The extended user guide for *LabGym* (v2.5.0)

Yujia Hu, Ph.D.

Table of contents

Installation of *LabGym* (https://github.com/umyelab/LabGym) [1] [2]

Please refer to: https://labgym.readthedocs.io/en/latest/installation/index.html

Use *LabGym* via its graphical user interface (GUI)

1. Initiate the graphical user interface (GUI) of *LabGym* for each use

Before v2.3:

a. Open the Terminal (Mac / Linux) or Command Prompt (Windows), activate Python3 [3] interactive shell by typing:

    python3

b. After the Python3 interactive shell is activated, type:

    from LabGym import gui

c. After the importing is done, then type:

    gui.gui()

After v2.3:

a. Open the Terminal (Mac / Linux) or Command Prompt (Windows), and type:

    LabGym

2. Extended guide on using the GUI of *LabGym*

Put your mouse cursor above each button to see a detailed description for it.

2.1. '*Preprocessing Module*'

Preprocess your videos to make them more appropriate for analysis.

2.1.1. '*Preprocess Videos*'

Use this to trim the videos to only keep necessary time windows, crop frames to exclude unnecessary regions, enhance contrast of the videos, or reduce video fps to increase processing speed.

(Button) 'Specify whether to enter time windows to form a trimmed video'

When trimming videos, if you want to set different time windows for different videos, use code '_stt_' and '_edt_' in the filename of a video, replacing the last 't' with the start ('st') time point and end ('ed') time point. For example, 'XX_st10_ed20.avi' (the '_' is unnecessary if it is at the end of file name) means the trimming will start at the 10th second and end at the 20th second. You can set multiple time windows for a video, just use multiple '_stt_' and '_edt_' following the time sequence.

2.1.2. '*Draw Markers*'

Use this to draw colored circle markers in your videos, which can be used to mark a region in videos. This function is useful when you want LabGym to identify interactions between animals and specific regions in videos, but the regions are not marked.

2.2. '*Training Module*'

The first 4 buttons in this module can be used to train and test Detectron2-based (https://github.com/facebookresearch/detectron2) [4] Detectors, which is the way to teach *LabGym* to recognize animals / objects of your interest. There are two detection methods in *LabGym*. One is the Detector-based, which is slow but versatile. It is also useful for distinguishing individuals when they have close body contact. The other detection method is background subtraction-based, which is much faster but requires the background to be static and illumination to be stable in a video. The background-subtraction-based method is your first choice if the videos meet the criteria. Note that to distinguish individuals when they have body contact, using trained Detectors is recommended.

The last 4 buttons can be used to train and test Categorizers, which is the way to teach *LabGym* to recognize behaviors that are defined by you.

2.2.1. '*Generate Image Examples*'

To train or test Detectors, you need to provide training or testing images in which the outlines of animals / objects of your interest are precisely annotated. But first, you need to generate some image examples (extract some frames from videos).

To train a Detector with high accuracy, 50~150 annotated images that covers different poses / states / appearance of the animals / objects are enough for a typical scenario of non-social behaviors. If you use 'Interactive advanced' mode, more annotated images are needed depending on the complexity of the scenarios. In 'Interactive advanced' mode, the annotation should be focused on images in which animals / objects have close body contact. Therefore, select as many such images as possible. In the other hand, you would only need 5~10 images in which the animals / objects are well separated. The more colliding scenarios the Detector sees during training, the higher accuracy of the detection, and the lower chance of identity-switching during analysis. If the Detector sees all the colliding scenarios during training, the identity-switching during analysis can be eliminated. Moreover, more iterations typically yield higher accuracy. If you have a powerful GPU or the training time is not a concern, always set training iterations >= 5,000. During training, pay attention to the 'total_loss', which should be ideally less than 0.2 at the end of training. If the 'total_loss' is too high, increase the training iterations.

2.2.2. '*Annotate images with Roboflow*'

After images are generated, you need to annotate them to train a Detector. The annotation is to label the outline of the animals / objects of your interest. You can do this by using online annotation tools such as Roboflow (https://roboflow.com) or CVAT (https://www.cvat.ai) or VGG Image Annotator (https://www.robots.ox.ac.uk/~vgg/software/via/). Roboflow is recommended because it is easy to use. When annotating images, make sure to select "Instance Segmentation" for the annotation type. When export the annotation file, make sure to select "COCO instance segmentation" format, which is a '*.json' file.

Importantly, if you use Roboflow, when you generate a version of dataset, do NOT perform any preprocessing steps such as 'auto orient' and 'resize (stretch)'. Instead, perform some augmentation based on which manipulations may occur in real scenarios.

2.2.3. '*Train Detectors*'

Trained Detectors will be stored in the Detector list in *LabGym* and ready to use.

(Button) 'Specify the inferencing framesize for the Detector to train'

It determines the speed-accuracy trade-off of Detector performance. You can try large size such as 960 or above if GPUs are used. If you don't use a GPU, start with small size such as 256 and increase the size if the accuracy is not ideal. If there are many animals / objects to detect in one frame, and they are relatively small compared with the entire frame, you may use larger inferencing size such as 640 or higher. If there is only one animal / object and it occupies large area of the frame, use smaller size such as 320 for faster processing speed. For general scenarios, size that is smaller than 192 or larger than 1600 is not recommended.

2.2.4. '*Test Detectors*'

To test a trained Detector, you need to prepare a ground-truth testing image dataset, just like what you do for preparing the training image dataset, i.e., annotate the images with the same object / animal names as those in the Detector to test. The mean average precision (mAP) of the selected Detector will be printed out at the end of testing.

2.2.5. '*Generate Behavior Examples*'

To teach *LabGym* to recognize behaviors that are defined by you, you need to use the sorted behavior examples to train a Categorizer. But first, you need to generate some unsorted behavior examples and sort them. **DO NOT** change the file names for the generated behavior

examples since they contain the key information for later training steps. The function of ‘Sort Behavior Examples’ will also be based on these file names.

(Button) ‘Specify the mode of behavior examples to generate’

There are 2 modes of interactive behaviors that *LabGym* can generate examples and analyze. "Interactive basic" is faster in analysis and is useful in the scenario when you don’t care about which individual does what in an interaction, or all the individuals do the same behavior in an interaction. It analyzes the behaviors of all animals / objects in an interactive pair / group as one and does not distinguish individuals. The behavior examples for this mode contain all the animals in a frame. The behavioral quantifications in this mode are behavior count, behavior latency and behavior duration.

The “Interactive advanced” is slower in analysis than "basic", but it distinguishes different social roles of individuals during close body contact. It can tell which animal does what in an interaction. The behavior examples for this mode contain all “characters” within a “social distance” defined by you, with a spotlight on the main character. The social distance should be an integer, for example, if enter “2”, all animals / objects whose distance from others is within “2 folds of the square root of an averaged animal area” will be included in one behavior example. You can do a rough estimation. If enter “0”, the social distance is set to be infinity and all the animals in a frame will be included. When annotate such examples, you annotate the behavior of the main character. The behavioral quantifications in this mode are behavior count, behavior latency, behavior duration, 3 length parameters, 3 areal parameters, and 4 locomotion parameters.

The “Static images” is for analyzing behaviors in images, which is different from the other three modes that analyze behaviors in videos. The behavior examples generated in this mode only contain images.

(Button) ‘Specify the method to detect animals or objects’

There are two options: ‘*Subtract background*’ and ‘*Use trained Detectors*’

‘*Subtract background*’ is the first choice for videos in which the background is static, the illumination is stable overtime, and the total behavior events are more important than animal IDs, because this method is fast and accurate in such videos but cannot distinguish entangled animals and the IDs might switch after they re-separate. When using this method, you need to specify the scenarios of their experiments: '*Animal brighter than background*', '*Animal darker than background*', or '*Hard to tell*'. A pop-up option '(Optional) load existing background?' will need to be specified. *LabGym* will output the extracted backgrounds (as images) for each video processed. Therefore, this pop-up option can be used to save the step of background extraction if the background for this video has already been extracted and output. You also need to specify whether the illumination in videos is unstable. If the illumination is very stable overtime, choose ‘No’ to increase the processing speed. Finally, you need to specify a time window for background extraction. An appropriate time window for background extraction should be a period (typically 10~60 seconds) during which the animals move around. This time window should be as short as possible for increasing processing speed. There are 3 options to specify this time window: (1) '*Use the entire duration of a video*': not recommended if processing speed is critical. (2) '*Decode from filenames: _xst_ and _xet_*': can be used if multiple videos are selected and the time window for background extraction is different for each video. Two code tags ‘_xst_ and _xet_’ (‘xs’ stands for ‘extraction start time’, ‘xe’ stands for ‘extraction end time’, and ‘t’ should be an integer) need to be added into the file name of each video to let *LabGym* decode the time window. For example, if a video name is ‘XX.avi’, to set the time window for background extraction from the 25th second to the 47th second, rename the video to ‘XX_xs25_xe47.avi’ (the ‘_’ is unnecessary if it is at the end of file name). (3) '*Enter two time points*': can used if only one video is selected, or multiple videos share the same time window for background extraction.

'*Use trained Detectors*' is useful in any kind of videos or experimental settings. It is also useful for differentiate individual animals when they entangle. The only caveat of this method is slow. When using "Static images" mode, you need to specify a detection threshold for the Detector, which is a number between 0 and 100. If you want to reduce false detections, make the threshold larger, for example, 80; if you don't want to miss any possible animal / object of interest, make it smaller, for example, 25.

(Button) 'Specify when generating behavior examples should begin (unit: second)'

- (Pop-up option) 'Illumination shifts?'

This option is only for background subtraction-based detection method. It is useful when there are sudden bright-to-dark or dark-to-bright illumination transitions in videos. If choose 'Yes', there will be 3 options to specify the beginning time: '*Automatic (for light on and off)*', '*Decode from filenames: _bt_*', and '*Enter a time point*'. If choose 'No', only the latter two options can be chosen.

'*Automatic (for light on and off)*' is for videos involving lighting on and off (e.g., optogenetics) and the first time point of illumination change will be automatically detected and used as the beginning time to generate behavior examples.

'*Decode from filenames: _bt_*' can be used if multiple videos are selected and the beginning time for each video is different. You need to add a code tag '_bt_' ('b' stands for 'beginning time' and 't' should be number) into the file name of each video to let *LabGym* decode the beginning time. For example, if a video name is 'XX.avi', to set the beginning time at the 12.35 second, rename the video to 'XX_b12.35.avi' (the '_' is unnecessary if it is at the end of file name).

'*Enter a time point*' can be used if only one video is selected, or multiple videos share the same beginning time.

(Button) 'Specify the number of animals in a video'

There are two options: '*Decode from filenames: _nn_*' and *'Enter the number of animals*'.

*'Decode from filenames: _nn_*' can be used if multiple videos are selected and the number of animals in each video is different. A code tag '_nn_' (the first 'n' stands for 'number of animals' and the second 'n' should be an integer) needs to be added into the file name to let *LabGym* decode the animal number. For example, if a video name is 'XX.avi', to set the number of animals in this video as 8, rename the video to 'XX_n8.avi' (the last '_' is unnecessary if it is at the end of file name). If there are multiple kinds of animals / objects in the videos to detect, you can put in multiple '_nn_' code. The order should be consistent with that of the 'animal / object name' in the Detector to use (when you select a Detector, the animal / object name will display in order, and you can also specify which ones to detect).

'*Enter the number of animals*' can be used if only one video is selected, or multiple videos share the same animal number.

(Button) 'Specify the number of frames for an animation / pattern image'

The animations and their paired pattern images in the behavior examples spans a duration (the number of frames, an integer) defined by you, which should approximate the duration of a behavior episode. This duration needs to be the same across all the behavior examples that are used to train one Categorizer. If the duration of different behavior episode is different, use the longest one.

(Button) 'Specify how many frames to skip when generating two consecutive behavior examples'

LabGym generates a pair of behavior example which spans a user-defined duration at a frame. If two consecutively generated examples are too close in time, say, one generated at the 10^{th} frame and the other at 12^{th} frame, suppose their duration is 10 frames, they will have 8 overlapping frames. These two behavior examples are too similar and will impair the training

efficiency when training a Categorizer. Generating too many such similar examples will also make the example sorting laborious. Therefore, you can choose to skip certain frames between the two consecutively generated examples. A practical recommendation to achieve a balance between getting the perfect examples that just span the behavior episode and reducing the labor is to set this number equal to the duration you set for a behavior episode.

(Button) 'Start to generate behavior examples'

- (Pop-up option) 'Including background?'

  Choose 'No' if the background information is behavior irrelevant.

- (Pop-up option) 'Including body parts?'

  Choose 'Yes' if the motion pattern of individual body parts such as limbs or noses are critical for behavioral identification. If choose 'Yes', a 'STD' need to be entered, which should be an integer between 0 and 255. The STD value decides the threshold to show the how many 'motion pixels' of the body parts in the pattern images. Lager STD, less motion pixels will be shown.

2.2.6. '*Sort Behavior Examples*'

The behavior examples need to be sorted into different folders according to the behavior types they belong to. This is the way how you teach *LabGym* the behaviors defined by you. You can select a number of behavior examples and move them into a folder, or use this button to watch each generated behavior example one by one and sort them using shortcut keys. The more behavior examples for each behavior type are included, the higher accuracy of the trained Categorizer; the more diverse (let the Categorizer see as many variations of each behavior type as possible), the higher accuracy of the trained Categorizer. For general scenarios, 100~200 pairs of behavior examples for each behavior type are enough for training a Categorizer with high accuracy.

Importantly, when you define behavior categories, make the list of categories as complete as possible. For example, there are ‘A’, ‘B’, ‘C’, and ‘D’, 4 types of behaviors showing frequently in all experimental sessions, and if you only sort behavior ‘A’ and ‘C’ because they are the behaviors of your interest. After training, the Categorizer can only recognize behavior ‘A’ and ‘C’ and when it sees type ‘B’ during analysis, it will forcibly categorize the ‘B’ into either ‘A’ or ‘C’, based on how similar ‘B’ is to ‘A’ or ‘C’. If a behavior type happens in a low frequency in all experimental sessions, then you may not need to include it since it will not affect the categorization accuracy too much.

The behavior examples within the same behavior category should share similar features or patterns that can be identified by watching them with your naked eyes. Likewise, the behavior examples belonging to different behavior categories should possess different features or patterns that can be identified by watching them with your naked eyes. *LabGym* distinguishes behavior examples based on visually identifiable features or patterns in them without knowing additional information from other sources. So, if you cannot identify the shared features or patterns of behavior examples within the same behavior category by just watching them, *LabGym* will not be able to learn this category well. Likewise, if you cannot distinguish the behavior examples of different behavior categories by just watching them, *LabGym* will not be able distinguish them either.

2.2.7. ‘*Train Categorizers*’

There are two parts: one is for preparing the training examples; the other is for using the prepared training examples to train a Categorizer. The Categorizers are the core of *LabGym*, which identifies behaviors defined by you during analysis. Trained Categorizers will be stored in the Categorizer list in *LabGym* and ready to use. The complexity of Categorizer is customizable. It takes some trial and errors to find the best setting that perfectly fits your behavioral data.

(Button) ‘Specify the type / complexity of the Categorizer to train’

- (Pop-up option) 'Categorizer types'

If the processing speed is critical to you, choose the Categorizer with only Pattern Recognizer. If you want to maximize the analysis accuracy and care less about the analysis speed, choose the Categorizer with both Animation Analyzer and Pattern Recognizer.

- (Pop-up option) Behavior mode'

You need to indicate which mode of behavior examples are used for training the Categorizer. For “Interact basic” and “Interact advanced” behavior examples, you can find the mode information in the filenames of the behavior examples. “_itbs” is “Interact basic” and “_itadv” is “Interact advanced”. If the behavior mode is “Interact advanced”, you also need to specify a social distance, which can also be found in the filenames of the behavior examples with “_scdtXX_” where “XX” is the number you need to enter for the social distance.

- (Pop-up option) 'Complexity level'

There are 7 complexity levels (1~7, from simpler to more complex) of either Animation Analyzer or Pattern Recognizer. You can start with simpler levels like 1 ~ 3 and go more complex if the accuracy is not ideal. If the Animation Analyzer is gray scale (animal / object color is behavior irrelevant), you can make the complexity levels of Pattern Recognizer a little higher than that of Animation Analyzer because the Pattern Recognizer analyzes pattern images containing additional color information, which is indicates the temporal sequence of a behavior.

(Button) ‘Specify the input shape for Animation Analyzer / Pattern Recognizer’

You can start with the smaller input frame / image sizes such as 16 or 32. If there are many detailed appearance features in the frames that are behavior relevant, go larger size such as 64 or 96 or even 128. And always go deeper (increase complexity level) first, rather than go wider (increase input frame / image size) first.

- (Pop-up option) 'Grayscale Animation Analyzer?'

  Choose ‘Yes’ if the color of animals is behavior irrelevant.

(Button) ‘Specify the methods for data augmentation’

Briefly, '*rotation*' will rotate the animal blobs; '*flipping*' will flip the animal blobs; '*brightening*' and '*dimming*' will increase and decrease the brightness of the animal blobs; '*shearing*' will distort the animal blobs; '*rescaling*' will change the width / height ratio of the animal blobs; '*deletion*' will delete one or two frames in the animations (mimic the scenario in which animals are not detected in one or two frames during analysis).

- (Pop-up option) 'Use default augmentation?'

  Default augmentation will use '*rotation*', '*flipping*', '*brightening*' and '*dimming*'.

- (Pop-up option) 'Augment validation data?'

  If the total number of behavior example pairs used for training a Categorizer is less than 1,000 before augmentation, choose “Yes” for this option.

2.2.8. ‘*Test Categorizers*’

To test a trained Categorizer, you need to sort a ground-truth testing folder, in which different behavior examples are stored in different subfolders that are named after the behavior name. The subfolder names (behavior names) should match the behavior names in the selected Categorizer. The precision, recall, and f1 score of each behavior category, as well as the overall accuracy will be printed out at the end of testing.

2.3. ‘*Analysis Module*’

Automatically track animals / objects of your interest, identify and quantify their behaviors in videos or images, and automatically mine the analysis results to display the data details that show statistically significant differences among groups of your selection.

2.3.1. '*Analyze Behaviors*'

If analyzing videos, will output a raster plot for all behavior events for all videos, an annotated video copy for each video, various spreadsheets storing quantification results for each selected behavior parameter. If analyzing images, will output each annotated images and spreadsheets storing the behavior counts and probabilities.

(Button) 'Select a Categorizer for behavior classification'

Choose a Categorizer for behavior classification in analysis. You can also let *LabGym* just track the animals and calculate their motion parameters and body kinematics (only for videos). If the former, you need to specify a 'Uncertainty level' for the Categorizer. This number in percentage determines the threshold for the Categorizer to output an 'NA' for behavioral classification. For example, when the probabilities of behavior A, B, and C are 60%, 10%, and 30%, respectively. If the uncertain level is set to be 31, the behavior classification will be 'NA', as the uncertainty level exceeds the difference between probability of the highest-likely behavior (A, 60%) and the second highest-likely behavior (C, 30%), which is 30%. Setting uncertainty level can reduce the possible false positives in behavior classification since ambiguous classification will be output as an 'NA'. If you want the behavior classification more precise, set the uncertainty level higher, for example, 50; if you don't want to miss any possible detections of a behavior (more sensitive), set it to 0. If choose to not classify behaviors, you need to specify a time window for calculating the motion parameters and body kinematics.

(Button) 'Select the video(s) / image(s) for behavior analysis'

- (Pop-up option) '(Optional) resize the frames / images?'

  Downsizing the frame / image will exponentially increase the analysis speed. This is **highly recommended** especially when using Detector-based detection method and GPUs are not

available. The analysis accuracy will not decline if the animal size after downsizing is still larger than the input size of the Categorizer used for analysis. For example, suppose the original size of a video frame is 1000 x 500, the size of an animal is approximately 1/4 to that of a frame, and input size of the Categorizer is 48 x 48. After downsizing the video frames to 500 x 250, the animal size is roughly 125 x 63 and is still larger than the input size of the Categorizer (48 x 48). In this scenario, the analysis accuracy will not decline since the animal blob will be downsized to 48 x 48 anyway when input to the Categorizer for analysis.

(Button) ‘Specify the method to detect animals or objects’

There are two options: ‘*Subtract background*’ and ‘*Use trained Detectors*’

‘*Subtract background*’ is the first choice for videos in which the background is static, the illumination is stable overtime, and the total behavior events are more important than animal IDs, because this method is fast and accurate in such videos but cannot distinguish entangled animals and the IDs might switch after they re-separate. When using this method, you need to specify the scenarios of their experiments: '*Animal brighter than background*', '*Animal darker than background*', or '*Hard to tell*'. A pop-up option '(Optional) load existing background?' will need to be specified. *LabGym* will output the extracted backgrounds (as images) for each video processed. Therefore, this pop-up option can be used to save the step of background extraction if the background for this video has already been extracted and output. You also need to specify whether the illumination in videos is unstable. If the illumination is very stable overtime, choose ‘No’ to increase the processing speed. Finally, you need to specify a time window for background extraction. An appropriate time window for background extraction should be a period (typically 10~60 seconds) during which the animals move around. This time window should be as short as possible for increasing processing speed. There are 3 options to specify this time window: (1) '*Use the entire duration of a video*': not recommended if processing speed is critical. (2) '*Decode from filenames: _xst_ and _xet_*': can be used if multiple videos are

selected and the time window for background extraction is different for each video. Two code tags '_xst_ and _xet_' ('xs' stands for 'extraction start time', 'xe' stands for 'extraction end time', and 't' should be an integer) need to be added into the file name of each video to let *LabGym* decode the time window. For example, if a video name is 'XX.avi', to set the time window for background extraction from the 25th second to the 47th second, rename the video file to 'XX_xs25_xe47.avi' (the '_' is unnecessary if it is at the end of file name). (3) '*Enter two time points*': can used if only one video is selected, or multiple videos share the same time window for background extraction.

'*Use trained Detectors*' is useful in any kind of videos or experimental settings. It is also useful for differentiate individual animals when they entangle. The only caveat of this method is slow.

(Button) 'Specify when the analysis should begin (unit: second)'

- (Pop-up option) 'Illumination shifts?'

Specify whether there are sudden bright-to-dark or dark-to-bright illumination transitions. If choose 'Yes', there will be 3 options to specify the beginning time: '*Automatic (for light on and off)*', '*Decode from filenames: _bt_*', and '*Enter a time point*'. If choose 'No', only the latter two options can be chosen.

'*Automatic (for light on and off)*' is for videos involving lighting on and off (e.g., optogenetics) and the first time point of illumination change will be automatically detected and used as the beginning time to generate behavior examples.

'*Decode from filenames: _bt_*' can be used if multiple videos are selected and the beginning time for each video is different. You need to add a code tag '_bt_' ('b' stands for 'beginning time' and 't' should be number) into the file name of each video to let *LabGym* decode the beginning time. For example, if a video name is 'XX.avi', to set the beginning time at the 12.35 second, rename the video file to 'XX_b12.35.avi' (the '_' is unnecessary if it is at the end of file name).

'*Enter a time point*' can be used if only one video is selected, or multiple videos share the same beginning time.

(Button) ‘Specify the number of animals in a video’

There are two options: ‘*Decode from filenames: _nn_*' and *'Enter the number of animals*'.

*'Decode from filenames: _nn_*' can be used if multiple videos are selected and the number of animals in each video is different. A code tag ‘_nn_’ (the first ‘n’ stands for ‘number of animals’ and the second ‘n’ should be an integer) needs to be added into the file name to let *LabGym* decode the animal number. For example, if a video name is ‘XX.avi’, to set the number of animals as 8, rename the video to ‘XX_n8.avi’ (the last ‘_’ is unnecessary if it is at the end of file name). If there are multiple kinds of animals / objects in the videos to detect, you can put in multiple ‘_nn_’ code. The order should be consistent with that of the ‘animal / object name’ in the Detector to use (when you select a Detector, the animal / object name will display in order, and you can also specify which ones to detect).

'*Enter the number of animals*' can be used if only one video is selected, or multiple videos share the same animal number.

(Button) ‘Select the behaviors for annotations and plots’

- (Pop-up option) Specify individual-specific behaviors?'

If the selected Categorizer is for “Interactive advanced” mode, you can select individual specific behaviors to initiate the “behavior-guided” identity correction. For example, behavior A only occurs in a specific individual whose initial ID is “0”, if its ID switches with another individual (e.g., ID “1”) because of false detection in a frame, after that, when behavior A occurs, its ID will switch back to “0” because behavior A only occurs in this specific individual “0”. In this way, this individual “0” can correctly maintain its ID during the entire analysis period.

- (Pop-up option) 'Specify colors for behaviors?'

Specify a color to represent a behavior category in the annotated videos and the raster plot for behavior events. In the annotated videos, the value of % confidence of behavior categorization will be shown; in the raster plot, the color intensity indicates the value of % confidence, from 0% of the color intensity (clear) indicating 0% of the confidence, to 100% of the color intensity indicating 100% of the confidence. If choose not to specify the colors, *LabGym* will use the default colors to represent the behaviors.

- (Pop-up option) 'Legend in video?'

  Specify whether to show the legend of behavior names in the annotated videos.

(Button) 'Select the quantitative measurements for each behavior'

There are 13 quantitative measurements (parameters) for each behavior for you to choose:

- ◊ The *count* is the summary of the behavioral frequencies, which is the occurrence number of a behavior within the entire duration of analysis. Consecutive single occurrences (at a single frame) of the same behavior are considered as one count.
- ◊ The *latency* is the summary of how soon a behavior starts, which is the time starting from the beginning of the analysis to the time point that the behavior occurs for the first time.
- ◊ The *duration* is the summary of how persistent a behavior is, which is the total time of a behavior within the entire duration of analysis.
- ◊ The *speed* is the summary of how fast the animal moves when performing a behavior, which is the total distance traveled (can be back and forth) (d) (between the two centers of mass of the animal) during the time window (t_w) for categorizing the behavior divided by t_w.
- ◊ The *velocity* is the summary of how efficient the animal's movement is when performing a behavior, which is the shortest distance between the start and the end positions (dt)

(between the two centers of mass of the animal) divided by the time (t) that such displacement takes place.

- ◊ The *acceleration* / *velocity reduction* is the summary of how fast the animal's velocity changes while performing a behavior, which is the difference between maximum velocity (v_{max}) and minimum velocity (v_{min}) divided by the time (t_v) that such velocity change takes place.
- ◊ The *distance* is the total distance traveled of the animal by performing a behavior within the entire duration of analysis.
- ◊ The *intensity (area)* / *intensity (length)* is the summary of how intense a behavior is, which is the accumulated proportional changes of the animal body area (a) / length (l) between frames divided by the time window for categorizing the behaviors (t_w) when performing a behavior.
- ◊ The *magnitude (area)* / *magnitude (length)* is the summary of the motion magnitude, which is the maximum proportional change in animal body area (a) or length (l) when performing a behavior.
- ◊ The *vigor (area)* / *vigor (length)* is the summary of how vigorous a behavior is, which is the magnitude (area) / magnitude (length) divided by the time (t_a or t_l) that such a change takes place.

Details on how they are calculated are in *LabGym* paper [1] [2].

- (Pop-up option) 'Normalize the distances?'

If choose 'No', all the distances will be output in pixels. The unit of all the distance related measurements will be 'pixel' or pixel related. If choose 'Yes', all the distances (calculated in pixels) will be normalized to (divided by) the size of a single animal (also calculated in pixels). In this scenario, all distance related measurements will be normalized measurements (e.g., normalized speed) and will not have a unit. In this way, you do not need to worry about the ratio

of pixel / actual size (length) across different videos with different recoding methods, if the size of animals used in these videos are consistent. The ratio of pixel / actual size (length) is not easy to obtain and is subject to change easily (e.g., when the zoom-in level changes). With the option of normalizing distances to the size of a single animal, you can compare the analysis results across different recordings or experimental sessions without worrying about the changes in the ratio of pixel / actual size (length).

2.3.2. '*Mine Results*'

Automatically performs parametric / non-parametric statistical analysis among groups that you selected, according to the data distribution, to compare the mean / median of different groups and display the data details that show statistically significant difference.

The Shapiro test will be first performed to assess the normality of data distribution. For normally distributed data, if unpaired, unpaired t-test for 2 groups, ANOVA for more than 2 groups, with either Tukey (comparing each pair) or Dunnett's (comparing all groups against the control group) posthoc comparison; if paired, paired t-test for 2 groups, ANOVA for more than 2 groups, with either Tukey (comparing each pair) or Dunnett's (comparing all groups against the control group) posthoc comparison. For data that is not normally distributed, if unpaired, Mann Whitney U test for 2 groups, Kruskal Wallis for more than 2 groups, with Dunn's posthoc comparison for both comparison of all groups and against control; if paired, Wilcoxon test for 2 groups, Friedman for more than 2 groups with Dunn's posthoc. The selections of the tests are consistent with those in GraphPad Prism 9.

2.3.3. '*Generate Behavior Plot*'

Generate a raster plot for all behavior events of every individual in 'all_events.xlsx' file, which is output by LabGym analysis. This function is used to re-generate the raster plot if the original

raster plot generated by LabGym does not meet the users' need (e.g., there are non-target animal / object falsely detected in the analysis).

Use *LabGym* via command prompt (requires *LabGym* v2.4.6 or later)

1. Initiate the Python3 interactive shell

Open the Terminal (Mac / Linux) or Command Prompt (Windows), and type:

```
python3
```

2. Documentation on using *LabGym* via command prompt

Note: this documentation is only for the functions that need to use Detectors, which can be accelerated with the usage of powerful GPUs, so that users can use command prompt to run these functions on servers or clusters that have GPUs. For other functions, please use *LabGym* GUI on a local computer, since a common computer is sufficient for those tasks.

2.1. Train a Detector

```
from LabGym.detector import Detector

DT=Detector()

DT.train(
        path_to_annotation,
        # the path to the .json file that stores the annotations in coco format
        path_to_trainingimages,
        # the folder that stores all the training images
        path_to_detector,
        # the path to store the trained Detector
        iteration_num,
        # the number of training iterations
```

```
        inference_size
        # the Detector inferencing frame size
        )

```

2.2. Test a Detector

```
from LabGym.detector import Detector

DT=Detector()

DT.test(
        path_to_annotation,
        # the path to the .json file that stores the annotations in coco format
        path_to_testingimages,
        # the folder that stores all the training images
        path_to_detector,
        # the path to the Detector to test
        output_path
        # the path to store the testing results
        )

```

2.3. Generate behavior examples (using trained Detectors)

```
from LabGym. analyzebehavior_dt import AnalyzeAnimalDetector

AAD=AnalyzeAnimalDetector()
```

```
AAD.prepare_analysis(
        path_to_detector,
        # path to the Detector
        path_to_video,
        # path to the video for generating behavior examples
        results_path,
        # the folder that stores the generated behavior examples
        animal_number,
        # the number of animals / objects in a video
        animal_kinds,
        # the catgories of animals / objects to be analyzed
        behavior_mode,
        # 0: non-interactive; 1: interactive basic; 2: interactive advanced; 3: static image
        framewidth=None,
        include_bodyparts=False,
        # whether to include body parts in pattern images
        std=0,
        # for excluding static pixels in pattern images
        t=0,
        # start time point
        duration=5,
        # the duration for example generation
        length=15,
        # the duration (number of frames) of a behavior example (a behavior episode)
        social_distance=0
```

```
# the distance determining which individuals in an interaction
)

# if generate behavior example in ‘non-interactive’ mode
AAD.generate_data(
background_free=True,
# whether to include background in animations
skip_redundant=1
# the interval (in frames) of two consecutively generated behavior example pairs
)

# if generate behavior example in ‘interactive basic’ mode
AAD. generate_data_interact_basic(
background_free=True,
# whether to include background in animations
skip_redundant=1
# the interval (in frames) of two consecutively generated behavior example pairs
)

# if generate behavior example in ‘interactive advanced’ mode
AAD. generate_data_interact_advance(
background_free=True,
# whether to include background in animations
skip_redundant=1
# the interval (in frames) of two consecutively generated behavior example pairs
)
```

2.4. Analyze behaviors in a video (using trained Detectors)

```
from LabGym. analyzebehavior_dt import AnalyzeAnimalDetector

AAD=AnalyzeAnimalDetector()

AAD.prepare_analysis(
        path_to_detector,
        # path to the Detector
        path_to_video,
        # path to the video to analyze
        results_path,
        # the folder that stores the analysis results
        animal_number,
        # the number of animals / objects in a video
        animal_kinds,
        # the catgories of animals / objects to be analyzed
        behavior_mode,
        # 0: non-interactive; 1: interactive basic; 2: interactive advanced; 3: static image
        names_and_colors=None,
        # the behavior names and their representing colors
        framewidth=None,
        # the width of the frame to resize
        dim_tconv=8,
        # input dim of Animation Analyzer
```

```
    dim_conv=8,
    # input dim of Pattern Recognizer
    channel=1,
    # color of Animation Analyzer
    include_bodyparts=False,
    # whether to include body parts in pattern images
    std=0,
    # for excluding static pixels in pattern images
    categorize_behavior=True,
    # whether to categorize behaviors or just track animals
    animation_analyzer=True,
    # whether to include Animation Analyzer
    t=0,
    # start time point
    duration=5,
    # the duration for example generation
    length=15,
    # the duration (number of frames) of a behavior example (a behavior episode)
    social_distance=0
    # the distance determining which animals / objects form a interactive group
    )

    AAD.acquire_information(
    batch_size=1,
    # for batch inferencing by the Detector
    background_free=True
```

```
# whether the background is included during analysis
)

# if the behavior mode is not ‘interactive basic’ mode
AAD.craft_data()

# if want to categorize the behaviors
AAD.categorize_behaviors(
path_to_categorizer,
# path to the Categorizer
uncertain=0
# a threshold between the highest the 2nd highest probability of behaviors to
# determine if output an 'NA' in behavior classification
)

AAD.annotate_video(
animal_to_include,
# animals / objects that are included in the annotation
behavior_to_include,
# behaviors that are included in the annotation
show_legend=True
# whether to show the legend of behavior names in video frames
)

AAD.export_results(
normalize_distance=True,
```

```
# whether to normalize the distance (in pixel) to the animal contour area
parameter_to_analyze=[]
# the behavior parameters that are selected in the analysis
# names are: 'count', 'duration', 'latency', '3 length parameters',
# '3 areal parameters', '4 locomotion parameters'
)
```

REFERENCES

1. Hu Y, Ferrario CR., Maitland AD, Ionides RB, Ghimire A, Watson BO, Iwasaki K, White H, Xi Y, Zhou J, and Ye B. (2023). LabGym: Quantification of user-defined animal behaviors using learning-based holistic assessment. Cell Rep Methods 3, 100415. 10.1016/j.crmeth.2023.100415.
2. Goss K, Bueno-Junior LS, Stangis K, Ardoin T, Carmon H, Zhou J, Satapathy R, Baker I, Jones-Tinsley CE, Lim MM, Watson BO, Sueur C, Ferrario CR, Murphy GG, Ye B, and Hu Y. (2024). Quantifying social roles in multi-animal videos using subject-aware deep-learning. bioRxiv. 10.1101/2024.07.07.602350
3. Van Rossum, G., and Drake, F.L. (2000). Python reference manual (iUniverse Indiana).
4. Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2.