderson, D. J., and Perona, P. (2012). Social behavior recognition in continuous video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1322–1329. IEEE.

Cichy, R. M., Khosla, A., Pantazis, D., Torralba, A., and Oliva, A. (2016). Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports*, 6:27755.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification.

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*.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941.

Frasch, M. G., Lobmaier, S., Stampalija, T., Desplats, P., Pallarés, M. E., Pastor, V., Brocco, M., Wu, H.-t., Schulkin, J., Herry, C., et al. (2017). Non-invasive biomarkers of fetal brain development reflecting prenatal stress: an integrative multi-scale multi-species perspective on data collection and analysis. *arXiv preprint arXiv:1801.00257*.

Henriques-Alves, A. M. and Queiroz, C. M. (2016). Ethological evaluation of the effects of social defeat stress in mice: beyond the social interaction ratio. *Frontiers in behavioral neuroscience*, 9:364.

Jhuang, H., Garrote, E., Yu, X., Khilnani, V., Poggio, T., Steele, A. D., and Serre, T. (2010). Automated home-cage behavioural phenotyping of mice. *Nature communications*, 1:68.

Kretschmer, F., Kretschmer, V., Köpcke, L., Helmer, A., and Kretzberg, J. (2012). Automated determinination of head gaze in rodents. In *Image and Signal Processing (CISP), 2012 5th International Congress on*, pages 1209–1213. IEEE.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., and Bethge, M. (2018). Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. Technical report, Nature Publishing Group.

Menezes, R. S. T. d., de Azevedo Lima, L., Santana, O., Henriques-Alves, A. M., Santa Cruz, R. M., and Maia, H. (2018). Classification of mice head orientation using support vector machine and histogram of oriented gradients features. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., and Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*.

Rasti, P., Uiboupin, T., Escalera, S., and Anbarjafari, G. (2016). Convolutional neural network super resolution for face recognition in surveillance monitoring. In *International conference on articulated motion and deformable objects*, pages 175–184. Springer.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2016). Yolo9000: better, faster, stronger (2016). *arXiv preprint arXiv:1612.08242*, 394.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Romero-Ferrero, F., Bergomi, M. G., Hinz, R., Heras, F. J., and de Polavieja, G. G. (2018). idtracker. ai: Tracking all individuals in large collectives of unmarked animals. *arXiv preprint arXiv:1803.04351*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer.

Unger, J., Mansour, M., Kopaczka, M., Gronloh, N., Spehr, M., and Merhof, D. (2017). An unsupervised learning approach for tracking mice in an enclosed area. *BMC bioinformatics*, 18(1):272.

Vu, M.-A. T., Adali, T., Ba, D., Buzsaki, G., Carlson, D., Heller, K., Liston, C., Rudin, C., Sohal, V., Widge, A. S., et al. (2018). A shared vision for machine learning in neuroscience. *Journal of Neuroscience*, pages 0508–17.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.