DLC Workshop

The machine learning model training will be carried out on Google Colab for training a DLC model – however, if you want to work with your own data in the future, installing the DLC GUI is necessary.
If you are interested in using DLC on your personal computer, please follow this tutorial for installing DLC. https://deeplabcut.github.io/DeepLabCut/docs/beginner-guides/beginners-guide.html The guide also contains instructions for how to use DLC.

For a video tutorial on how to use the GUI, see
https://www.youtube.com/watch?v=ofFx0vTMSxE

DeepLabCut Practical Introduction:

Video analysis using marker-less pose estimation has revolutionized the way researchers study animal and human behaviour, allowing for precise tracking of movement without intrusive methods. One of the most popular tools in behavioural neuroscience for this purpose is **DeepLabCut (DLC)**, a deep learning-based toolbox that enables efficient and accurate pose estimation. The output of DLC are x,y coordinates of animal key points, such as nose, tail, ears, etc. Using these coordinates, we can analyse movement and behaviour.
In this exercise, we will explore the core components of DLC, practice annotating behaviour videos to generate the data, and gain hands-on experience using DLC's graphical user interface (GUI) and Jupyter Notebook.
DLC can be used both with the GUI (which allows for easy labelling, training, and analysing of behaviours without coding experience) and in code using e.g. Jupyter Notebooks (which allows us to run DLC in Google Colab, thus not requiring installation of the DLC code environment for training models. However, the GUI is required in order to label the key points in the data).

DLC demo and introduction to using the GUI

Live demo of the DLC GUI will be conducted

Hands-On DLC Activities

In this section, you'll have the chance to interact directly with DLC using both the GUI and the Jupyter Notebook.

Activity 1: Running Analysis with Pre-Labeled Data

- **Objective**: Run a Jupyter Notebook with pre-labeled data to obtain key point tracking of motion. Notebook one provides a demo Jupyter Notebook for tracking a single mouse. Notebook two provides a demo for tracking 3 mice.
- **Notebook**: Single animal tracking – please download the folder with data and script from https://we.tl/t-ynGT0lGW8I or on Github https://github.com/AnnaStuckert/DeepLabCutWorkshupAlmer-a2025
- **Run the script DLC_training.ipynb**
-

- Alternatively, if the provided notebook does not work for you, you can try the DLC authored pre-made notebook https://colab.research.google.com/github/DeepLabCut/DeepLabCut/blob/master/examples/COLAB/COLAB_DEMO_mouse_openfield.ipynb
- **Instructions**:
    - In order to use Google Colab, you need to sign in with your Google account.
    - Detailed description of each step is provided in the notebook.
    - Google Colab allows usage of GPU for faster computing. To use this, in the top-right corner press 'connect to GPU'. This may automatically connect. Upon opening the Jupyter Notebook.
    - To run a cell on the notebook, simply press the 'play' button in the top-left corner of each cell.
    - If you get the warning '**Warning: This notebook was not authored by Google',** do not worry, you can proceed by pressing 'run anyway'
    - If you get stuck on a step, let Anna know.
    - Be aware that when using Google Colab, if there is any of your data you want to keep, it is important to save it from Colab. Once you exit the Colab session, the contents of the session will be lost.
    - You can download the entire folder produced in the COLAB script by adding the following lines of code at the end of the notebook. This will allow you to download everything in the folder, including trained models, analyzed videos, etc.
    - Zip the folder if using the DLC authored book to save to your local environment:
        - !zip -r /content/file.zip /content/cloned-DLC-repo/examples/openfield-Pranav-2018-10-30/
    - Download zipped folder:
        - from google.colab import files
        - files.download("/content/file.zip")
        - Alternatively go to the 'content' folder and press download

Activity 2: Using the DLC GUI for Data Labelling

Objective: Practice labelling data manually in DLC's GUI.

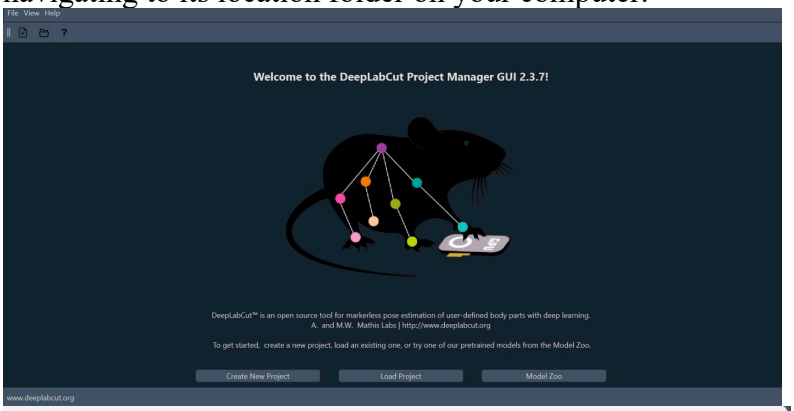
This component is aimed as support if you are going to use the GUI yourself. This acts as a companion piece to the GUI video guide https://www.youtube.com/watch?v=ofFx0vTMSxE

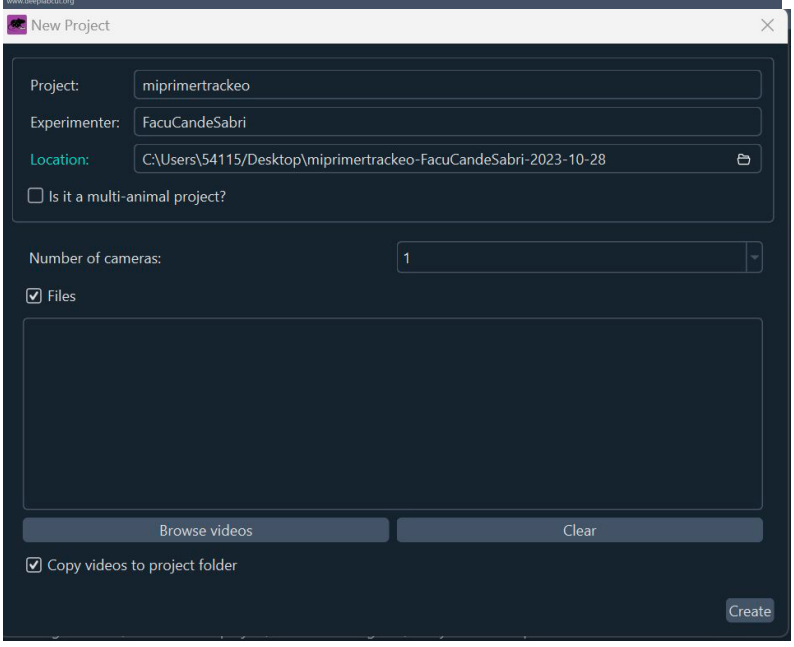
Instructions:

Step 0: Access the DLC GUI on your computer, if installed. (steps depend on how you installed DLC)

- Find Anaconda prompt (windows) / terminal (MacOS)
- Write **activate DEEPLABCUT (or chosen name of the conda environment)** and press enter
- Write **python -m deeplabcut** and press enter
- The GUI will open shortly

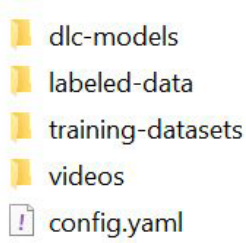
Step 1: Setting Up a New Project

1. Open DLC and start a new project
    - click 'Create New Project'. You'll be prompted to name your project and select a video file(s) to include in your dataset. Choose your video file(s) by navigating to its location folder on your computer.
    - 

    - 

    - In the folder which I set my project to be created in, a new project folder has been created, named:
      ProjectName-Experimenter-Date

        - It contains the following subfolders:

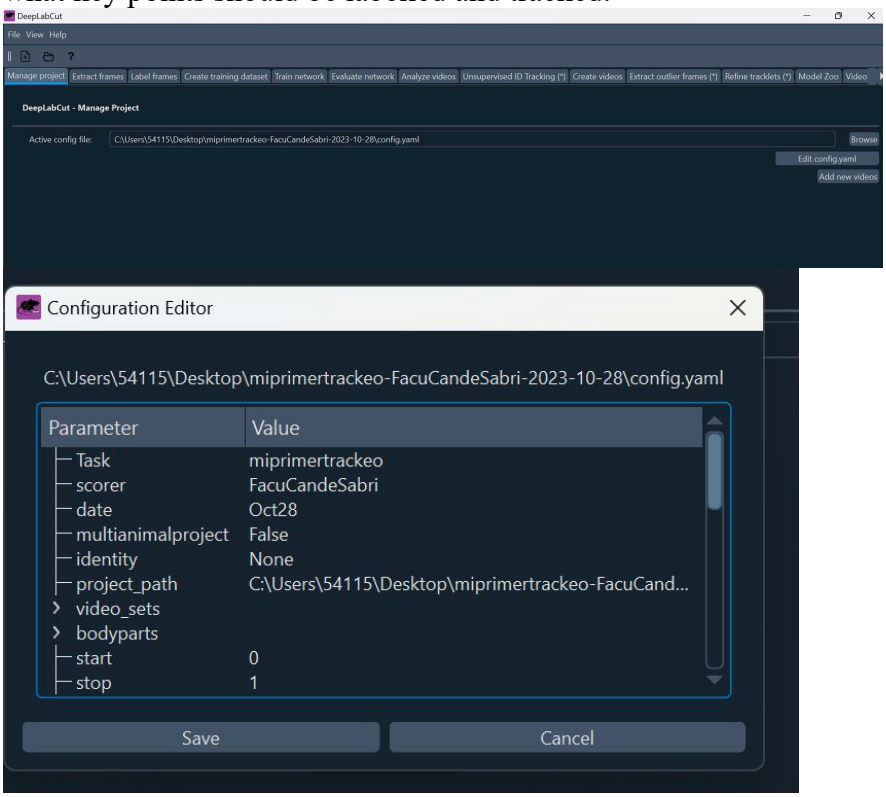


- **dlc-models:** This directory contains the subdirectories *test* and *train*, each of which holds the meta information with regard to the parameters of the feature detectors in configuration files. The configuration files are YAML files, a common human-readable data serialization language. These files can be opened and edited with standard text editors. The subdirectory *train* will store checkpoints (called snapshots in TensorFlow) during training of the model. These snapshots allow the user to reload the trained model without re-training it, or to pick-up training from a particular saved checkpoint, in case the training was interrupted.

- **labeled-data:** This directory will store the frames used to create the training dataset. Frames from different videos are stored in separate subdirectories. Each frame has a filename related to the temporal index within the corresponding video, which allows the user to trace every frame back to its origin.

- **training-datasets:** This directory will contain the training dataset used to train the network and metadata, which contains information about how the training dataset was created.

- **videos:** Directory of video links or videos. When **copy_videos** is set to `False`, this directory contains symbolic links to the videos. If it is set to `True` then the videos will be copied to this directory. The default

2. Configure Tracking Points
    - When in the ‘manage project’ section you can edit the config file, which essentially provides all the specifications for your DLC project, to point out what key points should be labelled and tracked.
    - 

    - Provided is an overview of the contents of the config.yaml 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 steps.

Parameters set during the project creation

- `task`: Name of the project (e.g., mouse-reaching). (Set in Step 1; do not edit.)
- `scorer`: Name of the experimenter (set in Step 1; do not edit).
- `date`: Date of creation of the project. (Set in Step 1; do not edit).
- `project_path`: Full path of the project, which is set in Step 1; 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. (Step 1; use the function `add_new_videos` to add more videos to the project; if necessary, the paths can be edited manually, and the `crop` can 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 `bodypart1, bodypart2, bodypart3, objectA`. 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`: This 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 refinement iterations used to create the training dataset. The first iteration starts with 0 and thus the default value is 0. This will auto-increment once you merge a dataset (after the optional refinement stage).

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]` represents the full video. The default is 0.
- `stop`: Same as `start`, but specifies 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] used to split the dataset into training and testing datasets. The default is 0.95.
- `resnet`: This specifies which pre-trained model to use. The default is 50 (user can choose 50 or 101; see also Mathis et al.[12]).

Used during video analysis (Step 13)

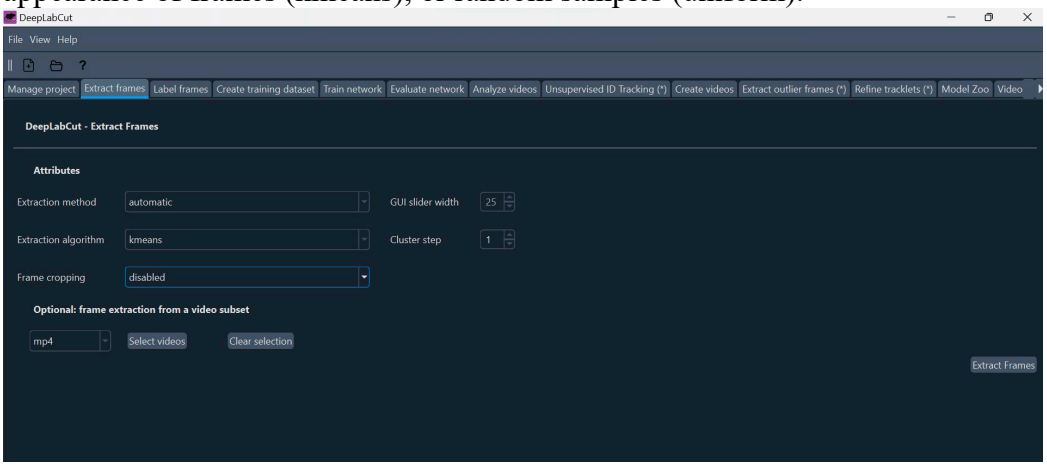
- `batch_size`: This specifies how many frames to process at once during inference (for tuning of this parameter, see Mathis and Warren[27]).
- `snapshotindex`: This specifies which checkpoint to use to evaluate the network. The default is –1. Use `all` to evaluate all the checkpoints. Snapshots refer to the stored TensorFlow configuration, which holds the weights of the feature detectors.
- `pcutoff`: This specifies the threshold of the likelihood and helps to distinguish likely body parts from uncertain ones. The default is 0.1.
- `cropping`: This specifies whether the analysis video needs to be cropped (in Step 13). The default is `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 (because of the location refinement shifts). This binary parameter ensures 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 `True`.
- `corner2move2`: This is the target location, if `move2corner` is `True`. The default is set to `(50,50)`.

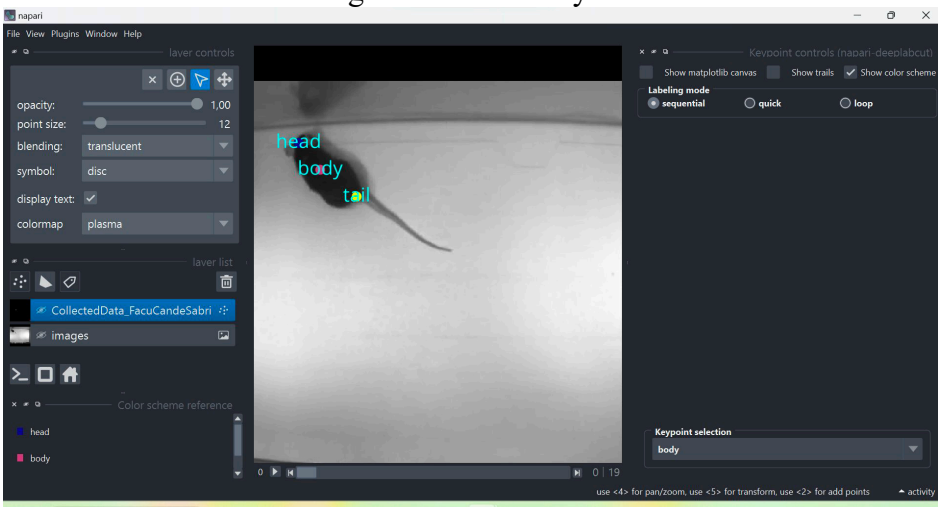
Step 2: Labeling Key Points in Frames

1. Extract Frames for Labelling
    - To train a model, we need labelled data for our model to learn from. DLC extracts frames from your video for labeling. Go the ‘extract frames’ section and select a number of frames to extract. I’d recommend starting with around 20-30 frames for a quick demo, but 100-200 if you want to create a useful model. You can do manual or automatic selection, and extract based on appearance of frames (kmeans), or random samples (uniform).



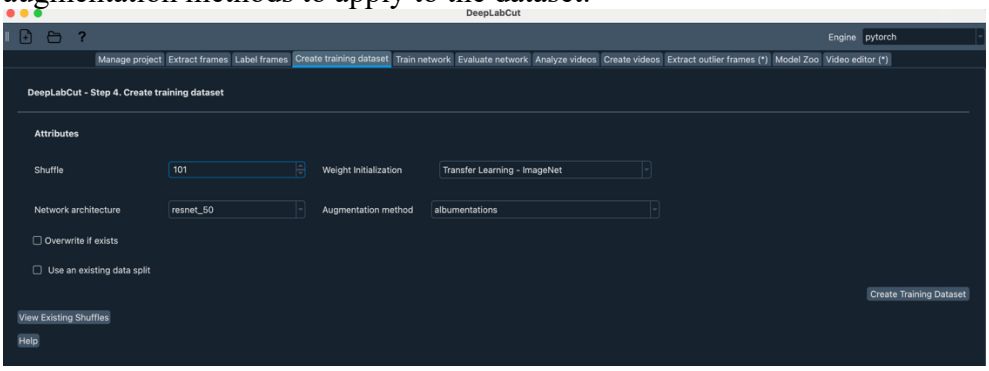
2. Label the Frames
    - Once the frames are extracted, we’ll begin labelling. Go to ‘Label Frames’ and start by placing markers on each key point for the subject in the frame. For example, if tracking a mouse, you could place a marker on its nose, front paws, and tail base. Labelling is crucial because it’s what the model will be learning from. Take your time to ensure the markers accurately match the key points you selected. In cases where the body parts are not visible (e.g.

occlusion, out-of-frame), it is recommended to not label them. In cases where the image might be blurry, but the body part is visible, it is generally recommended to label to the best of our ability, so the model will have data to learn from also in cases of greater uncertainty.

- 

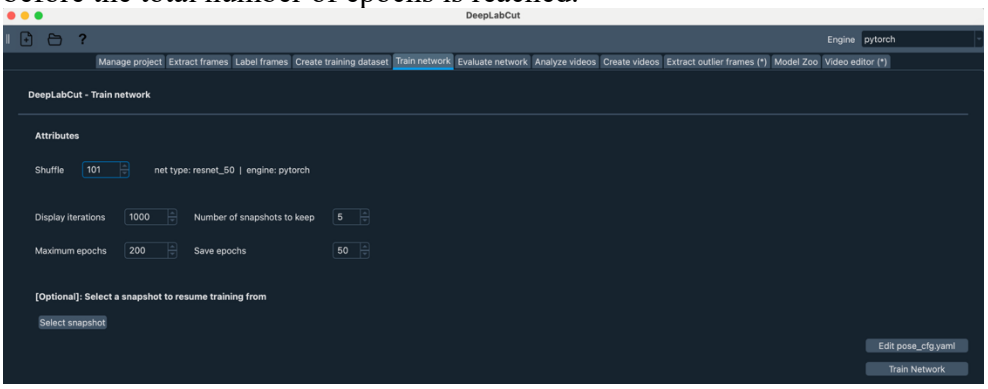
Step 3: Training the Model

1. Create Training Set.
   - Next, we'll set up how to train the model. Under 'create training dataset' we start by making our training dataset, which includes choosing the network architecture (e.g. resnet50, which is a type of convolutional neural network), and choosing which pretrained network weights to start from, and any augmentation methods to apply to the dataset.



2. Configure Training Parameters
   - We adjust training settings under 'train network'. DLC offers preconfigured settings, but you can adjust the training parameters, such as the total number of epochs (which is the number of times the model sees the full training data), how often to save, how many versions of the model to save, and how often we see the training progression of the model. You can always stop the model

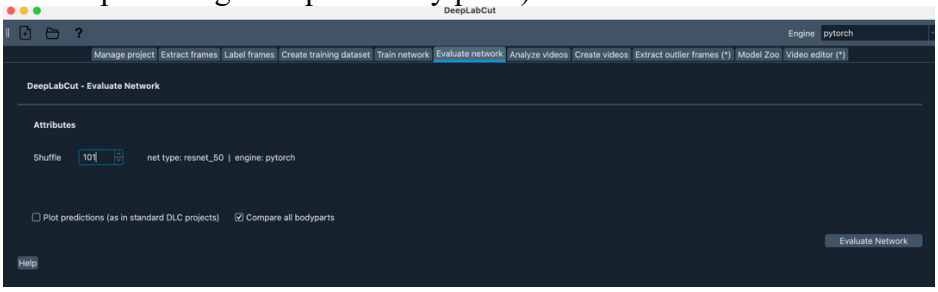
before the total number of epochs is reached.



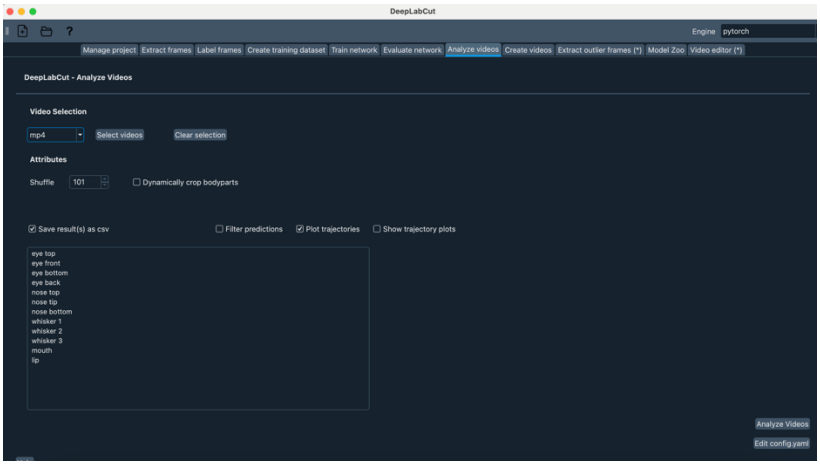
3. Start Training
    - "Click ‘Train Network’ to begin the process. The software will learn the labelled positions in the extracted frames. This may take a while. If running training on a graphics card with higher performance, this process can be significantly sped up.
4. Tip: If you do not have a GPU to train on, it can be a good idea to upload your project to Google Drive and run it through Google Colab, which provides use of a T4 GPU. From there you can download your trained project and proceed. Generally the project folder can be copied to Drive and as long as the path to the config file is set correctly in the script provided, it should run, but it may require setting the project_path parameter in the config file to the new folder location for it to run correctly.

Step 4: Evaluation and Prediction

1. Under ‘evaluate network’ you can test out how your network is performing. You can see how the model is doing quantitively in terms of e.g. ‘root mean squared error’ (RMSE) (how far predicted points are from actual labelled points), and qualitatively by opening the pictures and see how the labels and predictions look (e.g. the model having trouble predicting one specific key point).

    a. 

2. Under ‘analyse videos’ you can upload a video you want to analyse using your model.

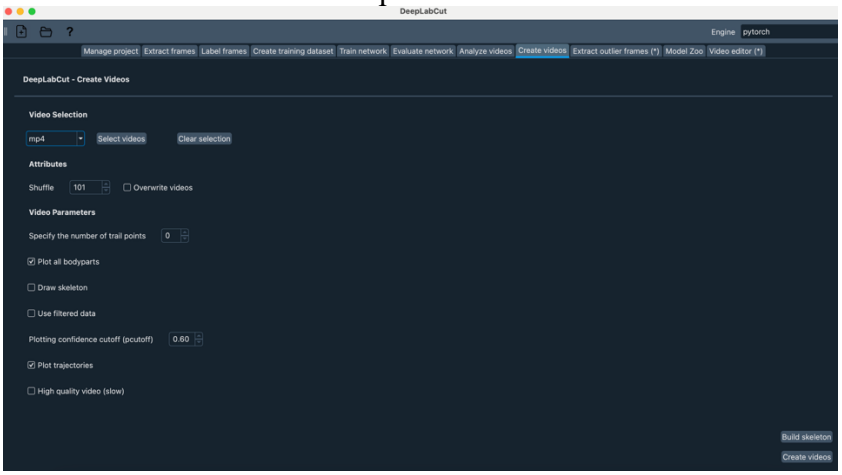
a. 

b. If you are going to use the KP coordinates in post-hoc analysis, make sure to save the csv files.

3. Create videos to save the tracked videos

a. Make sure to have the "plot all bodyparts" checked. If you do not see bodyparts, the cut off for plotting confidence may be set too high for any KPs to pass it – consider lowering it, but it could be an indication your model may need further training in order to predict KPs confidently.

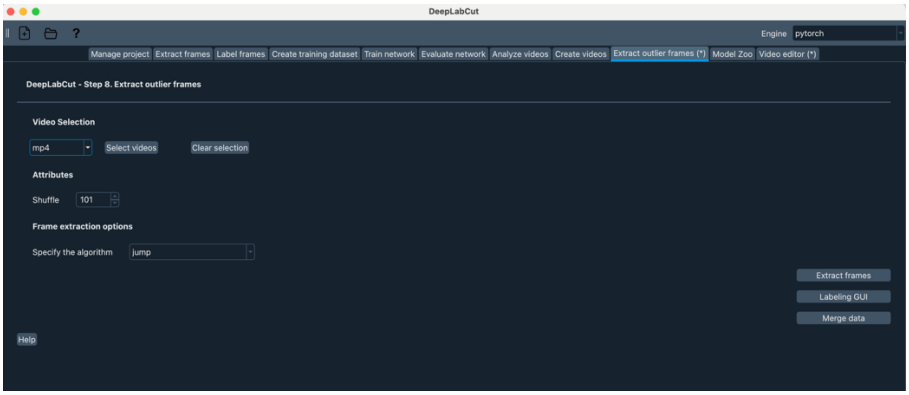
b. You can add a number of trailpoints to see the movement of KPs over time.

c. 

Step 5: Further Adjustments

1. You can do further adjustments of your model by iteratively checking how the model performs, and if it is not satisfactory, you can add more videos, label more data, merge data, re-train the model and evaluate it, until the result it satisfactory.
    a. First extract outliers frame
        i. Available methods for detecting outliers (outlieralgorithm):
            1. "uncertain" → frames where the likelihood of a body part is low.
            2. "jump" → frames where a body part "jumps" more than a threshold (epsilon) between consecutive frames.
            3. "fitting" → frames where the prediction deviates from a temporal model (ARIMA).
            4. "manual" → manual selection through visual inspection.

b. “Labelling GUI” – the GUI will pop up. Your extracted frames for refinement will be found in project folder → labeled-data → there will be a new folder with the extracted frames and their predicted labels for refinement. It will say in the terminal exactly where the files are located. Drag the folder into the GUI and adjust the labels. To the correct positioning.

c. 

d. Merge data

e. Afterwards you will need to create a new training dataset. Go to “create training dataset” and proceed as before.

   i. If you adjusted the original labels, it is best to train from fresh weights; if not, you can reuse previous weights.

2. Tip: in principle you can use videos you have analyzed in Google Colab, however sometimes DLC gives errors that the video has not been analyzed – however analyzing a video on a CPU should not take long (for a video with 2000 frames it should be <10 minutes), so in that case you can just analyze the video anew on your CPU and proceed to extract frames.

Activity 3: Running DLC Super Animals

Run the script DEMO_Superanimals.ipynb found on Github https://github.com/AnnaStuckert/DeepLabCutWorkshupAlmer-a2025 or downloaded from here https://we.tl/t-ynGT0lGW8I (link will be active for 3 days)

Designing Your Own Tracking Project

Consider a specific behaviour or set of movements that you would like to investigate. Consider what the relevant key points are for the behaviour/movement of interest.
An example could be walking. We might be interested in leg flexion during walking. This could be investigated by looking at the angle of the foot joint. For this, it would be relevant to look at markers on various positions of the leg to quantify how much or little an animal flexes the foot while walking.
Now, think about a behaviour (e.g., walking, grooming, reaching) that you would like to analyse. Think about which key points on the body (e.g., hands, feet, head) are necessary for capturing this behaviour.

Additional Ressources

Documentation and forums

- Cajal DLC course book
- DLC Documentation
- DLC forum (get assistance)
- DLC video tutorials

Papers

- Using DeepLabCut for 3D markerless pose estimation across species and behaviors
- Multi-animal pose estimation
- SuperAnimal pretrained pose estimation models
- Keypoint-MoSeq