LS/MCB100 | Computational Ethology

# WEEK 04 | INTRODUCTION TO POSE ESTIMATION

Souvik Mandal, PhD
Post-doctoral fellow
Center for Brain Science / Molecular and Cellular Biology
Harvard University

Since behavior can be denoted as the temporal sequence of body poses, estimating the changes/movement of different body parts, or the overall body poses of the animal is one of the most accurate ways to measure its visible behavior. Evidently, this needs tracking of its body parts. Earlier, many ethologists used to make videos of the animal(s) in focus, and later, went through each frame to annotate the intended body parts, or the behavior. However, whereas manually annotating body parts in thousands of frames is time-taking and tiresome, manually annotating behavior is prone to the error and bias of the annotator. With the advent of artificial intelligence, now we have tools that can track the body parts from an infinite number of frames from several videos (and several animals) with just a few hundred manually annotated frames. DeepLabCut, or DLC in short, is such a program (along with SLEAP, DeepPoseKit, etc) that is invented at Harvard and open-sourced. For this course, we will teach you “pose estimation” in animals using DeepLabCut.

# DeepLabCut: IN A NUTSHELL

- It’s a Python-based program.
- The program first extracts random (as well as user-defined) still image frames from a video (or several videos) provided by the user.
- The user then annotates or *labels* the intended body-parts of the animal(s) in these extracted frames.
- DeepLabCut "learns" the position of the body-parts from these manually labeled frames; we call these frames "training dataset", and the learning process as "training the deep-neural network".
- A network typically needs several iterations of training to be robustly trained.
- The training process requires a lot of computation and so, it is a lot faster when a GPU is used (instead of the CPU).
- After each iteration, a user can *evaluate* the quality of the trained network.
- Once the network is sufficiently trained, it can extract the position of body-parts from novel frames, and therefore from the entire video as well as similar novel videos. Again, using a GPU makes the process faster.

For a detailed overall view of DeepLabCut, please visit the site by the creators.

# Getting started with DeepLabCut:

## Step 1: Installation

Since GPU is an expensive component, we will do the labeling part in our personal computer and the training part in Google Colab. Next onwards, whatever you need to type in the terminal is typed in `green` in this document. Here is a video guide for installing DeepLabCut on your computer.

1. First, make sure you have Anaconda 3 installed on your computer. Visit this page to know how to verify it. Click here if you don't have it installed.
2. Then, make sure you have Python 3 installed on your conda environment.
    - If you have installed the full version of Anaconda 3, you should have Python 3 installed as well.
    - To check whether you already have Python 3 installed, open the Anaconda navigator and click on the Environments tab from the left menu. Under base (root), scroll down until you see "python". Check the version mentioned on the right side of the row.
    - However, your computer may run a different version of Python (and that's completely fine). If you want to check that, go to the terminal (you can do it through spotlight search, press Command and Space), type python --version and press return in your terminal, you may see a different version. You don't need to update this.
    - However, if you want to update the Python installed on your computer, you can download it through https://www.python.org/downloads/mac-osx/.
    - Alternatively, you can install Python 3 on your Mac using Homebrew. Click here for that.

3. Once you have Anaconda installed on your computer, open the terminal on your computer. If you are using a Mac, it will probably open in the base environment - if it is, you will see (base) at the beginning of the command line. If not, set it to the base environment. To do so, see 00:48 - 01:41 minutes of the guide video.

4. Set the directory to the folder where you want to save the DeepLabCut files. In my case, I chose “Documents”.

```
(base) Souviks-MacBook-Pro:~ souvik$ cd Documents  #this will set the directory to Documents
```

5. Next, clone DeepLabCut from git. For that, copy and paste the following command in green below. The download will take some time. Once completed, check the newly downloaded folders and subfolders.

```
(base) Souviks-MacBook-Pro:Documents souvik$ git clone https://github.com/DeepLabCut/DeepLabCut.git
```

6. Through the terminal, set the directory to the conda-environments folder inside the DeepLabCut folder.

```
(base) Souviks-MacBook-Pro:Documents souvik$ cd DeepLabCut/conda-environments
```

7. Now, open the DEEPLABCUT.yaml file using some text editor and add - wxpython<4.1.0 below the line 25 (where - ffmpeg is written).

8. Next, you need to create a virtual environment on your computer within which DLC will work.

```
(base) Souviks-MacBook-Pro:conda-environments souvik$ conda env create -f DEEPLABCUT.yaml
```

   - This step can take some time.
   - Also, instead of "DLC-CPU", you can use any name you prefer like “CompEtho”. Just don’t use any name with a space.

9. Next, you have to activate the newly created conda environment.
   - For Mac `(base) Souviks-MacBook-Pro:conda-environments souvik$ conda activate DEEPLABCUT` (or the name you chose for the conda environment)
   - For Linax, type `source activate DEEPLABCUT` (or the name you chose for the conda
   - For Windows, type `activate DEEPLABCUT` (or the name you chose for the conda environent)

9. Now start python.

   - For Mac (`DLC-CPU`)
   ```
   Souviks-MacBook-Pro:conda-environments souvik$ pythonw -m deeplabcut
   ```
   - On a Linax or Windows computer, type `ipython`
   - This will bring the command prompt (which looks like >>> ) and type the following

   ```
   >>> import deeplabcut
   >>> deeplabcut.launch_dlc()
   ```

   If a window like the figure below pops up, you have installed DeepLabCut on your computer successfully. Congratulations!


10. Once you are done with all of your work with DeepLabCut, close the GUI, and return to your terminal. Deactivate the conda environment by typing `conda deactivate` in your terminal command line. You may see that your command line will be back to (base).
11. To update DLC,  use the command `pip install --upgrade deeplabcut` inside your condo environment.
12. For more information on installation, please visit this. For more information about how DeepLabCut works, visit this.

# Step 2: Creating a project

Click here for the video tutorial.

1. First, open the terminal.
2. Activate the conda environment that we have created before.
3. Now, run Python within the conda environment.
4. Next, import DeepLabCut (DLC).
5. And launch the DLC graphical interface.
6. Click the “Manage Project” tab on the window that pops up on your screen.
7. First, you have to name the project and the name of the experimenter. Avoid using any space while naming anything.
8. Next you have to load the videos from which the frames for the training will be extracted.
9. Try to choose some videos that have diverse background, lighting conditions, animal body size and postures, and if possible and or needed, different animals.
10. Also, you may want to label about 200 frames for a good training data set.
11. If you want, you can add more videos later.
12. You can also select where you want to save this project.
13. Keep in mind that if you want to train the model using Google Colab, you should check the “Copy the videos” box as well. This copies all the videos within the DLC folder that will be created for the project.
14. Check the “Multi-animal project” box if it is so.
15. Now click OK.
16. And you have created the new DLC project.
17. Next you have to edit the config file.

# Step 3: Editing config file

Click here for the video tutorial.

1. In the **Manage Project** tab, click **Edit config file** and a new window shall pop up on your screen.
2. The config file contains several important features of your DLC projects, but the three most important features that require editing are the body parts, the number of frames to pick, and the skeleton. To know about other features, see next page.
3. First, label some body parts that move relatively less compared to other body parts in all of the frames throughout the videos.
4. Next, mark those body parts which are important for your question.
5. You also need to consider the morphology and or anatomy of the body-parts. So, you may need to add more body points.
6. You may want to include some other body parts which you may want to use in future.
7. The skeleton contains the information about the relative position of the body-parts. And thus, it helps the model to perform better in predicting the body-parts, especially similar looking ones.
8. To define a skeleton, we link the neighboring body parts. Connect all the body parts to the skeleton.
9. You can use abbreviations to name the body parts.
10. Names of the body-parts should start with some letter and not numbers. Also, the names do NOT contain any space.
11. Next, you can change the frames to pick from each video, which is 20 by default.
12. Now, edit the skeleton. Keep in mind that all the body-parts should be connected to the skeleton.
13. Save the file.

# BOX 1: Glossary of parameters in the project configuration file (config.yaml)

*The config.yaml file sets the various parameters for generation of the training set file and evaluation of results.*
*The meaning of these parameters is defined here, as well as referenced in the relevant step.*

**Parameters set during the project creation:**

**task:** Name of the project (e.g. mouse-reaching). ( do not edit)
**scorer:** Name of the experimenter (do not edit)
**date:** Date of creation of the project. (do not edit)
**project_path:** Full path of the project; edit this if you need to move the project to a cluster/server/another computer or a different directory on your computer
**video_sets:** A dictionary with the keys as the full path of the video file and the values, crop as the cropping parameters used during frame extraction.
(use the function add_new_videos to add more videos to the project; if necessary the paths can be edited manually, and the crop values are designed to be edited manually).

**Important parameters to edit after project creation:**

**bodyparts:** List containing names of the points to be tracked. The default is set to hand, Finger1, Finger2, Joystick. Do not change after labeling frames (and saving labels).
You can *add* additional labels later, if needed.
**numframes2pick:** This is an integer that specifies the number of frames to be extracted from a video or a segment of video. The default is set to 20.
**colormap:** It specifies the colormap used for plotting the labels in images or videos in many steps.
Matplotlib colormaps are possible (https://matplotlib.org/examples/color/colormaps_reference.html).
**dotsize:** Specifies the marker size when plotting the labels in images or videos. The default is set to 12.
**alphavalue:** Specifies the transparency of the plotted labels. The default is set to 0.5.
**iteration:** This keeps the count of the number of iterations used to create the training dataset. The first iteration starts with 0 and thus the default value is set to 0.
Do not change this manually.

If you are extracting frames from long videos:
**start:** Start point of interval to sample frames from when extracting frames. Value in relative terms of video length, i.e. [start=0,stop=1] is the full video. The default is set to 0.
**stop:** Same as start, but the end of the interval. Default is 1.

**Related to the Neural Network Training:**

**TrainingFraction:** This is a two digit floating-point number in the range [0-1] to split the dataset into training and testing dataset. The default is set to 0.95.
**resnet:** This specifies which pre-trained model to use. The default is set to 50 (user can choose 50 or 101, see also Mathis et al, 2018).

**Used during video analysis:**

**batch_size:** This specifies how many frames to process at once during inference (For tuning of this parameter see Mathis & Warren 2018).
**snapshotindex:** This specifies which checkpoint to use to evaluate the network. The default is set to -1. Use *"all"* to evaluate all the checkpoints.
Snapshots refer to the stored TensorFlow configuration, which holds the weights of the feature detectors.
**p-cutoff:** This specifies the threshold of the likelihood and helps distinguishing likely body parts from uncertain ones. The default is set to 0.1.
**cropping:** Specifies if the analysis video needs to be cropped. The default is set to False.
**x1,x2,y1,y2:** These are the cropping parameters used for cropping novel video(s). The default is set to the frame size of the video.

**Used during refinement steps:**

**move2corner:** In some (rare) cases the predictions from DeepLabCut will be outside of the image (due to the location refinement shifts).
This binary parameter makes sure that those points are mapped to a user defined point within the image so that the label can be manually moved to the correct location. The default is set to True.

**corner2move2:** This is the target location, if move2corner is True. The default is set to (50,50).

# Step 4: Extract still frames and labelling

1. Click the **Extract frame** tab.
2. You can either rely on the DLC algorithms to extract the frames, or you can manually do it yourself.
3. Click **OK**. The extracted frames will be saved in the sub-folder named after each video file within the "labeled-data" folder in the newly created DLC project folder.

   Quick notes

   While *kmeans* choose frames based on its visual dissimilarity compared to the rest of the frames, uniform method extracts frames that occur in a certain temporal interval in the videos. So, the *kmeans* method ensures that the extracted frames look different. As the *kmeans* method requires more computation than extracting frames using the uniform method, it may take some time for long videos.

4. Next, you need to label the frames. For that, click the **Label frames** tab and then click the **Label Frames** button.
5. Click the **Load frames** button on the newly popped up screen.
6. First, you have to label the frames extracted from one video. You can label the frames from the other videos later.
7. Now click **Open** and you will see the first extracted frame from the first video.
8. You can adjust the size of the markers (you can do the same in your config file as well). Also, you can zoom into some part, if you want, and you can pan the frame.
9. You can see the names of all the body parts in the side panel.
10. Once you mark one body-part, DLC automatically goes to the next body part.
11. You can readjust a marked body point as well. However, you cannot delete one.
12. Once you label all the parts in one frame, go to the next frame.
13. On the top of the frame, you can see which frame you are currently viewing and labeling.

14. Try to be as consistent as possible while labeling a body-part across frames. **Correct and consistent labeling is perhaps the most important part of the training process.**
15. Once you are done with labeling all the frames from one video, **Save** it and click **Quit**.
16. DLC will ask whether you want to label frames from another video.
17. If you have frames from other videos to label, click yes and load the frames from that video.

**Fun fact:**

You can share the task of marking frames with other people. For example, each team member can mark 80 frames and thus, each team will have 240 labelled frames. For that, all members create the DLC file in their own computer. One member can extract the frames from all the videos. These frames will be saved in the "labeled-data" folder. Now, share those folders with other members and chose which folders each member want to label. Once you save the labeling the frames in one folder, it will be saved in a separate folder named as the name_of_the video_file_labeled within the labeled-data folder. One member shall compile those _labeled folders and can use the whole dataset for the training.

18. After labeling all the frames from all the videos, you can check whether you have miss-labeled any frame.
19. Next, we need to create a training dataset and start training the neural network using Google Colab.

# Step 5: Training

1. To access Google colab, first, sign in to your Google account.
2. Upload the project folder to your Google drive.
3. Click here to open the Colab notebook for DLC and follow the instructions.

| | |
|---|---|
| **Computation (lab part)** | Click here to go to the course computation page and go through the Week 3 guide. Fill up your lab notebook as well. |
| **Before the next meeting** | Finish the first round of your training the neural network. |