COMP3850 Group 23 MVP Document

Revision History

Revision Number	Date	Person(s)	Changes
1.0	29/04/2024	Group 23	Initial version
1.1	16/05/2024	Michael Yee	Removed video pre-processing script section and added Video pre-processing notebook section to reflect change in solution architecture Added description and screenshots of instance tracking in SLEAP Updated screenshots of Jupyter notebooks to match current state

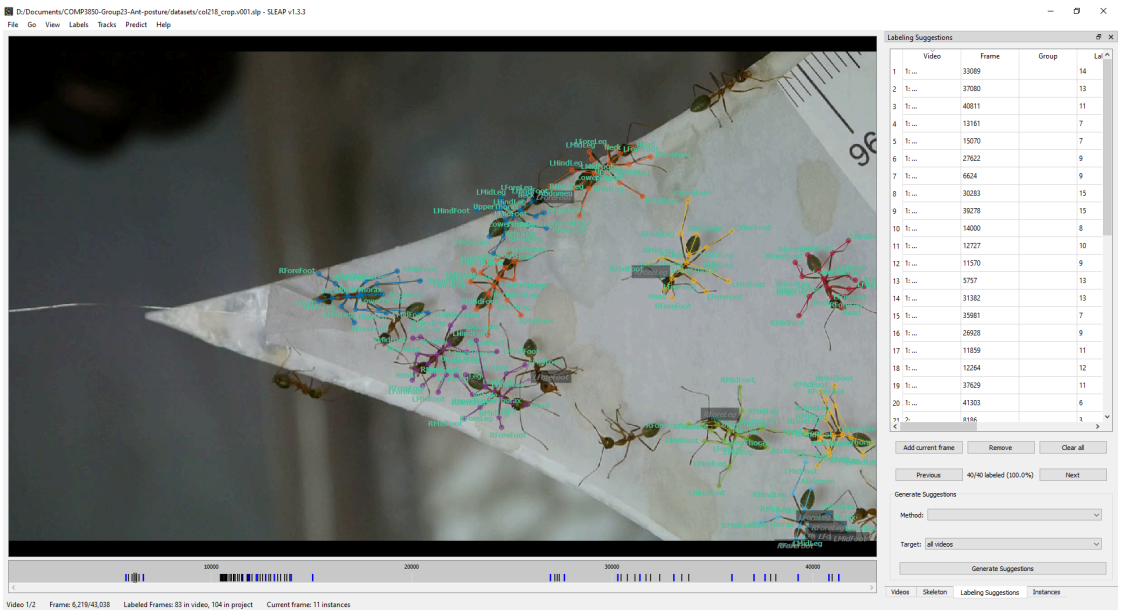
Prototype

The prototype for this project consists of the models developed to track the estimated postures in videos of weaver ants forming pulling chains when pulling on the tips of leaves, and the Jupyter notebooks that perform video pre-processing, labelled dataset merging, labelled dataset analysis, model comparisons, pose inference, and posture data export.

Model - training and inference

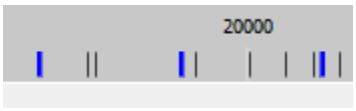
As per the client's requirements, the main goals of this project are to develop a model using the SLEAP pose estimation framework that can track ants engaging in a pulling chain on leaf tips, and to design a user-friendly process for building these models in SLEAP to track ants in other settings. SLEAP is a deep learning pose estimation framework which can track animal postures in images and videos. This involves three major steps, manually labelling frames in the video by matching a skeleton onto instances of animals in the frame, training the SLEAP model on these frames, and finally having the trained model label frames of previously unseen video data.

The image below shows multiple weaver ants that have been manually labelled in the 'sleap-label' application, a GUI application that is included with SLEAP that allows the user to label training data for use in training a model. On the right side can be seen a list of all frames that have been suggested for labelling. The user can either manually add frames to label or randomly generate a list of frames.



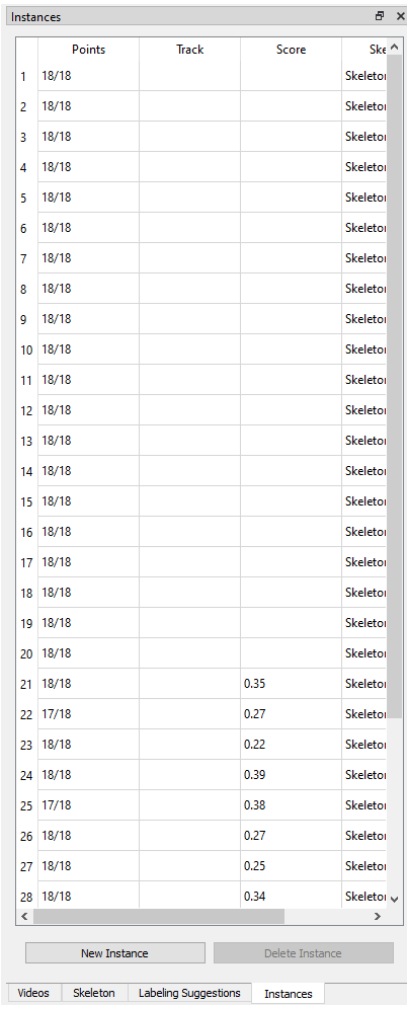
Example of a manually labelled video frame as shown in the 'sleap-label' application. Parts with a cyan-coloured name are considered visible for training phase and grey labels are considered non-visible.

To differentiate between manually labelled and model labelled frames SLEAP marks manually labelled frames with a solid blue line, and marks model labelled frames with a thin blue line.



Example of frames that have been labelled.

When a frame has been selected the user can see a list of instances in a frame. Each instance in the list has the number of points present in the frame and the name of the skeleton that has been used. Additionally, if the instance was generated by a model inference, the instance will be assigned a prediction score for the level of confidence in the prediction. This prediction score ranges from 0, representing low confidence in the prediction, to 1, representing high confidence in the prediction.



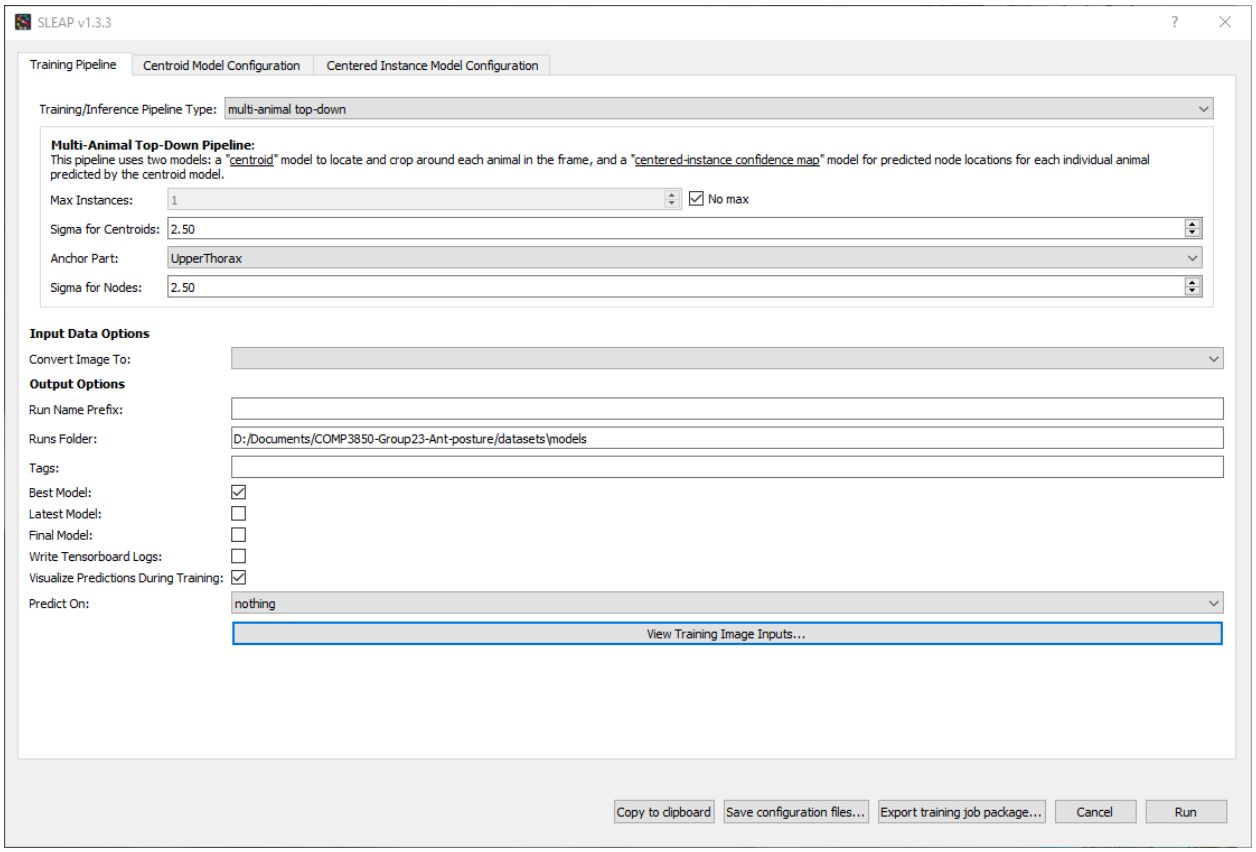
Example of a list of skeleton Instances

Training

SLEAP has two training modes available, top-down training and bottom-up training. In top-down training, two models are trained to estimate ant positions in a video frame. The first model, known as the Centroid model, locates instances of the animals in the frame and learns to identify the centroid of the animal instance. The second model, the Centred Instance model, then crops the frame to focus on individual instances using the centroid locations and finds the different points for the skeleton to be mapped to.

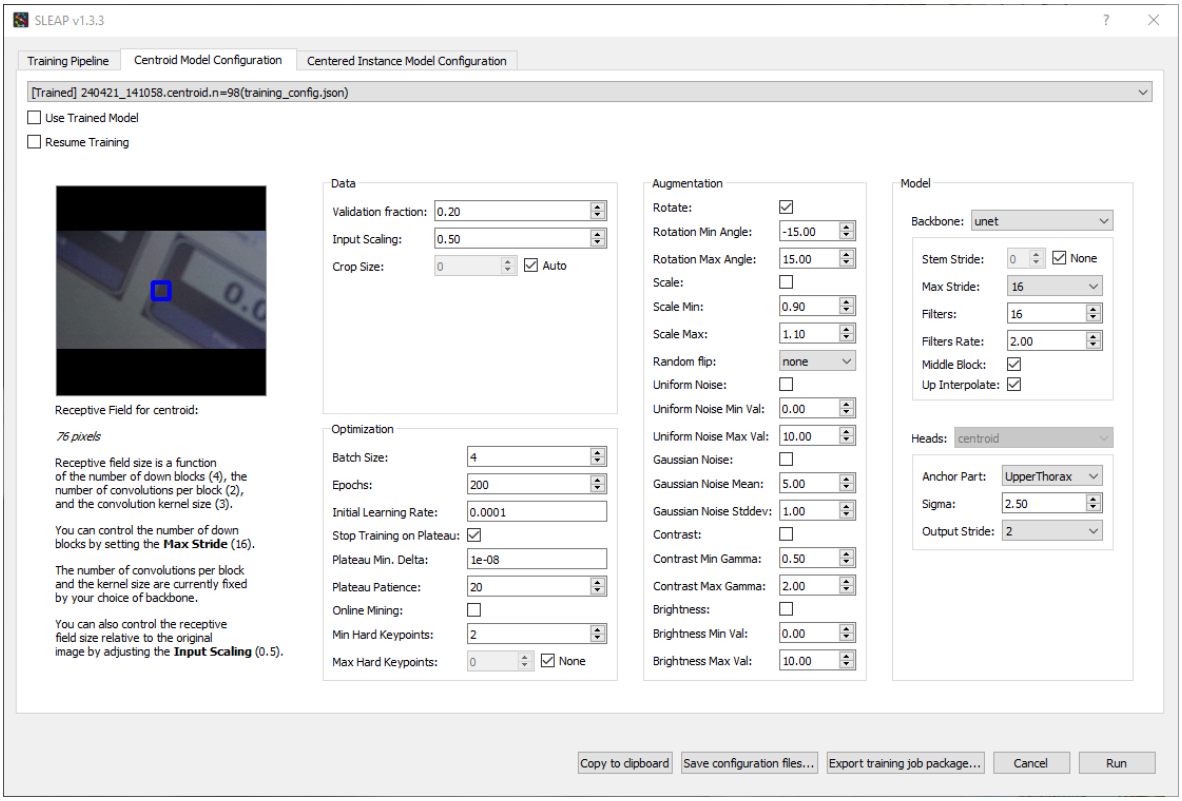
In the bottom-up training mode, a single model is trained by locating all of the body parts present in a frame before connecting the parts to each other and grouping them into a single animal instance using a learned part affinity map. However, due to the considerable amount of system resources required to perform bottom-up training using the high resolution videos being used in this project, this approach was not generally used.

Below is the configuration menu for the overall training pipeline. This menu allows a user to specify where to save the model once it has been trained as well as which model it saves, allowing the user to choose between different models for each training run.

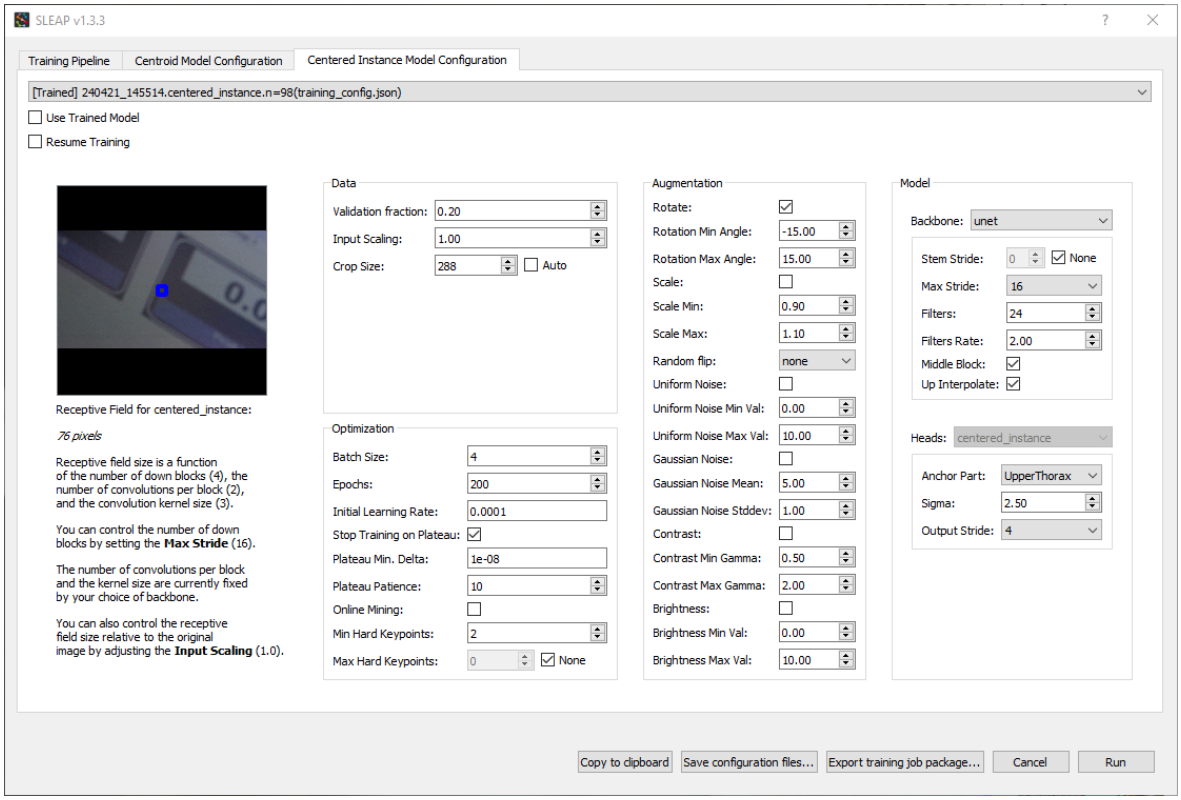


SLEAP Training Pipeline configuration menu

Depending on the type of training mode selected, the configuration for the corresponding model(s) is also available and allows for most hyperparameters in the backbone network and the model head, including the number of filters, the maximum stride which affects the depth of the network, and the sigma of the confidence map among others, to be set via the GUI. There are also a range of options for data augmentation, including rotating, cropping, and scaling the input, which effectively expands the training data set and allows the model to generalise to more diverse inputs without requiring additional labelling.



Centroid Model Configuration Screen

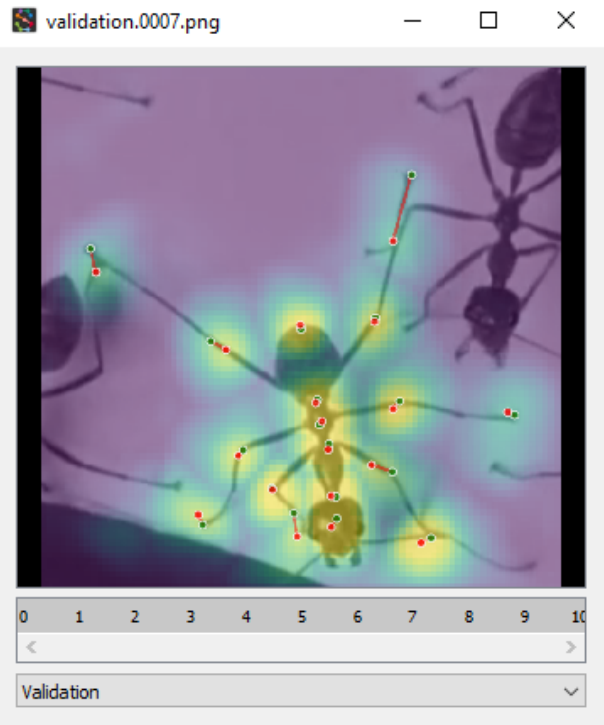


Centred Instance Model Configuration Screen

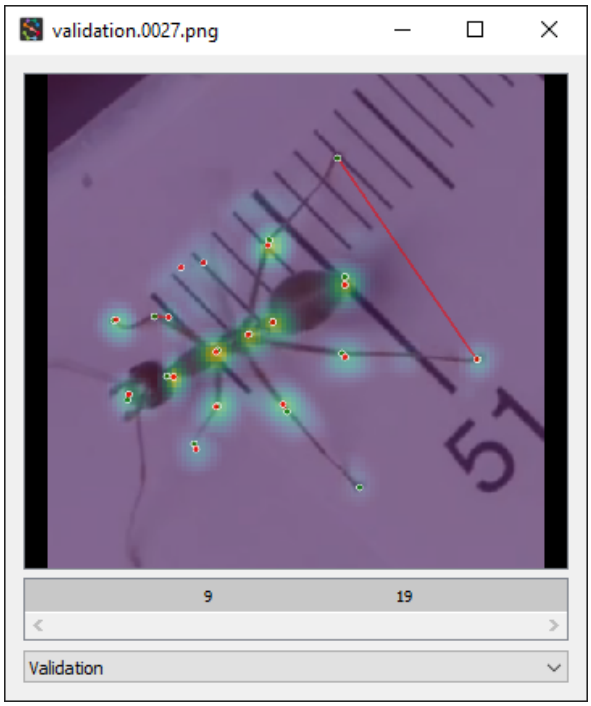
When training is initiated, SLEAP begins the process of optimising the model weights to produce the most accurate model by minimising the model loss function. This takes place over a series of epochs, and the last and best validation losses are displayed in a chart showing the reduction in training and validation loss over each epoch. This provides a useful way of identifying when overtraining is occurring, as the validation loss will eventually reach a plateau whereas the training loss will continue to decrease, indicating that is memorising the training data which could lead to poor generalisation and performance when inferring ant postures in unseen data.

Model training progress screen after 19 training epochs. The validation loss in orange has plateaued after the 6th epoch, however the training loss is continuing to decrease, indicating that overfitting is occurring for this run.

The ‘sleap-label’ application also has the option of displaying the predicted ant posture against the ground truth during training and validation, which can be useful for showing the user how closely the model is able to predict the location of the ant instances and body part positions and how these predictions improve in accuracy as the model learns. The confidence map as inferred by the model is overlaid with the image and the subsequent estimate of each body part position is displayed relative to the corresponding ground truth label.

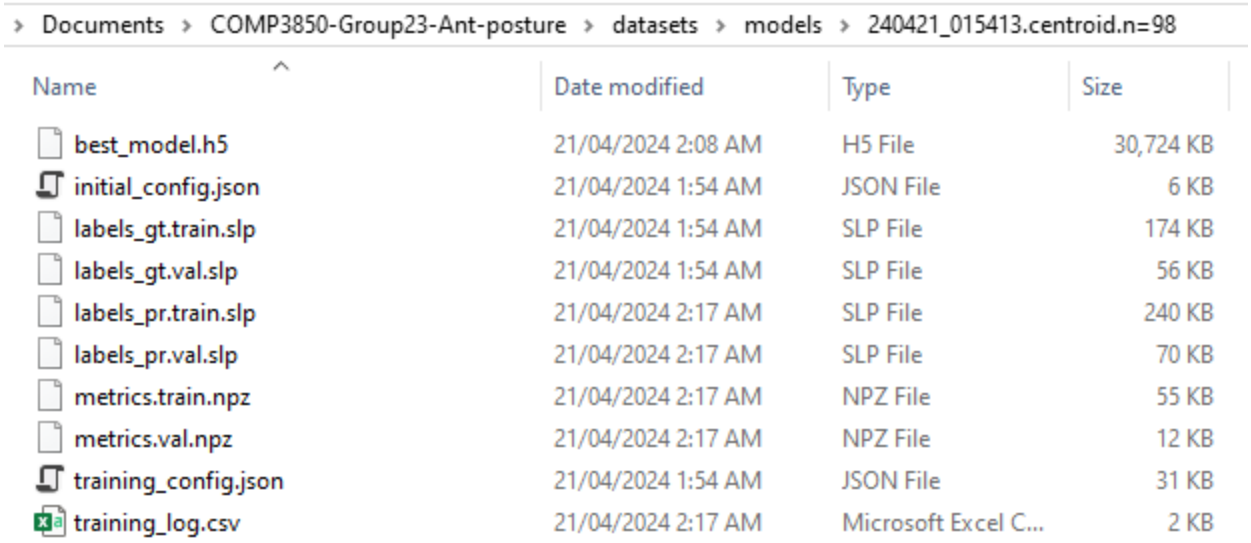


Example of a correct prediction during training evaluation in SLEAP. The background colours indicate the confidence map with colours closer to yellow indicating high confidence and purple representing lower confidence. The points in green are the ground truth and the points in red are the predicted part locations.



Example of an incorrect prediction during training evaluation in SLEAP. Note that the right rear foot has been predicted at the same position as the left rear foot as indicated by the red line.

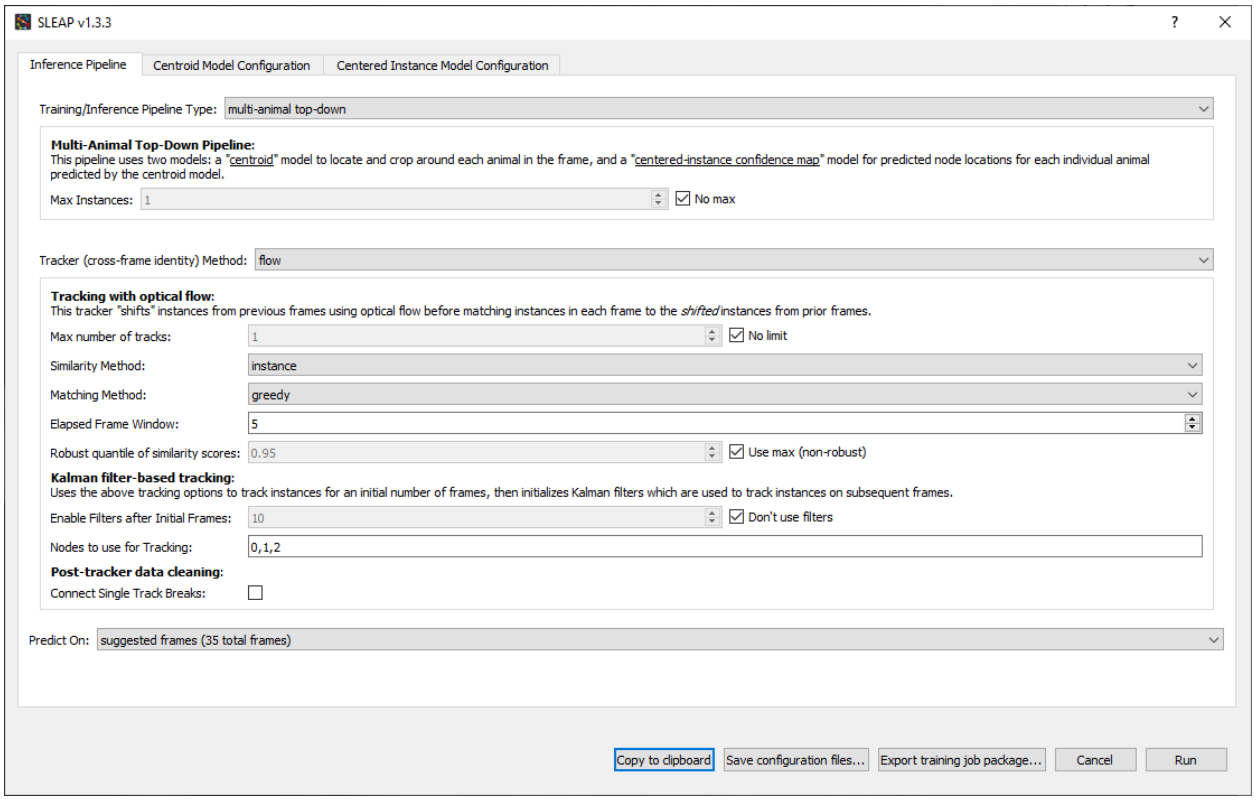
The best performing version of each model as measured by the validation loss is saved to the file system once training is complete and each model consists of several files necessary for further training the model, training other models using the same hyperparameters, or using the mode to run inference on unseen data. The ‘best_model.h5’ file and ‘training_config.json’ files are the most relevant for these purposes, as they contain the model weights and hyperparameter configuration respectively. The full details of these files including the file format are explained in the Detail Data Description section of the Analysis and Design document.



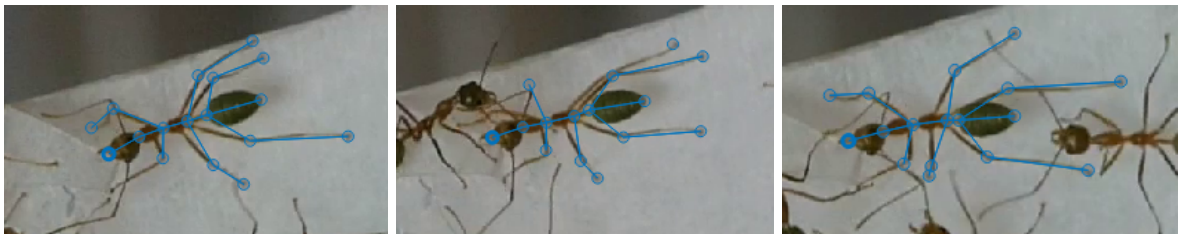
Example of the files produced by training a model.

Inference and tracking

When one or more models have been trained to a level of accuracy that the user is satisfied with, the model(s) can be used to run inference on unlabelled video frames. This is performed in the ‘sleap-label’ application, and the inference can be run on one frame, a set of frames, or even an entire video. There is also functionality for tracking ant instances between frames, which can be configured to track ant instances between frames based on their relative proximity, with predicted instances that have similar positions on different frames being considered as the same instance. These instances are assigned a “track” that can be used to analyse how individual ants move and behave over the course of a video. The inference process requires very little configuration otherwise, as most of the settings are defined by the configuration for the model(s) being used for inference.

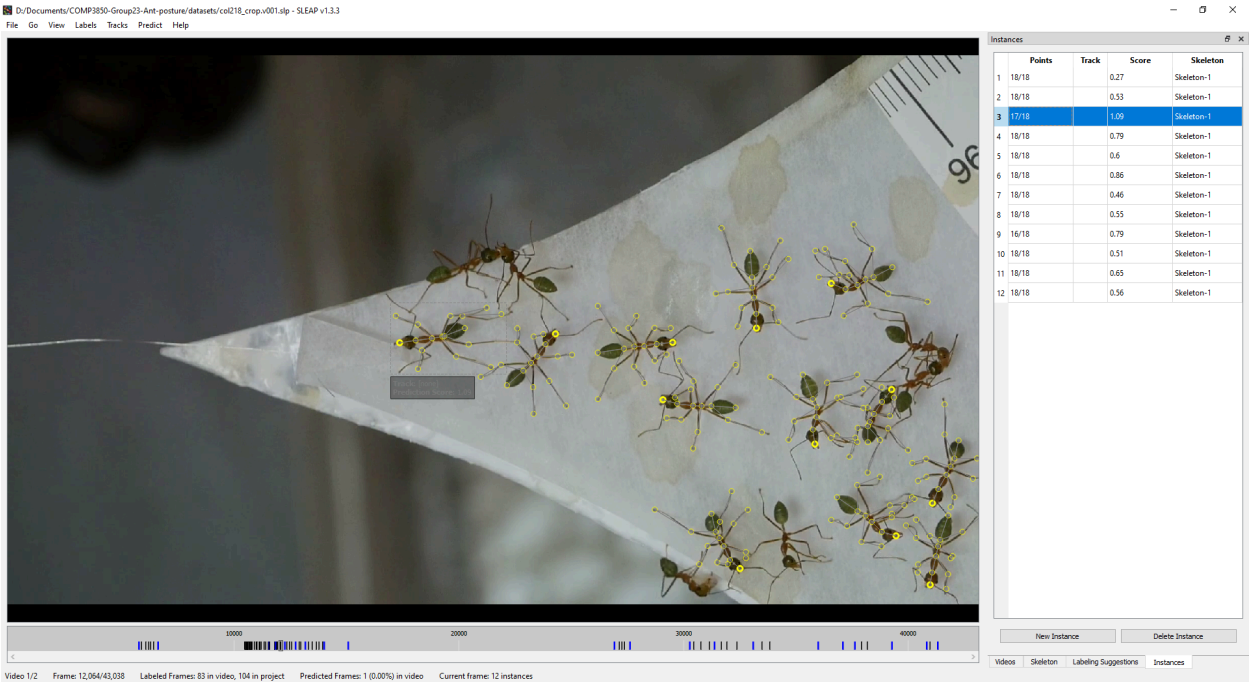


Screenshot of the SLEAP inference configuration menu with instance tracking options.



A single ant being tracked over multiple frames as it pulls on a leaf tip.

Once the inference and instance tracking (if configured) has been performed on one or more unlabelled video frames, the inferred ant positions are displayed on each frame, along with confidence scores for each predicted instance. The predicted instances can also be used as the basis for additional labelling by correcting the incorrect predictions, as theoretically the predicted instances should still be somewhat close to the actual posture which should in turn speed up the labelling process.



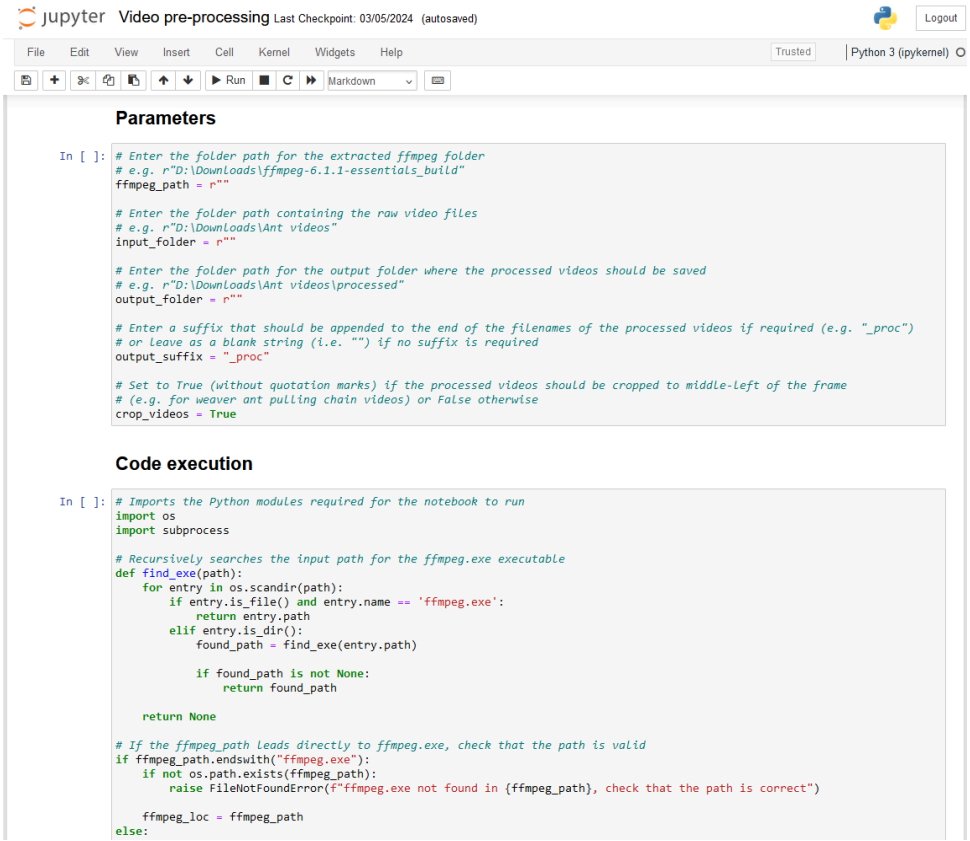
Example of a video frame with inferred ant postures using a model trained to recognise ants pulling on the leaf tip as shown in the ‘sleap-label’ application. The position of each inferred point is denoted by a yellow circle, with the head position having a more opaque circle as it is the root of the skeleton.

Jupyter Notebooks

Video pre-processing notebook

The video pre-processing notebook performs all necessary steps to convert the raw videos of weaver ants into a format that is most suitable for use as training data for a pose estimation model. For this project, this involves re-encoding the videos in a “reliably seekable” format and cropping the videos to only show the region of each frame in the general vicinity of the leaf tip that the ants pull on. For a video to be considered reliably seekable, it must produce the exact same frame at each frame index, which is not always the case for certain compressed video formats. Cropping the videos helps reduce the number of irrelevant animal instances in each frame as the primary goal of the project is to track the postures of ants pulling on the leaf tip. Thus, cropping the videos prior to labelling also minimises the number of ant instances that must be labelled.

In the notebook, the user must specify several parameters, including the location of ffmpeg on the user’s machine, where the unprocessed videos are located and where they should be saved, how the output files should be named and whether the videos should be cropped in the same way as other videos used during the project. Executing the notebook will then process all of the videos in the input folder and save them in the specified output folder.



Screenshot of the video pre-processing notebook

An example of a command generated and executed by the video pre-processing notebook is as follows,

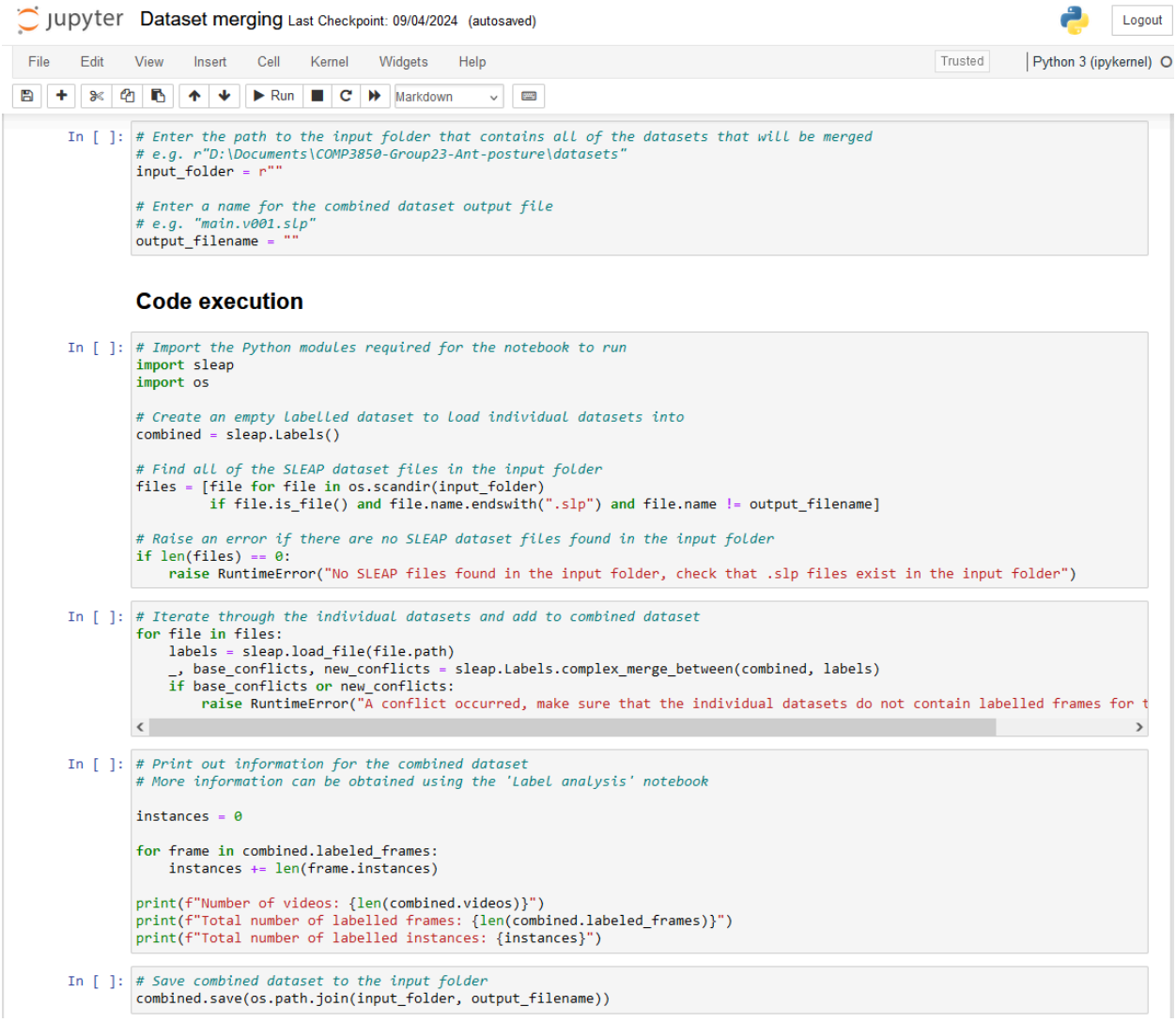
```
D:\Downloads\ffmpeg-6.1.1-essentials_build\bin\ffmpeg.exe -y -i
"D:\Downloads\Ant videos\Col202.mp4" -vf
"crop=in_w/2:in_h/2:0:in_h/4" -c:v libx264 -pix_fmt yuv420p -preset
superfast -crf 23 "D:\Downloads\Ant videos\rescaled\Col202_rec.mp4"
```

An explanation of each argument is as follows,

- -y - Overwrite files in the output directory without prompting.
- -i - Specifies the input file given immediately after the argument.
- -vf - Provide verbose output in the console window and force the output format based on the next part of the argument.
    - crop - Perform a cropping operation using a colon-separated list of values indicating the output width, output height, X-coordinate of the top-left corner of the crop location, and the Y-coordinate of the top-left corner of the crop location.
    - In this case, the size of the cropped frame will be half the width and length of the original frame, and will be focussed on the middle-left side of the frame.
- -c:v - Convert the video channel using the specified video codec, ‘libx264’ in this case.
    - This is part of converting the video into a reliably seekable format as mentioned previously.
- -pix_fmt - Use the given format for storing the pixel data in the out file, ‘yuv420p’ in this case.
    - This is another part of converting the video into a reliably seekable format as mentioned previously.
- -preset - Controls the trade-off between compression speed and compression size, values can range from ‘veryslow’ to ‘ultrafast’.
- -crf - Sets the constant rate factor that determines the bitrate of the output file, which affects the video quality and file size. Values for this argument range between 4 and 63.

Dataset merging notebook

The dataset merging Jupyter notebook is intended for use when merging multiple labelled datasets in cases where the labelling task is shared by multiple labellers, as was the case with this project. Having a single dataset to train models makes the process easier to perform as it can be managed using the ‘sleap-label’ application, whereas training using multiple datasets requires a complex configuration that must be manually written by the user and requires more technical expertise. The notebook takes the path to a folder containing all of the datasets that are to be combined into a single dataset as its input and will automatically produce a merged dataset as the output in the same folder location. Note that it is assumed that the individual datasets contained labelled datasets from different videos so that there are no conflicting labels.

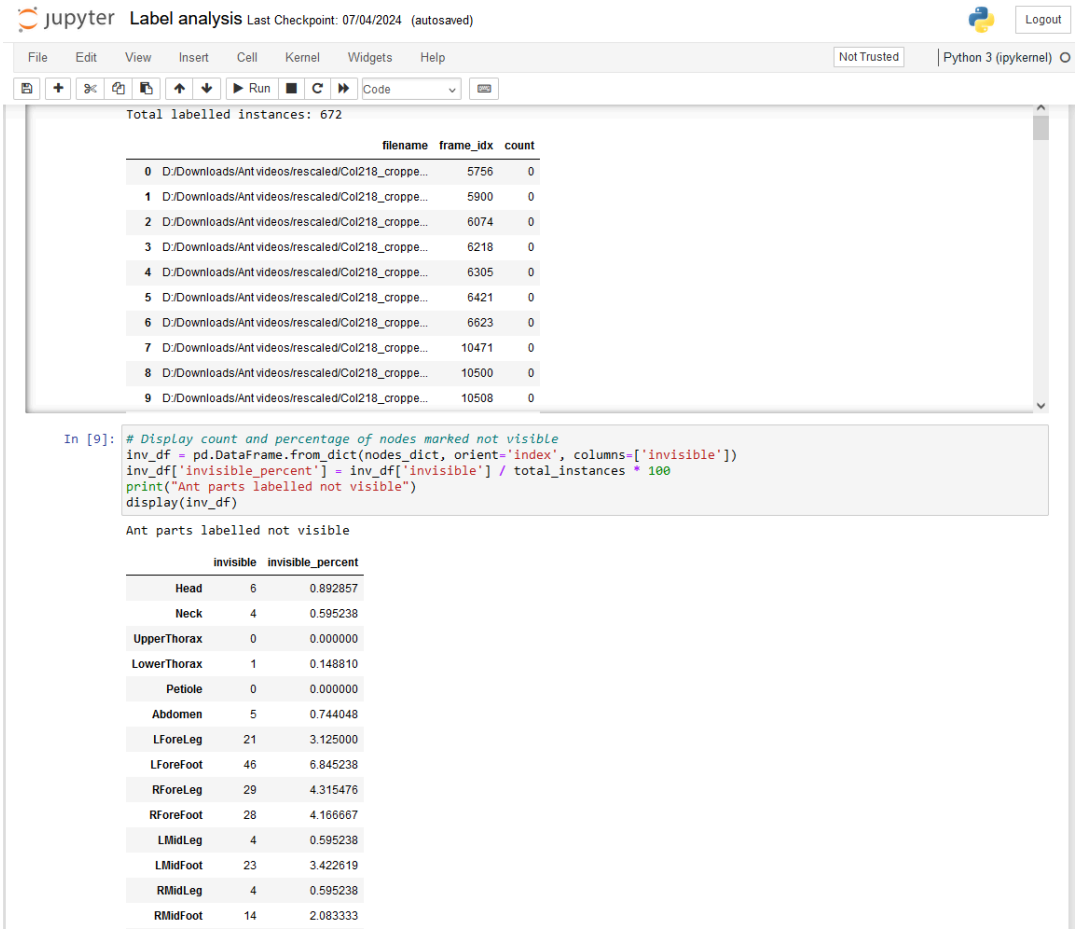


A screenshot of the dataset merging Jupyter notebook.

Label analysis notebook

The label analysis notebook is used for analysing the completeness of labelling for a given SLEAP dataset, as well as providing information that may be used during post-training analysis to potentially explain the results. The user must specify the file path to a SLEAP labelled dataset file, and the notebook will calculate the number of videos present in the dataset, the number of frames per video, and the number of labelled instances per frame. Having this information available in a summarised view that is not available through SLEAP itself is useful for determining whether the amount of labelled data in the dataset is sufficient for training accurate models.

The label analysis notebook also identifies any cases of labels that have been added to a video frame but not verified by a labeller, as would be the case if a skeleton instance was added to a frame but not moved into position. Since these unverified instances are still considered during the training process, failing to verify them by moving them into the correct location in the frame could result in poorer model performance as the model may learn incorrect or spurious features. The number and percentage of labels marked not visible by body part is also calculated and displayed, which may be useful for explaining why certain body parts are more difficult to identify during training.



A screenshot of a sample of the labelled dataset metrics in the label analysis notebook.

jupyter **Label analysis** Last Checkpoint: 07/04/2024 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

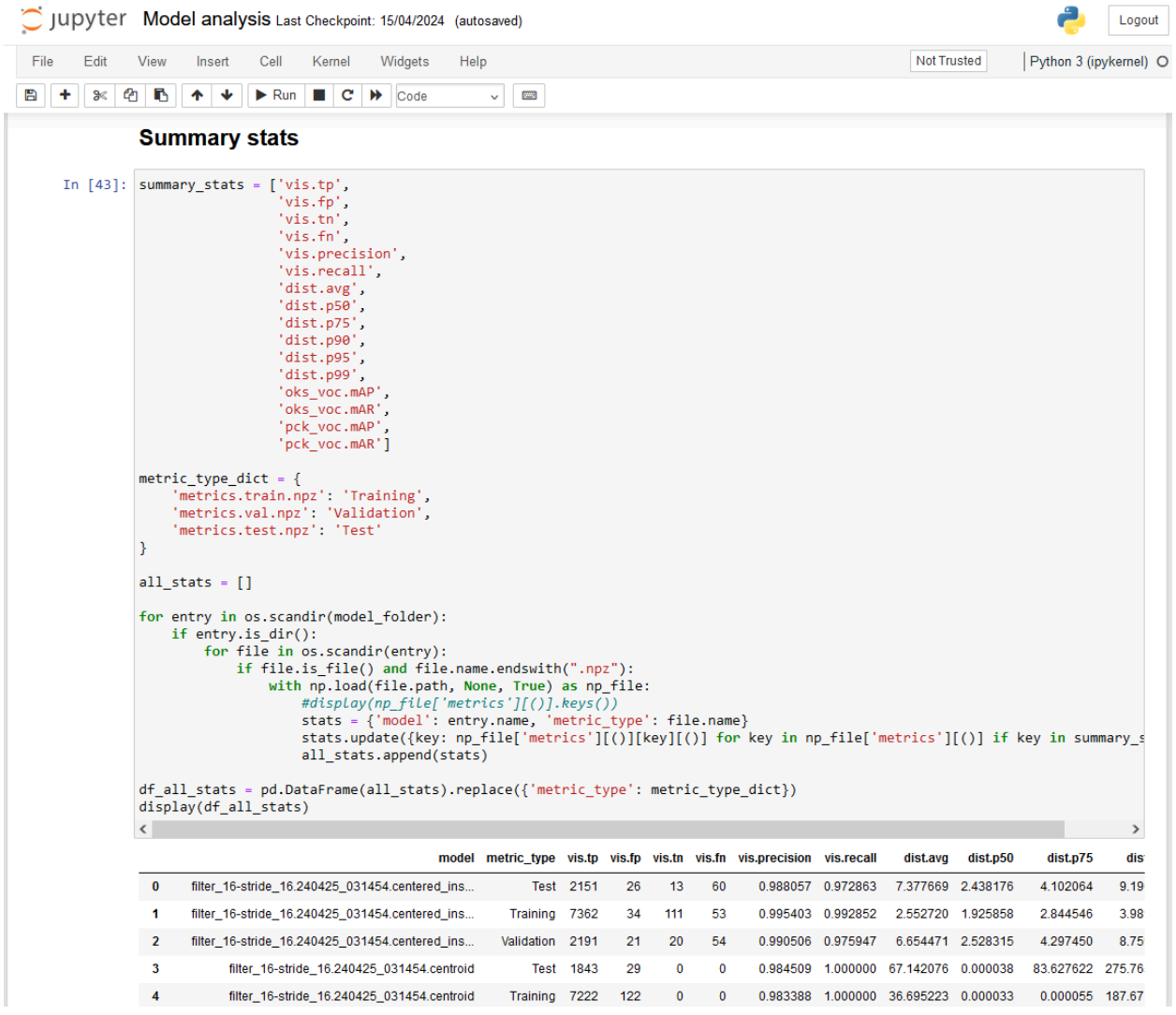
Run Code

In [33]:
```
# Display a sample labelled frome from the dataset
try:
    labels.labeled_frames[0].plot()
except FileNotFoundError as e:
    print(e)
    print("Make sure that missing videos have been replaced if the project file was created on another machine")
```

A screenshot of a sample labelled image in the label analysis notebook.

Model analysis notebook

The model analysis notebook compares different models that the user has trained and extracts the metrics that SLEAP has calculated to allow the user to determine the relative performance of each model. These metrics include the Percent of Correct Keypoints (PCK) score, the Object Keypoint Similarity (OKS) score, and the average Euclidean distances between the predicted and ground truth positions, both overall and for different percentiles of points. These measures assist the user in evaluating how closely each model is able to predict the actual position of each body part. The metrics also include the confusion matrix values for the part visibility predictions, which help evaluate how well each model is able to identify if a body part for a particular instance of an ant is visible in frame. Each of these metrics is displayed in a dataframe as well as in a graphical format to simplify comparisons between models.



	model	metric_type	vis.tp	vis.fp	vis.tn	vis.fn	vis.precision	vis.recall	dist.avg	dist.p50	dist.p75	dis
0	filter_16-stride_16.240425_031454.centered_ins...	Test	2151	26	13	60	0.988057	0.972863	7.377669	2.438176	4.102064	9.19
1	filter_16-stride_16.240425_031454.centered_ins...	Training	7362	34	111	53	0.995403	0.992852	2.552720	1.925858	2.844546	3.98
2	filter_16-stride_16.240425_031454.centered_ins...	Validation	2191	21	20	54	0.990506	0.975947	6.654471	2.528315	4.297450	8.75
3	filter_16-stride_16.240425_031454.centroid	Test	1843	29	0	0	0.984509	1.000000	67.142076	0.000038	83.627622	275.76
4	filter_16-stride_16.240425_031454.centroid	Training	7222	122	0	0	0.983388	1.000000	36.695223	0.000033	0.000055	187.67

Screenshot of the model comparison Jupyter notebook showing a sample of the metrics summary that can be used to compare model performance.

jupyter **Model analysis** Last Checkpoint: 15/04/2024 (autosaved)

Logout

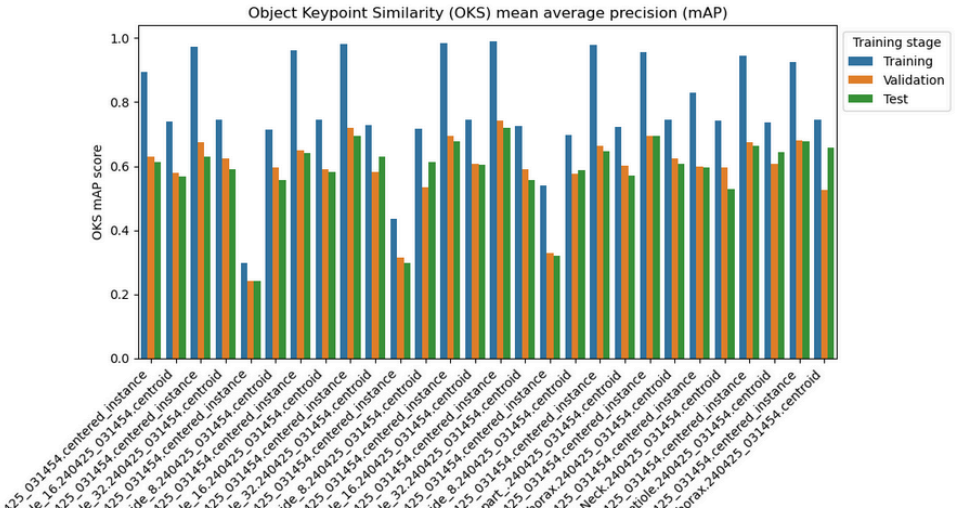
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

Comparison of all models

N.B. Different model types are not directly comparable to each other (e.g. accuracy measures for centroid models cannot be directly compared to centred instance models).

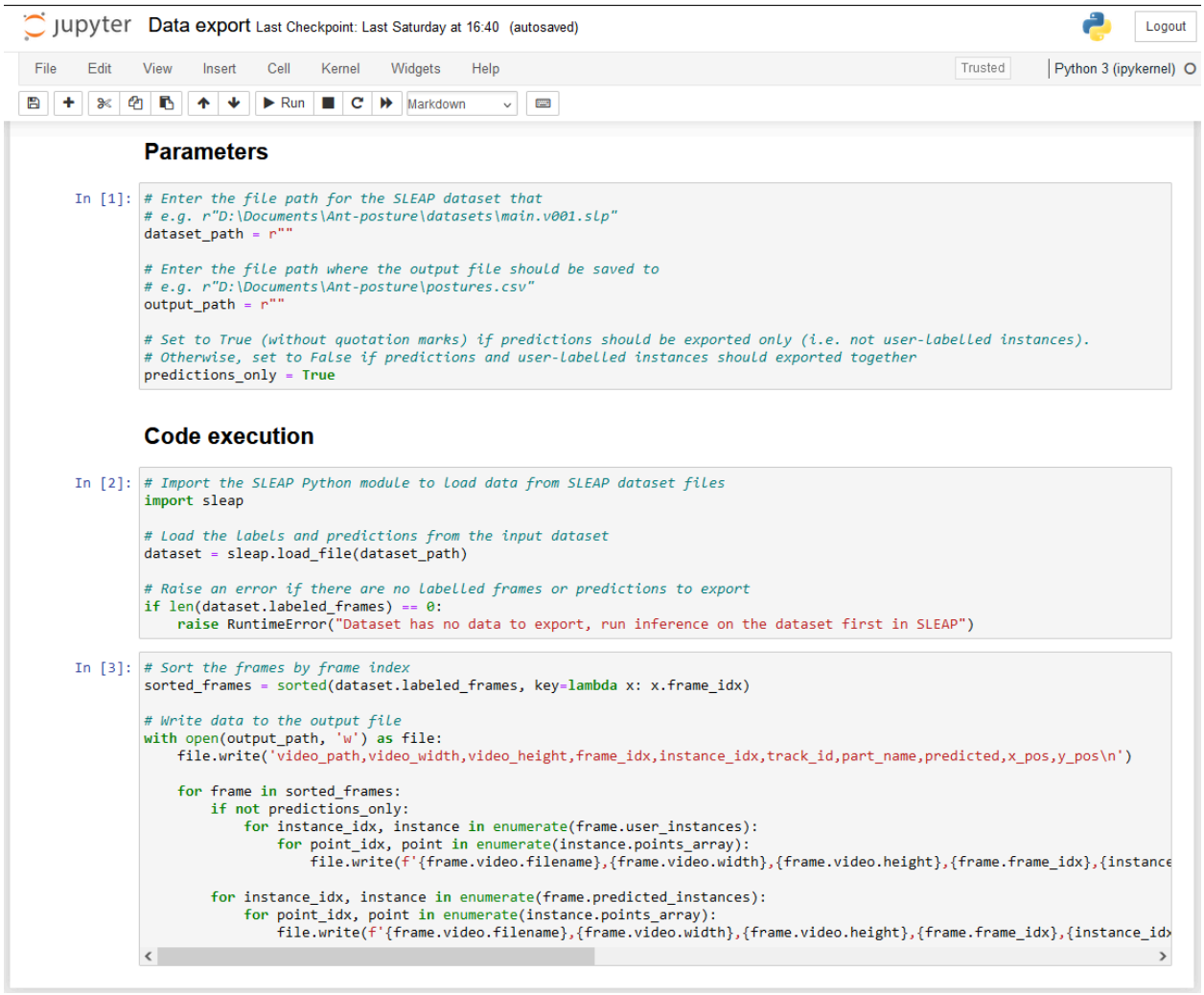
In [54]:
```
fig, ax = plt.subplots()
fig.set_figwidth(10)
sns.barplot(df_all_stats, x='model', y='oks_voc.mAP', hue='metric_type', ax=ax, hue_order=['Training', 'Validation', 'Test'])

# Set the legend name and reposition
ax.legend(title='Training stage')
sns.move_legend(ax, "upper left", bbox_to_anchor=(1, 1))

# Format the axes and set the axis names
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
ax.set_xlabel("Model name")
ax.set_ylabel("OKS mAP score")

# Add the plot title and display the plot
ax.set_title("Object Keypoint Similarity (OKS) mean average precision (mAP)")
plt.show()
```



Screenshot of the model comparison Jupyter notebook showing the differences in the OKS mAP scores between multiple models trained using different hyperparameter values.

Data export notebook

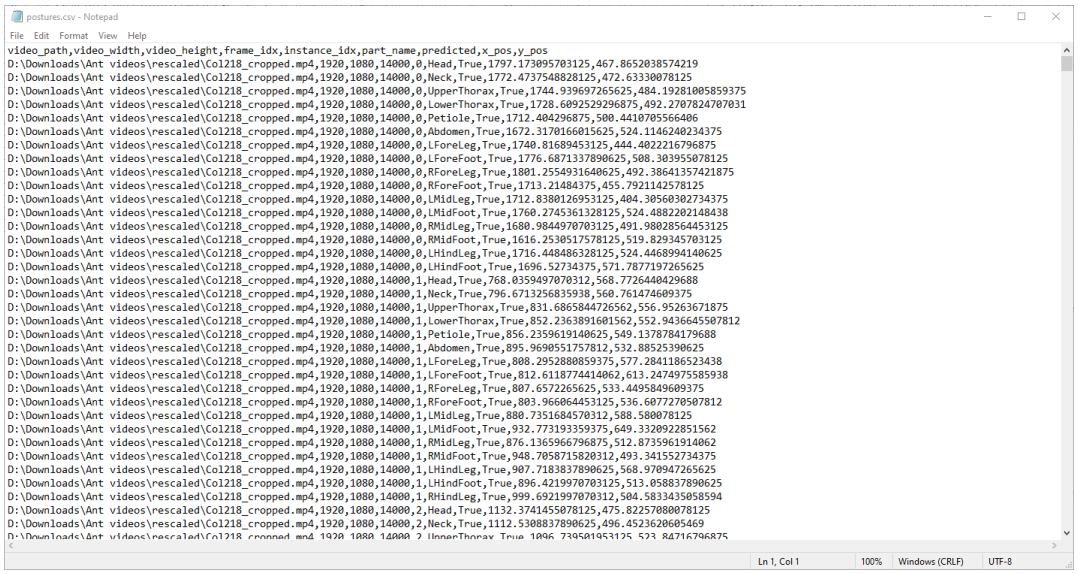
The data export Jupyter notebook exports the posture data from the predictions into an output file, which is functionality that is not readily available in SLEAP itself. The user must specify the path to the SLEAP dataset file containing the predictions and user-labelled instances (if required) that they want to use and the output location for the posture data file that will be exported by the notebook. They must also specify whether the output file should contain predictions only or if user-labelled instances should be included as well. Running the notebook will then generate the output file in the specified output path.



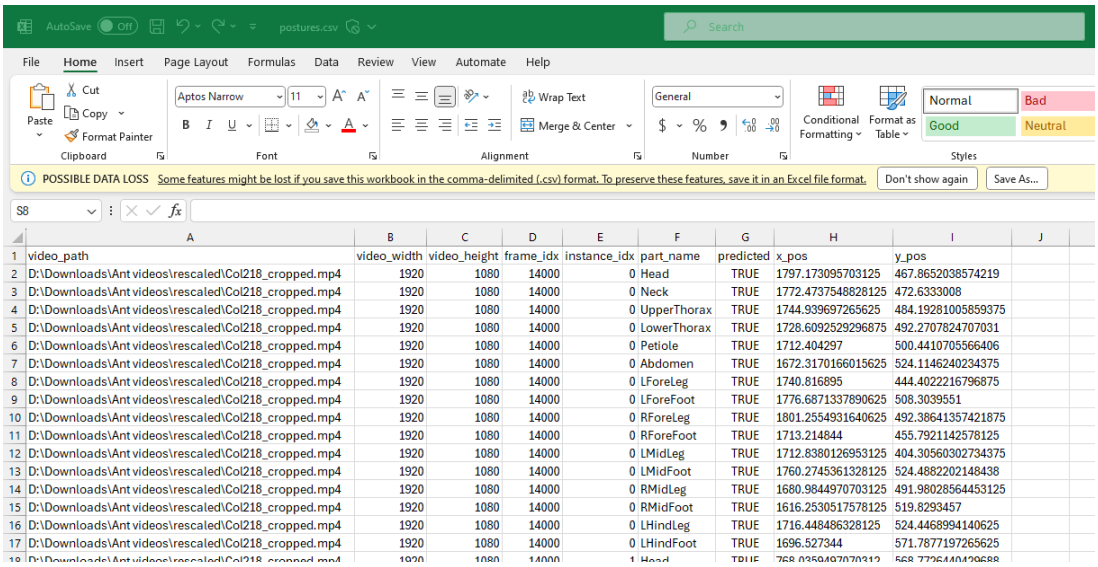
Screenshot of the data export Jupyter notebook

Posture data file

The posture data exported by the inference/data export notebook is saved to the file system in a comma-separated values (CSV) file format. This file format is widely supported by many data analysis applications as it is both human and machine-readable. The file contains a record on each row for each predicted or ground truth body part location that includes the video, video frame, and animal instance in the frame that the part location relates to, relevant details about the video dimensions, as well as the part name and location. Further details including the exact format of the file are given in the ‘Detailed Data Description’ section of the Analysis and Design document.



Sample of the posture data output CSV file in raw text format.



Sample of the posture data output CSV file as it appears when imported into Microsoft Excel.

Sponsor meeting, feedback and response to feedback

Date: 12 - 12:30pm 18th April 2024
Location: Online (Zoom)
Attendees: Michael Yee, Julian Teow, Rijul Khunger, Rhys Patton, Chris Reid
Feedback:

- The initial models showed good results for ant postures in general, but the client would ideally like to see a model that performs well specifically for ants pulling on the leaf tip.
- The Jupyter notebooks would help with the model training and inference pipeline, however the client's team is not as technically proficient as the project team and would require detailed documentation within the notebooks in addition to the user documentation that will be provided later in the project.

Response to feedback:

- Additional training data was labelled with over 100 frames and over 1,000 instances labelled, with a focus on the ant(s) at the leaf tip in each video. All subsequent models after the feedback was received have focused on improving the posture estimation accuracy for the ants pulling on the leaf tip specifically, and further labelling and training will continue for most of the remainder of the project.
- The Jupyter notebooks will be updated with internal code comments and Markdown cells clearly explaining their use, including where inputs are specifically required from the user. One or more in-person training and user acceptance testing sessions will also take place prior to the handover of the project to ensure that the client is comfortable with the use of all tools provided.