Application of a neural network approach to the task of arena marking for the "Open Field" behavioral test

A.I. Anikina,[a,*] **D.V. Podgainy,**[a] **A.V. Stadnik,**[a] **O.I. Streltsova,**[a] **I.A. Kolesnikova,**[a] **Yu.S. Severiukhin**[a] **and D.A. Savvateev**[b]

[a]*Joint Institute for Nuclear Research,*
6 Jolio-Curie st., Dubna, Moscow region, 141980, Russia

[b]*Dubna University,*
19 Universitetskaya st., Dubna, Moscow region, 141980, Russia

E-mail: aanikina@jinr.ru, podgainy@jinr.ru, stadnik@jinr.ru,
strel@jinr.ru, innakolesnikova@jinr.ru, yucucumber@mail.ru,
savvateeaa@yandex.ru

Within the joint project of MLIT and LRB JINR, aimed to the creation of an information system for the tasks of radiation biology, a module to study the behavioral patterns of small laboratory animals exposed to radiation is being developed. The module for behavioral analysis automates the analysis of video data obtained by testing laboratory animals in different test systems. The "Open Field" behavioral test is one of the systems. The considered behavioral test has a form of round arena with chequered-marked sectors and holes. The observation procedure on laboratory animals takes 3-6 minutes. The "Open Field" test system allows one to register the exploratory and general motor activities, orienting-exploratory reaction and emotional status of the animal. To this aim, we fix the number of crossed sectors together with the number of intersections of the marked center. In addition, the actions the animal performed, i.e., hole dipping, rearing/climbing, freezing and shifting, is taken into account.

Therefore, one of our tasks is to develop an algorithm for behavioral test arena marking. The paper presents algorithms for arena marking of the "Open Field" test system on the basis of computer vision methods together with the method of key points within a neural network approach.

---

*Speaker

1. Introduction

The automation of research in biology and medicine, including for the tasks of analyzing data of behavioral tests for small laboratory animals, is an actively developing direction [1]. To automate the studies of pathomorphofunctional changes in the central nervous system of rodents conducted at the JINR Laboratory of Radiation Biology [2], an information system is being developed [3]; it provides convenient tools for data storage and processing. Part of this system is a module that allows one to automate the analysis of video data obtained when testing rodents in various test systems, one of which is the Open Field [4]; its view is shown in Fig. 1 on the left.
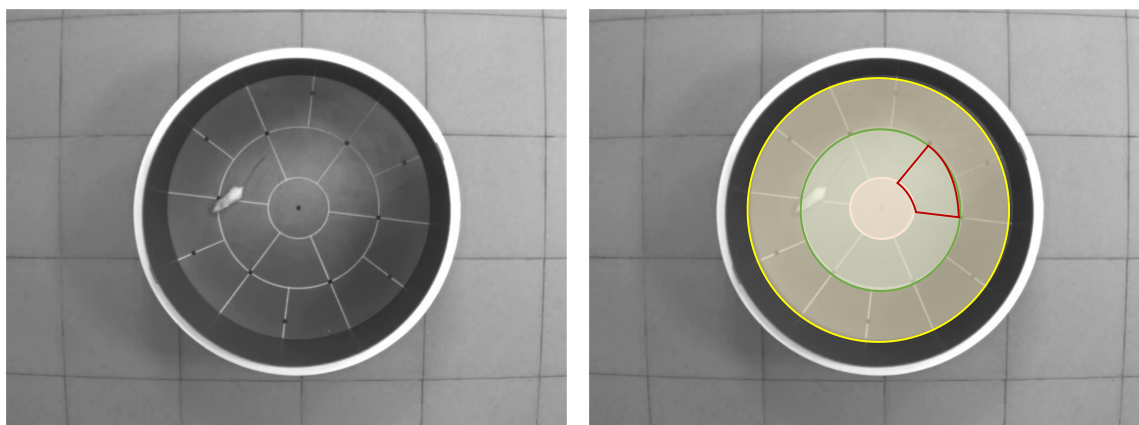
Figure 1: "Open Field" setup: type of setup (left), marking required according to the test protocol (right); the areas of the center, the inner and outer zones of the arena and one of the sectors are highlighted

2. Problem statement

The test system is an arena with drawn sectors and minks. The setups have different physical dimensions and marking parameters depending on the type of rodents: $63 \times 38 \times 14.5$ cm for mice and $97 \times 58 \times 22$ for rats. The observation of an experimental animal inside the arena takes from 3 to 6 minutes. The analysis of the video recording of the experiment allows recording the exploratory and general motor activities of the animal [5]. To assess the activities, the number of traversed sectors and intersections of the center is recorded, and minks, grooming, racks with an emphasis on the wall and without an emphasis on the wall, fading, movements on the spot are counted.

The task of the algorithmic module of the information system is to automate the process of processing data and obtaining the necessary data according to the protocols of conducting behavioral tests. To do this, the marking of the experimental field is required, which means defining the center of the physical setup, the radii of three circles on the video footage: central, middle and bounding arenas, as well as the marking lines of the sectors (Fig. 1 on the right).

To solve this class of tasks, there are approaches based on classical computer vision algorithms, such as the Hough transform for finding lines and circles [6], which have some limitations in applicability. The implementation of automated markup based on a neural network approach will significantly enhance the range of applicability and accuracy of the markup of the experimental field. The paper considers several modifications of one of the possible approaches to solving the

problem, based on the search for key points in the image. This technique is successfully used, in particular, in the task of number recognition [7] and other similar tasks.

3. Dataset

For conducting behavioral tests, LRB JINR is equipped with a specialized behavioral room [2], with installed video recording equipment. For the "Open Field" test system, the camera is mounted on top, and the setup itself can be shot at different angles and under different lighting modes.

To form a dataset, 46 video recordings were used with a video data format of 1024×1280, 768×1024, 1080×1920, 960×1280. For each of them, median images and the first frames were found, and in order to expand the dataset for 36 images, the data augmentation mechanism was used with the tools of the Keras framework [8], which implements rotation and reflection transformations and changes the illumination. For the final check of the models, 11 images were allocated to the test subset. The result set contained 231 images. For further implementation of the key point method, characteristic points with a pronounced change in intensity were selected as such, for this task these were "minks" in the arena, namely, 13 holes that are black in the image. Using the ImageJ program [9], the coordinates of the key points were marked in a certain order, starting from the center on the entire dataset, from which files in csv format were generated. An example of marked-up data is shown in Fig. 2.
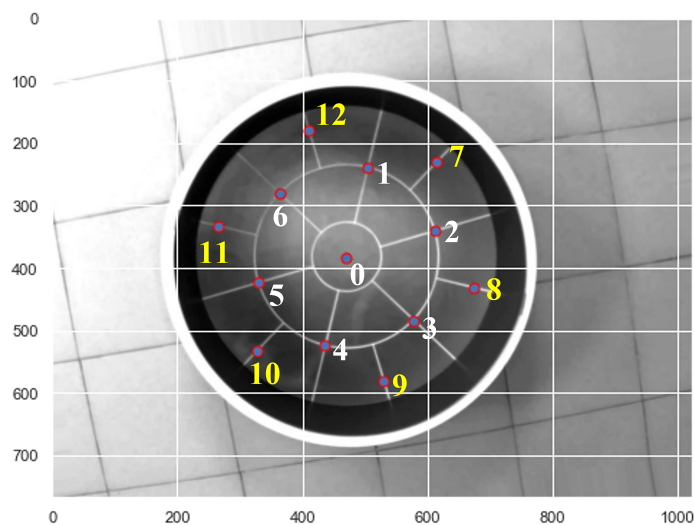
Figure 2: Example of an image with 13 key points marked

4. Neural network models

In recent years, the use of deep learning models has made it possible to achieve significant progress in solving many applied tasks [10]. To solve applied tasks, both models, the training of which is carried out on a given dataset "from scratch", are developed, and various modifications of transfer learning are elaborated. There are various high-performance deep learning networks that have been developed and tested in areas such as computer vision and natural language processing.

Deep learning systems and models are multi-level architectures that, as a result of training, form a representation of data at different levels of nesting. Then these convolutional layers are supplemented with final layers that form a decision rule for transforming the internal representation into the required classification or regression result, depending on the task format. The layered architecture formed in this way allows one to use a pre-trained network (e.g., InceptionV3 or VGG-16) without its final layer as a fixed feature extractor for other similar tasks. The key idea here is to simply use the weighted layers of the pretrained model to extract features, but not to update the weights of the model layers during training with new data for a new task. One of the fundamental requirements for transfer learning is to have models that perform well on the original tasks. Many of the common deep learning architectures that have been pre-trained on open datasets are publicly available. The set of parameters (weights) of the network obtained in the process of training the model can be used to solve similar tasks. Pre-trained models are available in the Keras deep learning library written in Python, which provides an interface for loading popular models. For computer vision, models such as VGG-16, VGG-19, InceptionV3, Xception, ResNet50 are often used. In addition, in Keras, an example of the implementation of the key point method using the transfer learning technique is presented [11].

For the task of finding key points for marking the experimental field, the median image or the first frame is fed to the input of the models considered in the work; the vector of the coordinates of the key points of the setup arena ($13 \times 2 = 26$ points) is expected as the output of the

Three models are considered in the work: based on the *"keypoint_detector_CNN"* convolutional architecture, which is trained on a prepared data set and two models using the *"keypoint_detector_VGG16"* transfer learning technique, as a convolutional basis for which the pretrained neural network model VGG-16 and the *"keypoint_detector_Xception"* model are used with a convolutional pre-trained Xception model.

Model 1: Model *"keypoint_detector_CNN"*. The architecture of the model is shown in Fig. 3. The size of the input layer is chosen 784×784 as the minimum of all dimensions of the available video data. The architecture introduces striped convolutional and sub-sampling layers of MaxPooling2D. The structure of the network and the number of weight coefficients in each layer are shown in Fig. 4 on the left.
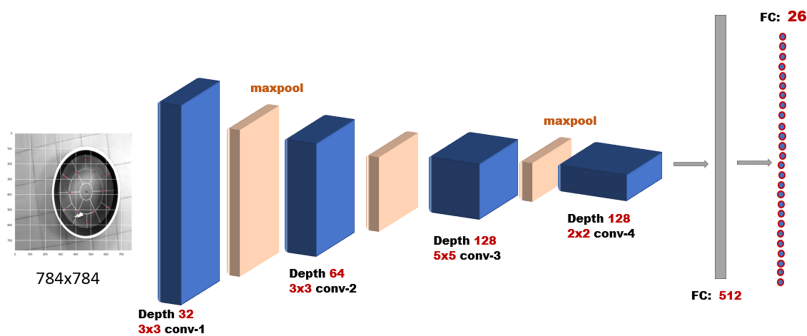
Figure 3: Architecture of a convolutional neural network

To train the image model, the coordinates of the key points were reduced to size 784×784, then the images were normalized to $[-1, 1]$: I := I/255, and the coordinates of the key points to the segments $[-1, 1]$: $(x, y) := ((x - 384)/384, (y - 384)/384)$. The mean squared error (MSE)

was chosen as a metric. The labeled dataset was divided into three parts: training, validation and test, in percentages of 67%, 28% and 5%, respectively. Fig. 4on the right s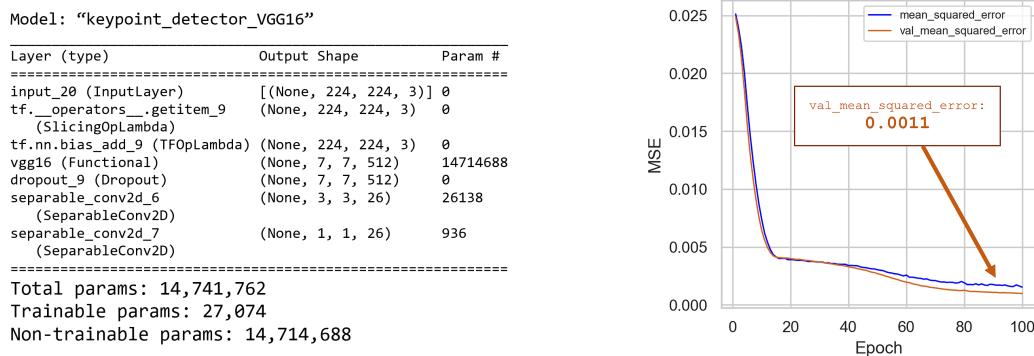hows the training and validation losses, where the rms training error is taken as a loss function. On the validation subset, the achieved value of the root-mean-square error was val_mean_squared_error = 0.0012.

```
Model: "keypoint_detector_CNN"

Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 768, 768, 1)]     0
conv2d_8 (Conv2D)            (None, 383, 383, 32)      320
max_pooling2d_8 (MaxPooling2D) (None, 191, 191, 32)    0
conv2d_9 (Conv2D)            (None, 95, 95, 64)        18496
max_pooling2d_9(MaxPooling2D) (None, 47, 47, 64)       0
conv2d_10 (Conv2D)           (None, 22, 22, 128)       204928
max_pooling2d_10(MaxPooling2D) (None, 11, 11, 128)     0
conv2d_11 (Conv2D)           (None, 10, 10, 128)       65664
max_pooling2d_11(MaxPooling2D) (None, 5, 5, 128)       0
flatten_2 (Flatten)          (None, 3200)              0
dense_4 (Dense)              (None, 512)               1638912
dense_5 (Dense)              (None, 26)                13338
=================================================================
Total params: 1,941,658
Trainable params: 1,941,658
Non-trainable params: 0
```

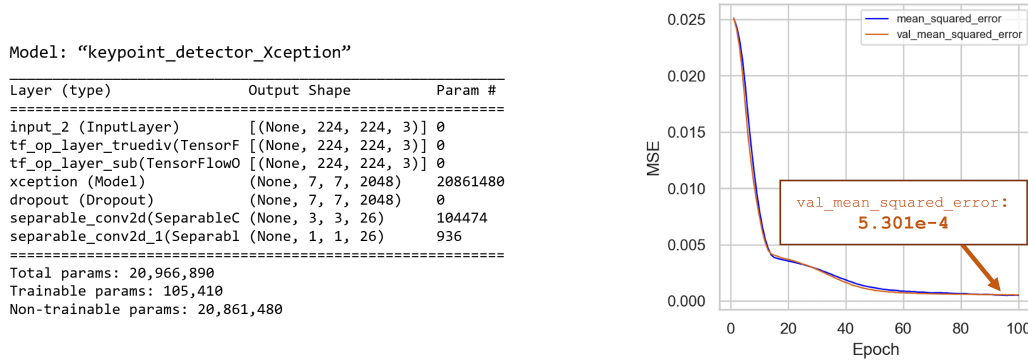**Figure 4:** Model parameters based on the CNN (left), training and validation losses (right)

Model 2: Model *"keypoint_detector_VGG16"* with a convolutional basis on top of the trained VGG-16 model [7]. The weights of the convolutional basis did not change during the learning process. The architecture of the neural network model with the number of weights is shown in Fig. 5 on the left. To train the image model, the coordinates of the key points were reduced to size 224×224, which corresponds to the dimension of the ImageNet dataset, on which VGG-16 was trained. Then the images were normalized to the interval $[0, 1] : I := I/255$, and the coordinates of the key points to the segments $[0, 1] : (x, y) := (x/224, y/224)$. The mean squared error (MSE) was chosen as a metric. The labeled dataset was also divided into three parts: training, validation and test, in percentages of 67%, 28% and 5%, respectively. Fig. 5 on the right shows a decrease in the mean squared error on the training and validation samples depending on the number of training epochs. The losses are shown at the stages of training and verification, where the mean squared learning error is taken as a loss function. The minimum value of the root-mean-square error on the validation subset was val_mean_squared_error = 0.0011.

```
Model: "keypoint_detector_VGG16"

Layer (type)                 Output Shape              Param #
=================================================================
input_20 (InputLayer)        [(None, 224, 224, 3)]     0
tf.__operators__.getitem_9   (None, 224, 224, 3)       0
  (SlicingOpLambda)
tf.nn.bias_add_9 (TFOpLambda) (None, 224, 224, 3)      0
vgg16 (Functional)           (None, 7, 7, 512)         14714688
dropout_9 (Dropout)          (None, 7, 7, 512)         0
separable_conv2d_6           (None, 3, 3, 26)          26138
  (SeparableConv2D)
separable_conv2d_7           (None, 1, 1, 26)          936
  (SeparableConv2D)
=================================================================
Total params: 14,741,762
Trainable params: 27,074
Non-trainable params: 14,714,688
```

Figure 5: Parameters of the model based on VGG-16 (left), losses at the training and validation stages (right)

Model 3: Model *"keypoint_detector_Xception"* with a convolutional basis on top of the trained Xception model [7]. To train the image model, the coordinates of the key points were reduced to

size 224×224 which corresponds to the dimension of the ImageNet dataset, on which Xception was trained. The mean squared error (MSE) was chosen as a metric. The labeled dataset was divided into three parts: training, validation and test, in percentages of 67%, 28% and 5%, respectively. Fig. 6 on the right shows the training and validation losses, where the standard error of training is taken as a loss function. The minimum value of the root-mean-square error on the validation subset was val_mean_squared_error = $5.3e - 4$.
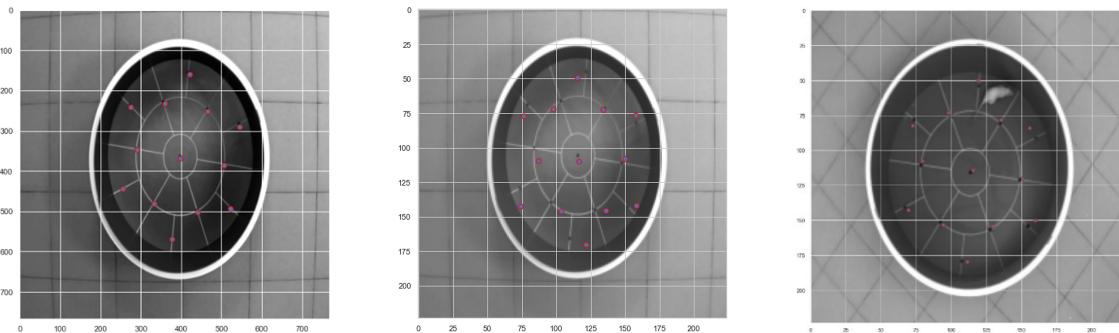

Figure 6: Model parameters based on Xception (left), training and validation losses (right)

The results of model testing on the test subset are shown in Fig. 7 and Table 1 below.

Table 1: Evaluation of the models on test data

Model	keypoint_detector_CNN	keypoint_detector_VGG16	keypoint_detector_Xception
MSE	0.0008417	0.001056	0.2903

Figure 7: Model validation examples on a test subset: model prediction results from left to right: *"keypoint_detector_CNN"*, *"keypoint_detector_VGG16"* and *"keypoint_detector_Xception"*

As seen from Table 1, the *"keypoint_detector_CNN"* model showed the best result in terms of accuracy. The *"keypoint_detector_VGG16"* and *"keypoint_detector_Xception"* models, built on complex architectures using the transfer learning technique, did not show such a good result, which is also seen in Fig. 7 that illustrates the results of testing the models on a test subset. From the obtained comparative results, there is a decrease in accuracy with the complication of the model, and the transfer of training assumed a decrease in the dimension of images by 3.5 times, which affected the final accuracy.

6

5. Marking the arena of the "Open Field" setup

To mark the setup field, namely, to determine the central and inner zones, the outer zone and sectors, the coordinates of the center of the arena, the radii of the circles R_1, R_2, R_3 and straight lines are sequentially defined on the basis of the obtained key points (Fig. 8) according to the following algorithm.

The result of the work of the neural network is scaled to the size of the original image, and an array of the coordinates of 13 key points, corresponding to Fig. 2, is formed.

The center of the arena of the experimental field is determined as the center of mass of the found key points, which allows leveling possible inaccuracies in defining the coordinates. Further, the circle radius R_2 is determined as an average distance to the center of the key points from the 1st to the 6th inclusive. According to the proportions of the experimental setup (arena radii for rats: 97 cm, 58 cm, 22 cm, for mice: 63 cm, 38 cm, 14.5 cm), the radii R_1 and R_3 are defined.
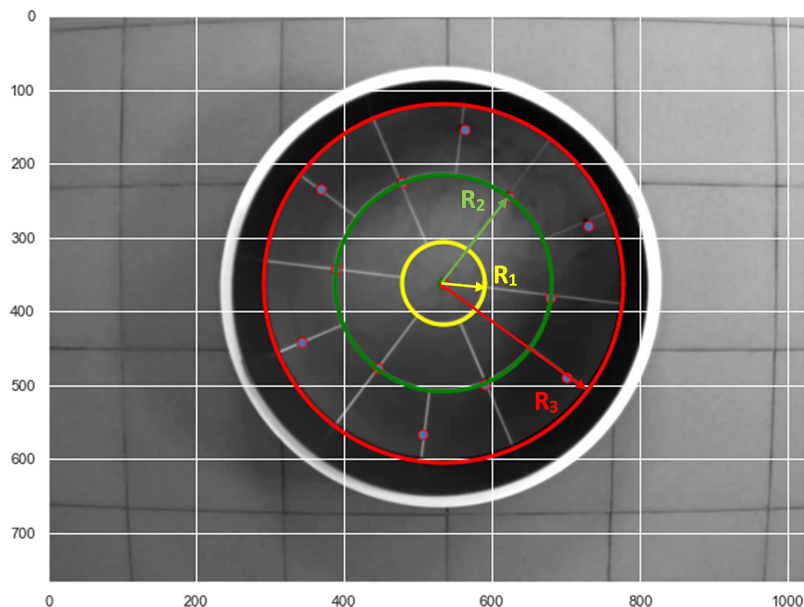
Figure 8: Architecture of a convolutional neural network

Sector marking lines are given as normalized vectors of unit length and determined as vectors from the center to the key points. The use of the equivalence of the angular size of the sectors equal to 60 degrees allows leveling errors in determining the coordinates when defining the marking lines.

As a result of the algorithm, we obtain a description of the sectors in the form of figures bounded by straight lines and circles, as shown in Fig. 1 on the right.

6. Conclusion

The proposed approach for marking the arena of the "Open Field" test setup based on the identification of key points showed the promise of its use for solving this task.

To determine key points, a neural network approach was chosen, and three neural network models, namely, a simple convolutional model and two models based on the VGG-16 and Xception

architectures trained using the transfer learning technique, were considered. The given neural network architectures showed comparable results, however, the best accuracy was provided by an approach based on the simple convolutional model *"keypoint_detector_CNN"*. The further development of the approach involves expanding the dataset for training and the set of key points to improve the accuracy of labeling.

References

[1] D. P. Chernyuk, A. G. Zorin, K. Z. Derevtsova et al., *Automatic analysis of the "Morris water maze" behavioral test data*, *Zh Vyssh Nerv Deiat I P Pavlova* **71 (1)** 126-135 2021 (in Russian).

[2] Yu. S. Severyukhin, M. Lalkovičová, D. M. Utina et al., *Comparative Analysis of Behavioral Reactions and Morphological Changes in the Rat Brain After Exposure to Ionizing Radiation with Different Physical Characteristics*, *Cell Mol Neurobiol.* 2022 Jan 17.

[3] I. Kolesnikova, A. Nechaevskiy, D. Podgainy et al., *Information System for Radiobiological Studies*, *CEUR Workshop Proceedings* **2743** 2020 (accessed 01.08.2022).

[4] Open Field test (OpenScience, Russia) [Electronic resource]: URL: https://www.openscience.ru/index.php?page=ts&item=001 (accessed 01.08.2022)

[5] Protocols for testing at the Open Field [Electronic resource]: URL: https://www.openscience.ru/index.php?page=ts&item=001 (accessed 01.08.2022)

[6] OpenCV [Electronic resource]: URL: https://opencv.org (accessed 01.08.2022)

[7] I. G. Zubov, *An Automatic Method for Interest Point Detection. Journal of the Russian Universities*, *Journal of the Russian Universities. Radioelectronics* 2021; 324:49 (in Russian).

[8] Keras: the Python deep learning API [Electronic resource]: URL: https://keras.io (accessed 01.08.2022)

[9] Image processing and analysis in Java ImageJ [Electronic resource]: URL: https://imagej.nih.gov/ij (accessed 01.08.2022)

[10] F. Chollet *Deep Learning with Python*, Manning Publications Co, 2018.

[11] Keypoint Detection with Transfer Learning. Keras official site [Electronic resource]: URL: https://keras.io/examples/vision/keypoint_detection (accessed 01.08.2022)