Using DeepLabCut for 3D markerless Handpose estimation

Overview of using DeepLabCut

DeepLabCut is organized according to the following workflow. The user starts by creating a new project based on a project name and username as well as some (initial) videos, which are required to create the training dataset. Additional videos can also be added after the creation of the project, which will be explained in greater detail below. Next, DeepLabCut extracts frames that reflect the diversity of the behavior with respect to, e.g., postures and animal identities. Then the user can label the points of interest in the extracted frames. These annotated frames can be visually checked for accuracy and corrected, if necessary. Eventually, a training dataset is created by merging all the extracted labeled frames and splitting them into subsets of test and train frames. Then a pre-trained network (ResNet) is refined end-to-end to adapt its weights in order to predict the desired features (i.e., labels supplied by the user;). The performance of the trained network can then be evaluated on the training and test frames. The trained network can be used to analyze videos, yielding extracted pose files. If the trained network does not generalize well to unseen data in the evaluation and analysis step, then additional frames with poor results can be extracted, and the predicted labels can be manually corrected. This refinement step, if needed, creates an additional set of annotated images that can then be merged with the original training dataset. This larger training set can then be used to re-train the feature detectors for better results.

Advantages

DeepLabCut has been applied to a range of organisms with diverse visual background challenges. The main advantages are that our code

(i) Guides the experimenter with a step-by-step procedure from labeling data to automated pose extraction in a fast and efficient way,

(ii) Minimizes the cost of manual behavior analysis and, with only a small number of training images, achieves humanlevel accuracy, (iii) eliminates the need to put visible markers on the locations of interest,

(iii) It can be easily adapted to analyze behaviors across species, and

(iv) Open source and free. Owing to the use of deep features, DeepLabCut can learn to robustly extract body parts, even with a cluttered and varying background, inhomogeneous illumination, or camera distortions12. and

(v) DeepLabCut is robust to video compression, potentially saving users >500-fold on data storage space27, making multi-camera use potentially more feasible.

Stage1: Creating new project & Opening DeepLabCut

Step 1: Open the program Terminal. Start an IPython session, and import the package.

Ipython

import deeplabcut

Step 2:

>> deeplabcut.create_new_project('Name of the project','Name of the experimenter', ['Full path of video 1','Full path of video2', . . .], working_directory='Full path of the working directory',copy_videos= True/False)

>> config_path = deeplabcut.create_new_project(…

>> deeplabcut.add_new_videos('Full path of the project configuration file',['full path of video 4', 'full path of video 5'],copy_videos=True/ False)

Stage2: configuration of the project

Step3:

Open the config.yaml file, which was created with create_new_project and then add the list of bodyparts.

Stage3: Data Selection

Step 4:

>> deeplabcut.extract_frames(config_path,'automatic/manual','uniform/ kmeans',crop=True/False, userfeedback=True/False)

Use the function extract_frames to extract frames from all the videos in the project configuration file in order to create a training dataset. The extracted frames from all the videos are stored in a separate subdirectory named after the video file's name under the 'labeled-data' directory. It is advisable to keep the frame size small, as large frames increase the training and inference times. It is also advisable to extract frames from a period of the video that contains interesting behaviors. This can be achieved by using the start and stop parameters in the config. yaml file. Also, the user can change the number of frames to extract from each video by setting the numframes2pick variable in the config.yaml file.

Stage 4: Labeling of the frames

Step 5:

```
>> deeplabcut.label_frames(config_path)
```

Use the toolbox function label_frames to enable easy labeling of all the extracted frames using an interactive GUI. The body parts to label (points of interest) should already have been named in the project's configuration file (config.yaml).

Step 6:

use the 'Load Frames' button to select the directory that stores the extracted frames from one of the videos. A right click places the first body part, and, subsequently, you can either select one of the radio buttons (top right) to select a body part to label, or use the built-in mechanism that automatically advances to the next body part. If a body part is not visible, simply do not label the part and select the next body part you want to label. Each label will be plotted as a dot in a unique color

Stage 5: creation of a training dataset

Step 7:

```
>> deeplabcut.create_training_dataset(config_path, num_shuffles=1)
```

The set of arguments in the function will shuffle the combined labeled dataset and split it to create a train and a test set. The subdirectory with the suffix 'iteration-#' under the directory 'training-datasets' stores the dataset and meta information, where the '#' is the value of the iteration variable stored in the project's configuration file (this number keeps track of how often the dataset is refined). If you wish to benchmark the performance of DeepLabCut, create multiple splits by specifying an integer value in the num_shuffles parameter.

Stage 6: training the network

Step 8:

```
>> deeplabcut.train_network(config_path)
```

The set of arguments in the function starts training the network for the dataset created for one specific shuffle. Example parameters that one can call are given below: train_network(config_path,shuffle=1,trainingsetindex=0,gputouse= None,max_snapshots_to_keep=5, displayiters=1000,saveiters=20000, maxiters=200000)

Stage 7: evaluation of the trained network

Step 9:

```
>> deeplabcut.evaluate_network(config_path, plotting=True)
```

It is important to evaluate the performance of the trained network. This performance is measured by computing the mean average Euclidean error (MAE; which is proportional to the average root mean square error) between the manual labels and the ones predicted by DeepLabCut. Setting plotting to True plots all the testing and training frames with the manual and predicted labels. The evaluation results for each shuffle of the training dataset are stored in a unique subdirectory in a newly created 'evaluation-results' directory in the project directory. You can visually inspect whether the distance between the labeled and the predicted body parts is acceptable.

Stage 8: evaluation of the trained network

The trained network can be used to analyze new videos. The user needs to first choose a checkpoint with the best evaluation results for analyzing the videos. In this case, the user can specify the corresponding index of the checkpoint in the variable snapshotindex in the config.yaml file. By default, the most recent checkpoint (i.e., last) is used for analyzing the video.

Step 10:

>> deeplabcut.analyze_videos(config_path,['Full path of video or video folder'], shuffle=1, save_as_csv=True, videotype='.avi')

Step 11:

>> deeplabcut.filterpredictions(config_path,['video_path'], videotype='.avi') >> deeplabcut.plot_trajectories(config_path,['Full path of video'])

Step 12:

>> deeplabcut.create_labeled_video(config_path,['Full path of video 1', 'Full path of video 2'])

This function has various parameters; in particular, the user can set the colormap, the dotsize, and the alphavalue of the labels in the config.yaml file, and can pass a variable called displayedbodyparts to select only a subset of parts to be plotted. The user can also save individual frames in a temp-directory by passing save_frames=True (this also creates a higherquality video). All parameters are listed in the related help function (deeplabcut. create_labeled_video?).

Stage 9: working with the output files of DeepLabCut

We have files that contain the predicted x and y pixel coordinates of each body part and the network confidence likelihood (per frame). Beyond extracting poses from videos, there are options to generate several plots and to generate a labeled video.

Command summary

Operation	Command
Open IPython and import DeepLabCut (Step 1)	ipython import deeplabcut
Create a new project (Step 2)	deeplabcut.create_new_project('project_name','experimenter', ['path of video 1','path of video2',...])
Set a config_path variable for ease of use (Step 3)	config_path = '/yourdirectory/project_name/config.yaml'
Extract frames (Step 4)	deeplabcut.extract_frames(config_path)
Label frames (Steps 5 and 6)	deeplabcut.label_frames(config_path)
Create training dataset (Step 7)	deeplabcut.create_training_dataset(config_path)
Train the network (Step 8)	deeplabcut.train_network(config_path)
Video analysis and plotting results (Step 9 and 10)	deeplabcut.evaluate_network(config_path) deeplabcut.analyze_videos(config_path, ['path of video 1 or folder','path of video2',...])
Video analysis and plotting results (Step 11)	deeplabcut.plot_trajectories(config_path, ['path of video 1', 'path of video2',...])
Video analysis and plotting results (Step 12)	deeplabcut.create_labeled_video(config_path, ['path of video 1', 'path of video2',...])