# Using the HPC for DeepLabCut:

Contents

[Using the HPC for DeepLabCut: 1](#_Toc14790964)

[1. Set Up Research Space in the HPC: 1](#_Toc14790965)

[2. Log In to the HPC: 1](#_Toc14790966)

[3. Setup and Start DeepLabCut in the HPC server: 2](#_Toc14790967)

[4. Create a Project in DeepLabCut Using the Interactive HPC Servers: 2](#_Toc14790968)

[5. Train the Network in DeepLabCut Using the HPC grid 4](#_Toc14790969)

[6. Get Results from Your Video Analysis 6](#_Toc14790970)

[7. Refining the Trained Network 9](#_Toc14790971)

## Set Up Research Space in the HPC:

In order to run files in the HPC, you need to have access to the microwave hpc-research space. To get a folder created in the hhorch research space, contact Dj Merrill ([deej@bowdoin.edu](mailto:deej@bowdoin.edu)). Any videos used, or projects created will have to be stored in this research space in order for them to have the correct HPC path name and thus be usable within the HPC server.

All files in the hpc-research folder will have the path name ‘/mnt/research/hhorch/your\_folder/xfile’

For any issues with the HPC, Dj is your best friend. He normally responds within minutes and knows the ins and outs of deeplabcut as well. If you’re getting an error message that you do not understand and is not deeplabcut-specific, I would definitely recommend reaching out to Dj for help.

## Log In to the HPC:

Logging in to the HPC user space requires your Bowdoin username and password. To login, open a terminal and when prompted, type

$ssh -Y [username@computername.bowdoin.edu](mailto:username@computername.bowdoin.edu)

And then your password when prompted, but you will not be able to see what you are typing when you enter your password.

The computer names are ‘dover’, ‘pauling’, ‘foxcroft’, and ‘moosehead’. Dover, pauling and Foxcroft are interactive computers meaning that you can run active commands on these servers. They are open to anybody to use so if it’s running kind of slow, you can switch to a different server and see if that is faster. While these servers are more powerful than the lab computer, you can only run the smaller training portions of DLC.

For more powerful projects, you will have to login to ‘moosehead’ which gives you access to the HPC Grid of more powerful graphics processor units (GPUs) and complete the larger training sessions. This server is not interactive, so you have to submit jobs via pre-written scripts.

## Setup and Start DeepLabCut in the HPC server:

Once you are logged onto the HPC server, you will have to go through a couple of steps to prime the server to use DeepLabCut. As of version 2.0.7.1 (July 11th, 2019), you need to use Python version 3.6 to run deeplabcut. Additionally, you will work within ipython to run deeplabcut on the HPC server. You will not get any response prompt if you enter the following codes correctly

(OLD: DO NOT NEED AS OF MAY 2020. DJ HAS UPDATED ALL SERVERS) To setup python version 3.6:

$ scl enable rh-python36 /bin/bash

If you forget any part of this, you will be prompted with the error message that you are missing certain commands.

(OLD: DO NOT NEED AS OF MAY 2020)To overcome an ipython bug and be able to open ipython on the server:

$ export IPYTHONDIR=/tmp

To enable the use of GUI whilst on the HPC

1. Make sure that XQuartz version 2.8 is installed on your Mac or use MOBAXTERM for PCs
2. $ xeyes

The ‘xeyes’ prompt should open up both xquartz and a window with cartoon eyes that will track your mouse if this command is read successfully. This is just a prompt to make sure that it is working properly.

Enter the DLC Virtual environment containing the proper software.

$[jeschole@dover ~]$ source /mnt/local/python-venv/dlc/bin/activate

Once you have completed these steps, you are ready to enter ipython on your computer. To do so, type

$ipython

If this command is read successfully, it will prompt you with the version of ipython being used and the ipython specific prompts “In [1]:”

Now that you are in ipython, you can load the deeplabcut software in order to run the program

In [1]: import deeplabcut

If this is done successfully, it will take a few seconds-minutes to load and then you will be prompted again with ‘In [2]:’

## Create a Project in DeepLabCut Using the Interactive HPC Servers:

To run any project in deeplabcut, you will first have to create a project and then you can manipulate the videos within that project.

When you create a project, you will have to name it, add the desired videos for training, and determine which folder you wish to store it in.

The command to do this is

In [#]: deeplabcut.create\_new\_project(‘projectname’, ‘username’, [‘videopath’], working\_directory=’yourhpcfolderpath’

This should create a project folder in your hpc-research folder containing four folders, ‘dlc-models’, ‘labeled-data’, ‘training-datasets’, and ‘videos’. Additionally, there should be a ‘config.yaml’ file which is the editable file that is used to define the points that you would like to tag on your video and the number of frames that you wish to label. Additionally, this file should list the paths to your project training videos and the path location of your project.

Once you have created your project, you will need to save the path of the config.yaml file. This file path will allow you to work within your project using the DeepLabCut. For ease of working with this path, the DLC makers suggest storing the project path as ‘config\_path’. I recommend also storing it as this name because it will allow you to work with and understand the official guides better. To store the config file path:

In [#]: config\_path = ‘/mnt/research/hhorch/username/projectname/config.yaml’

If you have ANY typo in this, when you try to run a command using config\_path, you will be faced with an error, config\_path file not found. Common things I normally leave out are the first forward slash (/) or the ‘config.yaml’ part of the path. Always triple check your spelling and syntax whenever you get error messages.

Once you have defined your config path, you can move on to actually working with your videos to train your network. The first step is to extract frames. There are a few ways that you can extract frames – you can let the computer automatically, or you can choose the frames manually. If you let the computer choose automatically, it can do it in two ways – uniform or via kmeans clustering. Uniform will select frames in constant increments while kmeans tells the computer to select frames from regions of the video that has a lot of movement. The default setting, and what seems to make the most sense to me, is to allow the computer to select frames automatically using the kmeans clustering. To do this:

In [#]: deeplabcut.extract\_frames(Config\_path)

The program should then run through all of the frames and then select some of them that you will label. Once this is complete, it will prompt that you can label the frames.

Labeling the frames requires the use of a GUI, or an interactive screen, that will appear when prompted by the labeling command. You will have the option to load frames from specific videos and label them. This labeling should be as exact as possible so that the computer can accurately learn where each body part is. If a body part is not visible you should not place the corresponding label on the frame. To open the labeling GUI:

In [#]: deeplabcut.label\_frames(Config\_path)

If you are not confident in where you placed the labels on the images, you can check and adjust the labels using:

In [#]: deeplabcut.check\_labels(config\_path)

After you have labeled the frames, you are ready to create a training dataset for the computer to learn the body parts. To do this, you first have to sort the labeled frames into two groups; a test subset and a train subset. The computer will use the train subset to learn the assignment of the labels to their respective component. Once it thinks that it knows the label assignment, it will see how well it has learned the components by using the ‘test’ subset. To create the two datasets:

In [#]: deeplabcut.create\_training\_dataset(config\_path, num\_shuffles=1)

This step should create subsets in the dlc-models folders that contain the ‘train’ and ‘test’ subdirectory folders with the respective labeled images.

## Train the Network in DeepLabCut Using the HPC grid

Once you have created the training dataset, you are ready to train the actual networks. This will require the use of the moosehead HPC grid rather than the interactive servers. You will not run these steps live as you did the previous steps. Instead, you will submit them via a shell script and a python script to much higher powered computers where your project will be queued and then run. I found it easiest to create a folder labeled ‘HPC\_Scripts’ that had a template for each of the necessary scripts that I could then copy into each project and adjust to each project. This folder of templates is stored in the hpc-research space under /hhorch/DeepLabCut/HPC\_Scripts

Your shell script should contain the following and will tell the HPC to open and run the python script it links to:

#!/bin/bash  
#$ -cwd  
#$ -j y  
#$ -S /bin/bash  
#$ -M username@bowdoin.edu -m be

scl enable rh-python36 'export DLClight="True"; python3.6 /mnt/research/hhorch/username/Project\_name/HPC\_Scripts/train\_network.py’

Your python script should contain the script that you would normally input manually:

import deeplabcut

config\_path = '/mnt/research/hhorch/jeschole/190708-Julie-2019-07-08/config.yaml'

deeplabcut.train\_network(config\_path,shuffle=1, saveiters=50000, maxiters=200000)

Before you can submit these jobs to the HPC grid, you have to the change the modifications of the shell (.sh) scripts. To do this, change into the directory where the scripts are stored while still logged on to the interactive server. Then, you will use a ‘chmod u+x name\_of\_shell\_script.sh’ to actually change the parameters

$cd /mnt/research/hhorch/username/project-name/HPC\_Scripts

$chmod u+x training\_script.sh

Once you have these scripts saved in your project space, you can submit jobs to the HPC grid.

Log onto the moosehead server using ‘ssh -Y [username@moosehead.bowdoin.edu](mailto:username@moosehead.bowdoin.edu)’ and give password when prompted.

Once on the moosehead server, change directories into the HPC\_Scripts folder of your project

$cd /mnt/research/hhorch/username/project-name/HPC\_Scripts

Submit your project by using the commands ‘qsub’ meaning submit, -l gpu2080=1 to specify that it is going to a powerful computer (could also use ‘-l gpup100=1’) and then include the name of your shell script. You will receive the script feedback from your submissions in the HPC\_Scripts folder as .osh###### files. If your command does not do what you expected, or ends prematurely, you should check in this file for feedback and error messages.

To initiate the training of the computer, you will submit the ‘training\_script.sh’ shell script linked to the training python script. This will be the longest time that any command will run. The default settings for the training are to run for 1,003,000 iterations. However, looking at the graph of error over time, the error seems to be fairly stagnant after ~400-500,000 iterations. To stop testing before 1,003,000 iterations, you can set maxiters=###### in your python train\_network.py script.

To submit, follow steps described above:

$cd /mnt/research/hhorch/username/project-name/HPC\_Scripts

$qsub -l gpu2080=1 training\_script.sh

Your project may not start immediately. TO see if it is queued to be run or if it is running, you can check by typing ‘qstat’ in the terminal. If it is queued, you will see ‘qw’. If it is running, you will see ‘r’. You should also receive an email once your project has started running.

The training should take between 12-24 hours to complete depending on how many iterations of training you wish to complete. Once it is complete, you should receive an email saying so. While the project is running, data will begin to accumulate in the ‘learning\_stats.csv’ file in one of the subdirectories in the dlc-models folder. DO NOT open the learning\_stats.csv file to see the progress of the training. If you want to check on the progress of your training, you can look at the ‘tail’ of the csv file without opening it using a command in moosehead. To do this, cd into the train subdirectory of the dlc-models folder and then use the command:

$ tail -f learning\_stats.csv

You will receive an output of a list of numbers. These numbers are the iteration numbers that the training program is on, so you will know where it is at in relation to the maximum number of iterations that you set up.

You will receive an email once your project has been completed. I recommend checking the training.osh#### file to see the returned script and make sure that everything was completed successfully, and you were prompted with “training complete, you can now evaluate your training using evaluate\_results” (or something like that).

Once you are ready to evaluate your results you will submit the evaluation script to the HPC grid using the same method:

$cd /mnt/research/hhorch/username/project-name/HPC\_Scripts

$qsub -l gpu2080=1 evaluate\_script.sh

This step should be much quicker (~5-10 minutes maximum) and will create a new ‘evaluation\_results’ folder in your project folder. It will also create a .csv or excel file which will contain the parameters of your training and the overall errors in pixels from the training. If you are happy with the pixel error (Jack O’Brien thought that there is likely a published paper that has parameters for what an acceptable pixel error might be…as of July 12th, we don’t know what this might be), you can move on to analyze your videos with the goal of extracting behavioral data. If you’re not happy with it, you will want to analyze your training videos and extract outlier frames, refine the labels given to them by the computer and retrain. But regardless, you will have to analyze videos.

Analyzing videos is also done on the HPC grid and you will submit your job the same way as already done. However, before submitting the job, you will want to edit your python script to include the pathnames for the video files that you wish to analyze. Once you have done that, you will submit to the grid:

$cd /mnt/research/hhorch/username/project-name/HPC\_Scripts

$qsub -l gpu2080=1 analyze\_script.sh

As before, you should receive an email when the script starts running and when it is complete. This script will take longer depending on how many videos you are analyzing, but it will probably take ~20-30 minutes at least. The analysis is the last step done on the HPC Grid. All subsequent steps are small enough that they can be done on the HPC Server.

## Get Results from Your Video Analysis

Log back onto the interactive HPC server (dover, Foxcroft, or Pauling) and open ipython and import DeepLabCut (see section 3 to review these steps).

Once you have analyzed your videos using your desired training iteration, there are a few options on how you can look at your data.

To visualize the labels that you have just placed, you can create a labeled video which will plot your labeled points on the entirety of the video. This is useful to see how accurate your labels really are and if they skip around a lot in the image or stay in place.

To add labels to a video:

In [#]: deeplabcut.create\_labeled\_video (config\_path, [‘video\_path\_file’])

In DLC version 2.0.7, you can add features like a skeleton connecting some of the labels, or adding trailing points to trace the previous label positions. I’m not sure how applicable these features will be for the videos taken in Hadley’s lab.

Once you have visualized the points on the video, you can plot the points of the labels and also plot the likelihood that each point is in the correct label position.

First, filter your predictions to ‘smooth’ the results using an ARIMA statistical model

Deeplabcut.filterpredictions(config\_path, [‘/mnt/research/hhorch/…videos/xxx.m4v’], videotype=’.m4v’)

Next plot the coordinates so that you can visualize the plotted xy coordinates

Deeplabcut.plot\_trajectories(config\_path, [‘/mnt/research/hhorch/…videos/xxx.m4v’], videotype=’.m4v’)

The graphs created by this command:

*Plot:* This is a plot of x coordinates over time (dashed line) and y coordinates over time (solid line). An ideal graph should have visibly traceable/followable lines without convoluted offshoots. A bad graph will have a lot of noise, spikes or jumps in the graph

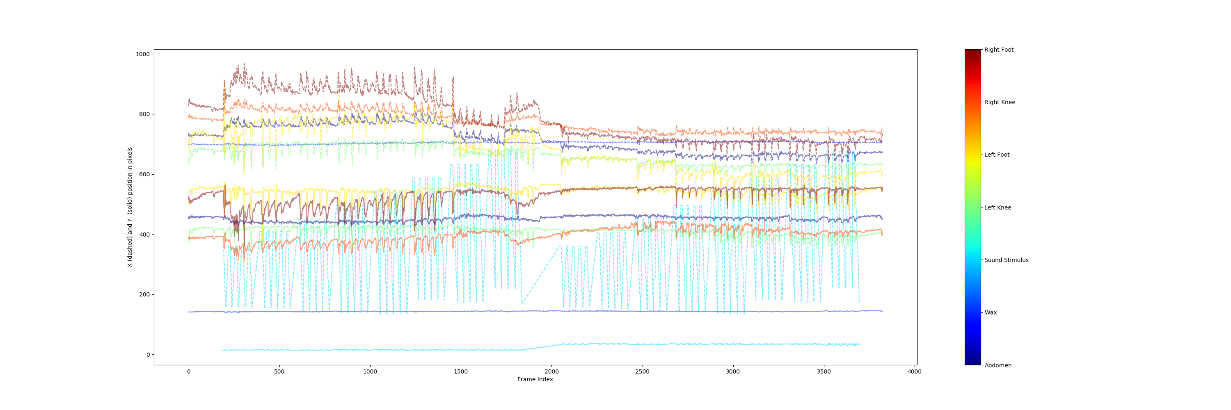


Figure : Ideal Plot

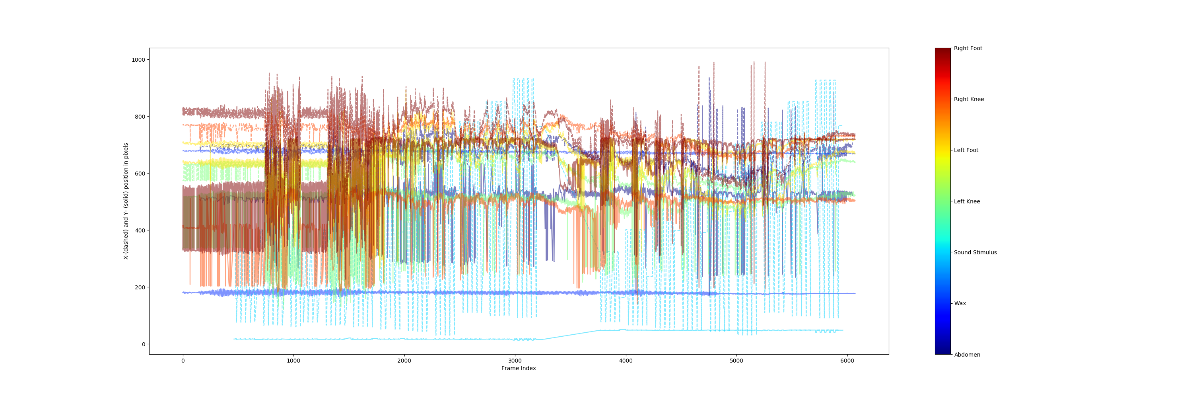


Figure : Bad Plot

*Plot likelihood:* This plot shows the likelihood (from 0-1) that a label is placed by the computer where it should be. An ideal plot would be concentrated at the top of graph and have very few spikes down from the optimal 1.0 area. A bad plot would have many spikes, meaning that the computer is fairly certain that most labels are plotted in different places than they should be

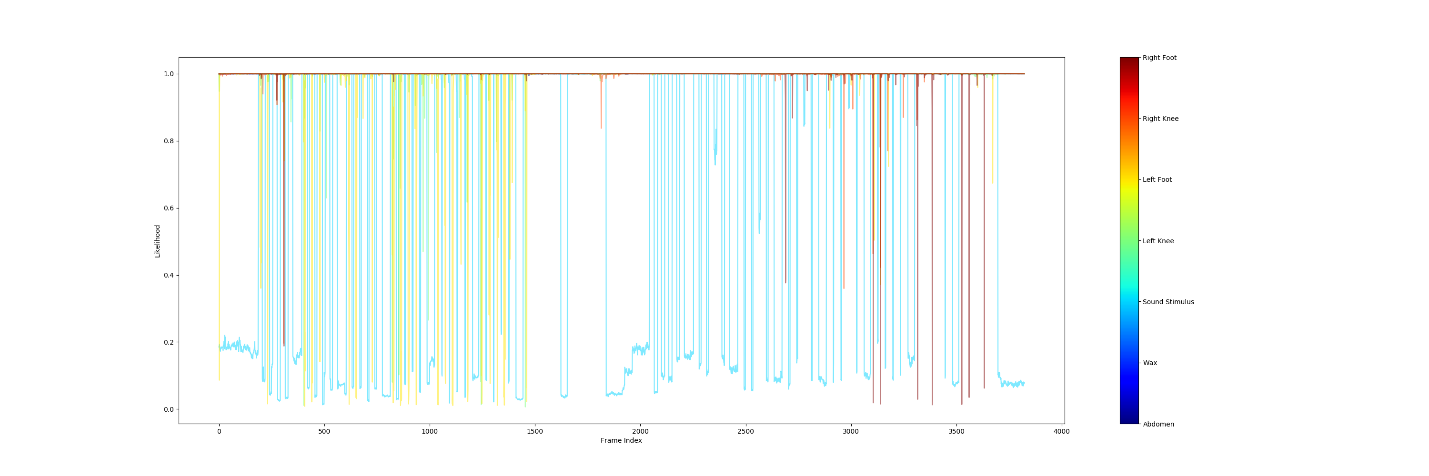


Figure : Reasonably Good Plot Likelihood

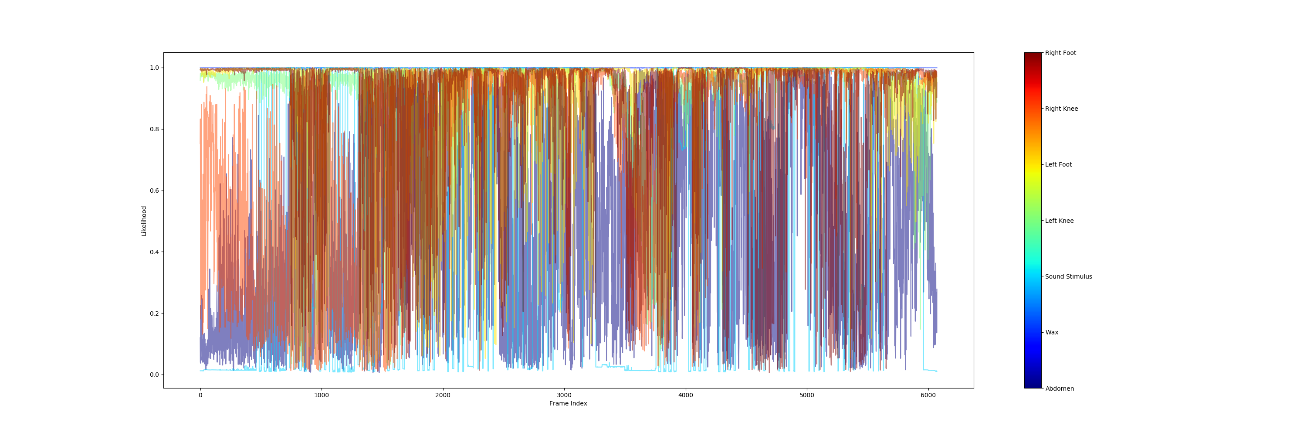


Figure : Bad Plot Likelihood

*Plot Trajectories:* This plot shows the (x,y) coordinates from each individual frame plotted on one graph. This allows you to visualize the overall movement of the cricket. Ideally, this plot should show a smooth area of movement around the area of each label. In a bad training, the plot will show ‘jumps’ of isolated labels, or dots in areas that do not correspond with the body part.

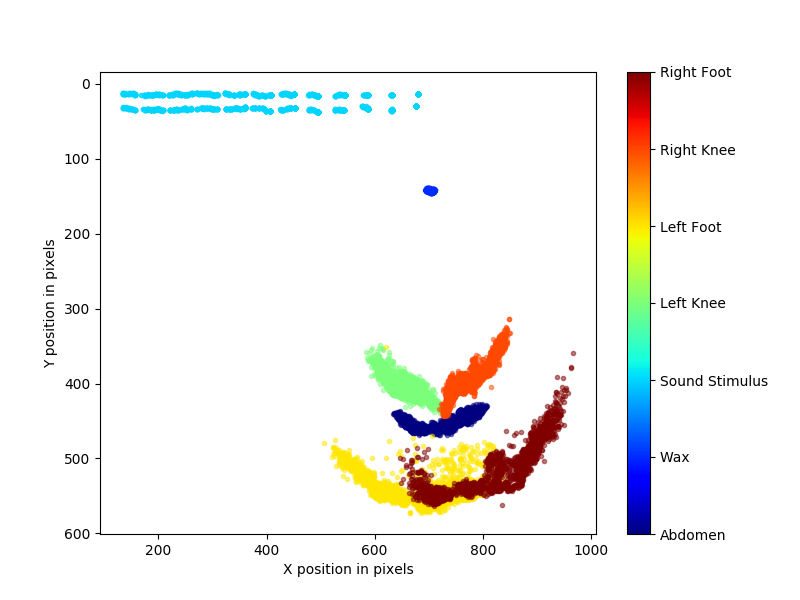


Figure : Good Trajectory

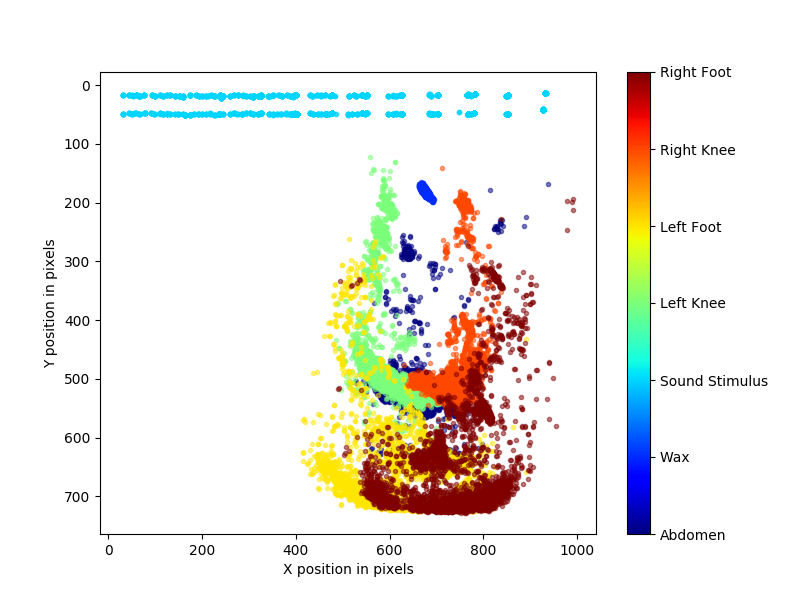


Figure : Bad Trajectory

## Refining the Trained Network

If you are unhappy with the training (have a bad trajectory plot, likelihood plot, or regular graph) you will want to refine the network. This process allows you to refine the computer-generated labels in the frames of your videos to move the inaccurately placed labels into their correct locations. To do this, you will use a DLC generated outlier algorithm which allows you to sort through outlier frames and select frames from that pool to refine.

To extract the outlier frames:

Deeplabcut.extract\_outlier\_frames(config\_path, [‘/mnt/research/hhorch/…videos/xxx.m4v’])

\*optional\* if there is an area of outlier frames or specific behavior pattern that you wish to refine the labels for, you can manually select frames to label. Add in ‘manual’ after the video path in the above command. This will open the GUI which has a scroll bar to scroll through the various frames of the video. Once you have scrolled to the area of interest, click on ‘grab frames’ to save the frame(s) that you would like to label.

Once you have extracted the frames that you wish to extract, you can adjust the labels using the ‘refine\_labels’ command

Deeplabcut.refine\_labels(config\_path)

This will open a GUI and you will have to select the video files that you wish to refine. Once you have opened the frames, click and drag the labels to where they should be placed. You will not have to move every single label in every single frame, just the ones that are not placed well. If the label is present but the feature is missing (i.e. there is no sound stimulus in the frame) you can delete the label (in this case it would be the sound stim label) by right clicking on the label and then selecting ‘yes’ when prompted to delete the label. Once you have worked through all your extracted frames, hit ‘save’ in the GUI and either ‘yes’ or ‘no’ to refining more videos, depending if you have extracted frames from multiple videos.

Once you have refined all of the labels that you need to, you will have to merge the newly labeled frames with the previously labeled training frames. You can do this with the following command:

Deeplabcut.merge\_datasets(config\_path)

Next, you will have to create a new training dataset that will separate the labeled frames into the ‘test’ category and the ‘train’ category so that the network can be trained again.

Deeplabcut.create\_training\_datasets(config\_path)

Now, you can submit the project for training again using the HPC Grid as explained in Section 5.

Once you have retrained the network, you will complete the following steps again – evaluate results, analyze video etc.

\*NOTE\* Because you have already analyzed the videos, the software may return that the videos have already been analyzed, filtered, plot poses created etc. For the videos that you want to reanalyze, you will have to delete the previous .csv files, filtered .csv files or plots before rerunning the corresponding commands.