Installation and Opening

If you need to install DeepLabCut on a new desktop/windows computer, please refer to these steps. Installing is probably the hardest part about using DeepLabCut, so don't despair if you're struggling with it :)

For installing on windows:

1. You will need to have Anaconda installed. You can install it here. Make sure to install the correct Window version (additional installers are all located at the bottom of the page under "Anaconda Installers").

2. Download this file

3. In Anaconda, you should see something called cmd.exe. Open it.

4. A terminal should pop up!

5. In the terminal, cd into the correct directory where the file you downloaded was.
    a. Cd-ing is when you tell the computer to go into a specific folder and look at the files there. For example, if I were to type "cd Documents" into the terminal (with no quotation marks) the computer would then have access and be able to 'see' anything you have in the Documents folder.
    b. If you don't know where the file ended up, you can look for it in the file explorer. It most likely ended up in downloads, so you should be able to type cd Downloads (make sure to verify this).

6. Once there, enter (still in the terminal): conda env create -f DEEPLABCUT.yaml

7. Great, now you just need to activate and open it! Follow the steps for Windows below.

If you need to install it on mac, follow these instructions:

1. You will need to have Anaconda installed. You can install it here (Install the Graphical Installer version). Follow the steps it provides to install it.

2. Download this file

3. On the sidebar of Anaconda, you should see a tab that says environments. Click on it, and then press the play button of the highlighted environment (ex: if Base was highlighted, you would click its play button).

4. Press the text that says Open Terminal.

5. In the terminal, cd into the correct directory where the file you downloaded was.
    a. Cd-ing is when you tell the computer to go into a specific folder and look at the files there. For example, if I were to type “cd Documents” into the terminal (with no quotation marks) the computer would then have access and be able to ‘see’ anything you have in the Documents folder.
    b. If you don’t know where the file ended up, you can look for it in finder. It most likely ended up in downloads, so you should be able to type cd Downloads (make sure to verify this).

6. Once there, enter (still in the terminal): conda env create -f DEEPLABCUT.yaml

7. Great, now you just need to activate and open it! Follow the steps for mac below.

If the above steps don’t work, try creating the environment in Spyder!

Every single time you want to use DeepLabCut you will need to open it.

To open DeepLabCut on Windows, simply go to the cmd.exe terminal in anaconda and type:

conda activate DEEPLABCUT
python -m deeplabcut

To open DeepLabCut on mac:

First you need to activate the environment. Type and enter in the terminal:

conda activate DEEPLABCUT

If you’re getting that conda is not a valid command, go to Spyder again. This time, type and enter in the command window “conda init zsh”. Then restart the kernel, and switch back to the terminal. Conda commands should now work, so re-enter **conda activate DEEPLABCUT**!

You should see on the left side that your environment has changed to DEEPLABCUT. Then type:

pythonw -m deeplabcut

If the last command is not working (and saying some sort of variation of module not found or no python), try typing this into the terminal:

conda install python.app

And then type **y** when asked if y/n

Then, re-enter: **pythonw -m deeplabcut**

It might take about a minute, but eventually the GUI will open!

Using DeepLabCut

We will be using DeepLabCut for the first half of the project, and then will be switching to Google Colab due to GPU constraints. It will be very hard to run the machine learning on your current computer's CPU, which is why it's essential that we use Google Colab in combination!

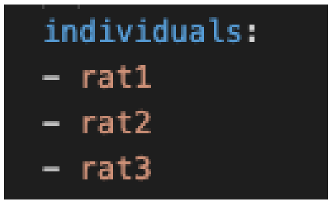
Creating a New Project

The first thing that we will be doing in DeepLab is creating a new project. Go to Manage Project and then "Create a new project" (if you have done this before you can skip these steps and load the config file of the previous project you created).

Fill in all of the blanks with the appropriate text before checking the multi-animal project box (IN THE PROJECT NAME DO NOT USE ANY "/" AS YOU WILL GET AN ERROR). You can also check the directory box if you want to specify where the project will be stored. Make sure to also add all of the video files you want to use by pressing "Load Videos". You can use as many as you want, but around five is a good amount (don't use less than 3 videos). You should also only use videos that have the same amount of rats in them (if one video has 3 rats then all of the videos should only contain 3 rats).

Once you have created a project, you now need to modify the config file. Select “Edit config file” and you should be able to directly edit the configuration file. At the very top you should see something called “Identity”. Make sure that this is set to false if you can’t tell the rats apart (if they are all the same color and have no distinguishing features). Scroll to the section that says “individuals” and “multianimalbodyparts”.

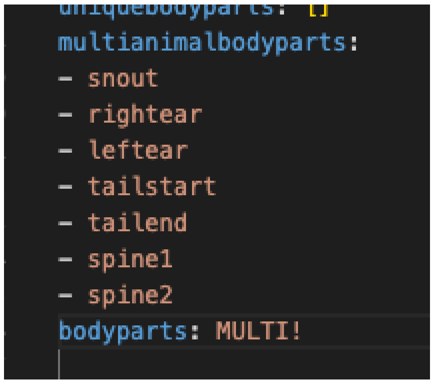
Under “individuals”, list the amount of rats you have. If I’m studying 3 rats in each video, I would type up something like this:



```
individuals:
- rat1
- rat2
- rat3
```

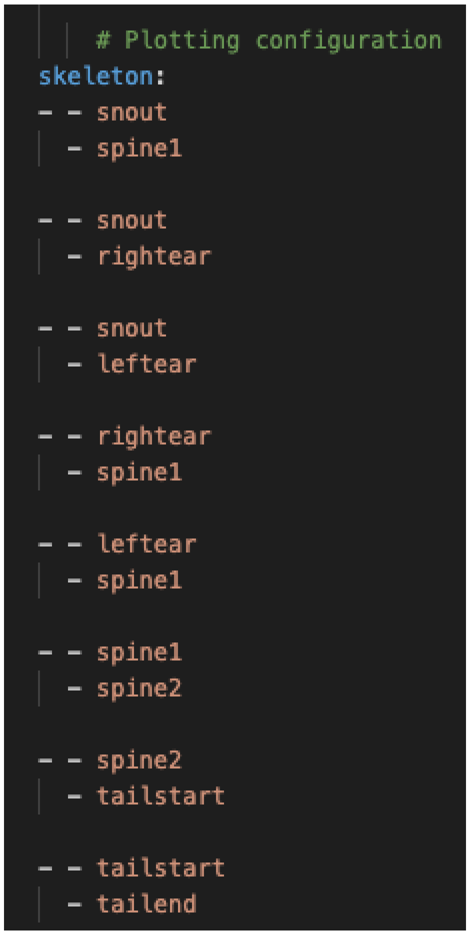
Make sure to keep this exact format with the space between the dash and the name.

Under “multianimalbodyparts”, you want to list every body part that you will be tracking. The parts that I typically did were “snout”, “rightear”, “leftear”, “tailstart”, “tailend”, “spine1” and “spine2”. You will later be marking these body parts on the pictures that we label. If you choose to have different names, there must not be any spaces between words.



```
uniquebodyparts: []
multianimalbodyparts:
- snout
- rightear
- leftear
- tailstart
- tailend
- spine1
- spine2
bodyparts: MULTI!
```

The last thing that we currently want to do here is update the skeleton. This is just so that the program knows what body parts of the rat are attached together! Here is the format I used:



```
    # Plotting configuration
skeleton:
- - snout
  - spine1

- - snout
  - rightear

- - snout
  - leftear

- - rightear
  - spine1

- - leftear
  - spine1

- - spine1
  - spine2

- - spine2
  - tailstart

- - tailstart
  - tailend
```

We will edit this file to make small changes later, but for now we have the basic outline done so we can move on! Close and save the file (On Mac you can do Command+S for a shortcut). Every single time you edit the file make sure to save.

REMEMBER THAT ANY CHANGE IN FORMAT (AS IN LOSING OR ADDING SPACES) WILL CAUSE THE PROGRAM TO NOT WORK.

Extracting Frames

Our next step is to choose the frames that we want to annotate. On average we want to choose about 150 frames total for a single project (you should evenly divide how many frames you take from each project so that it adds up to the total. Ex: If I had 3 videos, I would do 50 frames per video). You should do a mixture of manual selection and automatic selection so that you have a wide variety of the frames you want to get.

For the automatic frames: To edit the amount of automatic frames you want take from each video go back to the config file and change the number next to the words that say “numframes2pick”. How much you want to put is up to you, but I usually do about half the amount of frames that I plan to pick. So if I have 5 videos and am doing 30 frames from each video I would set numframes2pick=15. Now go back to the Extract Frames tab and select the “automatic”, “GUI” (for cropping), “No” (for user feedback. If you are re-training, then select “Yes” for this, as you may not want to re-do labels for every video), and “Yes” for OpenCV. Don’t adjust any of the other settings, and press the button labeled “Ok”.

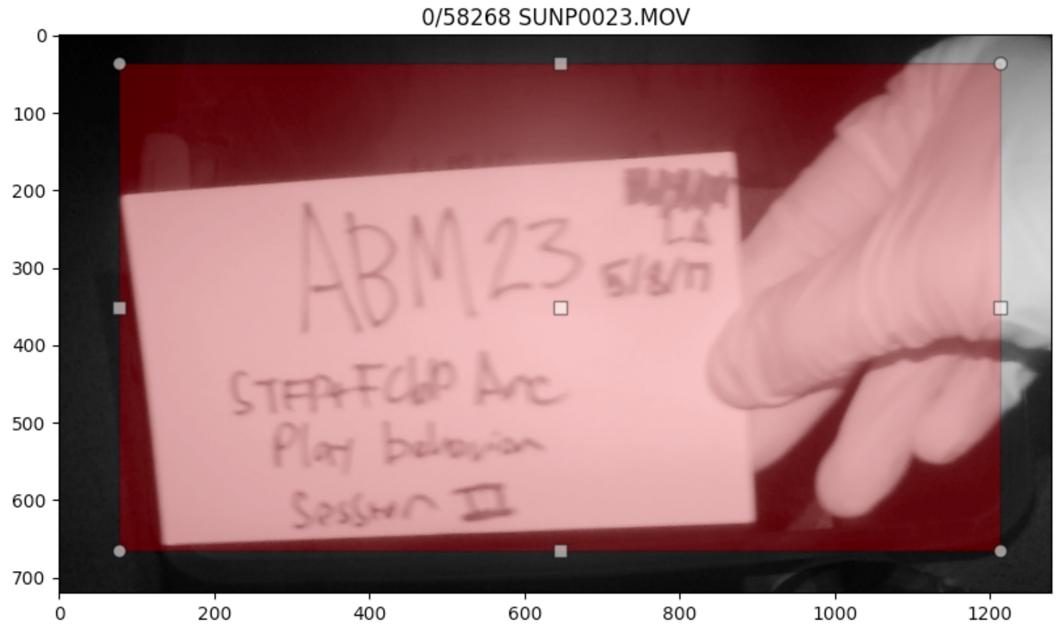
Since we’ve selected that we want to crop using the GUI you will get a pop-up with a screenshot of a video. The program expects you to now outline in the GUI what areas you want to keep in the frame. To do this you’ll have to drag your mouse over the area you want covered. The area that will be included in the frame will be highlighted in red. When you are satisfied, press crop. You will have to do this once for every video.

(If you want to see the progress of extracting, check the terminal! Anytime a loading symbol appears on your computer or the program is not responding, usually the terminal is running and executing something)

The automatic frames will take a bit to extract, so don’t be surprised if you’re waiting anywhere from 10-25 minutes for each video. If you have chosen “yes” for “user feedback" make sure to check the terminal often to see when the first video finishes extracting frames, as you will get a prompt about extracting from the other videos. If you have not already automatically extracted from the video in the past (i.e. you have not added any new videos to this specific project) then you should type “y” so that the program automatically extracts frames from every video you added (you will need to manually type “y” every single time a video is done being extracted).

Once the terminal finishes with the automatic frames, you can move on and extract frames manually!

For the manual frames: Select the manual button and press Ok. You should see at the bottom there is a button that says “Load Video”. Press it, and look for the video file that you first want to extract frames from. You will then get a pop up that asks if you want to crop the frames. Select yes, and drag your mouse over the area you want to keep before pressing “set cropping parameters” (the area kept will be highlighted red when cropping).



As soon as you finish cropping you will see that there is now a slider at the bottom. This allows you to go through each part of the video, frame by frame. It may look like a lot, but we're only extracting a small amount of frames! For manual selection, we're going to want to try to get frames where the rats are directly interacting with each other (they don't all have to be interacting though). This is because the Machine Learning model is going to need to understand how to track when the rats run into each other! Once you find a frame that you like, click "Grab Frames", and keep grabbing them until you think you have enough! You can see the amount that you currently have by looking in your directory. Once you're done, press quit. You will then repeat this process for every video that you have in the project!

Annotating/Labeling Frames

The next step that we want to do is annotate each of the frames that we got! Go to the "Label Data" tab and press the "Label Frames" button. Press "Load Frames" and select/open the video folder in "labeled-data" that you want to do first. Once the folder is open, you should be able to go through every frame that you extracted for a specific video in the last step. Adjust the marker size (pressing the check box first. If doing it on the small rats (not zoomed in) adjust until the right side says 6, and if doing on the rats that look large and extremely zoomed in, keep at 12) and appropriately label every rat in the frame. If you can not see a body part do not label it. It does not matter which rat you call "rat1", "rat2", etc. What only matters is that if you call a rat "rat1" in a frame, then all of the body parts you label for that rat in that frame must consistently be labeled as belonging to rat1 (ex: if you label the rightear of a rat and say it belongs to rat1, then that same rat must have have rat1 leftear, rat1 spine1, etc. in that specific frame).

When you're annotating, make sure to use the magnifying glass to zoom in on rats and then label (for accuracy). To exit out of magnifying (and go back to full zoom), press the back arrow. Make sure to always deselect the magnifying glass when you're not using it (like when you have zoomed into an area and are annotating).

To place a label, select the appropriate buttons and then right click the location. To drag a label, hold it by left clicking. To delete a label, press it and the delete button on your keyboard.

When you finish labeling one frame, press the next button and move on to the next frame! Your progress will be shown at the top.

Once you're finished annotating all of the frames in a folder click "Save". Congrats, you have finished labeling one video file! Repeat the process for the rest of the video files that you have (this will be asked when you press the "Quit" button). Once you're done you can press "Check Labels" to make sure that all of the images you have are saved and labeled correctly. You can see them in a sub-folder in "labeled-data" that is called the video name plus the words "_labeled".

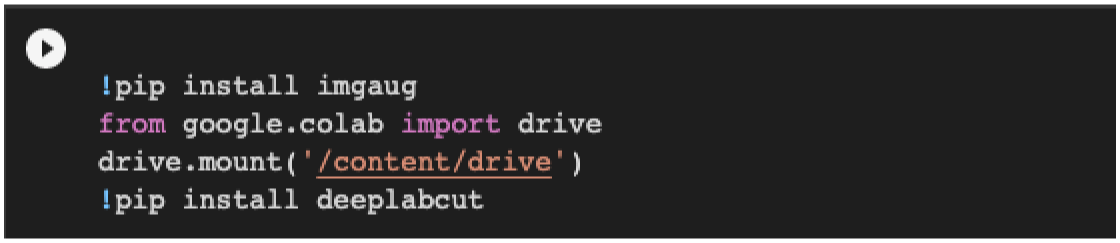
This is the last step that we'll be doing in DeepLabCut, as the next few steps will require a lot of computational power and time.

Using GoogleColab

The entire second half of the project will be done here. You will not need the GUI again unless you need to re-train your model.

The first thing that you need to do is go to your GoogleDrive and upload the project folder that you've been working on. This should contain everything in it: from the config file to the labeled frames. You can potentially not upload all of the videos if you want to save space, but you must have the videos that you want to analyze in your drive (in other words, the videos that you want to have your machine learning algorithm ran on).

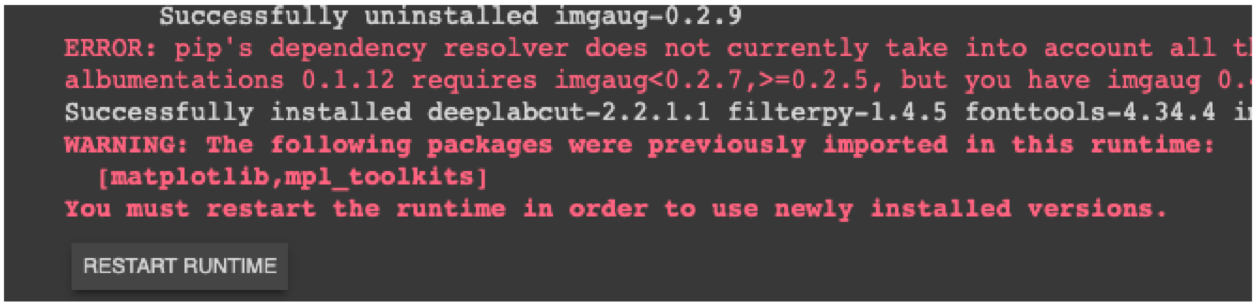
Once the folder is open, go to Google Colab and create a new notebook. Under "Runtime" (in the top left corner), select "Change runtime type". You should get a popup that asks for the Hardware Accelerator. Select "GPU" and press Save. Now add some code blocks and run these commands:



```
!pip install imgaug
from google.colab import drive
drive.mount('/content/drive')
!pip install deeplabcut
```

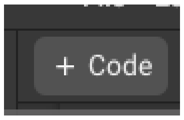
You will get a popup that asks about permitting the notebook to access your google drive files. Select “Connect to Google Drive” and choose the email account that you uploaded your project folder to. Allow Google Colab to access what it’s requesting.

Once that segment of code is finished you should get a red warning. Don’t worry about it and just press the restart runtime button that it has.

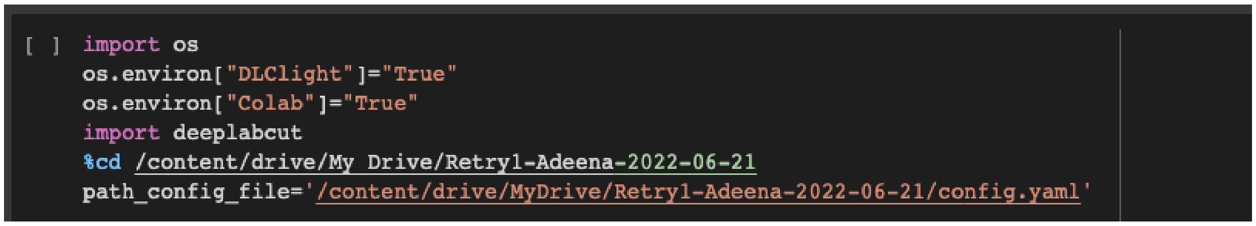


```
      Successfully uninstalled imgaug-0.2.9
ERROR: pip's dependency resolver does not currently take into account all tl
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.
Successfully installed deeplabcut-2.2.1.1 filterpy-1.4.5 fonttools-4.34.4 i
WARNING: The following packages were previously imported in this runtime:
  [matplotlib,mpl_toolkits]
You must restart the runtime in order to use newly installed versions.
```

Once it has restarted, add a new code segment to run by pressing the “+ Code”.



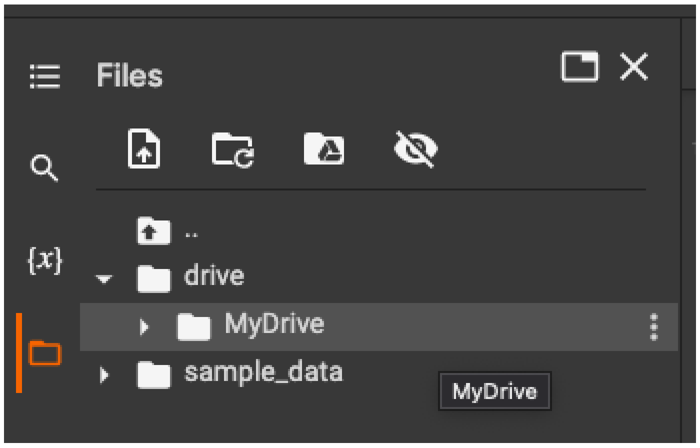
Add the lines of code below, except you should replace the “%cd /content/drive/My Drive/Retry1-Adeena-2022-06-21” and “path_config_file='/content/drive/MyDrive/Retry1-Adeena-2022-06-21/config.yaml'” with their correct paths/names. For the cd line you should be cd-ing into the project folder, and for path_config_file you should set it equal to the path of your config file. Below the code I have shown how to get both of these values.



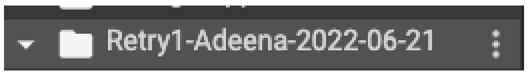
```
import os
os.environ["DLClight"]="True"
os.environ["Colab"]="True"
import deeplabcut
%cd /content/drive/My Drive/Retry1-Adeena-2022-06-21
path_config_file='/content/drive/MyDrive/Retry1-Adeena-2022-06-21/config.yaml'
```

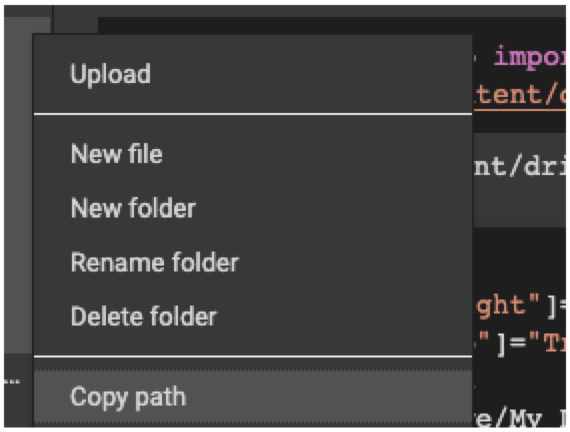
Note that you will have to re-run these two code blocks every single time you open Google Colab to connect DeepLabCut

To get the path of the project folder:
Open the files folder on the side of Google Colab and then press the folder that says drive and then MyDrive. You should then be able to see all of the folders that you have stored in your drive.

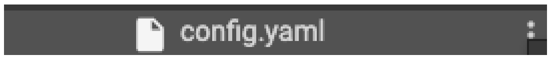


Click the dotted lines on the folder's name and press copy the path of. You can now paste the path!

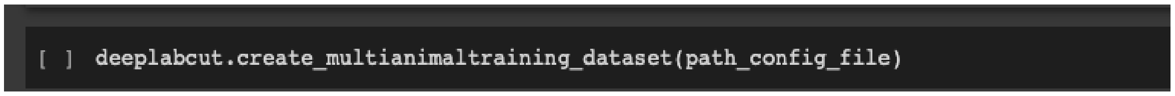




To get the path of the config file: Go back into the project folder and find the file called config.yaml. You are able to copy the path the exact same way!

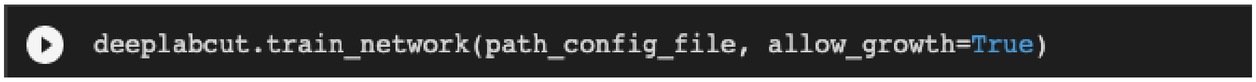


Run the code block (by clicking on the play button on the side) and wait for it to finish executing. Once it's done, we now need to create a training dataset. Run the following code in its own block:



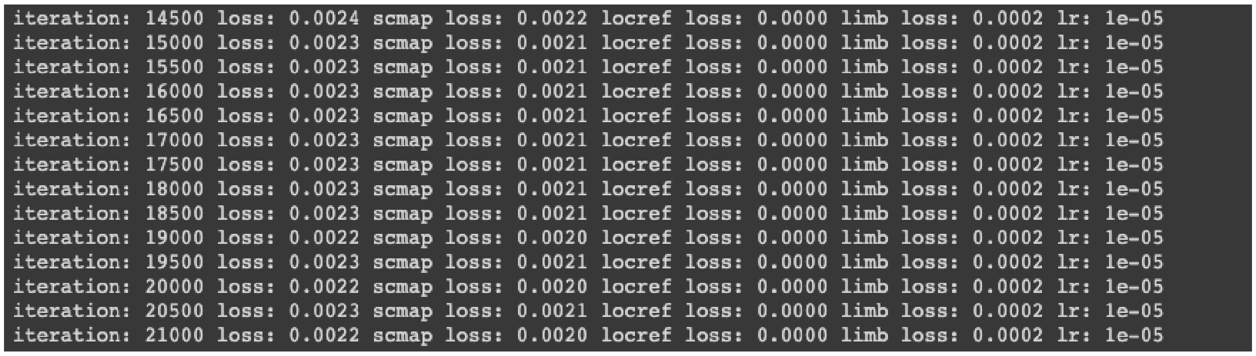
```
[ ]  deeplabcut.create_multianimaltraining_dataset(path_config_file)
```

Then run:



```
deeplabcut.train_network(path_config_file, allow_growth=True)
```

Note that training will take a while (anywhere from 4+ hours to a few days). You will need to keep the code running until it looks like the loss is plateauing. A good estimate is somewhere from 20k-100k (iterations should at least be 20k and at max 100k). Don't worry, even if you did not train enough you can always go back and re-train with new snapshots!



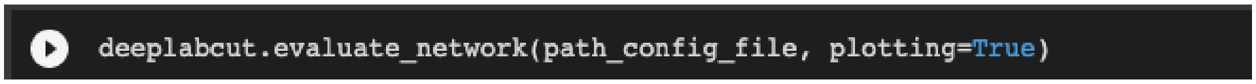
```
iteration: 14500 loss: 0.0024 scmap loss: 0.0022 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 15000 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 15500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 16000 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 16500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 17000 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 17500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 18000 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 18500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 19000 loss: 0.0022 scmap loss: 0.0020 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 19500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 20000 loss: 0.0022 scmap loss: 0.0020 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 20500 loss: 0.0023 scmap loss: 0.0021 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
iteration: 21000 loss: 0.0022 scmap loss: 0.0020 locref loss: 0.0000 limb loss: 0.0002 lr: 1e-05
```

Pictured above is an example of loss plateauing.

If GoogleColab turns off on you during training due to inactivity don't worry: we can load snapshots that resume our progress! Snapshots are progress markers that are automatically saved every 10,000 iterations. You can change this if you want by editing in the pose_cfg.yaml file, but for now we're going to leave that alone. To load a snapshot and resume training progress you need to go to your drive and find the pose_cfg.yaml. This should be in dlc-models →iteration-0 → project name + some other words→ train→ pose_cfg.yaml. Open the pose file with text editor and look for a line that says "init_weights: ". Replace the text that follows it with the path to the snapshot you want to use (this should be in the train folder as well!). Remove the . extension at the end of the name (.index, .meta, etc). An example of a valid snapshot path is: '/content/drive/MyDrive/research/Rats/dlc-models/iteration-0/ratsJun2-trainset95shuffle1/train/snapshot-78000'. Please don't forget to put it in single quotation marks!

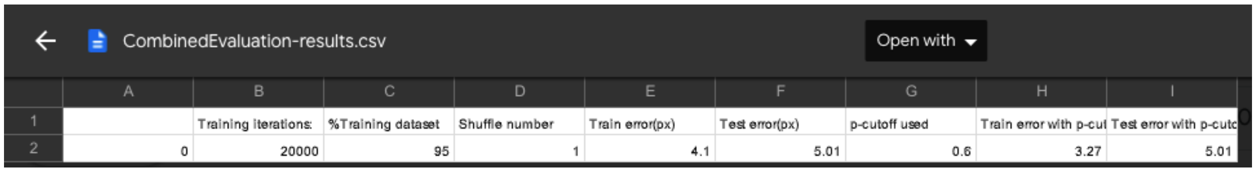
When you're ready to resume training, simply run the train network command again! You should see that this time your first iteration should be resuming from where you last left off.

Once training is done, run the command below to evaluate the network.



```
deeplabcut.evaluate_network(path_config_file, plotting=True)
```

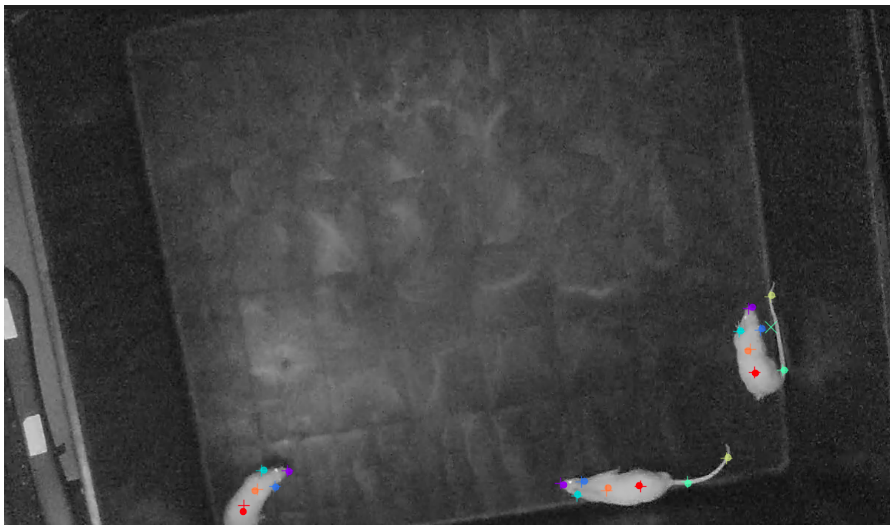
This will give you some plots and predictions for labeling (somewhere in the evaluation-results folder, you'll have to navigate back to the main folder to see it) that will let you know if you need to re-train the model. Check the pixel errors in the csv file and make sure that they are reasonable amounts before proceeding (my train error was 4.1 pixels and test errors was 5.01 pixels with 20k iterations, so your value should be around there or lower).



CombinedEvaluation-results.csv

	A	B	C	D	E	F	G	H	I
1		Training iterations:	%Training dataset	Shuffle number	Train error(px)	Test error(px)	p-cutoff used	Train error with p-cut	Test error with p-cutc
2	0	20000	95	1	4.1	5.01	0.6	3.27	5.01

Also, look at how the machine learning model predicted where the body parts would go in the "LabeledImages_DLC…." folder. The circles represent where you labeled it, while the xs are the machine learning model's predictions. Some small errors are fine, and you might notice that sometimes the x marks are on the box instead. If you want to make this more precise you can retrain and do more labeling!



If you think the model looks good, you can proceed onwards! Otherwise, skip to the next section to re-train the model and then come back to do this later.

Now that we have our fully finished training set, we want to check one more thing before we output the labeled videos: pose estimation!

To do this, all you have to run is

```
deeplabcut.analyze_videos(path_config_file,'/content/drive/MyDrive/Retry1-Adeena-2022-06-21/videos/SUNP0023.MOV',save_as_csv=True)
deeplabcut.create_video_with_all_detections(path_config_file,'/content/drive/MyDrive/Retry1-Adeena-2022-06-21/videos/SUNP0023.MOV')
```

Where you replace the videopath with your actual video's path in the drive (remember, you can copy this directly from Google Colab!) and set videotype= '.MOV'. In analyze_videos, you should also add a parameter that says save_as_csv=True. Analyzing the video will take a few hours (about 2-6ish I think) so make sure that your computer is running the command until it's finished.

When it's done, the create_video_with_all_detections will execute. This will create a labeled video, which might seem repetitive since we're going to be later making another labeled video (that may or may not look the same). The purpose of this though is to check and see how accurate your current tracking is. The good thing about seeing all the detections is that if there is something wrong with your tracking, you will be able to execute this command (ONLY EXECUTE THIS IF YOUR TRACKING IS BAD AS ITS FOR RE-TRAINING):

```
deeplabcut.find_outliers_in_raw_data(config_path, pickle_file, video_file)
```

The pickle_file will be near wherever you had your video stored, and its end will be …_full.pickle. All of these parameters (excluding config_path since we already defined it earlier in our program) will be the full path to the file.

What this will do is get frames that the machine learning algorithm didn't know how to track and will save them to your google drive folder. You will then be able to label these later (more info on that in the next section!).

If this looks good, great! We are now going to analyze the video again and create a labeled video. Don't worry, if you've already analyzed the videos the first step should be super fast! Run these two commands in a code block:

```
deeplabcut.analyze_videos(path_config_file,'/content/drive/MyDrive/Retry1-Adeena-2022-06-21/videos/SUNP0023.MOV',save_as_csv=True)
deeplabcut.create_labeled_video(path_config_file,'/content/drive/MyDrive/Retry1-Adeena-2022-06-21/videos/SUNP0023.MOV')
```

Once these commands are finished executing, you should see a fully labeled video in your google drive folder (in the video folder most likely). If it's not uploading, try re-mounting your google drive through this command:

```
[ ] from google.colab import drive
    drive.mount("/content/drive",force_remount=True)
```

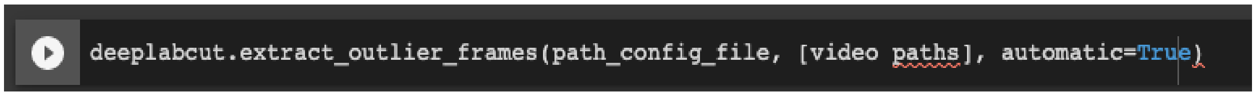
You might have to wait a bit and check up on it in a few hours, but eventually you will see the video! (If you're still not seeing the video it might be a storage issue, which means you'll have to delete a few files so that you'll have room for it to upload). The file name should be extremely long and have DLC in it. You might need to download the video to see it.

Once that video is created, you want to make sure that it doesn't work only for that video. Try using it on other videos that you did not extract frames from (remember, all you have to do is analyze a video and then label it to get the tracking to work on it). You can even use this model on videos that have a different amount of mice than the amount that you trained on! (Ex: if my machine learning model was made using videos that only had 3 rats, I can now label videos that have 6 rats by analyzing them and then running the create label function). If your model does not work on a video with a different environment/different amount of rats, you will need to extract frames from that videos (using extract outlier frames) and retrain it!

Re-training Model (Optional)

If you're noticing that your program is not tracking well, you will need to re-train your model. Full disclaimer: I have not re-trained the model before so there might be some inaccuracy in this section. Please feel free to add to/modify this section if you need to!

To retrain your model, you will have to label additional frames. You can get these frames by either running the find_outliers_in_raw_data from earlier or by doing the method below.



```
deeplabcut.extract_outlier_frames(path_config_file, [video paths], automatic=True)
```

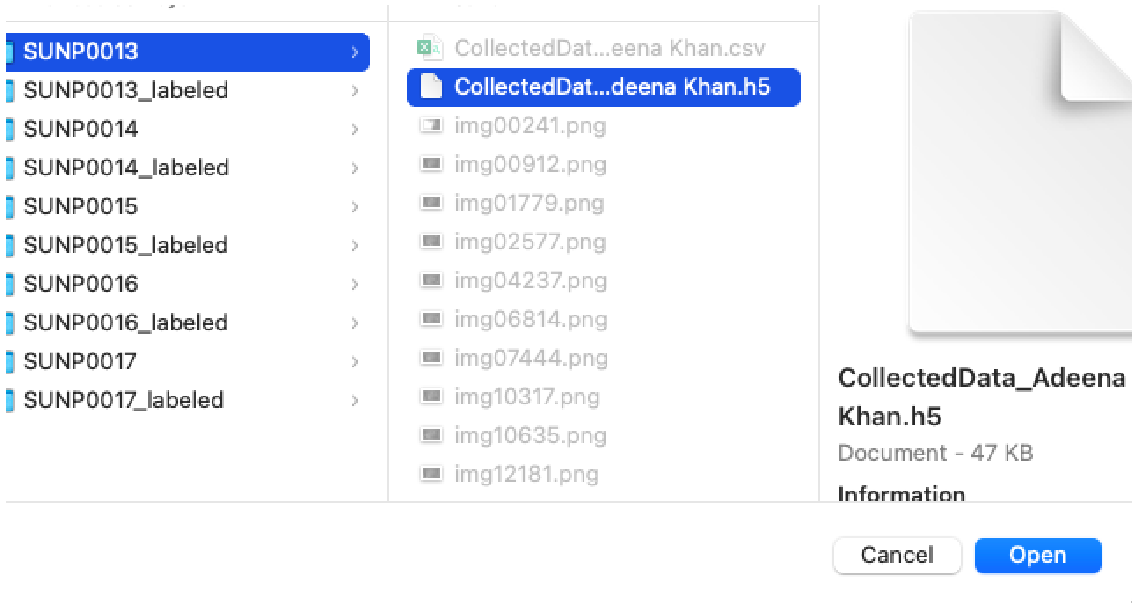
Where [video paths] is a list of the video paths that you want to run it on.

Where the amount of frames is determined by what you have numframes2pick set to in the config file (remember that you can directly edit this! You can do as many as you want, as long as it's within a reasonable range for the amount of frames you have per video.) Either of these methods should be fine, but you can only run find_outliers_in_raw_data if you ran the create_all_detection method from earlier.

(You are also welcome to extract additional frames like we did earlier (at the very beginning) and label those regularly. You can even add new videos to the project and extract labels from those as well! (not recommended unless you're running this model on a new video that was not included in the original creation).)

These new frames should be in the labeled-data folder for that specific video. To label those frames, we're going to have to switch back to the GUI. Go to your google drive and download the entire folder. Then, open up the GUI. When it asks you to load your project, use the config file of the project from your google drive (this is because this is the updated one with the labels and machine learning model).

There should be a tab/section in DeepLabCut that is titled "Optional: Extract and Refine labels on Outlier Frames". Press "Launch GUI" and select Load Labels. You then can choose label folders like before, but this time you must choose an h5 file inside of the label directory you want to use. Remember, you must have already extracted frames to do this step. If you need to, you can also extract frames instead in the GUI.



Open it and read the instructions. Set the threshold to 1, and begin to relabel! Save when you finish relabeling a set. Also be careful about deleting labels: once deleted, they can't be undone!

This step is important: once you're finished with re-labeling, quit and press the "merge button". This will put all of your labels together in a dataset.

Once you're done with that, upload the project to google drive again and create a new training dataset with your config file (same steps as last time)! Then, you need to change the init_weights in the pose_cfg.yaml file to the last snapshot you saved when you first trained it (this is extremely important). You can finally train again with the new model!

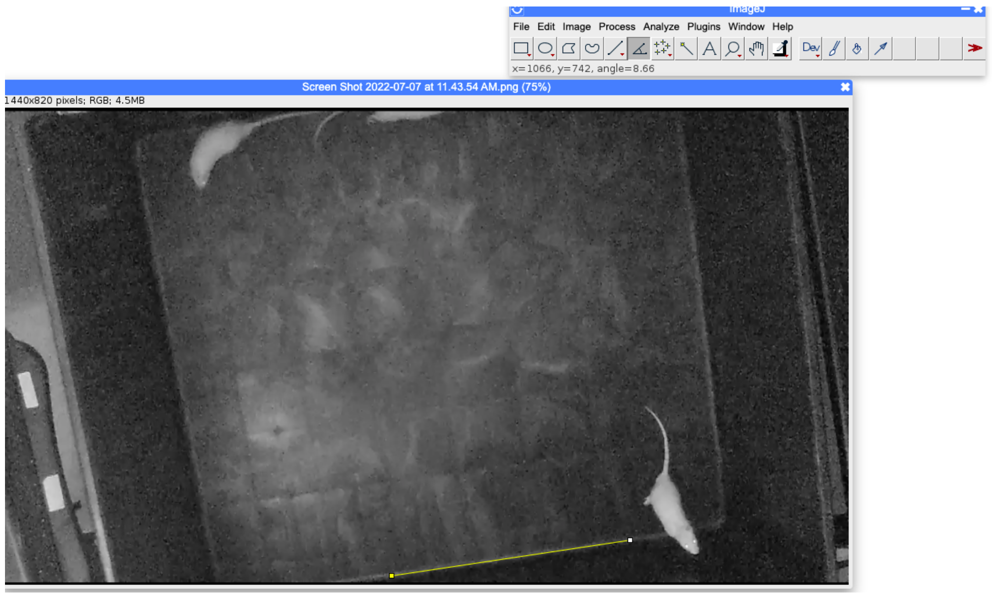
Using the CSV file

We now have a CSV file that contains all of the information we obtained from tracking (this should be next to the video it belongs to and will have the video title in its name). Download this file, and make a note of where you store it. The CSV file is a list of coordinates (x,y) for each body part you tracked, plus the likelihood that those coordinates are correct. We can use these coordinates to get the average velocity and total distance traveled for each rat in a video, plus the total amount of seconds each rat interacted with each other by running the data through the *getting_values.py* file. To do this though, you're first going to need some values.

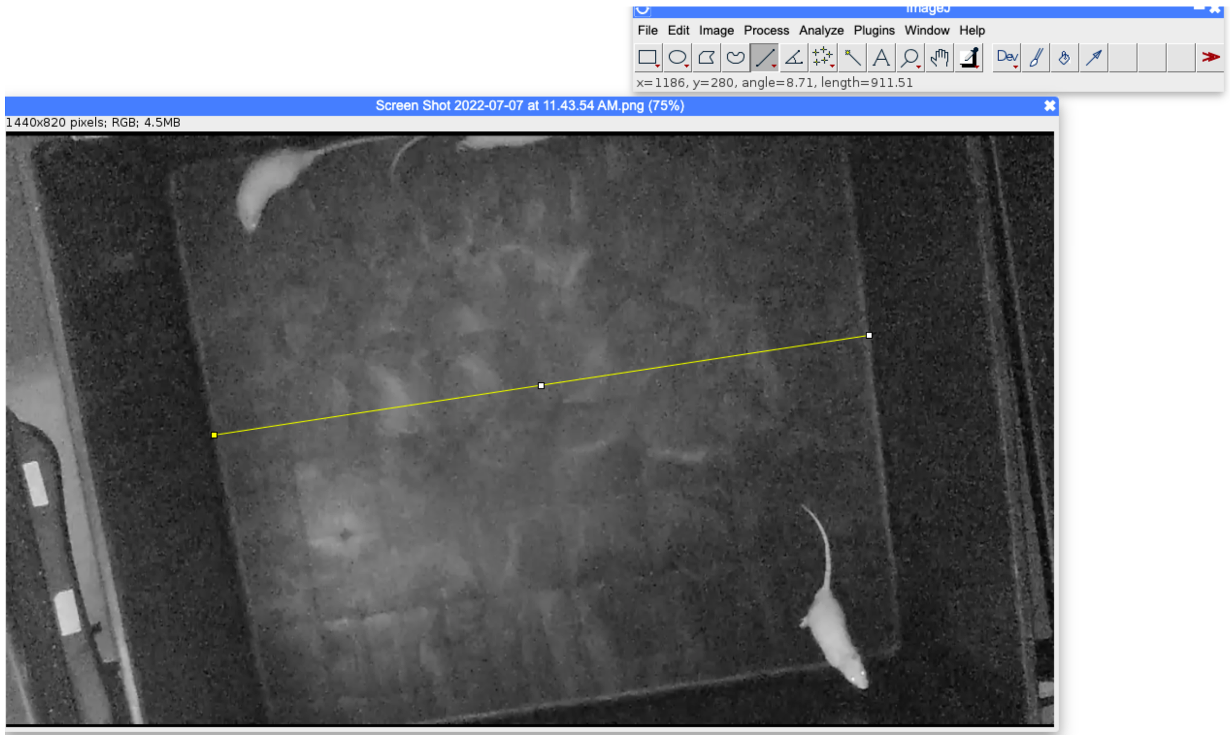
Currently, the code only examines one video at a time, so you will need to choose one video's csv file that you would like to run. It should be in the same folder/directory as *getting_values.py*, otherwise the code won't have access to it.

Once that's selected, you now need to find out what the conversion between pixels to inches is for your specific video. This is pretty easy to do! All you have to do is take a screenshot from the video you're currently examining and open the file here (File→ Open→ Select Local File → Find Image).

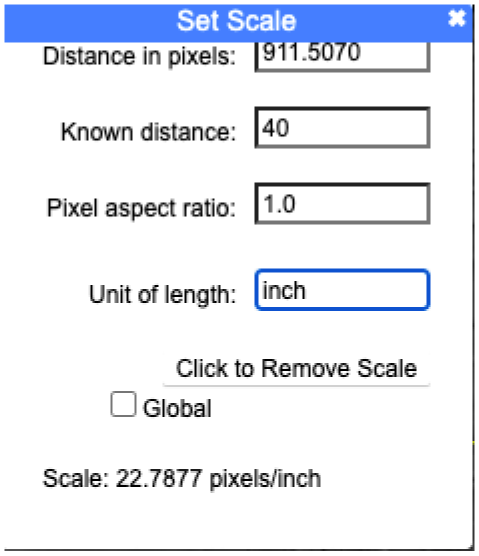
Once you have it uploaded, you now want to see what angle the box is currently at. You can check this by selecting the protractor tool and then drawing a line along the edge of the box (I believe drawing a regular line will also work). An example is depicted below, where the angle is approximately 8.66 degrees (told by the ImageJ window bar at the top right corner).


ImageJ
File Edit Image Process Analyze Plugins Window Help
x=1066, y=742, angle=8.66
Screen Shot 2022-07-07 at 11.43.54 AM.png (75%)
1440x820 pixels; RGB; 4.5MB

Next we want to measure a line across the box at (approximately) the angle we just got. Remember what the length of the line is (in the example, the length is 911.51).


ImageJ
File Edit Image Process Analyze Plugins Window Help
x=1186, y=280, angle=8.71, length=911.51
Screen Shot 2022-07-07 at 11.43.54 AM.png (75%)
1440x820 pixels; RGB; 4.5MB

Now, go to Analyze and press “set scale”. A pop up should appear that asks you to enter a variety of values. We are going to be measuring the box across so enter 40 for “known distance” and inch for “unit of length”. For distance in pixels enter the length that you just got from the line (it might already be in the box). You should get a conversion value from pixels to inches as soon as you type this in!



Distance in pixels: 911.5070

Known distance: 40

Pixel aspect ratio: 1.0

Unit of length: inch

☐ Global

Scale: 22.7877 pixels/inch

The last thing for you to do is run the python file! You can find the code here, in this GitHub respiratory. Make sure that the *getting_values.py* file is in the same place as the csv file that you downloaded earlier.

The program will ask you a few questions about the file that you're entering and the conversion scales, so answer those appropriately (you don't need to put your answer in quotation marks because the program will already do that).

These are the questions you will be asked:

"What is the file name of the csv file you want to input? Ex: Video17.csv". → Enter the name of the csv file of the video that you want to look at.

"How many pixels are in an inch (according to ImageJ)? Ex: 22.75 " → Please enter in the number that you got earlier from the scale. Don't enter any spaces.

"What second do you want to start the analysis at? Ex: 25" → Enter the second that all of the rats have been put in, after the researcher's hand has disappeared/is no longer visible in the frame. The current code assumes that all of the videos are 60 fps.

Once you enter each of these, you should get a really large output! The structure for it is so that the first large segment of code is a dictionary that contains the total distance traveled by each rat, plus their velocity every twenty seconds. There will also be a dash line, and after that there will be another dictionary, this time saying how long each rat interacted with each other in seconds.

If you have a question on how the code works, be sure to check the comments on the file (these are marked with #). They will explain what most of the code is doing/implementing.