



# Deepbrain Project

מגישה: נופר ברט

## תוכן עניינים

2-4	.....התקנה על מחשב חדש
5	.....הרצה של הפרויקט
6-8	.....תוכן הקבצים
6-7	..... <i>backend</i>
8	..... <i>frontend</i>
9-10	.....מיקומים של פרטים חשובים
11-15	.....יצירת רשת נוירונים חדשה
15-20	.....עבודה עם ה- <i>User Interface</i>
21	.....שגיאה במערכת
22	.....דוקומנטציה ( <i>documentation</i> )

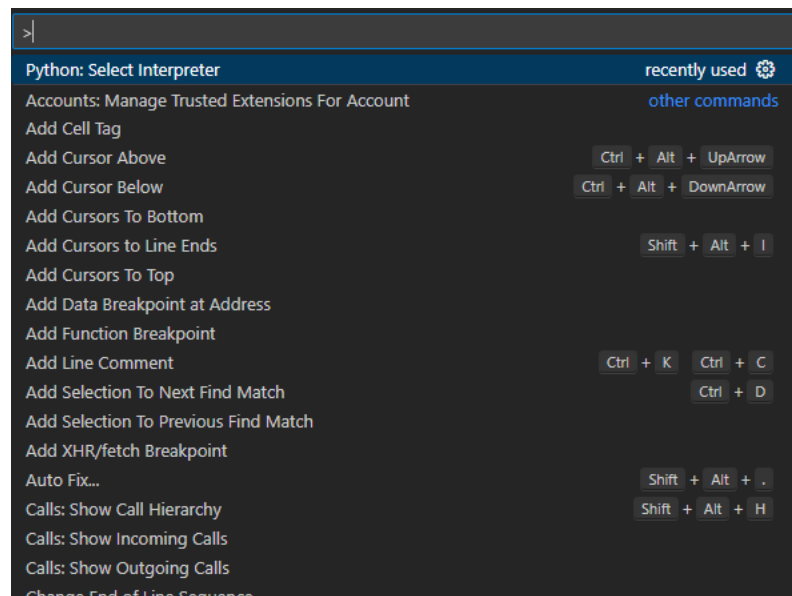
# Deepbrain

## התקנה על מחשב חדש

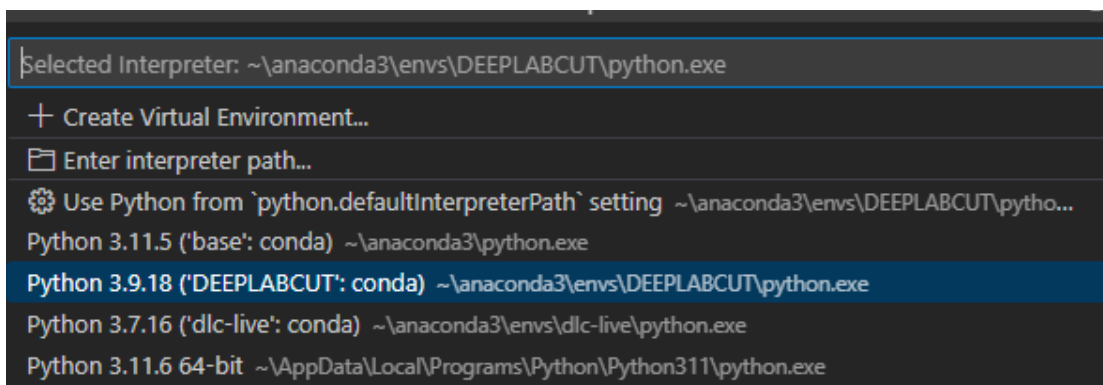
ראשית וודאו שיש לכם *python* ו-*VScode* מותקנים. יש צורך בשניהם על מנת להריץ את הפרויקט. בנוסף, וודאו כי יש לכם *Anaconda* מותקן על המחשב, הוא הכרחי על מנת להשתמש ב-*DeepLabCut*.

## הוראות להתקנת DeepLabCut:

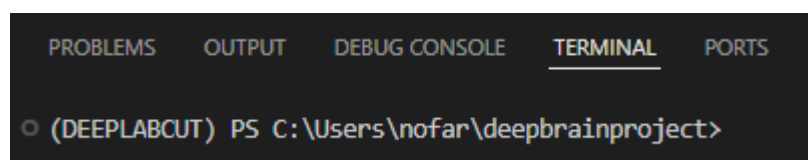
1. הורידו את ה-*Conda file* ומקמו אותו במיקום שאתם יודעים שלא יימחק מהמחשב.
2. פתחו את תוכנת *Anaconda Navigator*, בחרו משם את *CMD.exe Prompt*, והריצו את הפקודות לפי הקישור שבכותרת.
3. אם תרצו להשתמש ב- *DeepLabCut* בהרצת קובץ מסוים תצטרכו [לשנות את האינטרפרטר](#), כך שהוא יזהה שסביבת העבודה שלכם משתמשת בספרייה. לחצו על *Ctrl + shift + P* בתוך *VScode* וייפתח לכם החלון הבא-



לאחר לחיצה על האופציה הראשונה (הכחולה בתמונה), ואם אתם לא רואים אפשר להקליד (והוא ימצא), תוכלו לבחור מתוך כל גרסאות ה-*python* את זאת שתומכת ב-*DeepLabCut*.



תוכלו לראות שהשינוי הצליח בטרמינל לפי השם *DeepLabCut* לפני ה-*path*:



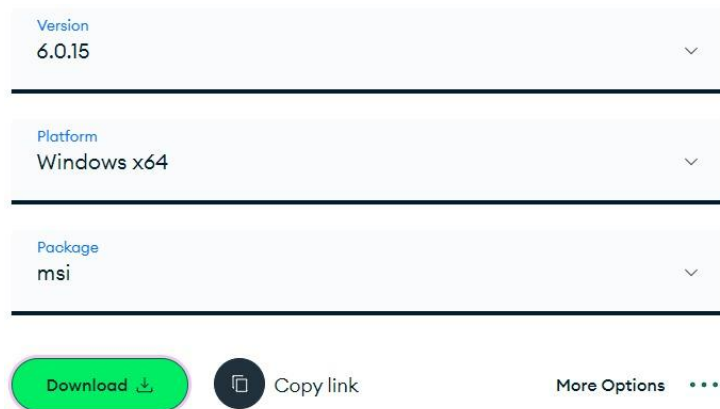
### [הוראות להתקנת NodeJS:](#)

הפרויקט משתמש ב-NodeJS לשפת שרת- תקשורת בין חלקים שונים בפרויקט.  
ההורדה פשוטה- לחצו על כפתור ההורדה ובשלב ההתקנה סמנו ✓ על להתקין הכל.

### [הוראות להתקנת MongoDB:](#)

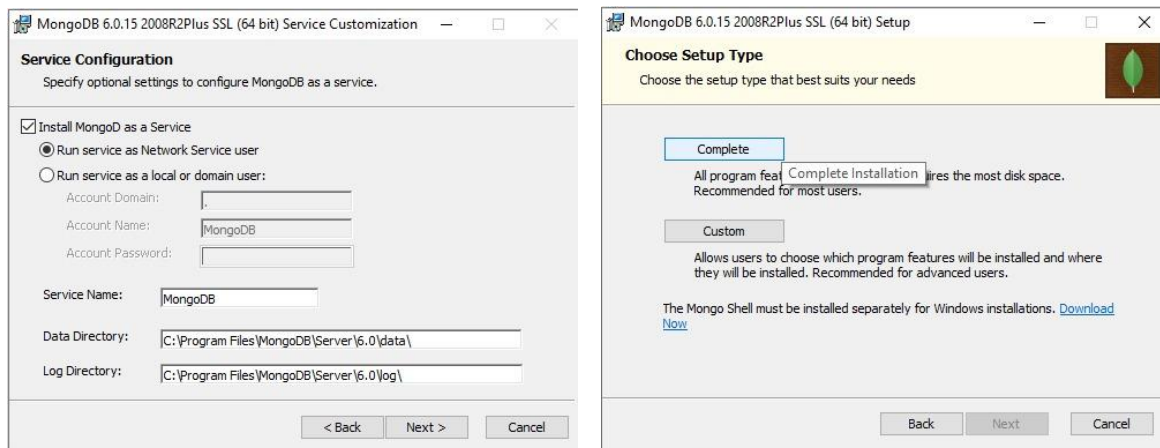
הפרויקט משתמש בבסיס נתונים לאחסון יעיל ומהיר יותר של חלק מהנתונים. בנוסף סמנו ✓ על התקנת *compass*.

שימו לב שאתם מתקינים את הגרסה הזו ולא אחרת!



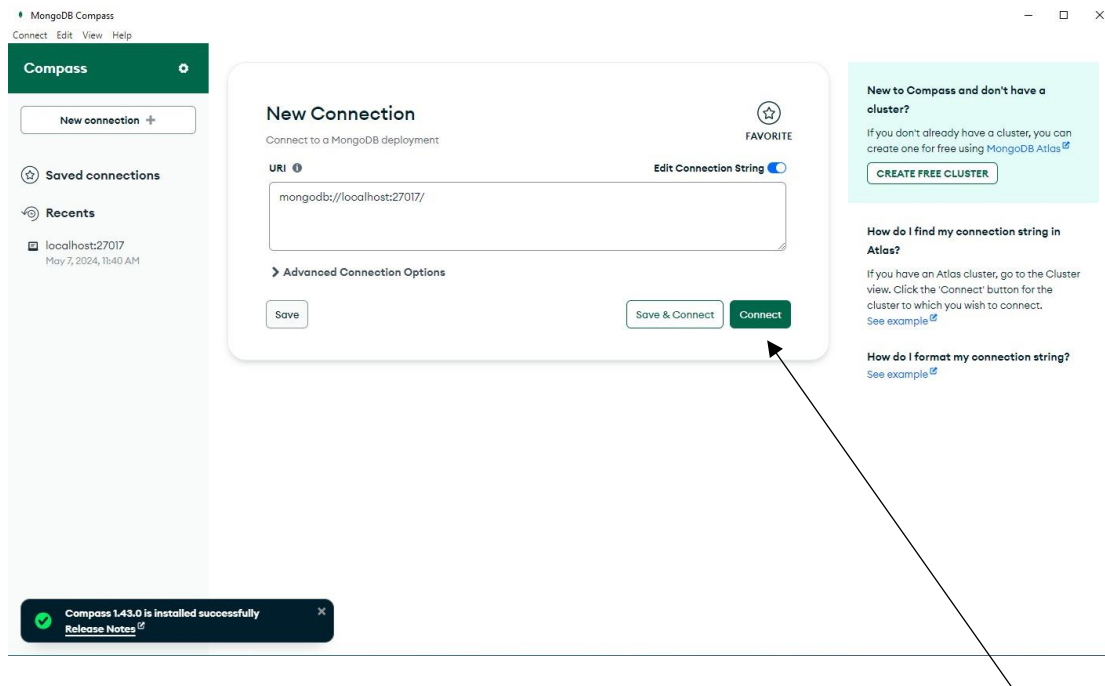
ולאחר מכן סמנו בצורה הבאה-

לחצו על *complete*-



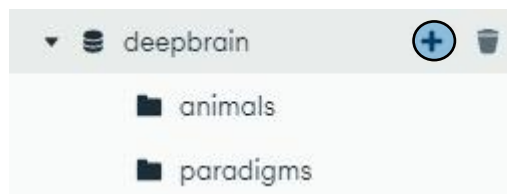
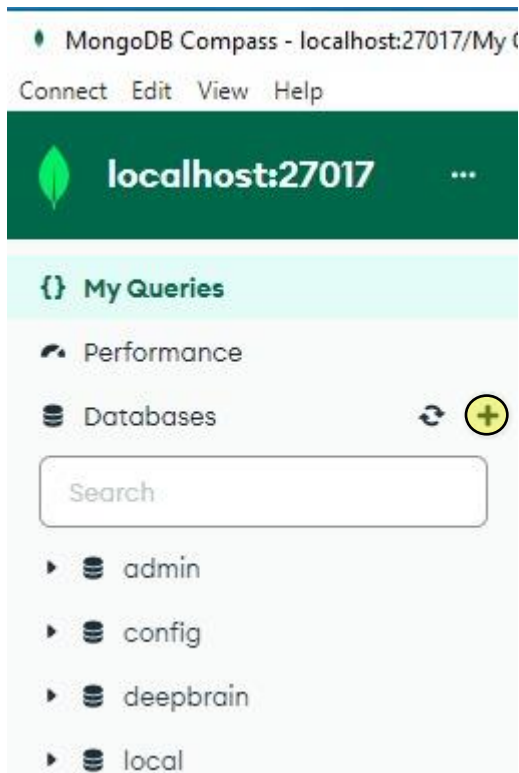
בעמוד האחרון סמנו ✓ על התקנת *compass*. [התקינו גם את mongosh](#) בגרסה הישנה יותר.

לאחר שהורדתם את *MongoDB* נותר לכם ליצור דאטא בייס.



לחצו על *Connect*.

בעמוד שנפתח לכם לחצו על הפלוס (מודגש בצהוב) ליד המילה *Databases*, והוסיפו דאטא בייס בשם *deepbrain*.



בתוך הדאטא בייס תצרו שני *Collections*, על ידי לחיצה על הפלוס (מודגש בכחול).

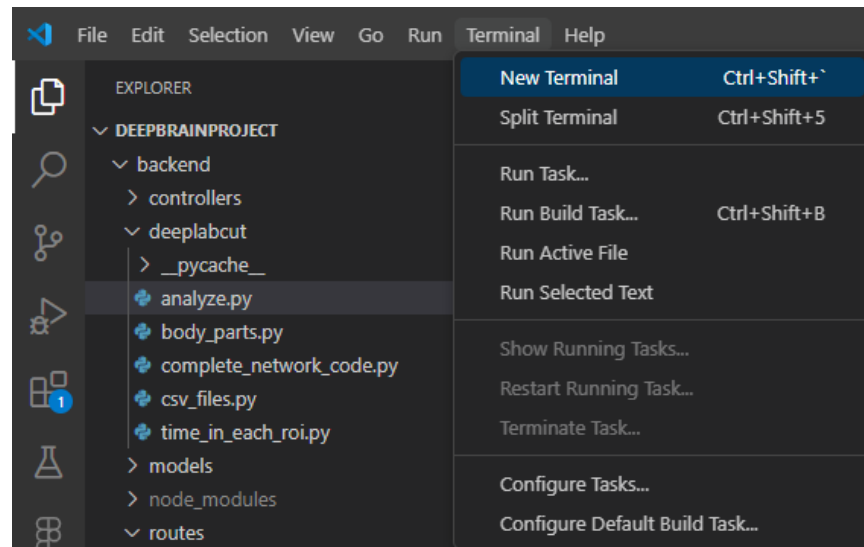
אחד בשם *animals* ואחד בשם *paradigms*.

**וודאו כי כל השמות באותיות קטנות בלבד!!**

מעתה ניתן לסגור את החלון של *Compass* אין צורך לפתוח את *MongoDB* כלל. הוא יפעל ברקע התוכנה מעצמו.

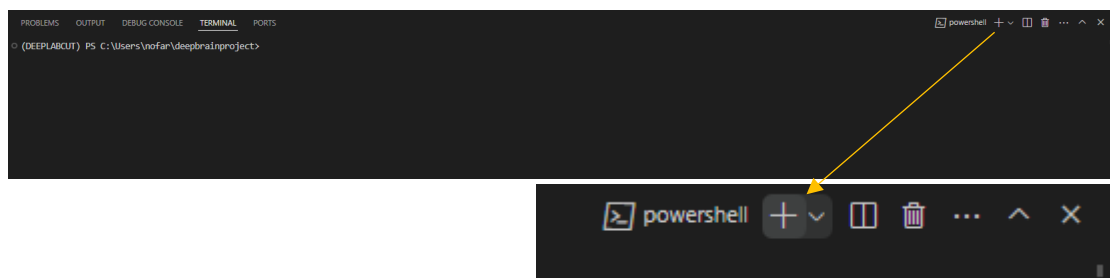
## הרצה של הפרויקט

פתחו את תיקיית הפרויקט. הפרויקט מורכב משני חלקים אותם תצטרכו להריץ: ראשית לחצו על לחצן ה-*Terminal*.



ופתחו טרמינל חדש כדי להריץ את הפרויקט.

בצד ימין של החלון ייפתח לכם טרמינל להרצה.



לחצו על כפתור הפלוס לפתיחת טרמינל נוסף. תצטרכו להשתמש בשניים. תראו מצב כזה מימינכם:

הטרמינל הראשון יישמש אותנו לבקשות מהשרת ולבקשות מבסיס הנתונים ואילו הטרמינל השני יישמש לתצוגה של האתר.

בטרמינל הראשון תכתבו `cd backend/` ואז `npm start`

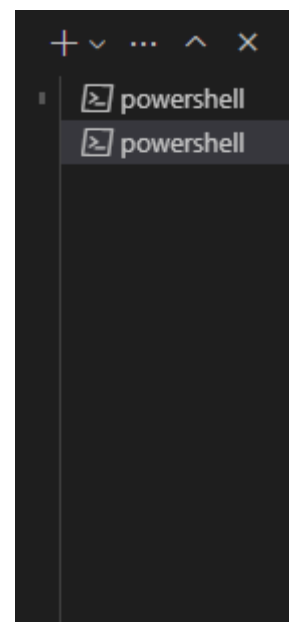
כרגע אתחלתם את בסיס הנתונים.

בטרמינל השני תכתבו `cd frontend/` ואז `npm start`

יקפוץ לכם מסך האתר בדפדפן (שימו לב שהפרויקט יעבוד גם ללא חיבור לאינטרנט).

במקרה שהאתר לא מגיב שגיאה תופיע בטרמינלים לפי סוג השגיאה.

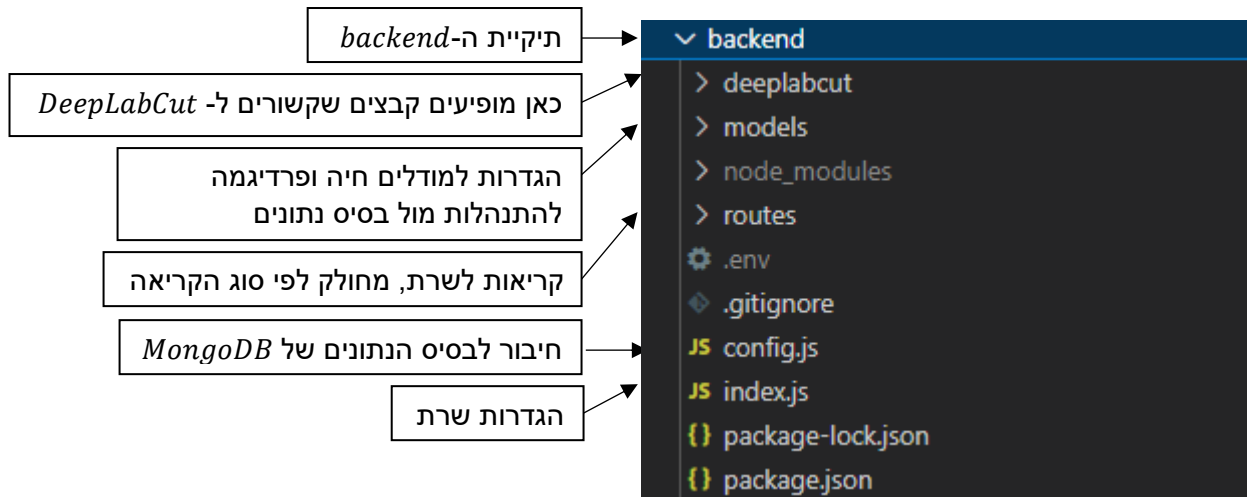
שימו ❤️ ששני הטרמינלים מאותחלים, אחרת האתר לא יעבוד.



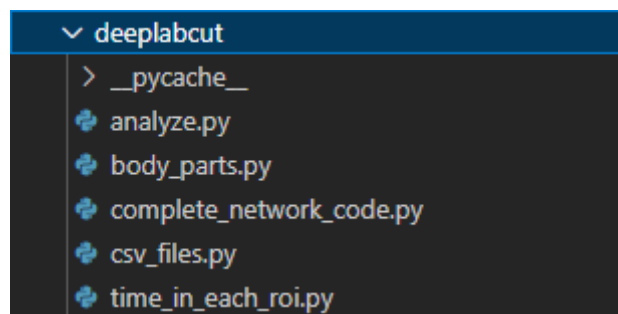
## תוכן הקבצים

הפרויקט מחולק לשתי תיקיות מרכזיות אותן הרצתם בטרמינל, נעבור על תוכן כל תיקייה רלוונטית. בנוסף, ישנם קבצים שאינם חלק מהפרויקט עליהם נרחיב בהמשך-

-backend



התיקייה הראשונה היא תיקיית ה-DeepLabCut.



שימו, ❤️, הקבצים בתיקייה כתובים בשפת התכנות פייתון. נעבור על המטרה של כל אחד מהם ומה הוא מכיל מבחינת הלוגיקה.

**analyze.py** - ניתוח של המידע שהתקבל מהסרטונים עם ה-*labels* כלומר יצירת גרפים. הקוד מקבל מספר פרמטרים:

1. **h5** - הקובץ ש-DeepLabCut מייצר לאחר שהוא ניתח סרטון מסוים (לאחר שהוסיף לסרטון תוויות). בקובץ הזה נמצאים כל הנתונים על הסרטון, כלומר מה המיקום  $(x, y)$  של כל אחד מאיברי הגוף של העכבר בכל פריים לאורך הסרטון ומה הלייקליהוד שלהם. נשתמש בו ליצירת הגרפים.
  2. **bpt** - חלק הגוף אותו נרצה לנתח.
  3. **graph\_generator** - אילו גרפים נרצה ליצור (הקוד נקרא גם כשמשנים כותרת לאחד הגרפים ולכן מייצרים רק את הגרף בו נעשה השינוי).
  4. והלאה. כותרת הגרף. יכולה להיות גם ריקה, נגיע לזה בהסבר על השימוש בממשק המשתמש.
- נשתמש בספרייה בשם *DLC2Kinematics* שמאפשרת מציאת מהירות ותאוצה עבור כל איבר. הקוד מנקה רעשים ומייצר שלושה גרפים - מהירות, מיקום ותבנית התנועה, כפי שיוסבר בצורה מפורטת בתיעוד הקוד.

`body_parts.py` - החזרת סוגי האיברים (רשימת האיברים) שמופיעה בקובץ הקונפיגורציה `config.yaml` כפי שיפורט בהמשך.

`complete_network_code.py` - יצירת סרטונים עם תוויות (`labels`) עבור תיקייה נבחרת.

הקוד מקבל מספר פרמטרים:

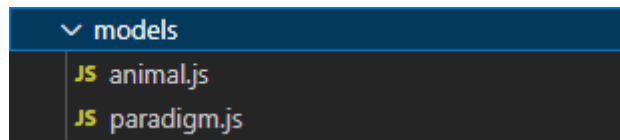
1. `config` - ה-`path` של קובץ הקונפיגורציה של רשת הנירונים המאומנת.

2. `video_path` - התיקייה שמכילה את הסרטונים להם נרצה להוסיף תוויות.

`csv_files.py` - יצירת קבצי אקסל עם המידע מהקובץ `h5` של סרטון ספציפי. הקוד מייצר שלושה קבצי אקסל- מיקום, מהירות ותאוצה של כל האיברים לאורך כל פריים ופריים בסרטון.

`time_in_each_roi.py` - קובץ מהאתר של `DeepLabCut` לשרטוט גרף לאורך זמן במקום לאורך `frames` (למקרה שתמצאו).

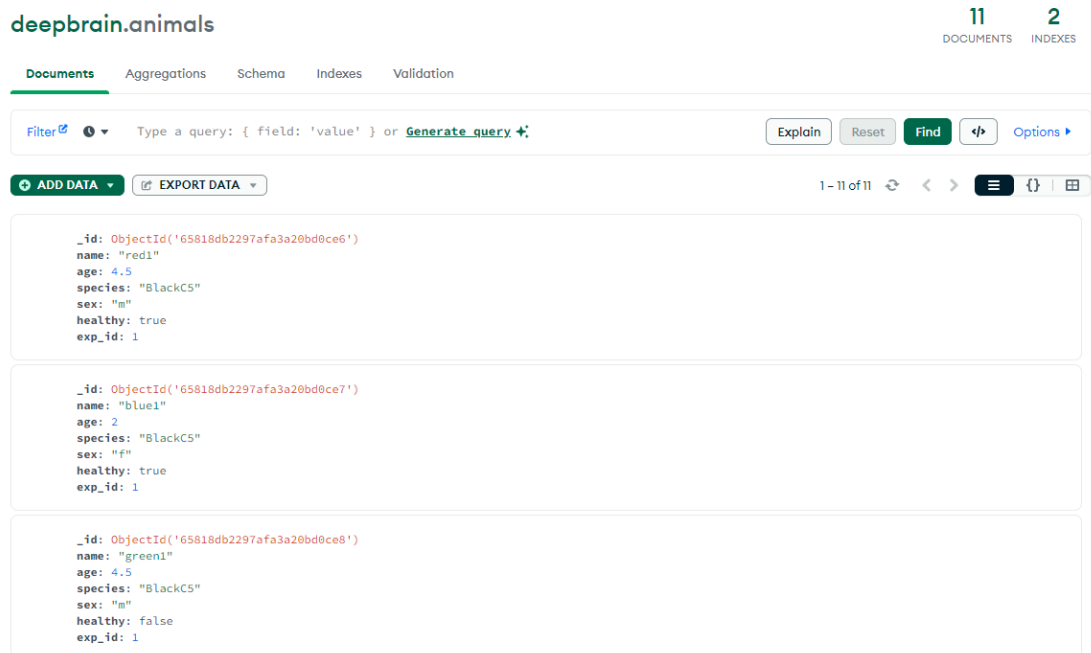
התיקייה השנייה היא תיקיית ה-`models`.



התיקייה עוזרת לקשר בין בסיס הנתונים (דאטא בייס) לבין האתר, כאשר הקובץ הראשון מכיל מבנה של השדות לחיה מסוימת והקובץ השני לפרדיגמה מסוימת. (כתובים בשפה `JavaScript`)

ניתן לפתוח את הדאטא בייס בכל שלב ולראות את הנתונים-

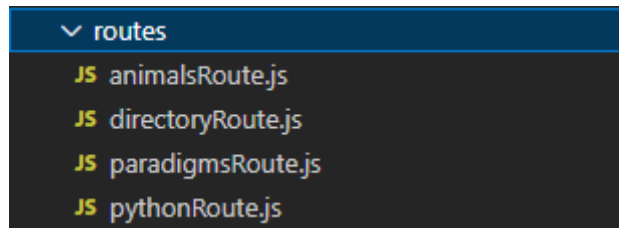
לדוגמא כאן מופיעות החיות שנמצאות בבסיס הנתונים (הזמני, אתם תזינו לפי הצרכים שלכם)-



The screenshot shows the 'deepbrain.animals' web application. At the top, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. The 'Documents' tab is active. Below the tabs, there is a search bar with a 'Filter' button and a 'Generate query' button. To the right of the search bar, there are buttons for 'Explain', 'Reset', 'Find', and 'Options'. Below the search bar, there are buttons for 'ADD DATA' and 'EXPORT DATA'. The main content area displays a list of animal records, each with a unique ID, name, age, species, sex, healthy status, and experiment ID.

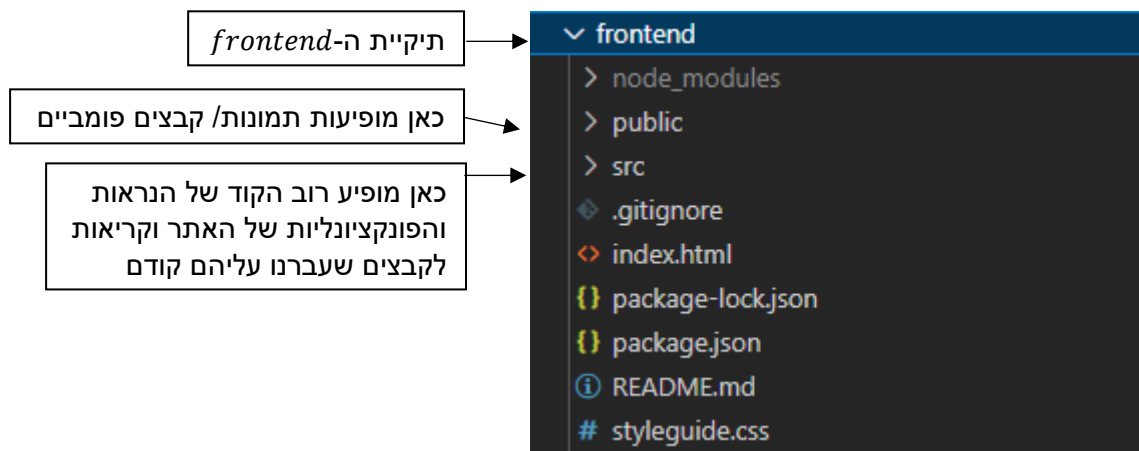
_id	name	age	species	sex	healthy	exp_id
ObjectId('65818db2297afa3a20bd0ce6')	red1	4.5	BlackC5	m	true	1
ObjectId('65818db2297afa3a20bd0ce7')	blue1	2	BlackC5	f	true	1
ObjectId('65818db2297afa3a20bd0ce8')	green1	4.5	BlackC5	m	false	1

התיקייה השלישית היא תיקיית ה-*routes*.

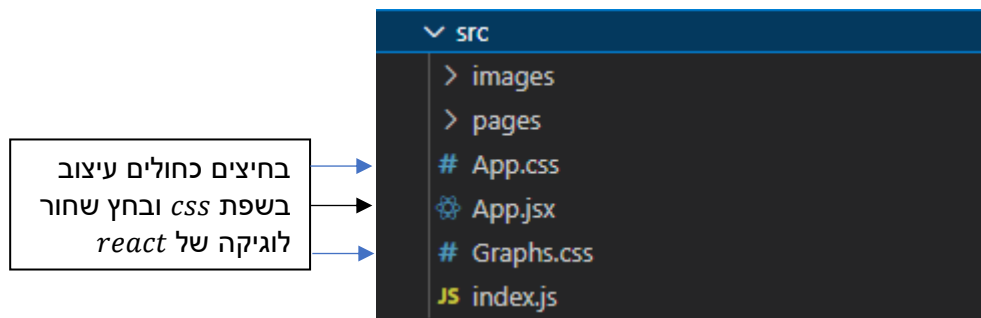


כל אחד מהקבצים בתיקייה הזאת מקבל קריאות מסוימות לשרת ומטפל בהן. לדוגמא בלחיצת כפתור נרצה ליצור גרף או ליצור סרטון עם תוויות אז נקרא לשרת עם בקשה והוא יקרא לתוכנית הפיתוח המתאימה.

-*frontend*



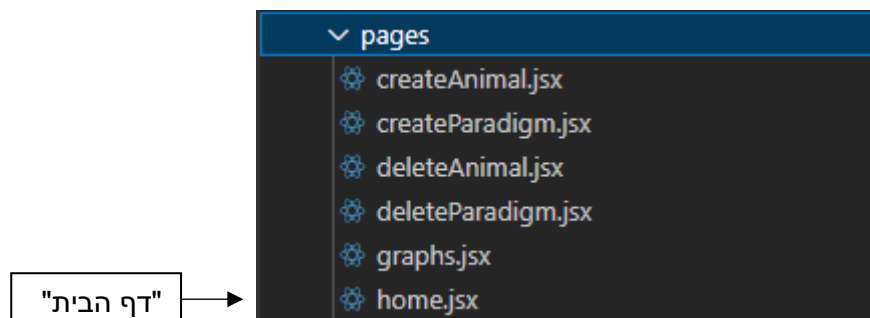
נסתכל בתיקיית ה-*src*.



התיקייה מכילה שתי תתי תיקיות וארבעה קבצים. נעבור עליהם.

*images* - מכילה את הלוגו של האתר ואת הגרפים על מנת להוריד אותם למחשב.

*pages* - מכילה את כל הדפים של האתר.





## מיקומים של פרטים חשובים

- קביעת רשת הניירונים בה משתמשים-

על מנת לקבוע מהי הרשת בה תרצו להשתמש לחצו על התיקיות בסדר הבא:  
*frontend* → *src* → *pages* → *home.jsx*

```
const [list, setList] = useState([]);
const [listBodyParts, setListBodyParts] = useState([]);
const [showModal, setShowModal] = useState(false);
const [selectedOption, setSelectedOption] = useState('');
const [isButtonDisabled, setIsButtonDisabled] = useState(true);
const [showBodyPartsModal, setShowBodyPartsModal] = useState(false);
const [selectedBodyPartOption, setSelectedBodyPartOption] = useState('');

let config_path = 'C:\\Experiment18-Tester18-2024-02-29\\config.yaml'

// Function to toggle modal visibility
const toggleModal = () => {
  setShowModal(!showModal);
};
```

בתוך הקובץ *home.jsx* השורה המודגשת היא השורה שקובעת את רשת הניירונים. השורה מכילה את המיקום של קובץ הקונפיגורציה של הפרויקט.

איך מוצאים את קובץ הקונפיגורציה של רשת ניורונים?

בתוך התיקייה שנוצרת לכם באימון רשת ניורונים חדשה (נפרט לגבי אימון חדש בהמשך) נוצר באופן אוטומטי קובץ קונפיגורציה. לשם המחשה, נשים ❤️ למבנה התיקייה:

Experiment18-Tester18-2024-02-29		Experiment18-Tester18-2024-02-29		Windows-SSD (C:)		מחשב זה	
שם	תאריך שינוי	סוג	גודל	שם	תאריך שינוי	סוג	גודל
dlc-models	29/02/2024 13:23	תיקית קבצים		dlc-models	29/02/2024 13:23	תיקית קבצים	
evaluation-results	03/03/2024 18:37	תיקית קבצים		evaluation-results	03/03/2024 18:37	תיקית קבצים	
labeled-data	29/02/2024 13:23	תיקית קבצים		labeled-data	29/02/2024 13:23	תיקית קבצים	
training-datasets	29/02/2024 13:23	תיקית קבצים		training-datasets	29/02/2024 13:23	תיקית קבצים	
videos	03/03/2024 22:37	תיקית קבצים		videos	03/03/2024 22:37	תיקית קבצים	
config	17/04/2024 17:32	Yaml Source File	2 KB	config	17/04/2024 17:32	Yaml Source File	2 KB

הקובץ מסומן בצהוב. חשוב להוסיף ל-*path* של הקובץ סיומת של *config.yaml* \\ כמו שמופיע בדוגמא ולהפוך *slash* ל-*backslash* כפול (במקום / לכתוב \\)!

- שינוי מספר הנקודות בסרטון עם ה-*labels*-

על מנת לשנות את פרמטר ה-*trailpoints* כלומר את מספר הנקודות בפריים לחצו על התיקיות בסדר הבא:

*backend* → *deeplabcut* → *complete\_network\_code.py*

```
deeplabcut.create_labeled_video(config, [video_path], videotype=VideoType, trailpoints=5, save_frames = True)
```

שנו את *trailpoints* למספר הרצוי. אם אתם רוצים מעקב של נקודה יחידה עבור כל איבר מחקו את הפרמטר מהרשימה ואת הפרמטר שאחריו, בצורה הבאה:

```
deeplabcut.create_labeled_video(config, [video_path], videotype=VideoType)
```

**חשוב-** אם סוג הסרטון שהוקלט הוא לא *mp4* שנו את סוג הסרטון במשתנה *VideoType* בקובץ.

○ הגדרות חשובות ב-*config.yaml*

סף וודאות לשרטוט נקודה (ככל שיותר נמוך יותר טוב)

גודל נקודה, בקפיצות של 4 (כלומר 4, 8, 12, ...)

[מפת צבעים לנקודות](#)

אחוז הדאטא שהולך לאימון (האחוז הנותר הולך ל-*test*)

סוג הרשת שמאמנים

מספר ה-*frames* בכל איטרציה

```
# Plotting configuration
skeleton: []
skeleton_color: black
pcutoff: 0.4
dotsize: 12
alphavalue: 0.7
colormap: rainbow

# Training, Evaluation and Analysis configuration
TrainingFraction:
- 0.95
iteration: 0
default_net_type: resnet_50
default_augmenter: default
snapshotindex: -1
batch_size: 4
```

כלומר אם תרצו לשנות את גודל הנקודות או את הצבעים של הנקודות השינוי מתבצע ישירות מקובץ הקונפיגורציה. שימו ❤️ שאתם שומרים את השינויים ב-*config.yaml*, על ידי *ctrl + s*.

• מבנה הקבצים ומיקום כל קובץ-

כל מי שמשתמש בפרויקט צריך ליצור את התיקיות הבאות בהיררכיה הנוכחית:

info

|

subjects

|

paradigms

|

session

analysis

|

subjects

|

paradigms

|

session

דוגמא למבנה קיים:

info

|

red1

blue2

|

run

test

|

1

2

1

analysis

|

red1

blue2

|

run

test

|

1


2

1

כאשר *info* מכיל את הסרטונים עם ה-*labels* (ו-*h5*) ואילו *analysis* מכיל את קבצי ה-*csv* ואת הגרפים.

## יצירת רשת נוירונים חדשה

מספר נקודות חשובות-

1. יש מספר פרמטרים בקבצים שיש לשנות, בהתאם למיקום במערכת הקבצים במחשב הלוקאלי ולמספר האיטרציות הרצוי, מיד נעבור על כולם.
2. מרגע ההרצה בטרמינל הריצה מתבצעת אוטומטית, שימו  שהמחשב צריך להישאר פתוח על מנת לסיים את הריצה (ללא שינה/ כיבוי).
3. ה-*labelling* לא מתבצע בצורה אוטומטית, יש צורך בהשמה ידנית של התוויות על חלקי הגוף, בהמשך הסבר מפורט על השימוש ב-*napari* וקביעת ה-*labels*.
4. כיוון שיש כאן שימוש ב-*DeepLabCut* כמו שראינו ב"התקנה על מחשב חדש" יש לוודא שהאינטרפטר הנכון נבחר.

מצורפים חמישה קבצים בתוך תיקייה ליצירת רשת נוירונים חדשה (ובדיקה שלה):

*mini\_projects.py* - הקובץ הראשוני ואותו יש להריץ על מנת לאמן רשת (השאר רצים באופן אוטומטי וקוראים אחד לשני. הקובץ עוזר באוטומציה של המערכת, הוא יוצר המון פרויקטים קטנים ומאחד אותם תחת פרויקט אחד. בצורה הזאת אין צורך בלכתוב תגובות ישירות של *yes* לטרמינל. על מנת להפעיל את התוכנית פתחו טרמינל חדש כמו ב"הרצה של פרויקט" רק עם פקודת `python -i mini_projects.py` שתריץ את התוכנית.

הפרמטרים שצריך לשנות בקובץ:

התיקייה שבה נמצאים הסרטונים לאימון	<pre>DIR = "D:\\testing_videos_new" # for creating a new project for all files TEMP_DIR = 'C:\\neural_network1' # temp directory for all mini projects ZERO = 0 ONE = 1 MINUS_ONE = -1 PARADIGM = "Experiment18" # name of paradigm TESTER = "Tester18" # name of the experimenter FILES = os.listdir(DIR) count = ONE</pre>
מיקום תיקייה זמנית (לא קריטי לשנות)	
שם רשת הנוירונים- שם הניסוי ושם הנסיין	

*say\_yes.bat* - תוכנית של *script*, ממש כותב שורות קוד של המילה "yes" לתוך הטרמינל כשמתבקש. קורא עבור כל מיני פרויקט ל-*frames\_from\_mini\_projects.py* שיוציא עשרים *frames* מסרטון.

*main\_project.py* - הקובץ שנקרא לאחר מכן מקבל את הקובץ הסופי, את הפרויקט עצמו. הוא עושה מודיפיקציה לקובץ הקונפיגורציה (*config.yaml*) ומגדיר בו מספר פרמטרים:

הגדרת הנקודות, ה- <i>labels</i> שנרצה לסמן	<pre># Define hierarchical structure for body parts cfg["bodyparts"] = ["right_front",     "right_front",     "right_back",     "right_back",     "left_front",     "left_front",     "left_back",     "left_back",     "tail_base",     "tail_upper",     "tongue"]  cfg["skeleton"] = []  cfg['pcutoff']=0.4  cfg['batch_size']=4</pre>
הגדרת סף וודאות לשרטוט נקודה (ככל שיותר נמוך יותר טוב)	
הגדרת מספר ה- <i>frames</i> בכל איטרציה באימון	

כמובן שניתן לגשת לערכים נוספים ולשנות אותם כבר בשלב האימון אך הפרמטרים שלא נמצאים כאן יכולים להשתנות גם במערכת מאומנת (ראו הגדרות חשובות ב-*config.yaml*).

עבור הפונקציה הזו-

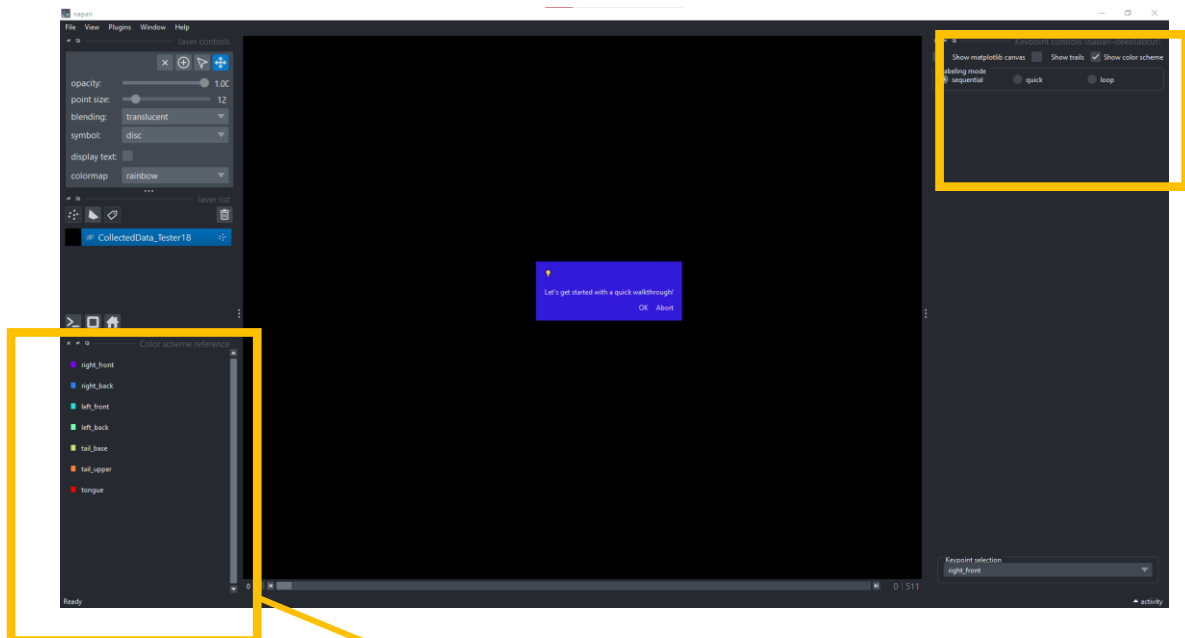
```
deeplabcut.train_network(path_config_file, shuffle=shuffle, displayiters=DIS_ITERS, saveiters=SAVE_ITERS, maxiters=MAX_ITERS)
```

עריכת הפרמטרים קריטית לתוצאת האימון של רשת הניורונים. כרגע הרשת שומרת מידע כל 100 איטרציות ורצה במשך 75000 איטרציות אך הנתונים הללו ניתנים לשינוי.

*complete\_network.py* - כמו הקובץ שמופיע בפרויקט (ראו "תוכן הקבצים"), למקרה שתצטוו ליצור סרטונים עם *labels* מחוץ לפרויקט. שימו ❤️ שצריך לשנות כאן את הפרמטרים של קובץ הקונפיגורציה ושל מיקום הסרטונים שתצטוו לנתח.

## עבודה עם *napari*

החלון ייפתח לכם מעצמו לאחר מספר דקות מתחילת ההרצה.



ניתן ללחוץ במלבן הסגול על *Abort*.

שימו ❤️ שמצד שמאל יופיעו האיברים שיש לסמן.

על מנת להתחיל אתם נדרשים לבחור תיקייה מהפרויקט. כל תיקייה שתבחרו מכילה 20 *frames* נבחרים מסרטון מסוים (מהסרטונים שאיתם אתם מאמנים את הרשת).

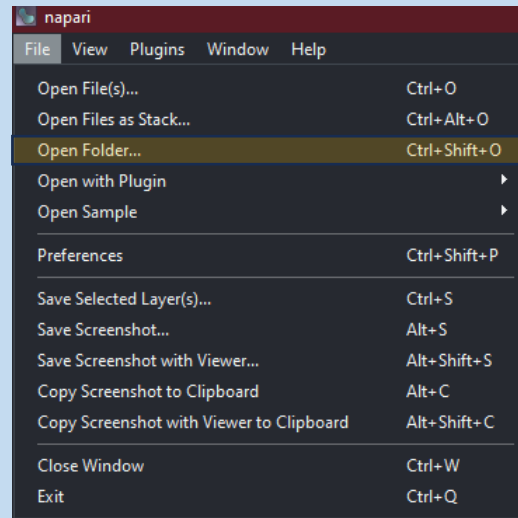
תצטרכו לחזור על השלבים הבאים עבור כל תיקייה (חוץ מפתחת התוכנית, בחירת התיקייה הראשונה, בה יש החרגה כיוון שהתוכנה נפתחת אוטומטית עם קובץ הקונפיגורציה ולא תצטרכו את שלב 2).

בנוסף וודאו שאתם על מצב *loop* להשמה קלה יותר של הנקודות.

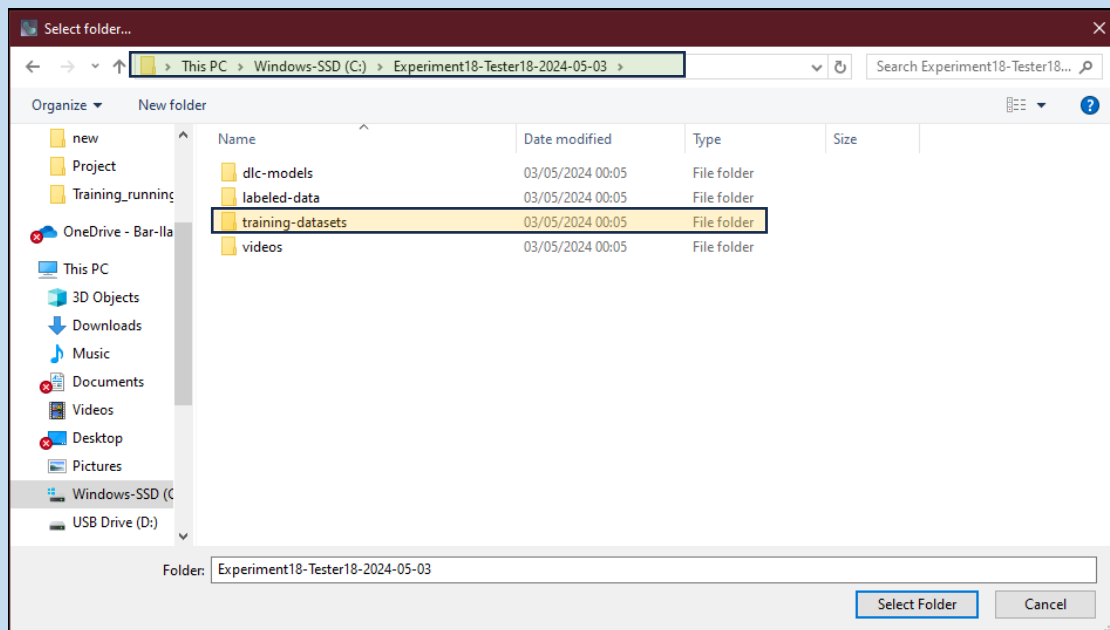
משמאלכם מעל רשימת האיברים ניתן לשנות את הפרמטרים, הם מוגדרים בהתחלה אוטומטית לפי קובץ הקונפיגורציה (*config.yaml*).

# 1. כדי לבחור תיקייה לחצו על הסימון הצהוב:

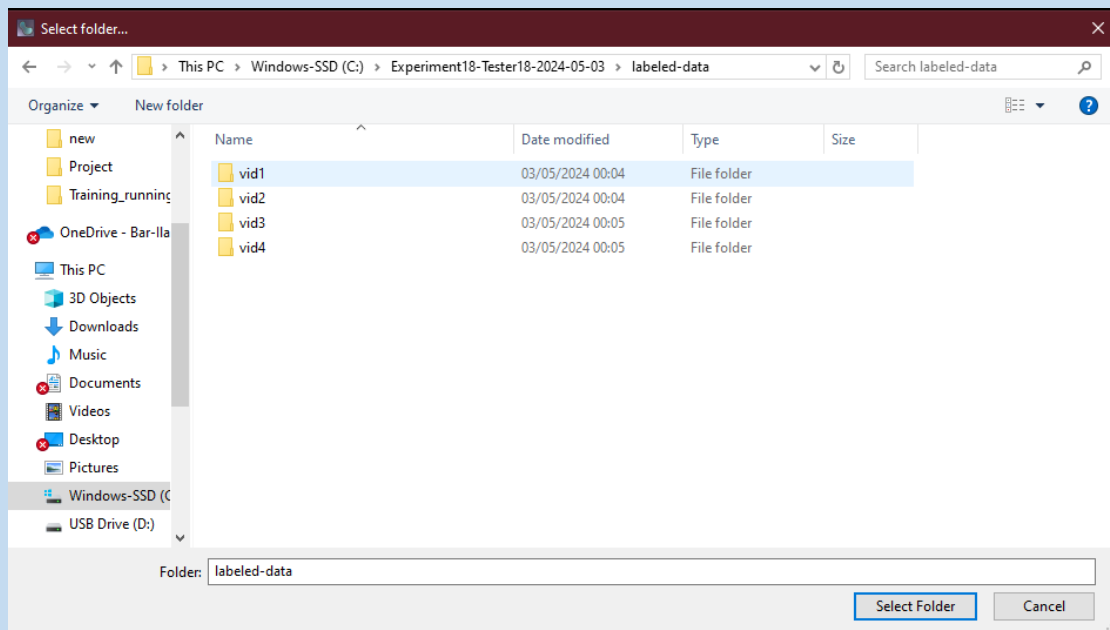
יפתח לכם חלון בחירה. כנסו לתיקיית הפרויקט לפי הפרמטרים מקובץ `mini_projects.py` והתאריך של יצירת הפרויקט (מסומן בירוק בתמונה א') בתמונה א' ניתן לראות ארבע תיקיית שהפרויקט יוצר באופן אוטומטי. כנסו לתיקיית `data – labeled`.  
בתיקייה זו כרגע המידע עוד לא `labeled` אבל היא מכילה את התמונות שיש להשים עליהן `labels`.  
כעת יוצגו לכם תיקיות של סרטונים כמו בתמונה ב', כנסו לתיקייה של אחד הסרטונים (בדוגמא הזו יש ארבעה) ולחצו `select folder`.



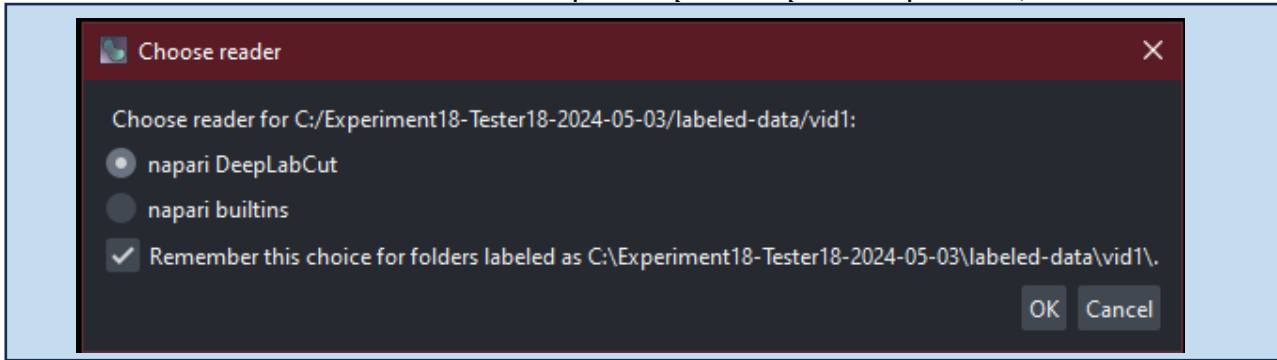
א.



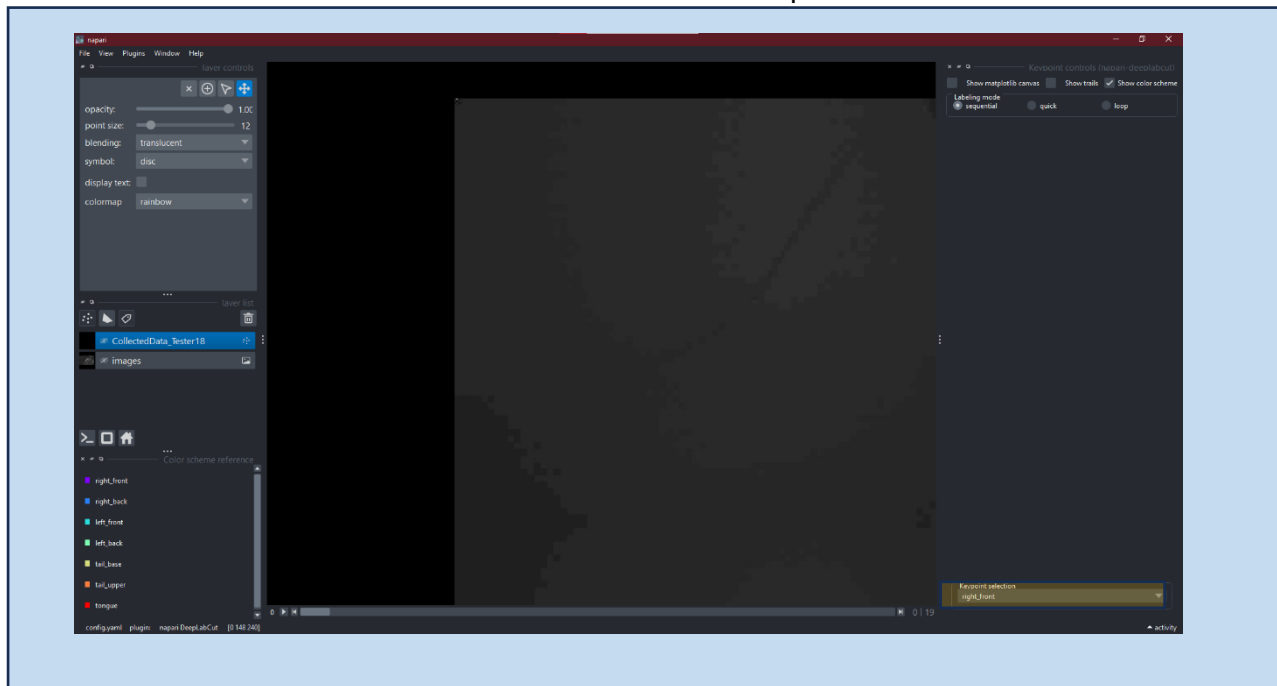
ב.



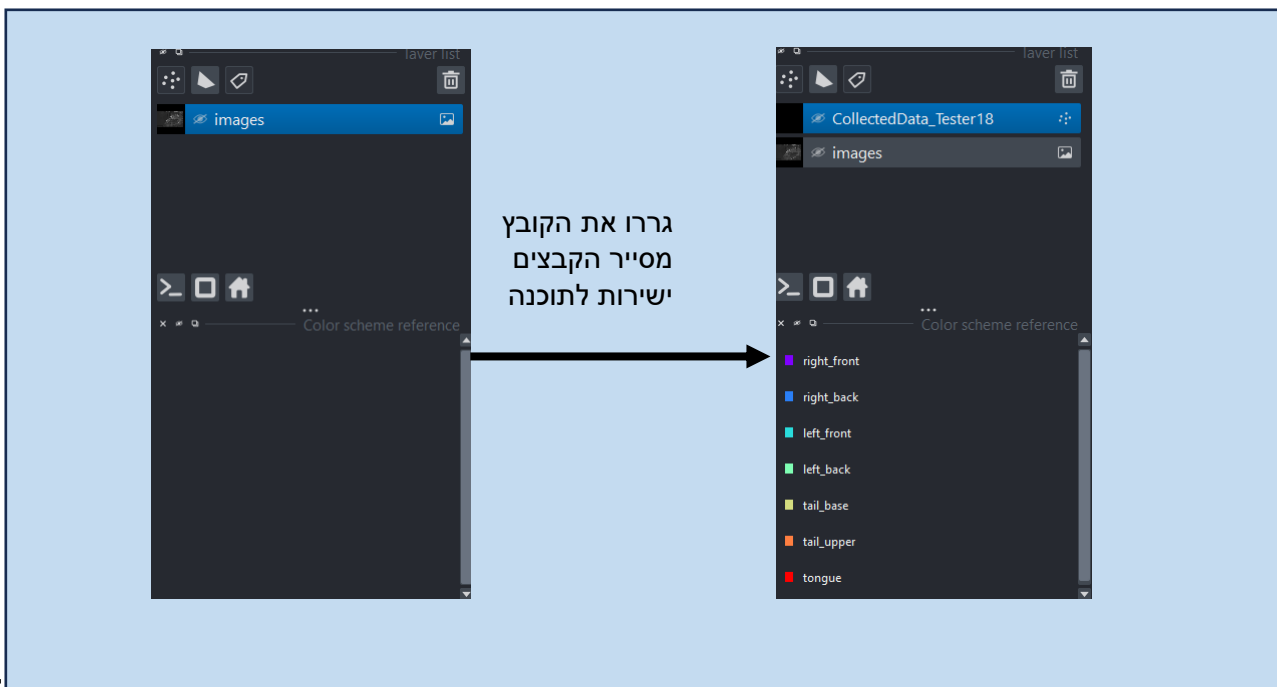
2. שימו ❤️ שמסומן *napari DeepLabCut* בחלון ולחצו על OK.



3. כעת יופיעו התמונות שממוספרות מתמונה 0 עד תמונה 19 (20 סך הכל). על מנת להקטין את התמונות לחצו על החלק האמצעי בעכבר וגללו אותו לשינוי הזום.



4. בפעם השנייה ואילך בכל פעם לאחר שתבחרו תיקייה תצטרכו להוסיף את *config.yaml*



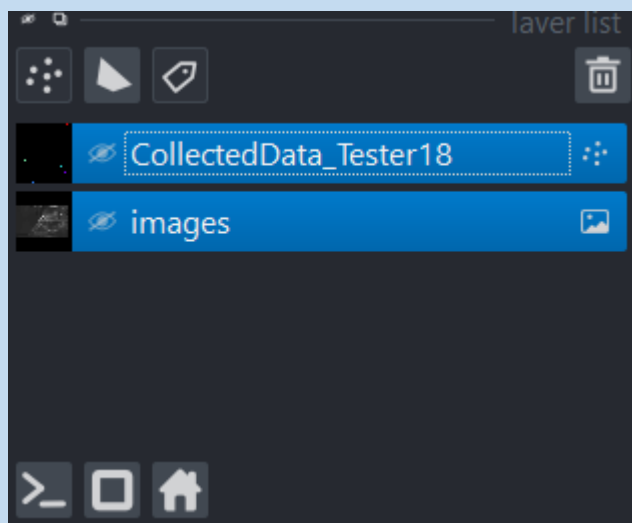


5. שימו לב לכלים שבהם תשתמשו ל-*labels*.

כמה נקודות פחות טריוויאליות:

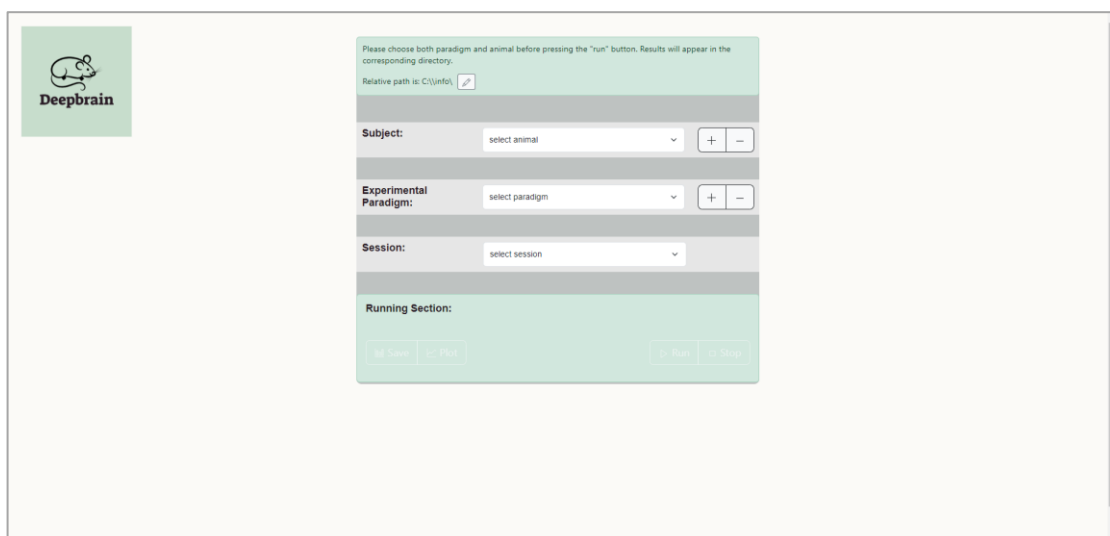
- למחיקת נקודה סמנו אותה עם הכפתור השני מימין ורק לאחר מכן יתאפשר לכם ללחוץ ולמחוק.
- ניתן לדלג על סימון *label*, אם האיבר לא מופיע בתמונה.
- בחירת ה-*label* הנכון מתבצעת מימין (מודגש בצהוב בסעיף 3). אם דילגתם על סימון *label* מסוים התוכנה תחזור אליו באופן אוטומטי ולכן שימו ❤️ איזה *label* כתוב לפני שאתם מסמנים אותו על התמונה.

6. אם סיימתם לסמן את כל ה-*labels* בכל אחד מה-*frames* של הסרטון סמנו עם העכבר את שתי השכבות ולחצו על סימון הפח (מימין).



7. בצעו את כל השלבים שוב ושוב עד למעבר על כל תיקיות הסרטונים ולאחר מכן סגרו את התוכנה. הקוד ימשיך באופן אוטומטי עד לסיום.

### עבודה עם ה-*User Interface*




ברוכים הבאים ל-*Deepbrain*! דרך האתר תעשו את ניתוח הנתונים העיקרי. נעבור על כל אחד מהחלקים בו.

### קביעת ה-*relative path*:


ייתכן ולאחר יהיו מספר משתמשים עם נתונים שונים. עבור כל משתמש תהיה היררכיית התיקיות כפי שצוין בחלק "מיקומים של פרטים חשובים". *info* יכיל את הסרטונים עם ה-*labels* (ו-*h5*) ואילו *analysis* יכיל את קבצי ה-*csv* ואת הגרפים.

לכן עבור כל *user* חשוב שישנה את ה-*path* כך שיתאים למיקום הקבצים שלו.

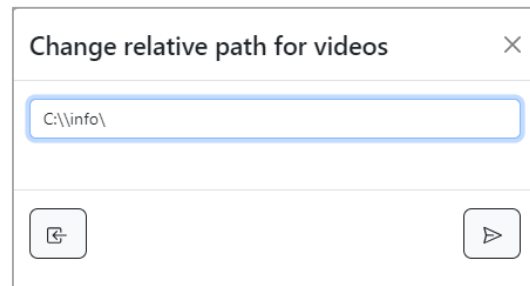
Relative path is: C:\\info\\ 

ה-*path* הנבחר יכלול בתוכו את המילה "info".

איך עורכים את ה-*path*?

בלחיצה על כפתור העיפרון ייפתח לכם חלון לעריכה. שימו  שעד שלא לחצתם על לחצן השליחה ה-*path* לא ישתנה.

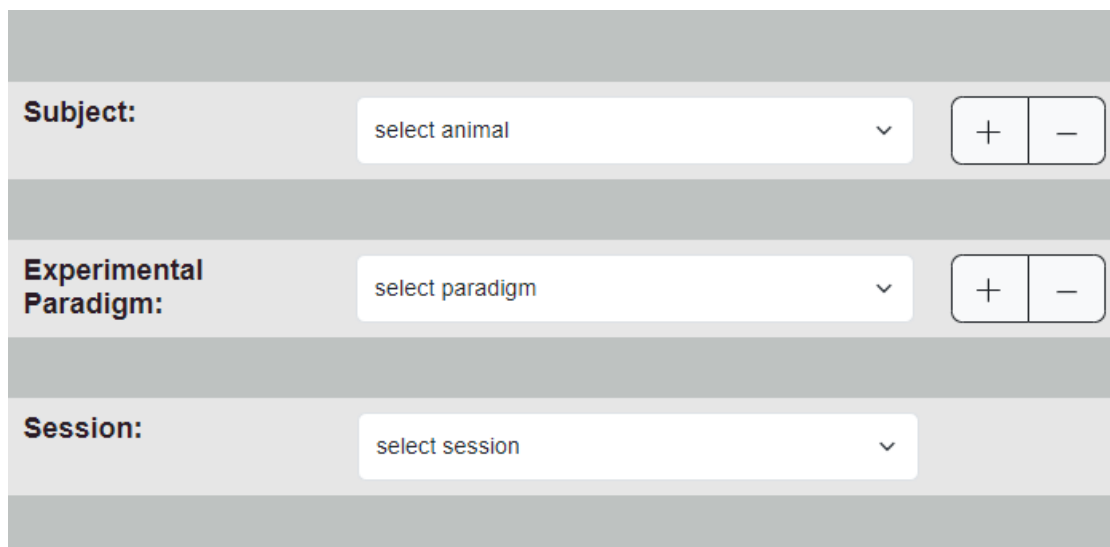
כפי שנכתב ב"מיקומים של פרטים חשובים" בתוך תתי התיקיות של התיקייה *info* יהיו הסרטונים המוקלטים אותם תרצו לנתח.



סגרתי ופתחתי את התוכנית. האם אני צריך לשנות את ה-*path* שוב?


לא. התוכנית תזכור את המיקום האחרון בו השתמשתם. אבל חשוב להשים לב שמדובר במיקום הנכון אם מספר סטודנטים עובדים על התוכנה.

בחירת סרטון לניתוח:



הבחירה עצמה מורכבת משלושה חלקים, לפי היררכיית התיקיות.

עבור כל חיה/ פרדיגמה חדשים יש להוסיף את השמות כפי שמופיעים בשם התיקיות (ע"י לחצן ה-+ כפי שיתואר בהמשך). אם חיה/ פרדיגמה אינם רלוונטיים יותר לניתוח יש להסיר את השמות (ע"י לחצן ה- - כפי שיתואר בהמשך).



שימו  שכיוון שהמערכת עובדת חלקית בלבד מול מערכת ניהול הקבצים שבמחשב, כאשר חיה או פרדיגמה נמחקים הנתונים שלהם לא נמחקים. את המחיקה הזו יש לבצע ידנית במערכת הקבצים.



## הוספה/ מחיקה של חיה ופרדיגמה (הדגמה עם חיה)

Please fill the fields below:

Name of animal:



להוספת חיה (לחיצה על מקש ה- + ליד שדה החיה) יש להקליד את שם החיה. כיוון שהחיות unique בשם ברגע שתנסו ליצור חיה עם שם קיים תקבלו הודעת שגיאה.

Please fill the fields below:

Name of animal:



Can't use the same name twice

כך ימנעו מכם כפילויות וחילופי מידע לא נכונים.

Please fill the fields below:



select animal

למחיקת חיה יש לבחור את החיה שרוצים למחוק.

Confirm Delete

Are you sure you want to delete white1?

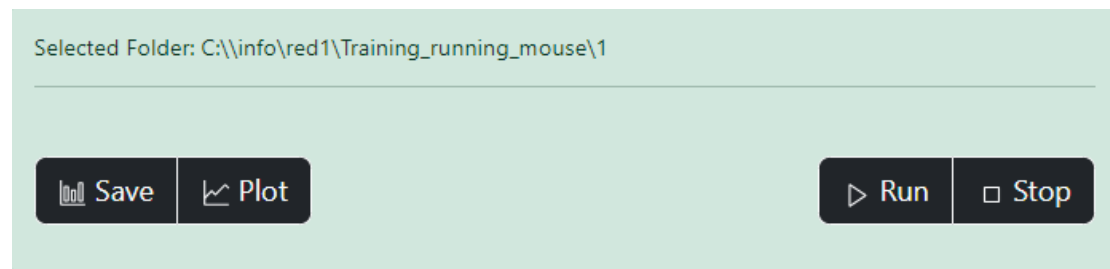
 

ברגע שתשלחו את הבחירה שלכם תופיע חלונית שתשאל האם אתם בטוחים. לחיצה כאן היא סופית ותביא למחיקה של החיה מבסיס הנתונים.

למה ל-*session* אין הוספה/ מחיקה?

השדה הזה נקבע בהתאם למערכת הקבצים. התוכנה מחפשת את ה-*path* (האם קיים) ואם כן מחזירה את רשימת תתי התיקיות של ה-*sessions*.

יש לבחור את שלושת השדות. על מנת לנתח את הקבצים המסלול (*path*) צריך להיות קיים ושיהיו בו סרטונים לניתוח. רק אז ייפתח סרגל ה-*Running Section*.



כפי שניתן לראות מסלול התיקיה הנבחרת מופיע אוטומטית. נעבור על הכפתורים בסרגל.

#### יצירת סרטונים עם labels:

ברגע שתלחצו על כפתור ה-Run התוכנה מנתחת את הסרטון המקורי בעזרת DeepLabCut ויוצרת סרטון עם labels וקובץ h5 לניתוח הנתונים של תנועת העכבר.

בכל תיקייה של trial יכולים להיות מספר סרטונים. עבור כל סרטון בתיקיה שכבר נותח התוכנית מזהה שנותח וממשיכה.

שימו - ❤️ התוצאות של הסרטונים נשמרות באותה תיקייה ממנה הסרטונים נקראו.

אם תרצו לעצור את ההרצה ויצירה של הסרטון המנותח, תוכלו בכל שלב (לפני סיום הריצה) ללחוץ על כפתור ה-Stop.

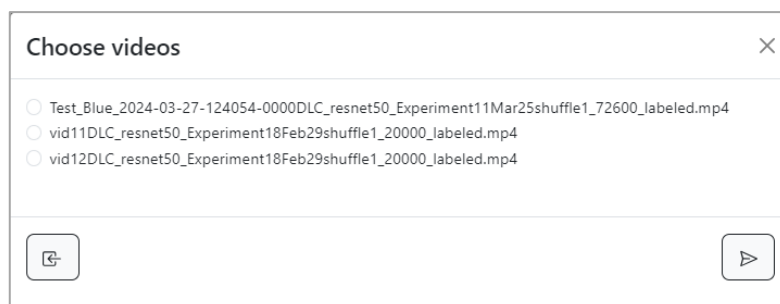
#### שמירת המידע בקבצי csv:

ברגע שתלחצו על כפתור ה-Save המידע נשמר בשלושה קבצי אקסל שונים- קובץ בו יש מידע עבור המיקום, קובץ עם מידע על מהירות וקובץ למידע על תאוצה. תוכלו להשתמש בנתונים הללו ליצירת גרפים נוספים.

שימו - ❤️ המידע על התנועה נשמר באותו מיקום אבל תחת תיקיית analysis. למידע נוסף ראו "מיקומים של פרטים חשובים".

#### יצירת גרפים:

ברגע שתלחצו על כפתור ה-Plot ייפתח לכם חלון נוסף-



למקרה שקיים יותר מסרטון אחד לניתוח בתוך הסשן. סמנו את הסרטון אותו תרצו לנתח. לאחר מכן ייפתח לכם עוד חלון-

Choose bodypart

×

☐ right\_front
☐ right\_front\_Z
☐ right\_back
☐ right\_back\_Z
☐ left\_front
☐ left\_front\_Z
☐ left\_back
☐ left\_back\_Z
☐ tail\_base
☐ tail\_upper
☐ tail\_lower
☐ tail\_tip
☐ tongue

↩

➤

כעת בחרו את חלק הגוף אותו תרצו לנתח לפי *config.yaml* הנתון.  
 המערכת יוצרת שלושה גרפים- גרף מהירות, גרף מיקום וגרף דפוס הליכה של העכבר (דפוס תזוזה אם מדובר באיבר שאינו כף רגל).

VELOCITY

POSITION

WALKING PATTERN

Walking pattern of the mouse right\_back\_Z

↩

Download

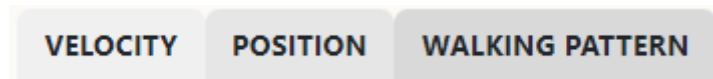
Title:

Fill in your new title

➤

19

נעבור על החלקים השונים במערכת-



ראשית, לניווט בין הגרפים לחצו על שמות הגרפים ב-tabs. הלחיצה תציג לכם את הגרף הרלוונטי.

Title:

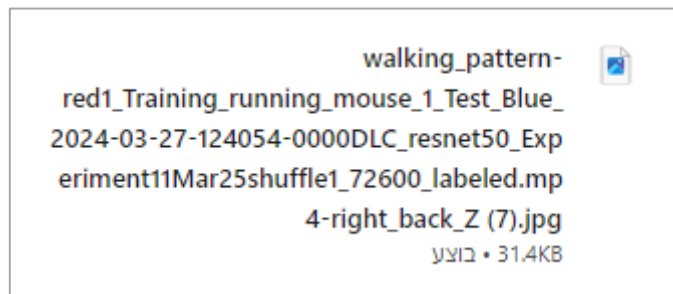


שנית, לשינוי כותרת הגרף אין צורך בייצור מחדש של כל הגרפים. מלאו את הכותרת המתאימה ולחצו על לחצן השליחה. אם התוכן יהיה תו רוח (רווח אחד בלבד!! שימו ❤️ שעבור יותר מרווח בודד המערכת תוציא שגיאה ולכן לא תעדכן את הגרף) המערכת תייצר לכם גרף ללא כותרת.



Download

להורדת הגרף הספציפי לחצו על כפתור ההורדה והגרף יישמר לכם במיקום הרלוונטי המערכת הקבצים.



שם הגרף מורכב מסוג הגרף, הנבדק, הפרדיגמה, מספר הסשן, שם הסרטון והאיבר ששורטט.

אם תרצו לשנות את שלושת הגרפים שמוצגים באופן יותר קיצוני מאשר הכותרת בלבד, מוזמנים לשנות את קובץ `analyze.py` כפי שהוסבר עד כה. קחו בחשבון שאם תשנו את הלוגיקה עצמה ולא את הגרפים ייתכן ותקבלו שגיאה. (לדוגמא אם תוסיפו/ תחסירו גרף).

מצורפים גם הקבצים של ניתוח המידע בצורה נפרדת מהקוד של הפרויקט (כולל `analyze.py`) ותוכלו לבצע שינויים בהם (כולל הוספת/ הורדת גרף). גם עבורם קיים גיבוי!

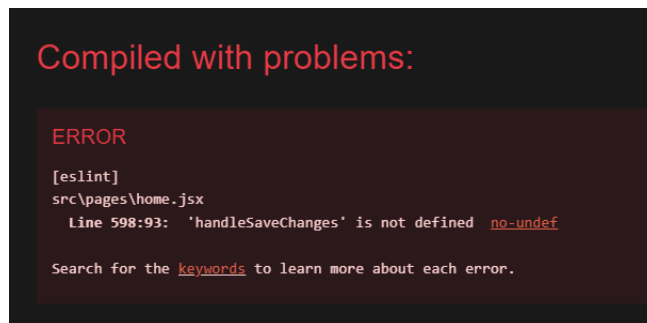
אם תרצו להריץ אותם בצורה נפרדת חזרו על "הרצה של הפרויקט" אך במקום פתיחת שני טרמינלים פתחו אחד, ובמקום `cd backend/` כתבו `python -i analyze.py` לדוגמא (כלומר `python -i` ולאחר מכן את השם המלא של הקובץ).

סיימנו לעבור על כל רכיבי המערכת 😊

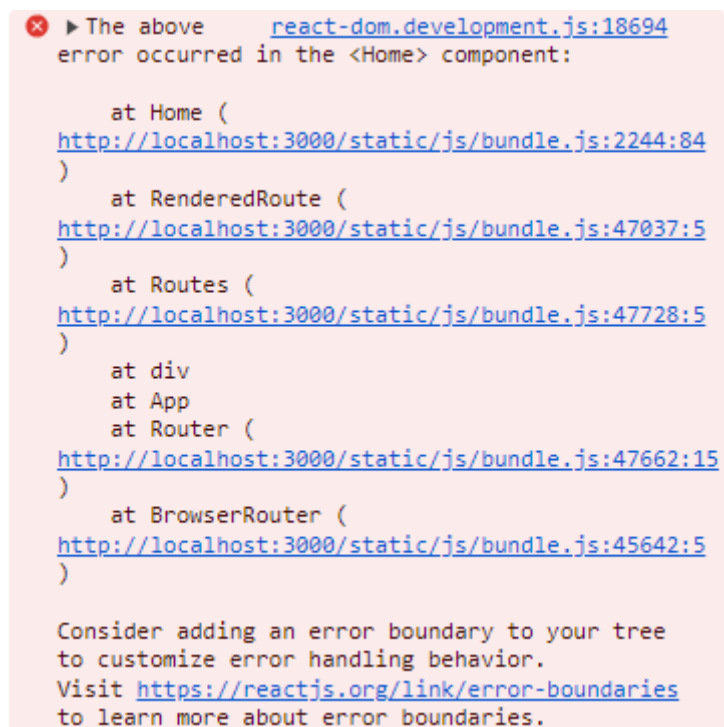
## שגיאה במערכת

מה נעשה כאשר בכל זאת יש שגיאה במערכת (למרות שהמערכת אמורה לטפל בכל השגיאות)?

- בדקו האם השגיאה נובעת משינוי שעשיתם בקוד. אם כן, תוכלו לשחזר את הקוד המקורי שיישמר אצלכם במיקום מיועד במחשב.
- בדקו האם שתי המערכות שהפעלתם בטרמינל פועלות, אם לא, חזרו על "הרצה של הפרויקט" באופן מלא או חלקי.
- במקרה ושני הדברים לא עובדים ייתכן שהשגיאה תופיע על המסך באחת משתי דרכים- ישירות על המסך- (במקרה הזה באופן יזום חלק מהקוד בהערה והוא חסר)



או על ידי לחיצה על לחצן ימני עכבר ועל כפתור בדיקה או *inspect* ולאחר מכן *console*:



כפי שניתן לראות גם כאן כתוב בדיוק איפה ביקוד התרחשה השגיאה.

אם השגיאה היא בחוסר הצגה של הגרפים, כנסו ל-*public* → *frontend* ומחקו את שלושת התמונות שמופיעות (*output1.jpg*, *output2.jpg*, *output3.jpg*). שימו ❤️ שיש למחוק את התמונות בתיקייה זו בלבד ורק את התמונות בתיקייה!

## דוקומנטציה (documentation)

### מספר אופטימלי של לקיחת frames מסרטון:

נשמע שמספר ה-frames תלוי ברעש שיש ב-*data*. עבור *data* רועש יותר מספר ה-frames גדל. מעבדות רבות השתמשו בכ-25 frames (במאמרים נראה שבין 20 ל-30, חלק גדול השתמשו ב-25) לסרטון לא רועש (תנאי מעבדה, כפי שמתקיים במקרה שלנו).

כתלות באורך הסרטון- לא נמצאה התייחסות לאורך הסרטון בהתאם למספר ה-frames שנקחו. חלק מהמאמרים השתמשו בסרטונים של חצי שעה, חלק בסרטונים של מספר דקות וחלק ב-45 שניות בממוצע וכולם השתמשו בכ-25 frames לסרטון בלי קשר לאורך הסרטון.

*Frames vs. videos* - בנוסף, באתר של *DeepLabCut* הייתה התייחסות לעובדה שהעדיפות היא למספר קטן של תמונות מכל סרטון מאשר מספר גדול של תמונות ממעט סרטונים.

### אילו frames עדיפים?

לפי מאמר שמתייחס גם לסוגיה הזאת, האפשרויות הן להביא frames שנבדלים אחד מהשני ומייצגים זוויות שונות, או לנתח אותם ולחשב את ה-MPE או לקחת רנדומית. החישובים שווים בערך לדגימה יוניפורמית כלומר העובדה ש-*DeepLabCut* דוגם רנדומית לא משפיעה על איכות התוצאות לרעה.

### כלי שיחליף את napari בצורה אוטומטית:

נכון לרגע זה כל המאמרים שנסקרו בחרו את ה-frames ידנית ולא השתמשו בכלי אוטומטי, כלומר אם קיימים כלים מתאימים בשוק הם לא מתאימים או לא נבחנו ונבדקו במספר רב של ניסויים. לכן הפרויקט ישתמש בהגדרה ידנית של *napari*.

### Track method:

Ellipse Tracking: מומלץ ברוב המקרים. מתאים למעקב אחר פיצ'רים עם צורות או כיוונים מורכבים, כגון חלקי גוף או אובייקטים שיכולים להיות מיוצגים היטב על ידי אליפסות. מעקב אליפסה מתאים אליפסות לתכונות שזוהו בכל פריים, ומספק גמישות רבה יותר בלכידת הצורה והשינויים בכיוון של התכונות לאורך זמן.

מעקב גמיש אחר דברים מוגדרים.

Box Tracking: מומלץ לפרויקטים ללא סמן חד-נקודתי (ma). מתאים למעקב אחר תכונות של נקודה אחת שאין להן צורה או כיוון מוגדרים, כגון מיקום סמן בודד או נקודת עניין בסרטון. מעקב אחר תיבות מתאים לתיבות תוחמות סביב התכונות שזוהו בכל פריים, ומספק דרך פשוטה וישירה יותר לעקוב אחר המיקום של מאפיינים חד-נקודתיים.

מעקב ספציפי אחר דברים לא מוגדרים (נקודה אחת לא מוגדרת למשל).

### מסקנות מההרצות:

מינימום איטרציות בשביל להשיג תוצאה סבירה - 5000-10000. בשביל תוצאה טובה - 15000-30000 לפחות.

*pcutoff* - ערכים גבוהים עדיפים, ערך שנבדק ונראה שטוב 0.4.