Machine Translated by Google


Eötvös Loránd University

Faculty of Informatics

Artificial Intelligence

Department

# Automatic annotation for pose estimation and for measuring movement speed in spinal cord injury on the rats database

Supervisor:

Dr. Anna Gelencsér-Horváth

Research Associate, PhD

Author:

Lorand Dallos

BSc in Computer Science

Budapest, 2025

Machine Translated by Google

# EÖTVÖS LORÁND UNIVERSITY
**FACULTY OF INFORMATICS**

# THESIS TOPIC REPORTER

**Student details:**
**Name:** Loránd Dallos
**Neptune code:** XIKVIF

**Training data:**
**Major:** software engineer, bachelor's degree (BA/BSc/BProf)
**Category :** Living room
I have an internal supervisor.

**Supervisor name:** Dr. Anna Gelencsér-Horváth
Name of workplace, department: **ELTE IK, Department of Artificial Intelligence**
Address of workplace: **1117, Budapest, Pázmány Péter sétány 1/**
**C.** Position and educational qualification: **research associate, PhD**

**Thesis title:** Automatic annotation for pose estimation and movement velocity measurement on a database of spinal cord injured mice

**Thesis topic:**
(In consultation with your supervisor, provide a description of the thesis topic in 1/2 - 1 page)

The aim of this thesis is to create an application that facilitates research based on the analysis of mouse movement. In medical research, annotation is a lengthy and monotonous process. Researchers at Semmelweis University want to study the movement of the hind legs of mice in the framework of research aimed at rehabilitation after spinal cord injury. To replace human annotation, I train a deep network using artificial intelligence, deep learning and machine vision methods that can automatically label the remaining part of the data set after annotation of up to 10% of the entire data set. Based on the predicted keypoints, I create an implementation of a method that generates the movement pattern/velocity of relevant keypoints from the image data in a format suitable for further analysis.

A user-friendly graphical interface is being created for the algorithm, on which: the trained model and the video for which the user would like the program to perform the analysis can be loaded; data related to key points can be generated for further analysis.

Budapest, 11 October 2024.

Machine Translated by Google

# Table of contents

Machine Translated by Google

TABLE OF CONTENTS

Machine Translated by Google

# Chapter 1

# Introduction

## 1.1. Motivation

Medical and pharmaceutical research often examines living, treated, and behavioral or movement patterns of untreated experimental animals. Typically recordings are made of the examined specimens, which are then provided with human annotation, to extract the necessary patterns and relationships from the data. This process is a lengthy, unskilled, monotonous and expensive task, therefore the Automation can greatly help the efficiency of research.

The Institute of Biophysics and Radiation Biology of Semmelweis University of Medical Sciences The leg movements of rats are being examined. During the experiment, Sprague-Dawley rats were They performed a surgical procedure on the calves, during which they destroyed the spinal cord. at the level of T9-T10 vertebrae. As a result, the hind parts of the animals became numb, paralyzed. The different groups in different treatments received, the effectiveness of these treatments was monitored for 20 weeks. The The possible healing of the lesions was monitored using various methods, one of which was so-called Open Field Test [1, 2]. This test is used by animals (mainly rodents) to destructive behavior, exploratory activity, and general mobility The test is performed on a circular track that is open at the top and closed at the sides. was carried out in complete darkness, under infrared light. The animals were examined weekly The recordings are 4 minutes long. The animals' movements are scored on a scale from 1 to 5. tuk, the number 1 meant the completely paralyzed legs and the immobile hindquarters, the and 5 represents completely healthy, active animals. In Figure 1.1, a sample I present a frame that well illustrates the nature of the recordings.

Machine Translated by Google



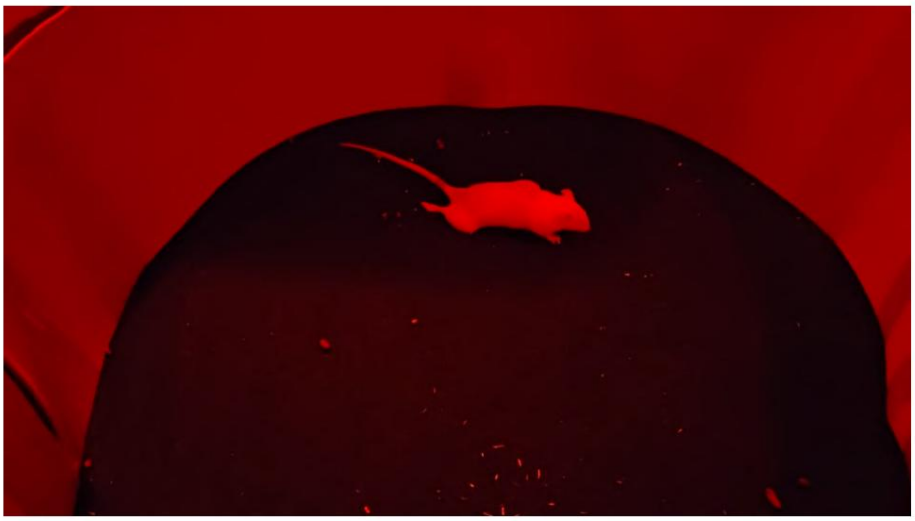


Figure 1.1. Sample frame that illustrates the nature of the recordings.

The goal is to achieve this without human intervention or with minimal human time investment. can categorize 20 videos of each of the 20 rats according to their current state Each category has a classified video as reference data. The We distinguish five categories based on movement patterns: category 1 includes those rats, which are almost completely unable to retract their legs, while Category 5 corresponds to completely healthy individuals with fast leg movements. Since detailed annotation would take an extremely long time, the goal is to develop an application that automatically categorizes the rat should be developed status, based on a small amount of annotated data.

Machine Translated by Google

# Chapter 2

# User documentation

The following chapter shows the entire operation of my program, in stages.
I will show you how to run the application on a new computer.

## 2.1. Purpose of the program

The aim of the application is to make it possible to automatically
to classify a video of a rat's movements. For correct operation
An input video is required that has one of the following formats:
'.mp4', '.avi', '.mov', '.mkv', '.flv', '.wmv'. The program has a user interface
returns a rating for the input video, along with an illustration of the chosen class
graphs created from the data along with graphs generated from the input video.
More about the mentioned input and output data and additional functions
I discuss this in section 2.5.

## 2.2. System requirements

The application can be used on computers with Linux operating system, or
I developed it to run on remote servers. The machine I use is Ubuntu
It runs the 22.04 distribution and has an NVIDIA GeForce RTX 4050 graphics card, 6GB
has virtual memory. Running is possible without a dedicated video card
also, but in this case it is worth considering a significant increase in running time, therefore
The configuration given is the recommended minimum requirement. The program is installed on a server,
can also be run remotely in a server-client format. The installation and running instructions described

Machine Translated by Google



I performed the steps on a server with two NVIDIA TITAN RTX cards.
each had 24GB of virtual memory. Of this, approximately
The program required six or seven GB.

## 2.3. Installation

During the installation of the program, it is assumed that the user has a
a machine running configured Linux, or a remote server.

### 2.3.1. Application file structure

The installed application has the following file hierarchy (Figure 2.1):

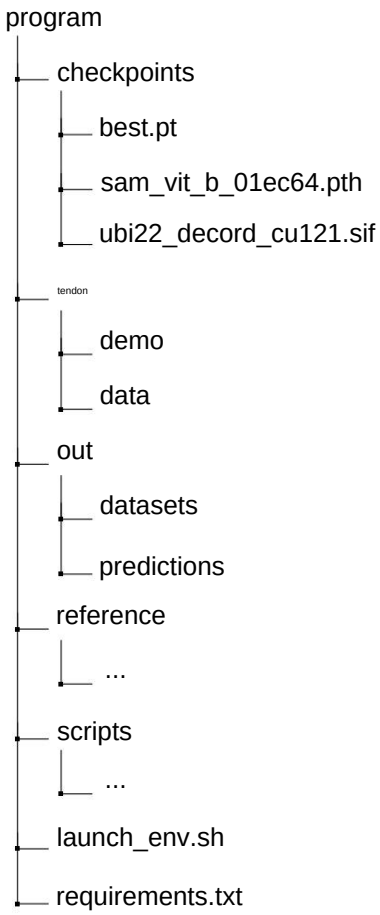



Figure 2.1. Structure of the program library.

I will introduce the mentioned libraries in more detail in the documentation. The
scripts and reference content in the developer documentation (Figures 3.4 and 3.14)

Machine Translated by Google



You do not need to interact with these as a user.

### 2.3.2. Steps

Steps to operate the application after downloading its files:

(1) Install Apptainer:

Apptainer is required to create a virtual environment for
program. Issue the following commands in the console:

```
$ sudo add-apt-repository -y ppa:apptainer/ppa
$ sudo apt update
$ sudo apt install -y apptainer-suid
```

More information about installing Apptainer is available here: [3].

(2) Setting up an Apptainer environment:

Checkpoints/ubi22_decord_cu121.sif and launch_env.sh shown in Figure 2.1
we use it to create the virtual environment required for execution, which is used to
to create conditions for the program on the machine such as-
I developed it in . This guarantees deterministic behavior. Let's run the
launch_env.sh file, and then issue the following commands:

```
./launch_env.sh
apptainer shell instance://yolo_v12
source .envs/venv_yolo_v12/bin/activate
```

Success is indicated when the command line looks like this:

```
(venv_yolo_v12) Apptainer>
```

(3) Install the required python packages into the environment:

These are contained in the requirements.txt text file, with the following command
the packages included in it can be installed:

```
pip install -r requirements.txt
```

The program is ready for use, which I will demonstrate in section 2.5.

Machine Translated by Google



### 2.3.3. Downloading data to the project directory

The program loads the input data from the in folder. In the data folder
must download or copy the annotated data necessary for teaching, paying attention to the
for their structural correctness; and the video files you want to classify in the demo folder.

## 2.4. Surface

In the following subsections, I will discuss the appearance and capabilities of the application, as well as
with labels for transparency.

## 2.5. Usage

### 2.5.1. Launching an application

To launch the application, in a terminal, go to the program's installation directory,
Within this, you need to navigate to the scripts folder and run the following command:

python -m streamlit run index.py

This will run the program's entry point, after the command a console message similar to this will appear
is displayed (Figure 2.2):

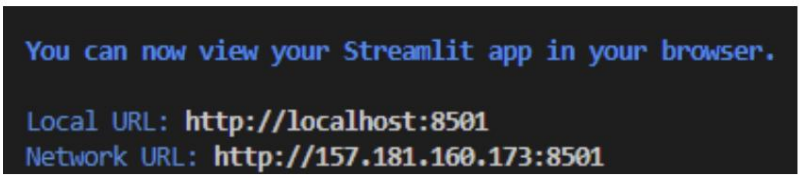



Figure 2.2. URLs that access the application.

Open the URL above, which will take you to the main page of the application, where you will find another summary.
We get a description of the operation of the application (Figure 2.3b).

### 2.5.2. Presentation of the work session

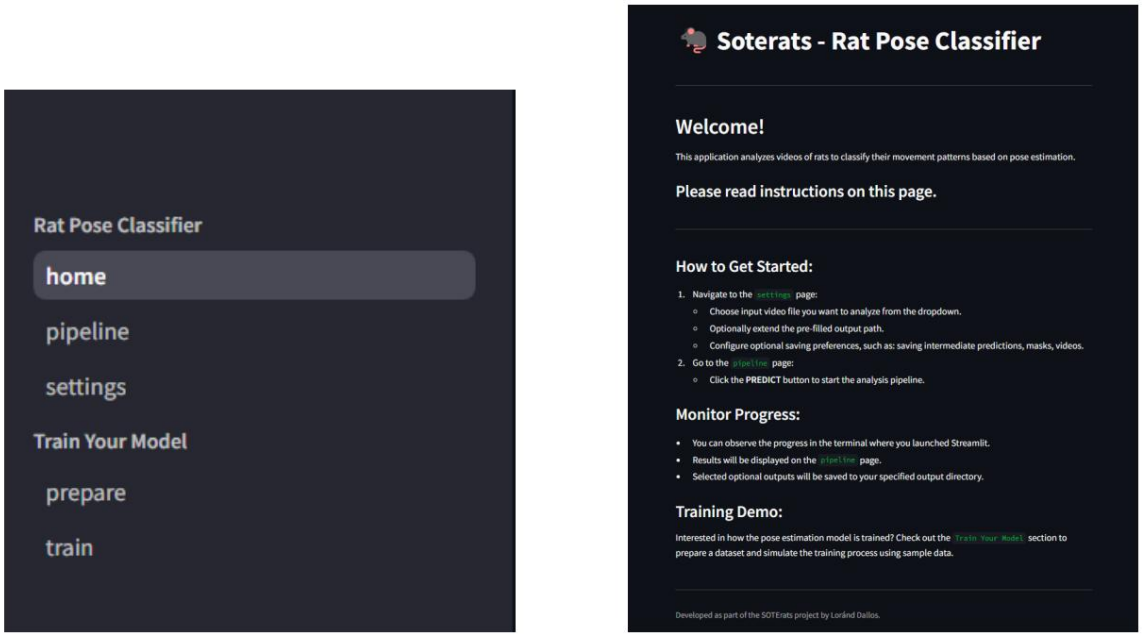
Home page

This is the default home page of the application, where it drops the user.
The pages are navigated using the menu on the left side of the screen (Figure 2.3a).
It is possible to navigate with the . Here we can check that the home page is active. On the page

Machine Translated by Google



guidance information is included (Figure 2.3b) that can help you decide the next step in its storage.




(a) The program's navigation menu.

(b) The main page of the program.

Figure 2.3. A striking interface when starting the program.

After reading the information on the main page, we enter it on the settings page. the settings required to run the program.

Configure settings

The page is shown in Figure 2.4, and is located in the program under the name settings. in the side menu.

You will see two route entry controls and eight switches. The top drop-down menu offers the videos stored in the in/demo folder, the second controller automatically kushan names the output directory based on the input video, leaving it optional individual renaming. In case of re-running the same data, it is worth di to extend the folder name with an additional string of characters so that it does not overwrite the previous save The folder for output paths is located within the project directory at out/predictions, this is created by the program and saves the output in a sub-folder with the same name as the input video. The third field offers the option to select the model used, defaults to the one I use the path of the trained model. These were the mandatory settings. The eight switches are allows you to save intermediate states that occur during processing. Their setting is not

Machine Translated by Google



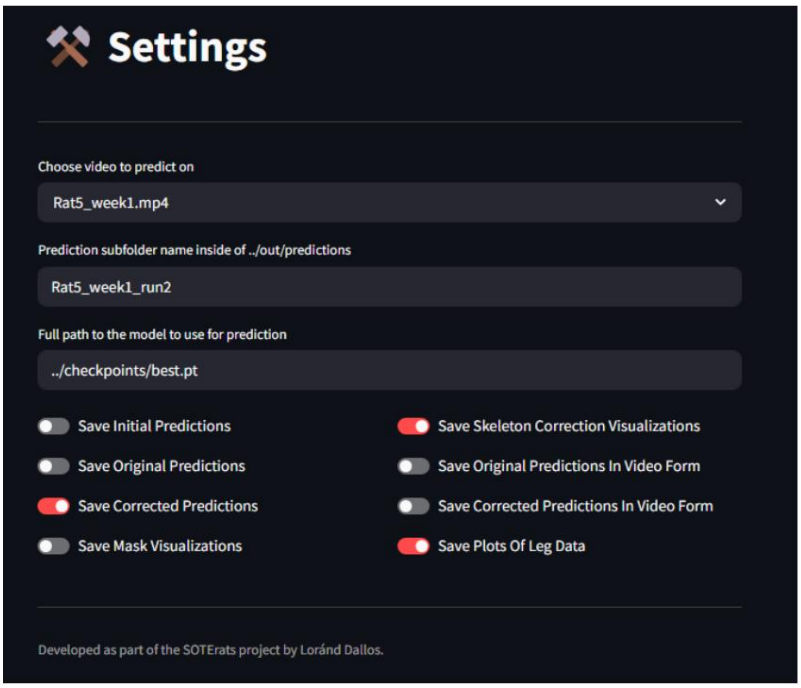



Figure 2.4. The settings page.

are separate and independent from each other, except for the settings related to video stitching, which have the basic requirement of saving the frames needed for the video. If this is the prerequisite is not met, the program will run, but the videos and files will not be saved.
The program draws attention to a possible malware. The settings are explained in section 2.1. summarized in the table:

To run, enter the required data and select any intermediate steps.

save states. After specifying the settings, the program can be run in the pipeline

starting from the side.

Machine Translated by Google



| Switch name | Function description |
|---|---|
| Save Initial Predictions | Saves the primary, raw YOLO predictions (even before the processing steps, drawn on the images in a default style). |
| Save Original Predictions | Saves uncorrected but already processed predictions with their own drawing style. |
| Save Corrected Predictions | Saves final predictions corrected based on toe positions with your own drawing style. |
| Save Mask Visualizations | Saves SAM masks in image format, visually displaying the outline of the object. |
| Save Skeleton Correction Visualizations | It saves the images generated during skeletal correction, showing the endpoints and any corrections, as well as the search radius. |
| Save Original Predictions In Video Form | It joins the frames containing the original predictions into a video. |
| Save Corrected Predictions In Video Form | It joins the frames containing the corrected predictions into a video. |
| Save Plots Of Leg Data | Saves graphs depicting time series data (amplitude, frequency, etc.) derived from leg movement. |

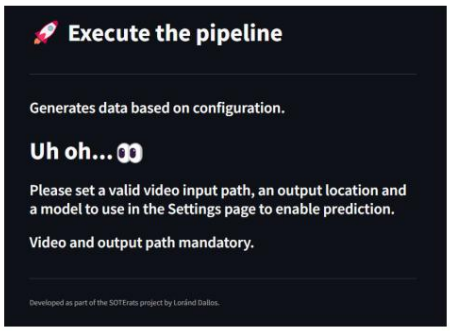
Table 2.1. Meaning of output options in the application.

Processing

The pipeline page is simple, with a single interactive element. This is the PREDICT button, which can be used to start processing the video given as input, provided that ve that we have set a valid input and output path (Figure 2.5b), and we have specified the path to a model to be used. If this is not the case, A message prompting you to make a correction is displayed on the page, or the button is unavailable (Figure 2.5a). After the run, the prediction result is displayed on this page.

Machine Translated by Google






(a) Skipping page settings in case.


(b) Correct page settings in case.

Figure 2.5. Possible layouts of the pipeline page.

Clicking the processing button will start processing the specified input data. processing, which is completely automatic. Depending on the data, the calculations are time-consuming may be present, so a warning and a rotating element will appear on the screen, indicating the The user can follow the main steps of the process from the console where from where you can follow the program interface. The menu items during runtime pressing may lead to inconsistent operation, so running a given task In the meantime, interactions should be avoided.

After processing, the selected movement category number will appear on the interface. today, and as an illustration, the time series graph generated from the input video is the response- next to the graph of the time series of the category tott. If we saved, the output The data is created and can be viewed in the structure shown in Figure 3.10. The output The result is saved in text format in the directory verdict.txt.

For illustration purposes, you can view the page and console while it is running in 2.6. and in Figure 2.7; after running down in Figure 2.8:



Figure 2.6. Logs displayed in the console.

Machine Translated by Google



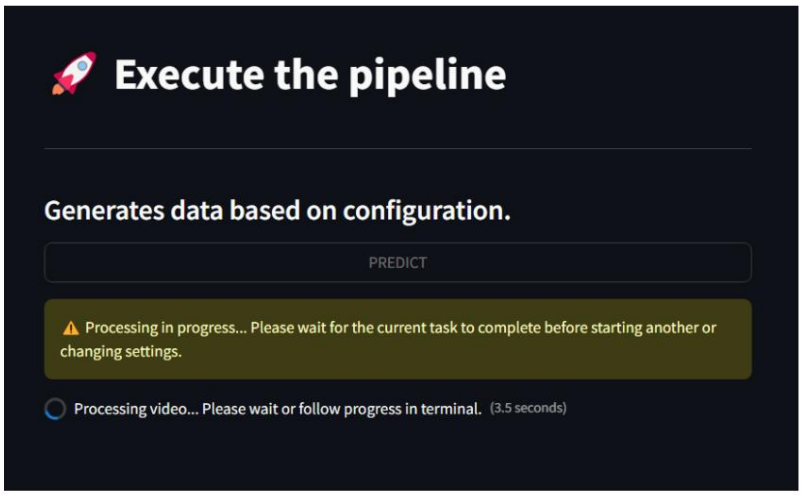



Figure 2.7. Processing during runtime.

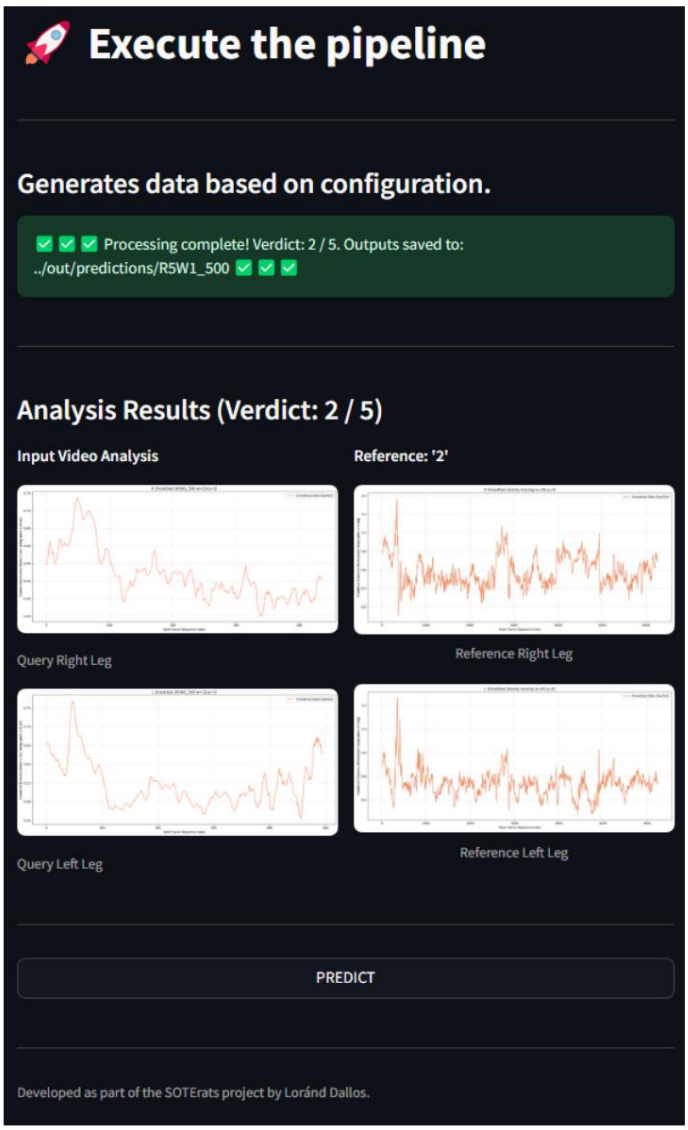



Figure 2.8. Processing result. The graphs can be enlarged by clicking.

Machine Translated by Google



### 2.5.3. Model training

This component includes two pages: prepare, where the user can enter the
can generate a training data set based on input data presented in the
to train your own model; train, where you can start the model based on the data set
teaching.

It is important that the user uploads before using this part of the application.
directories with a predefined structure (the in/data folder shown in Figure 2.1)
ba), which contain the frames and annotations used as the basis for the training data.
If the user wants to add additional data, they can upload it here.
the formatted libraries that the program recognizes.

The keypoint annotation was made with the DeepLabCut [4] software, this application
creates the expected '.csv' files for the image data on my input. In the '.csv' files
The following key points were annotated for each rat:

| Index | Key point name |
|---|---|
| 0 | left ear |
| | right ear |
| 1 2 | nose |
| 3 | spine |
| 4 | tailbone |
| 5 | tail middle |
| 6 | tail end |
| 7 | left leg thigh |
| 8 | left leg knee |
| 9 | left foot |
| 10 | right leg thigh |
| 11 | right leg knee |
| 12 | right foot |

Table 2.2. Number and name of key points.

A sample annotation can be seen in source code 2.1, which is a sample of the annotation program.
The annotation protocol was agreed with biologists.
We have compiled it to fit the task. Each key point is a two-dimensional
Annotated in the (x, y) coordinate system.

Machine Translated by Google



```
1 scorer , , , HP , HP , HP , ...
2 bodyparts , , , left ear spine , tail01 leg , left , left ear , right ear , right ear tail02 tail02 tail03 tail03 , nose , nose , spine ,
  leg knee left leg knee , tail01 , , , , , left leg, left
  leg , right leg , right leg knee leg toe , , left leg toe, right , left leg toe, right
  leg knee , right leg toe , right
3 coordinates , , ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y ,x ,y
4 labeled - data , Rat01_week19, img0122. png
  ,1090.3079738565527 ,806.4054245451202 , ...
5 labeled - data , Rat01_week19, img0123. png
  ,1104.091166218707 ,809.1620630175511 , ...
6 ...
```

Source code 2.1. Structure of a CSV file containing key points.

Data preparation

Located on the prepare page (Figure 2.9), it is used to prepare a Dataset needed to train YOLO pose estimation model. Switches visible We can choose from the teaching data uploaded to the in/data directory.

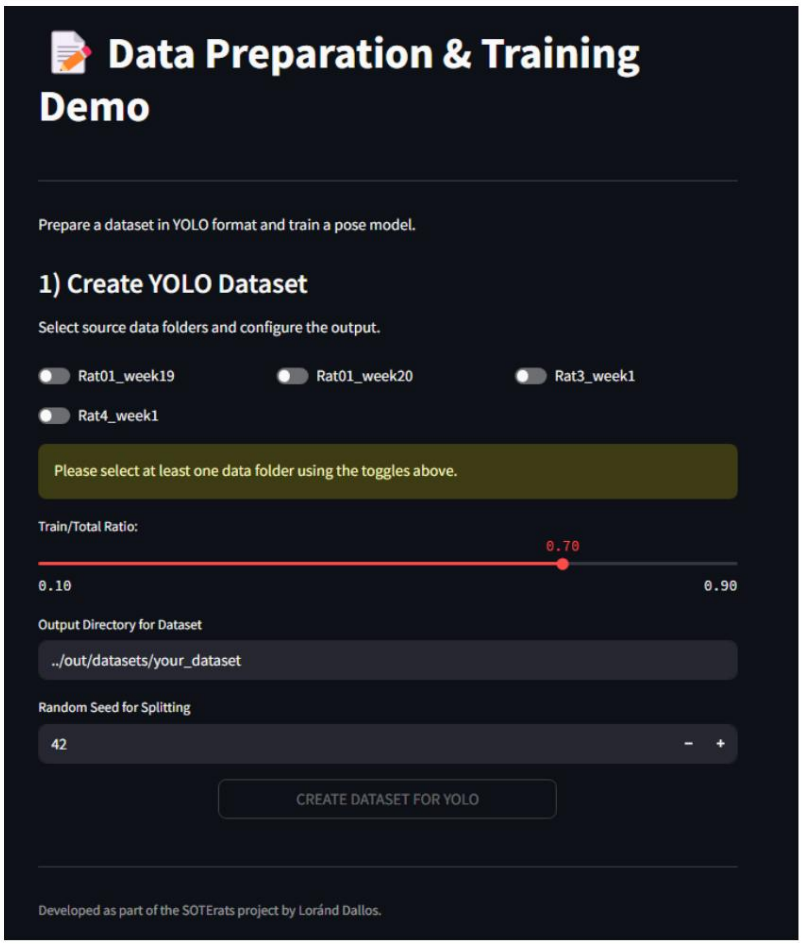



Figure 2.9. The training data generation page.

Machine Translated by Google



In the in/data folder in the program structure, based on the figure above, the following is
The following libraries are included:

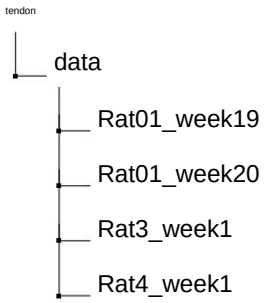



Figure 2.10. Demo training data structure based on Figure 2.9.

Below these is a sliding control with which we can select the teaching data.
ratio of the case to the total data. By default, the data
70% of the data is used for teaching. It is recommended that up to 80% of the data be used for teaching.
that the remaining 20% is used for validation and testing and helps the model
avoiding overfitting. Furthermore, the storage path of the data set can be specified,
which must be supplemented with the unique name of the dataset, for example: out/datasets/ds1).
This is pre-filled with an example name, which can be edited freely. Finally
a random number that determines the random number generation and ensures that
the random distribution of the data should be reproducible.

We can start the generation with the CREATE DATASET FOR YOLO button, then strictly
After the run-down, we can move on. As a result of the run-down, the following structure comes
We create a dataset:

Machine Translated by Google



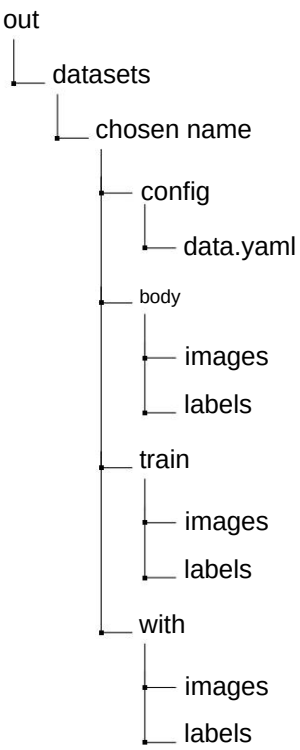



Figure 2.11. Structure created by data set generation.

data.yaml is a configuration file whose contents specify where to find
the data set necessary for teaching.

Configuring teaching

On the train page (Figure 2.12), we specify the configuration generated on the prepare page.
the path to the file; the name of our future model; the access to the model storage directory
si path (recommended "out/datasets/dataset generated in previous step)
save the model in a "access path" location to transparently access the learned data
the number of training cycles. The latter setting affects the
It indicates how many times the learning model iterates over all the training data. A
A higher cycle number may result in higher accuracy, but the learning time is
can be extended as follows. The path to the configuration file and the model
The program fills in the save path based on the settings on the prepare page.

We can start training our model with the TRAIN YOLO MODEL button.

Machine Translated by Google



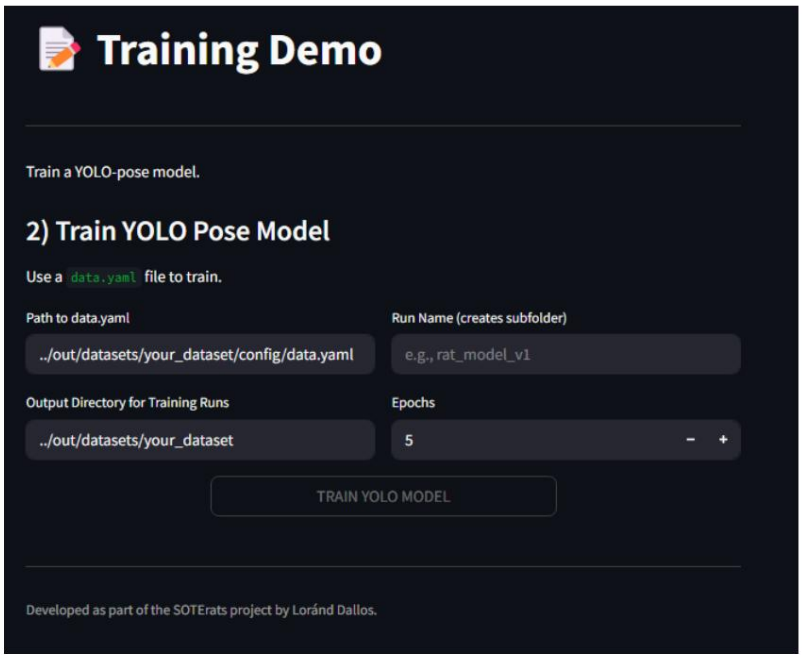



Figure 2.12. The model teaching page.

### 2.5.4. Alternative run option

The prediction part of the application can also be used directly from the command line (nogui_pipe.py). During operation, the input required for video analysis is
Various switches can be used to control which outputs are saved.
These switches are the same as the save options that can be set in the graphical interface.

In addition to video processing, I also created two additional scripts (nogui_prepare.py and nogui_train.py), which allow data collection
The generation of the model and the training of the model are done in the same way as the user with superficial possibility.

## 2.6. Errors due to incorrect use

The program design helps to avoid misuse. If the
If the data in the expected format exists at the specified path, the program will run correctly.
generates output to folders specified in output paths, as needed
creating the folders.

The most serious error is when the user exits a task while it is running.
This cannot be disabled explicitly from the framework used. The
Therefore, the program always displays a warning during execution, stating

Machine Translated by Google



prompts the user not to do this. If the page change does occur,
the program can still be followed in the console, but the user interface is lost.
the connection with the running state, therefore, in case of a possible return navigation, the initial input
displays statements. Also, if we switch to a page while another task is running
while, and then navigate back to the original page, the interface will restart the process
tot, multiplying the system requirements and overwriting the output directory of the original run.

If the processing pipeline page is not specified with a video or output path
If you try to run it, the button will not appear. If the video to be processed is too
The program will notify you briefly, as shown in Figure 2.13:

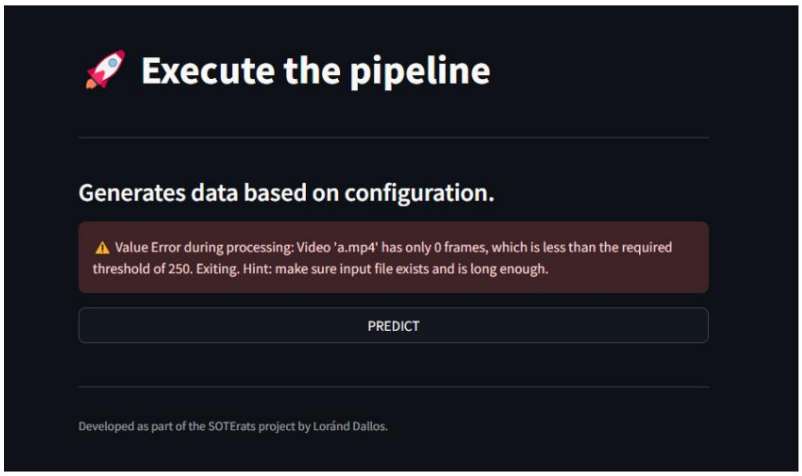



Figure 2.13. Error message for short or non-existent video.

Access to all controls on the page while any process is running
denied to the user, so he cannot influence their value, which is from the page
would lead to a similar result to navigating.

If none of the components selected when creating the training data set make
meets the imposed structural constraints, then the error message shown in Figure 2.14 will be displayed.
I show the user the way with net:

Machine Translated by Google



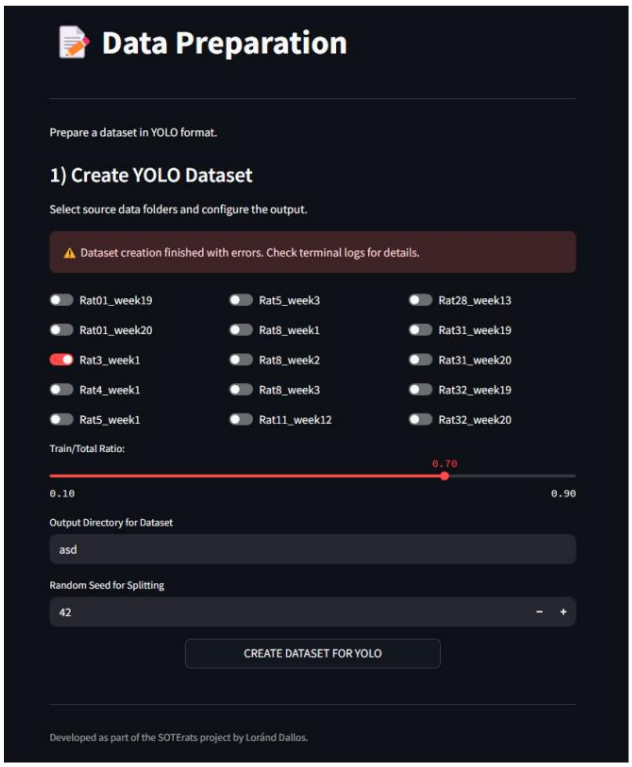



Figure 2.14. Error message in case of incorrectly structured training data.

If only some of the specified libraries have an unrecognized structure, they will be program is skipped.

If the user has not yet uploaded training data, as shown in Figure 2.15. the program warns effectively.

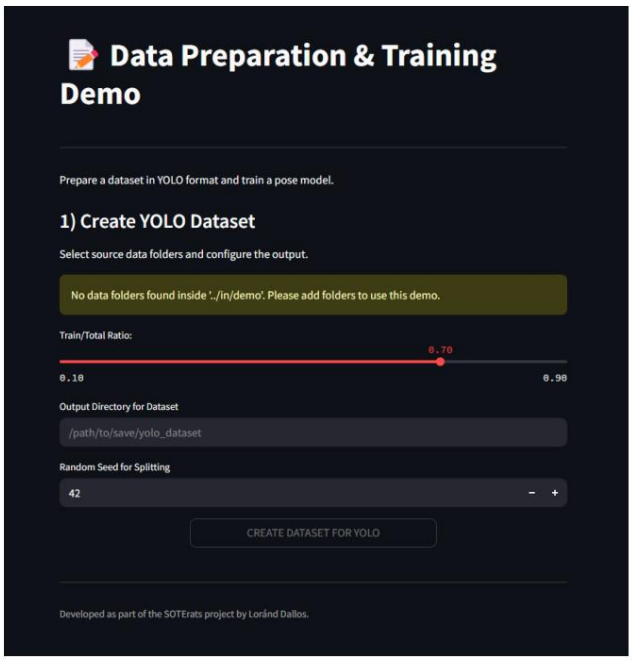



Figure 2.15. Warning for empty training data library.

Machine Translated by Google

# Chapter 3

# Developer documentation

## 3.1. Planning

I chose the Python [5] programming language to create my program. The
Python is a widely used language in scientific research, with wide support
its research branches out into the fields of image processing, artificial intelligence and data visualization
It is easy to read and ensures rapid development.

During the design of the software, I tried to break it down into sub-units and modules.
sam so that each module is responsible for one phase of data processing. The trans-
In addition to visibility, there are more opportunities to expand the application area, new, even
for adaptation to a different type of recording.

The studies and applications of DeepLabCut [4], Sleap [6], MMpose [7], and Ultralytics [8], as well
as a similar research [9], provide a significant starting point and
they gave me guidance.

## 3.2. Main components and basic concepts

There are basically several pre-built libraries available for the Python language.
with a simple import statement.

In the following section, I would like to introduce some basic principles and concepts that
occur in my application.

Machine Translated by Google



### 3.2.1. Machine learning

Machine learning is a field of artificial intelligence in which computers solves tasks not based on predefined algorithms, but from the data learn the patterns and rules needed to solve them.

Its main types are supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the training data is labeled and The goal is to create a model that can improve in the case of new, unknown data. In unsupervised learning, there are no predefined labels. The purpose of the model is to recognize patterns and groups (clusters) from the data. In reinforcement learning, an agent makes decisions in an environment, and receives feedback (reward or punishment) based on how well your decisions were correct. Its goal is to gain the most rewards in the long run.

Neural networks

Neural networks within machine learning are a way of learning about the human brain. patterned mathematical models that contain many interconnected “neurons” (computing units). An artificial neuron is a simple computing unit, which models the functioning of a biological neuron. Each neuron receives multiple inputs gad, which have weights attached to them. Figure 3.1 provides a visual illustration of what has been described. The neuron first calculates the scalar values of the input values and their corresponding weights. The sum obtained is an activation goes through a function that determines the output of the neuron. The activation The introduction of the function is crucial as it allows the network to be nonlinear. also learn relationships, thereby increasing the expressiveness of the model. The non-linear nearness means that the relationship between inputs and outputs cannot be written with a simple linear (straight line) function, but rather with complex patterns and develops along connections.

Machine Translated by Google



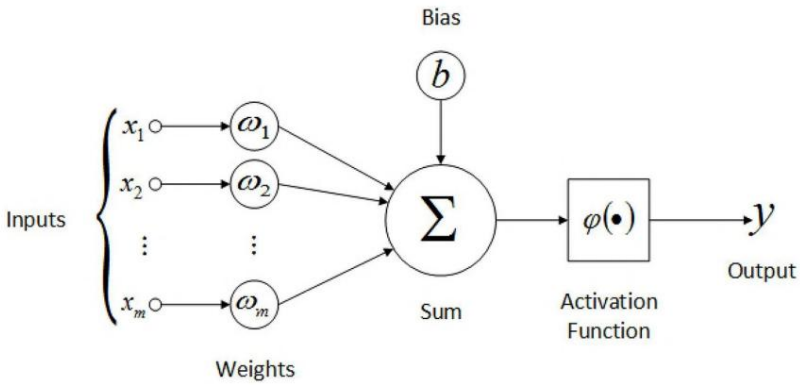



Figure 3.1. The structure of the artificial neuron based on [10].

By arranging neurons into layers, where the outputs of each layer are as follows become the inputs of the layer, build the so-called neural network, ensuring the network information flow within the network. The learning process fine-tunes the weighting of and shifts, typically using the backpropagation algorithm and some optimization method (such as gradient descent). As a result,
As a result, the network is able to make increasingly accurate predictions based on the inputs. A semantic diagram of a neural network is shown in Figure 3.2.

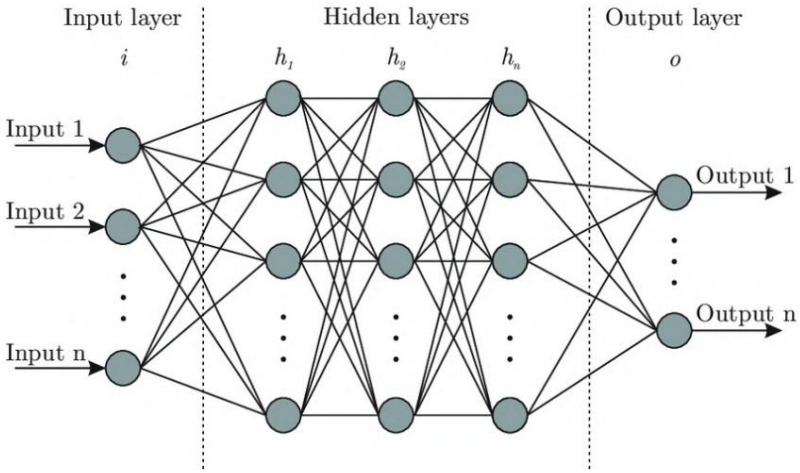



Figure 3.2. General structure of neural networks based on [11].

Convolutional neural network

Machine vision is a field of science that aims to enable computers to be able to interpret and process visual information. As an informative website [12] also mentions:

> Machine vision basically means when image-based information is (even video) as input data, and we start with this data

Machine Translated by Google



something. This something could simply be the collection and evaluation of data,
but very often, analyzing the image, we are faced with some kind of decision-making situation
We use a computer, and the decision is some kind of mechanical solution
For example, at the exit of the parking garage, the machine lifts our car
the barrier in front of us, if the camera image of our license plate matches the toll booth
with information coming from tomato you get the result that we have paid
the parking.

In machine vision, a common approach to object detection and pose detection is con-
application of convolutional neural networks (CNN).

The basic idea of CNNs is based on the workings of the human visual system: instead of each pixel would be treated separately, local patterns would be examined, for example edges, corners, textures [13]. For this, so-called convolutional layers are used. in which smaller, learnable filters are slid across the image, and the similarity is calculated at each position. Typical CNN architectures have multiple convolutional layer, followed by activation functions and coverage layers follow. This combination allows the network to become increasingly recognize more specific features while reducing data size and computational effort The final layers are usually fully connected neural layers that are predictions or classification are made from the collected features. CNNs typically trained with supervised learning and gradient-based error backpropagation, where the error in the network's predictions is fed back through the layers to update its weights to minimize the error.

In the section below I have collected the packages I used for in order to create my application, and I would like to explain some basic principles, concepts that occur in this.

## 3.2.2. Packages used

The packages include productivity enhancement, image processing, data visualization, file management, They supported reading and performing mathematical operations.
Packages focused on efficiency include numpy [14], which allows for fast execution of array-based numerical operations, and torch [15], which enables GPU-accelerated computing with the PyTorch deep learning framework and training neural networks.

Machine Translated by Google



Traditional image processing algorithms are mainly implemented in the opencv [16] library.
I implemented it using a software that allows you to scan and process images.
The PIL (Pillow) [17] library is also used for image processing tasks.
provided, mainly during loading and converting images.
Matplotlib [18] was used to visualize the results, creating graphs and
I plotted time series. To train my own models, the data was processed using the pandas [19] package.
I analyzed and treated it.
I used a combination of several modules for file management: os, shutil and pathlib
I used packages to perform file operations and path management, while gc and
tempfile helped with memory management and creating temporary files. yaml
and I used the same method to read and save json format files.
libraries with the name.
The math package provided the basic functions for mathematical calculations.
For more complex operations such as filtering and time series transformation, scipy
[20], including the scipy.signal submodule. The skimage [21]
The morphology package extension made it possible to use the center of mask objects
The random package is used to introduce randomness.
served, among other things, in the selection of teaching data.
In this paper, two modern deep learning-based neural network architectures are used.
I used: the YOLOv8 (You Only Look Once) [22] pose estimation model, and the
Segment Anything Model (SAM) [23], which is used for image-based segmentation. The
Using the segment_anything package, I easily integrated the SAM model into
The YOLOv8 version is implemented in the Python package provided by Ultralytics.
I used it with mag, which provides a high-level application programming interface
offers a way to load and run trained models and display the results of detections
for treatment.

There are many variations of YOLO models. They are described by two properties:
their size (n, s, m, l, or x, i.e. nano, small, medium, large, or extra large) and
the function of the model (pose, seg, cls, etc.). The larger a model, the more accurate it is
It can produce results, but it also requires more resources to run. The v8 engine
Dell family includes detection, segmentation, pose detection and classification models
architectures. Pose models not only recognize objects in the image, but also
They are also able to estimate key points on the body. This is especially useful when it is necessary to
information, for example, in the case of an animal, in addition to its location, what pose it is in,

Machine Translated by Google



where he looks or how he moves.

Among the YOLO model versions, I chose the v8 model. The official documentation states

based on the reference measurements and comparisons presented in their presentation, it appears promising

(Table 3.1), and practical experience also confirmed this in the later, v11 version

Compared to the previous study: the results shown in Figure 3.3. show clearly that the

YOLOv11 outperforms the v8 edition, but this measurement is not based on the pose estimation task.

born. More on the comparison of the performance of the models is given in section 3.5.2.

information found.

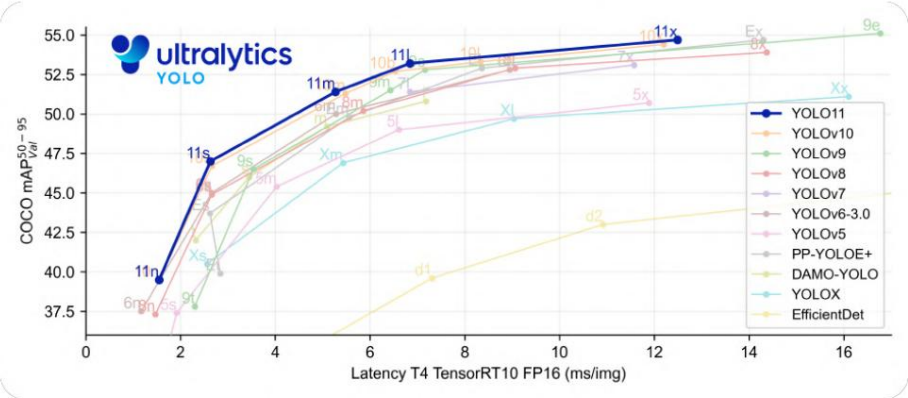



Figure 3.3. Benchmark highlighting the performance of YOLOv11. [8]

| Model mAP pose 50-95 mAP pose 50 | | |
|---|---|---|
| YOLOv8l-pose 67.6 YOLOv11l-pose | | 90.0 |
| 66.1 | | 89.9 |

Table 3.1. YOLOv8 vs YOLOv11 pose estimation performance.

## 3.3. Description of the structure

Below I will explain the thinking behind the structure of the application,

first summarized and then explained in detail. My application is divided into two main parts-

effective, these are the following:

- 3.5 - Training the model
- 3.6 - Data processing

### 3.3.1. Teaching your own model

The goal of the module is to transform data received from biologists into something usable for us.

format (source code 3.2). After that, the Ultralytics YOLOv8l-pose model

Machine Translated by Google



Using the data converted into the appropriate format, we can further teach this.
This will create a specialized model for estimating key points in rats.

### 3.3.2. Processing

Its purpose is to cut the input video into movie frames so that I can process them in the sa-
with my YOLO model; using the aforementioned model to fit the key points to the rat-
ra; generating a mask for each image using a SAM model; the masks
filtering out images where the rat is not moving; also masks
based on fitting an artificial skeleton to the mask shapes; the foot is the key point
improving estimates using the aforementioned skeletons; comparing the corrected data
compared with five existing sample data categories, and based on this result, the original
classification of an individual in a video. In my implementation, I give a name to the moving
si categories, but I return a number at the end of processing. The categories are: (1)
crippled (2) barely moving (3) intermediate (4) almost healthy (5) healthy.

## 3.4. Structural structure

In this section, I would like to outline the structure of the application in Figure 3.4.
for making the later details more understandable.

Machine Translated by Google



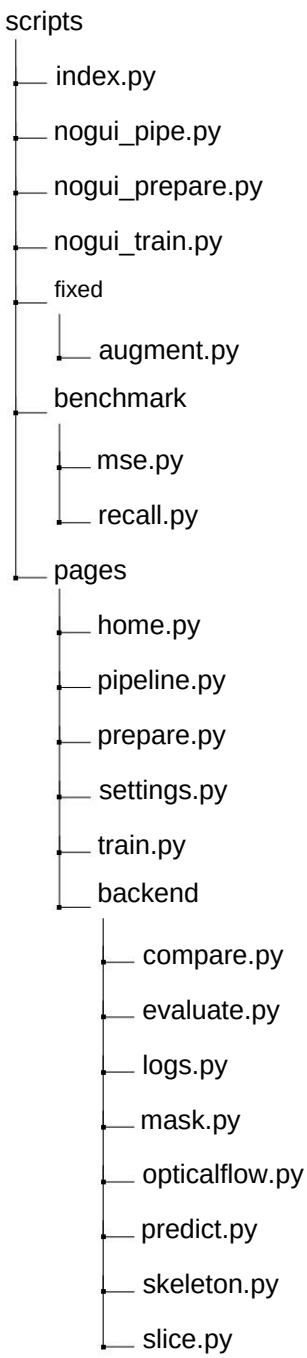



Figure 3.4. Structure of the project library.

## 3.5. Training the model

This part of my application is divided into two further subsections.

Machine Translated by Google



### 3.5.1. Dataset preparation

Before programming my application, I tried something similar as part of the project. model-training applications (such as DeepLabCut, Sleap mentioned in the introduction). Of the mentioned, I found DeepLabCut (hereinafter DLC) software to be the most more user-friendly. The contributing biologists thus use the software I recommend. vert was used for annotation, the DeepLabCut annotation application. It was useful to me, that this saves the prepared data in a '.csv' file, so biologists they could easily give me the hand-crafted teaching data. An important first step is I refined the data I received.

#### DeepLabCut annotations

To create annotations, you need to install the DLC software, preferably a virtual one. environment, such as a conda [24] environment. After opening it, you need to create a new project with any settings you want, but the video path you want to tag The next step is to select whether to automatically The goal is to label frames selected by software or manually. The In the latter case, you need to configure the project's config.yaml file so that How much of the video is continuously cut up by the software.

After the settings, you can start annotating the images, a separate interface is provided for this. DLC. It saves the tagged data in a comprehensive '.csv' file. Biologists use such '.csv' summaries and the images mentioned in the files were provided to me for training the model. For more comprehensive documentation, see [25].

#### CSV scan

The actual application model training part begins at this phase. The data I always received it broken down into weeks. The format of the library containing the data In Figure 3.5, I illustrate with an example library:

Machine Translated by Google



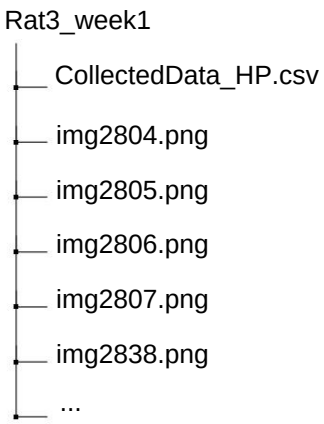



Figure 3.5. The library containing the rat data.

Module 3.5.1 expects an input folder with subfolders as shown in Figure 3.5. It is made up of folders with an observable structure, and the '.csv' files contained in them files follow the structure shown in source code 2.1. If it differs from this the input folder, the module generating the training dataset does not take into account the given library.

The original format of the data can be observed in source code fragment 2.1. The uniform In order to create data in the form of .csv, I read it from the first '.csv' file in the input folder. the individual body parts, and then in later files I will discard them based on this, whose headers do not match the established body parts. The keypoint data is During the winning process, I also constantly pay attention to the predicted number of body parts. keypoint quantity is maintained in the data file. Where the given point, I replace it with a zero value. By converting the keypoint data to YOLO format, I store lom, which I present in source code 3.2.

Based on the above, I will collect all the images and their corresponding names from each '.csv' file. keypoint data into a comprehensive image or data set. The images I will rename it using the folder name in the '.csv' file, so it is guaranteed to be The image will have a unique identifier (RatX_WeekY_imgZ.png) and the collector they will not overwrite each other in a set.

YOLO data format

In computer vision, especially in object detection and pose analysis tasks, During the process, it is crucial to properly annotate the image data. There are two common forms of I use Ultralytics YOLO and COCO for this. Within the framework of my thesis, YOLO

Machine Translated by Google



I will introduce a data format: a simple, text-based annotation system,
which is specifically optimized for efficient training of YOLO architecture models
Each image has a file with the same name and extension '.txt'.

In this file, each line represents a single detected object. The de-
representation of a tected object in a row: the object identifier; the object
coordinates of the center of the bounding box (hereinafter referred to as the bounding box), width
latitude and altitude. All coordinates and distance data are in the dimensions of the image
It is in a normalized state, meaning that each value is a number in the interval [0, 1].
This is important because the model resizes the images during training, so the pixel
measured distances and coordinates would be moved to a different location in the image. Normalization
allows you to store a position in the image without specifying a specific size.
The general structure of the data format can be observed in source code 3.1:

```
1 < class_id > < bbox_center_x > < bbox_center_y > < bbox_width > <
      bbox_height >
```

3.1. Source code. Ultralytics YOLO format

However, the source code 3.1 above is a more general, object recognition model.
The format required for teaching pose tracking models is
In comparison, the math is supplemented with the coordinates of the key points you want to annotate,
optionally with their visibility. The visibility value can be 0, 1, 2. The zero value
value means that the keypoint is not annotated or is not visible (e.g., it is in masking, poor
quality). According to some values, the keypoint is annotated, but not
visible (for example, a location given by a human estimate). Finally, it is marked with a value of two if
the key point is annotated and visible in the image (clearly visible). In 3.2.
The resulting structure can be studied in the source code. My program is not visible.
uses an external approach, as the biologists' annotation tool does not support it
indicating the visibility of the points.

```
1 < class_id > < bbox_center_x > < bbox_center_y > < bbox_width > <
      bbox_height > <px1 > <py1 > <px2 > <py2 >                    ... <pxn> <pyn>
2 OR
3 < class_id > < bbox_center_x > < bbox_center_y > < bbox_width > <
      bbox_height > <px1 > <py1 > <p1 - visibility > <px2 > <py2 > <p2 -
      visibility >          ... <pxn > <pyn > <pn - capacity >
```

3.2. source code. Ultralytics YOLO format with keypoints in 2 and then 3 dimensions
version.

Machine Translated by Google



Above I discussed the '.txt' file that carries annotated data for an image.
file structure. I would like to show you how we have at our disposal
How can images and text files be arranged to be transferred as teaching data?
a YOLO-pose model. Within a data set, the three main parts are divided into
images and text files, which must always be treated as a single unit (a given image and
(the associated text file must be in the same category). These
The parts are the train, val and test subfolders, which are further subfolders of images and labels.
The data is stored in a buffer, as shown in Figure 3.6.


Figure 3.6. Data set structure.

The training set contains the data on which the model is directly trained.
During the teaching process, the model learns through these examples
patterns, relationships between visual information and the desired output.
The network weights are updated iteratively based on this data. Typically, the full
It accounts for seventy to eighty percent of the data set.

The validation set is used to monitor the training process.
model performance on data that the model has not seen directly
during learning. It can be used to detect the phenomenon of overlearning: if the model is the teacher
data, but its performance deteriorates on validation data, overfitting

Machine Translated by Google



refers to the learning process. The training can be stopped when the validation performance is no longer satisfactory.
continues to improve within a predefined limit, saving time and resources
va. This is called early stopping in the literature. This set is the complete data
ten to fifteen percent.

The test set is used only after training is complete to obtain a final, objective
to obtain a subjective assessment of the model's generalization ability based on completely unknown data.
This gives the most realistic picture of how the model will perform in real-world conditions.
This data should not be used during training or validation.
should be used to ensure that the evaluation is unbiased. Its size is the same as the validation
with its set.

Additionally, to train your own YOLO-pose model, you need a data.yaml
configuration file that specifies the paths to the dataset to be loaded.
The YAML (YAML Ain't Markup Language) format is readable by a human eye.
is a scalable data serialization standard and describes data using key-value pairs. The
The YAML file I used contains the following:

- path - The relative or absolute path to the root directory of the dataset.
  The train, val, and test paths are relative to this root.
- train - The path to the training folder.
- val - Path to the validation folder.
- test - The path to the test folder.
- kpt_shape - The number of keypoints a and the dimension size b in the format [a, b].
- flip_idx - This is a list that specifies which image to flip horizontally.
  keypoint index corresponds to which other. This is an essential symmetrical object-
  For example, if index 0 is the left eye, then
  If the right eye is 1, then in the flip_idx list the zero position is 1, and the first position is 0
  If a keypoint has no counterpart, its index refers to itself.
- names - A dictionary containing the names of the various recognizable classes.

This structure is much less dynamic in my case than it might seem at first.
het. Several invariants exist during the research: The kpt_shape is replaced by [13, 2], because
I follow thirteen key points during the experiment, and there is no visibility associated with it.
related data, so my keypoint data is two-dimensional (x,y). The names of the keypoints

Machine Translated by Google



and index-corresponding to them are shown in Table 2.2, according to which flip_idx contains still consistently today [1, 0, 2, 3, 4, 5, 6, 10, 11, 12, 7, 8, 9], since the ears and the feet are the ones which can be affected by a horizontal mirroring.

The names field with the value {0: rat} still represents that there is only one My model, the rat, must have a reasonable class. It is important to mention that if I had multiple classes, the order would have to be the same with the order found in '.txt' files containing data. The only variable factor is the path in the entire data.yaml file, as it is possible to combine data from different weeks creating mixing datasets, which is always worth a new, configurable path Save. Source code 3.3 is an example, showing a complete data.yaml contents:

```
1 path : / path / to / datasets / example_dataset
2 train : train / images
3 values : values / images
4 text : text / images
5 kpt_shape : [13 2] ,
6 flip_idx : [1 0 2 7 names : {0: rat } , , 3 , 4 , 5 , 6 , 10 , 11 , 12 , 7 , 8 , 9]
```

Source code 3.3. YOLO-pose model configuration data.yaml.

The algorithm I wrote produces a data structure that respects these structures. set and a configuration file, provided the input directory is available The complete code for the component can be viewed in the prepare.py script.

## 3.5.2. Teaching your own model

Model variants

During development, in addition to my final decision, I tried the second-largest model with better performance on pose estimation task, the newer version YOLOv11.

Furthermore, I tried to increase the accuracy of these model types by I teach and evaluate them on cut pictures. This method is significantly more and requires post-processing. First, I used an existing and complete image null model to reliably find and cut out the rat from the original image using a Segment Anything Model (SAM, more in subsection 3.6.3). To supplement the new, smaller image, I saved some arbitrary data that helps...

Machine Translated by Google



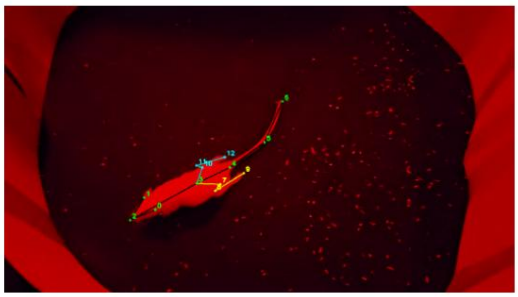
you can insert the cut out image back into the original image. In my case, this data
is the coordinate of a point that marks the upper left corner of the crop in the original image. The
information needed for re-fitting, such as the dimensions of the cut rectangle
can be extracted from the dimensions of the cropped image. The post-processing of the cropped image is
is necessary to be able to merge the produced images into a video format
data, this is not hindered by the different sizes of the cropped images.

As a final attempt, I subjected my images to a new kind of pre-processing, color-
using SAM. After I calculate a mask that represents the rat,
I set the values of all pixels outside this to zero, thus generating a dark image,
where only the rat is visible. My hope was that by filtering out the background, the model would
learns the rat's specific traits more effectively, making more accurate estimates
then give.

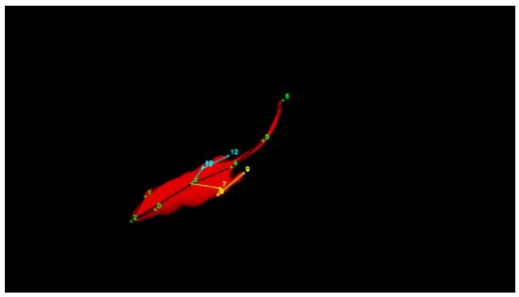
I combined the aforementioned pre-processing during my research, although in a careful way.
A major disadvantage of these methods is that they can only work on data that
than the one they were trained on. This means that a model trained on dark images will
Proper operation can only be expected for video that has been pre-processed and
every frame of film is darkened. A model trained on cropped images has an extra
would have required a step: after splitting the input video, searching for each frame
and cut out the rat. Figure 3.7 shows a sample output image, which is-
It is clear what specific procedures are required to generate their input pairs, but
Small differences in estimates can also be observed.

Machine Translated by Google

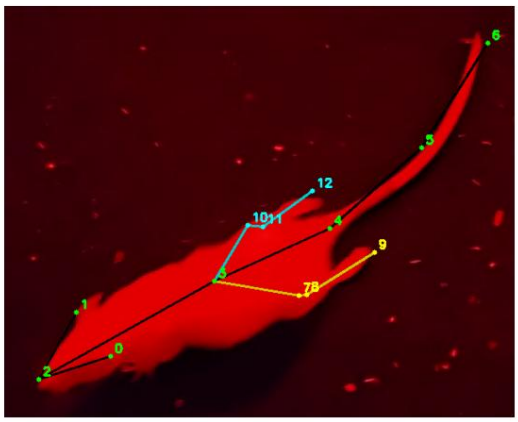




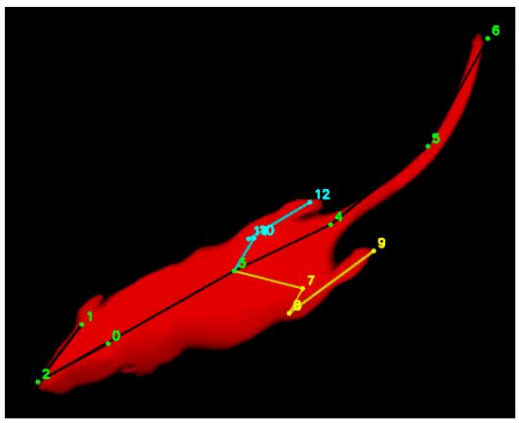
YOLOv8l-pose full picture.



YOLOv8l-pose full darkened image.




YOLOv8l-pose cropped image.




YOLOv8l-pose cutout darkened picture.

Figure 3.7. Comparison of YOLOv8l-pose model variants.

Augmentations

Before starting the lesson, a modification must be made to the Ultralytics package. in the core library. During the writing of the thesis, the ultralytics package I used core version 8.3.88. I have added the script that runs the model training with augmentation methods that were not built in by default. The I wanted to increase the generalizing ability of the model in this way. This means that In the program files, you can find the script scripts/fix/augment.py must be overwritten with the file with the same name in the virtual environment script. The path to this is: .envs/venv_yolo_v12/lib/python3.10/site-packages/ultralytics/data/ augment.py. If this override does not occur, models trained on the same data may make different predictions. will be made compared to mine. The augmentations that are included as an addition are The albumentations package used natively by ultralytics provides additional transformations They include the following functions, their description can be seen in Table 3.2, and their effects are

Machine Translated by Google



in Figure 3.8. The use of augmentations increases the generalization ability of the model by providing input images with a variety of visual variations. Augmentation These help the model not to adhere too strictly to the given training data. characteristics, but rather robustly handle natural variance, thus providing more accurate provide position estimation even in unfamiliar environments (e.g. with a different type of camera) recorded videos).

| Augmentation | Description |
|---|---|
| A.ChannelDropout | Randomly resets one or two color channels, so Certain parts of the image may disappear, which increases robustness against loss of color information. |
| A.ColorJitter | Randomly adjusts colors (brightness, contrast, saturation, hue) to give the model different looks. manage lighting conditions better. |
| A. Emboss | It contours the edges of the image, highlighting the shapes. |
| A. Equalize | Optimizes image contrast by smoothing the histogram, making details more visible. |
| A.HueSaturationValue | Randomly modifies hue, saturation and brightness, mimicking different lighting conditions. |
| A.ISONoise | It adds noise to the image (ISO-like noise), which is a weak simulates lighting conditions. |
| A.InvertImg | Inverts the colors of the image, completely reversing the color values. |
| A.MultiplicativeNoise | It applies a random multiplier to the pixels, different for each channel, thus creating small color distortions. |
| A.RGBShift | It modifies the values of the red, green, and blue channels separately, causing subtle color distortions. |
| A.RandomBrightnessContrast | Randomly changes the brightness and contrast of the image to reflect different lighting conditions. model it. |
| A.RandomFog | Adds a foggy effect to the image, reducing sharpness and visibility. |
| A.RandomToneCurve | Modifies the tone curve of the image, changing the color temperature and tone distribution. |
| A.Sharpen | Sharpens the edges of the image, highlighting details. |
| A.ToSepia | Converts the image to a sepia (warm brown) tone, imitating old or warm-toned photographs. |

Table 3.2. Applied image processing augmentations and their effects.

Machine Translated by Google



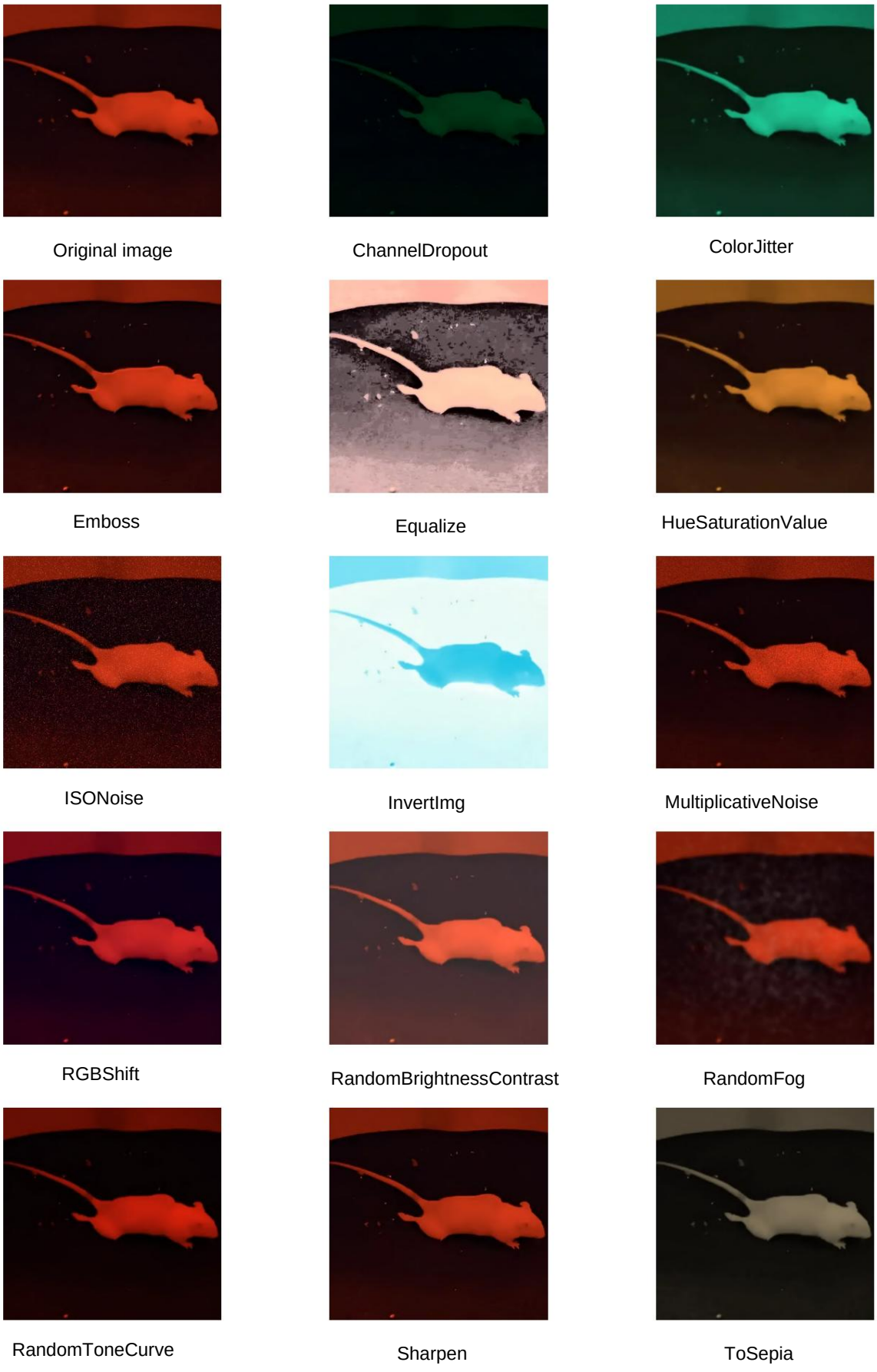



Figure 3.8. Additional augmentations shown on a sample image.

Machine Translated by Google



Teaching

Training a pose estimation model is easy with the Ultralytics library with its built-in function. Source code example 3.4 demonstrates using the YOLO class. Here's how I trained my actual used model:

```
1 from ultralytics import YOLO
2
3 def train_model ( data_yaml_path , epochs , save_path ,
      name_of_future_model ) :
4     model = YOLO ('yolov8n - pose .pt ')
5
6     model . train (
7         data = data_yaml_path ,
8         epochs = epochs ,
9         imgsz =640      ,
10        save_period =5          ,
11        batch =8     ,
12        degrees =15       ,
13        flip =0.5             ,
14        augment = True     ,
15        patience =50         ,
16        project = save_path ,
17        name = name_of_future_model
18    )
```

Source code 3.4. Training the YOLOv8 model.

For teaching, you need to provide the configuration file that points to the language you want to learn. Several important parameters can be specified to fine-tune the model performance:

- epochs - the number of training cycles. One cycle means that the model is trained one-times the entire training data set. In the case of three cycles, the model sees all training images three times. More epochs generally lead to better learning means, but it also takes longer.

- imgsz - the size of the input images. The images are converted to a square of imgsz pixels. are resized before teaching. The YOLO infrastructure monitors that the images should not be distorted when resized, but should retain their original aspect ratio. If necessary, the image is padded to achieve the correct size. This helps

Machine Translated by Google



uniform processing of the model and speeds up calculations. Larger image size
it may mean more detail, but slower teaching.

• batch - This determines how many images the model processes at a time.
For example, a batch of eight means that each step of the teaching
In this case, it takes eight images into account for weight update. The larger batch size
it may be faster, but it requires more memory usage.

• degrees, flipud, augment - various augmentation techniques that increase
This increases the generalization ability of the model.

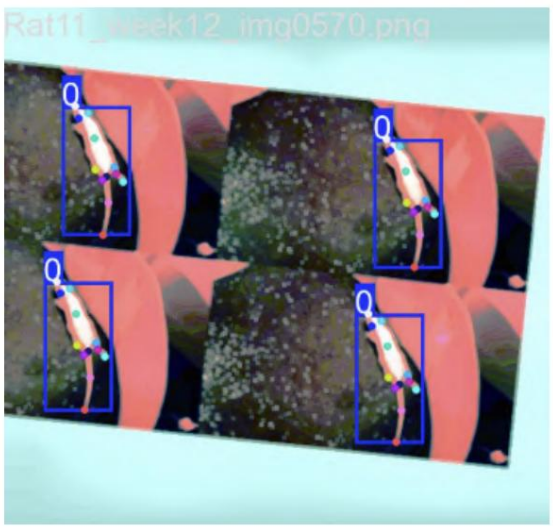
– degrees=15, for example, rotates images by a maximum of 15 degrees.

– flipud=0.5 means there is a 50% chance of flipping an image vertically.

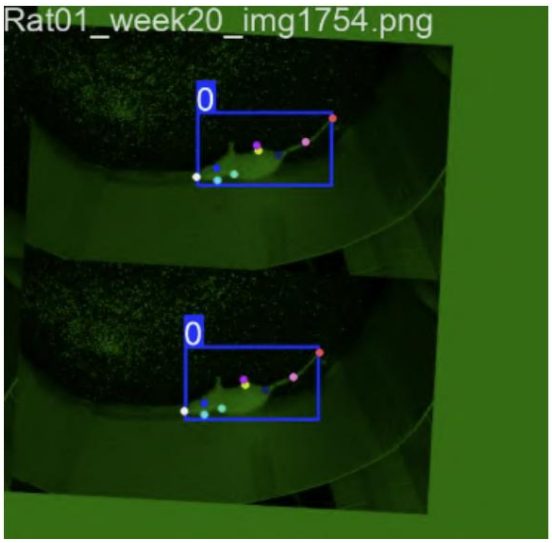
– augment=True turns on all available distortions (e.g. brightness,
rotation, mirroring) so that the model does not only learn on the original images,
but also on more diverse examples. This helps to make it more general
cry on unknown images. Illustration in Figure 3.9.

• patience - This is the early stopping parameter. If no response is received for this many cycles,
If the validation performance improves noticeably, then the training will be automatically
stops, saving time and resources.

• save_period - how many epochs to save the model state every.



Contrast augmentation.




Color channel augmentation.

Figure 3.9. Teacher data augmentations.

Machine Translated by Google



Several files will be automatically created in the save_path folder specified during the tutorial. It turns out that the following are important for my application:

- weights folder, which contains the weights of the trained model with .pt extension-
  cell.
- results.png, results.csv, which contain metrics and statistics
  files.
- train_batch.jpg images to check the applied augmentations
  presence.
- val_batch_labels.jpg and val_batch_pred.jpg images, where you can compare
  The performance of the model shown in the validation images with the measurement base.

Teaching can also be controlled from a Streamlit-based graphical interface, where the user can badon you can specify the path to the configuration file, the name of the trained model as ID, the number of teach cycles, and the output folder. A single button-pressing the button starts the process, which is called by the function call in source code fragment 3.5. represents:

```
1 train. train_model ( yaml_input , yolo_epochs , yolo_output , yolo_name )
```

Source code 3.5. The function called by the graphical interface to train its own model.

This user-friendly method allows biologists to be able to to train a new YOLOv8-pose-based model, if deemed necessary and available with the previously defined initial data structure.
The full code for the component can be viewed in the train.py script.

## 3.6. Data processing

The second main component of my application is the actual input data processing. It deals with. Its operation is enclosed in the pipeline.py script, which is used by the user It also uses a user interface. The main execution function in this is called execute(). This includes the invocation of the following subcomponents. The meaning of its parameters is In explanatory table 3.3. you can see:

Input video and output directory are required; boolean values are are passed as false by default.

Machine Translated by Google



| Parameter name | Explanation |
|---|---|
| input_file | The absolute path to the input video file. |
| output_folder | The absolute path to the output directory. |
| save_initial_predictions | Boolean value that determines whether the YOLO pose is model-estimated results are displayed as images in the program save it according to the drawing procedure in the Ultralytics library. |
| save_original_predictions | Boolean value that determines whether the YOLO pose model-estimated results are displayed as images in the program whether to save it according to specific procedures. |
| save_corrected_predictions | Boolean value that determines whether the YOLO pose Should the program save the results estimated by the model as images after correction according to specific procedures? |
| save_masks | A logical value that determines whether the program Rat masks estimated by SAM model as images should he save it? |
| save_skeletons | A logical value that determines whether the program should display the skeletons built on the rat mask as images. should he save it? |
| save_original_vid | A logical value that determines whether the save_original_predictions images from the program whether to save it in video format. |
| save_corrected_vid | A logical value that determines whether the save_corrected_predictions images from the program whether to save it in video format. |
| save_plots | A logical value that determines whether the program whether to save the graphs and the values they are based on lists that visualize the rat's movement rate and influence the decision. |

Table 3.3. Explanation of executive function parameters.

As an output, classify the rat in the video into the five described movement levels one of them and returns this decision, or depending on the set logical values saves various data as a side effect. Depending on whether it is long enough or not input video (see the restriction mentioned in section 3.6.1), but regardless of the program The result of running ram is the following file structure. The files contained in it The contents of the folders depend on the settings before running.

Machine Translated by Google



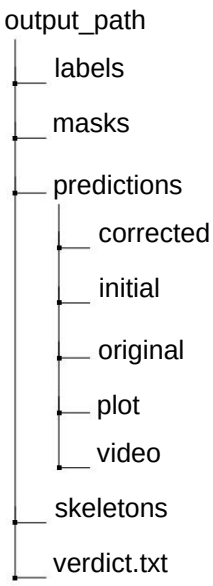



Figure 3.10. Output library structure.

The verdict.txt is only created upon successful execution, this is where the video is saved.
name of the final classification in text format. Hereinafter referred to as subsections
I will provide a more detailed insight into the components of the principal component within the framework of

### 3.6.1. Input fragmentation

The first step in processing is to divide the input video into frames. To do this,
I am using the OpenCV library, which allows me to smooth the frames of a video file.
reading out the video and saving it as an image. With cv2.VideoCapture
I load the video, then I go through the frames one by one, and the data is homogeneous.
In order to preserve the quality, each frame was resized (by default to 1280x720 resolution)
and saved in PNG format to a temporary folder.
Resolution setting allows processing of videos of different resolutions
also, thus expanding the set of videos on which my application can be run. The
The running time of the program is significantly affected by the size of the input images, so the above
This procedure may also cause an increase in performance.

I used the Lanczos4 interpolation method to resize the frames,
which is a high-precision, sine-based interpolation technique. Its advantage over other popular
compared to other methods (nearest, bilinear, bicubic) to achieve excellent sharpness and
ensures detail retention in resized images. Despite its higher computational requirements
This quality advantage is particularly beneficial in machine vision tasks, where the input

Machine Translated by Google



The quality of the data has a direct impact on the performance of the model. In my program
I chased this detail retention advantage because of the rat's feet.

Although the YOLO model can run directly on video files, this type of
In case of inputs, the returned results do not include the original frame
cat. In contrast, if we run it on an image folder, then each estimate can be associated with
original image as well. Since further steps in my application (such as masking and visualization)
require these original images, I chose the slower but more robust solution
I chose.

The input video must meet the following constraints: At least
It must contain 250 frames, otherwise the processing will be interrupted. The idea behind this
The reason behind this restriction is that my program wouldn't be able to convert a video that short anyway.
can provide a reliable estimate.

Once the cutting is complete, the system logs the successful processing and returns
gives the path to the temporary image folder that will be used in the subsequent steps.
The full code for the component can be viewed in the slice.py script.

## 3.6.2. Predicting key points

This program fragment receives the frames extracted in the previous module, including the content-
temporary folder path. Using the pre-trained model, it estimates
key points located on the rat's body. This step is relatively simple to imp-
can be omitted, as the Ultralytics library provides a high-level YOLO class.

Predict and Results

The model can be run by calling a single predict method. My job is to...
was to connect the device with the previous and subsequent parts of my program in the appropriate
by specifying parameters:

- source - The path to the selected input data. In this case, the source
  temporary folder containing input video frames from the previous module
  output.

- conf - The confidence threshold. Only those detected
  holds values whose certainty exceeds this value. For example
  If conf=0.5, the model only returns matches with a probability greater than 50%.
  give it back.

Machine Translated by Google



- batch - How many images to process at once. A higher value speeds up the processing.
  if enough memory is available.

- device - Select the computing device. Its value can be "cpu", 0, or other
  number for multiple GPUs. If 0, the model uses the first available GPU.
  Since I developed the program on the NIPG13 server, I had access to
  powerful graphics cards, reducing runtimes.

- verbose - Boolean value, if true, detailed logging during execution
  displays information on the console, which can be useful for debugging or
  during development.

The model returns a list of Results objects. The stream
It is possible to influence the operation of the predict method with the parameter, because if this
the default false logical value is converted to true, then memory is
returns only one Results generator. Lists are data structures that
which generate all the elements in advance and store them in memory. In contrast, the
generators are lazy evaluated, meaning that elements are only created when
they are actually needed, for example during iteration. This behavior is particularly
advantageous when processing large data sets, as generators require significantly less memory
They require a lot of memory. However, their disadvantage is that they are not indexable and are usually only
can be walked on once. This disadvantage prevented the generators from providing
I want to take advantage of the advantages of this, because the modules of my application use the
results returned by the YOLO pose model multiple times. Thus, a single pass
was not possible. The Results objects contain a lot of useful information-
for further analysis. The following list presents the data used in the program
data members:

- keypoints – coordinates and confidence values of keypoints detected in the image.

  Important data members of a Keypoints type object:

  – xy – a Tensor of size (N, K, 2), where N is the number of detected objects,
    K is the number of keypoints. Each keypoint consists of (x, y) coordinates.

  – cpu() – copies the tensor containing the key points to CPU memory.
    Useful if the prediction was done on the GPU, but we want to process it on the CPU
    the data.

Machine Translated by Google



– numpy() – converts the tensor to a NumPy array, which is compatible with many other with processing library (matplotlib, opencv).

• boxes – the bounding boxes surrounding the rats.

Important data members of a Boxes type object:

– xyxy – containing the boxes in the format (xmin, ymin, xmax, ymax) Tensor.

– cpu() – copies the tensor containing the bounding boxes to the CPU.

– numpy() – converts the tensor to a NumPy array.

• path – the path to the input file from which the prediction was made.
It can be used to find out which frame a given Results belongs to.
object.

The output of the YOLOv8-pose model I use therefore directly contains-
za the key points, so no further processing is required to use them
these for the next steps. Replacing the specific model in case of dissatisfaction is easy. After
training a new model, you need to replace it in the line in source code 3.6.
the model path passed to the YOLO class.

```
1 model = YOLO ("/ home / dalloslorand / YOLO_lori / runs / pose / full_v8 /
      weights / best .pt")
```

Source code 3.6. Specifying your own model.

The full code for the component can be viewed in the predict.py script.

### 3.6.3. Mask production

Masks

As an introduction to the module, I would like to explain what an object is.
mask. This is a visual representation that hides the individual objects in the image.
The mask contains all pixels that are
which belongs to a given object while setting the background pixels to zero
(see Figure 3.11). This allows objects to be separated from their surroundings.
Object masks are widely used in image processing.

Machine Translated by Google



and in computer vision, for example in object recognition, segmentation,
in and tracking. Masks can be used to easily isolate and
individual objects can be identified, which is essential for accurate image analysis and
for automated decision-making. In later phases of my program, the rat mask
I will use it for various calculations.

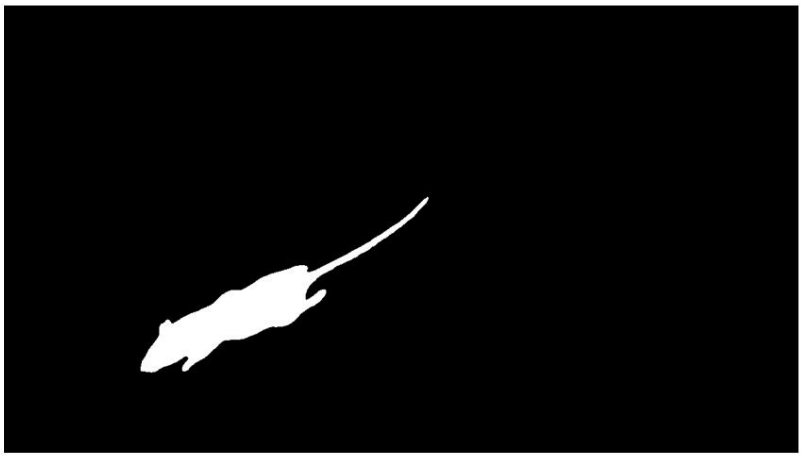


Figure 3.11. The rat mask.

Segment Anything Model

In my thesis, the Segment was used to produce masks representing the silhouette of a rat.
I chose Anything Model (SAM). SAM is an advanced image processing model,
which was developed by Meta AI specialists to be able to automate
to segment any object in the input image. The model aims to
minimizes the need for human intervention while enabling rapid
and accurate segmentation in a wide range of applications. The SAM model works in a
is based on a pre-trained network that can recognize and segment individual
objects without the need for special annotation. Other alternatives,
segmentation models such as FastSAM or Ultralytics YOLO are also
would have been accessible and easy to integrate, however, their accuracy, especially
on animal-based (rat) images, the studies did not prove to be sufficiently reliable
during.

Similar to Ultralytics' rich YOLO model set, SAM models are also available
There are three versions, but the difference between them is more related to the size and complexity of the networks.
The more complex a model is, the more accurate results it can produce.
but also requires more resources. The models in order of magnitude:

Machine Translated by Google



- SAM-H (huge)
- SAM-L (large)
- SAM-B (base)

I started planning my program with the SAM-H model, but I was forced to change it. I switched to the SAM-B model because it turned out that my original choice was impractical. After the switch, it creates noticeably faster, but also less accurate masks. I was able to generate it, so I had to adjust my instruction method.

Prompt search

During the operation of the module, you will receive a list of Results objects and all element proceeds as follows: It uses the estimated spine and tail points as foreground points. A background is determined by a proprietary algorithm points, and using the three points as SAM instructions, produces an estimated mask on the rat in the picture.

The algorithm for calculating the background point is based on the width of the given image. ten percent, and searches for a point at this distance from the estimated ridge point first to the right, if it falls outside the boundaries of the image then to the left, finally it checks whether this point is inside the bounding box surrounding the rat. If it is, then it increases the percentage of the distance by five, then try again. The point is attempted ten times find, if not, then only the above-mentioned spine and tailbone points use it without background points (thus reducing the chance of successful mask calculation). The The presented algorithm can be seen in Figure 1. Algorithm.

The module returns a list of masks while iterating over the Results list. which is passed on to the next program section at the end of its operation. The mask list Its elements are of type numpy.ndarray, which corresponds to a binary mask. Its size is matches the size of the original image, with each pixel value replaced by a 0 or 1 depending on whether the rat is under the position in the original image, or If SAM cannot find a mask for an image, I replace it in the list with a None value, which is handled appropriately by the other modules.

Machine Translated by Google



Algorithm 1. Calculation of SAM negative prompt for rat mask.

```
Require: kp3_point = (kp3x, kp3y), bbox = (xmin, ymin, xmax, ymax),
imgh, imgw, initial_d_perc = 10, increment_d_perc = 5, max_iterations =
    10
current_d_minutes ÿ initial_d_minutes
for attempt = 1 to max_iterations do
    d ÿ round(current_d_minute · img_w)
    Wednesday ÿ kp3y
    negx ÿ kp3x + d
    point_valid ÿ true
    if negx < 0 or negx ÿ imgw then
        negx ÿ kp3x ÿ d
        if negx < 0 or negx ÿ imgw then
            point_valid ÿ false
        end if
end if
    if point_valid then
        inside ÿ (xmin ÿ neg_x < xmax) and (ymin ÿ neg_y < ymax)
        if not inside then
            return [round(neg_x),round(neg_y)]
        else
            current_d_minutes ÿ current_d_minutes + increment_d_minutes
        end if
    else
        current_d_minutes ÿ current_d_minutes + increment_d_minutes
end if
end for
return None
```

The full code for the component can be viewed in the mask.py script.

### 3.6.4. Filtering film frames

Sample input videos and future live input videos intended for the program were also recorded in a specific way. The perspective of observation is not a fixed, The footage was taken with a hand-held camera, which was used for the top-view camera. It also mixes with a small amount of side view, with noticeable tremors. It happens that the camera view follows the rat, always keeping it in focus, and the zooming in and out of what the camera sees. The phenomena listed are all factors that may interfere with the accuracy of measurement and classification, introduce noise into the system. Noise caused by camera movement is described in the next subsection. I tried to filter it out using a method. The noise introduced into the recording by zooming in was filtered out by a I tried to mitigate it in a later module.

Machine Translated by Google



### Optical flow

Optical flow is a fundamental method of computer vision,
which is used to estimate pixel movements between two consecutive frames.
The aim of the technique is to determine how objects in the image change
positions in time, i.e. how pixels "flow" from one frame to the next.

Optical flow is often used in video-based analytics, especially
in cases where a given object (such as an animal or vehicle, for example)
The movement of a rat (e.g. a rat) must be separated from background noise or camera movement.
In the system I developed, optical flow helps decide whether the pat-
has actually moved relative to the background, so that those frames where
There is no significant movement. With this processing, not only the quality and accuracy of the data are improved,
improves, but we also lighten the burden of the upcoming program details, because
they have to calculate on fewer frames of film.

#### Frame filtering

To implement the filtering, I put together an algorithm that iterates through
on the mask of all images, while building a dictionary object whose keys are
the path to the currently examined image (I extract this from the estimates of the YOLO pose model),
and its values are "discard" or "keep", depending on how the
algorithm.

The logical value taken as the main condition of the filtering compares the rat mask opti-
kai flow with its background. I calculate a difference between the two, which if greater,
as a threshold value I define, then I keep the given frames
(flow can always be calculated between two frames). Optical flow calculation
I use the cv2.calcOpticalFlowFarneback function, which is the Farneback-
implements algorithm [26].

I implement this with a state machine solution (algorithm 2), which has two states:
pot switches between: rat_moving and rat_still. The state transitions are handled by a buffer and a
tolerance counter: the algorithm does not react immediately to a single deviation
but only if the pattern persists for a long time. This helps to filter out
random or irrelevant fluctuations in movement patterns. The buffer is temporary
It stores the "contradictory" frames, which are then used retrospectively.
retains or discards, depending on the direction of the state change. This

Machine Translated by Google



the approach results in more reliable and accurate decision-making, especially
for noisy or poor quality videos.

After creating the dictionary containing the decisions, I iterate through it and create new
I create YOLO estimates and mask lists that only include those elements
from the original lists, where the image corresponding to the element in the decision dictionary
value is "keep". Finally, these two filtered lists are returned by the module and used
the next model, building skeletons.

The complete code for the component can be viewed in the opticalflow.py script.

Machine Translated by Google



Algorithm 2 State machine-based decision based on optical flow.

```
Require: P reds, M asks, Tf low
BUF_SIZE ÿ 25, T OL ÿ 5, Decisions ÿ {}, Buffer ÿ [], T olCount ÿ 0
State ÿ "rat_still", P revM ask ÿ None
for all paths in P reds; i = 0 to |P reds| ÿ 1 do
Mask ÿ Masks[i]
    if M ask = None then
        Decisions[path] ÿ "discard", buffer deletion, T olCount ÿ 0, P revM ask ÿ None
        continue
end if
    if P revM ask ÿ= None then
        calculation of rat flow, bg flow
        Diff ÿ rat flow - bg flow
        if State = "rat_moving" then
            Decisions[path] ÿ "keep"
            if Diff ÿ Tf low then
                into buffer ( path , "discard") , T olCount ÿ 0
            else
                into buffer ( path , "discard") , T olCount++
                if T olCount > T OL then clear buffer, T olCount ÿ 0
                end if
            end if
            if buffer is full then
                the status of all buffered images should be retroactively set to "discard", State ÿ "rat_still", buffer flush
            end if
        else if State = "rat_still" then
            Decisions[path] ÿ "discard"
            if Diff > Tf low then
                into buffer ( path , "keep") , T olCount ÿ 0
            else
                into buffer ( path , "keep") , T olCount++
                if T olCount > T OL then clear buffer, T olCount ÿ 0
                end if
            end if
            if buffer is full then
                the status of all buffered images should be retroactively set to "keep", State ÿ "rat_moving", buffer clearing
            end if
        end if
    else
        Decisions[path] ÿ "discard"
end if
P revM ask ÿ M ask
end for
if buffer is not empty then
If "rat_moving" ÿ all "discard", otherwise "keep"
end if
return Decisions
```

Machine Translated by Google



### 3.6.5. Toe Key Point Correction with Synthetic Skeletons

Detecting the toes of rats is particularly challenging due to their rapid movements.
on images with noise and blur. Key point estimation of the YOLO model
in the case of such difficult-to-remove body parts, it is not always reliable, often
returns correct positions. To address this problem, a skeleton-based correction algorithm is used.
I introduced a step that attempts to calculate the medial axis based on
to specify the position of the toes.

The medial axis is a binary shape (rat mask) with internal, topological spines.
means: a set of pixels that are equidistant
from at least two different boundary points of the shape. Starting from the binary mask,
I use the skimage.morphology.skeletonize function to generate the medial axis-
t, which represents the midline of the rat's body with a thin, one-pixel width
in a concise form.

During correction of the original 9th and 12th index toe points estimated by the model
the algorithm finds the endpoints closest to them in the generated skeleton
zone (algorithm 4). To do this, I first calculate the dimensions of the rat's bounding box
a dynamic search radius, which is a fixed percentage of the diagonal of the box
derived from the value of (Algorithm 3). This approach allows us to separate
proportionally even for rats of different sizes or recorded from different distances
use an adaptive search range.

If the toe point is located within a circle with a dynamically calculated radius centered at
if the skeleton has an endpoint, then I select this, or in case of multiple hits, the average of them
as a new keypoint. Thus, the original estimates are overwritten on a topological basis
with a more probable position, which is expected to improve the accuracy of the entire system. The fo-
The model can also be optionally visualized, where the various key points, skeletons and
decision circles are marked with different colors on the original images. An example
A visualization can be seen in Figure 3.12.

In this phase, the first mask from the list filtered by the previous module is
I calculate a list of skeletons and then run the correction algorithm on a thousand
rhythm, and finally I produce a dictionary for further analysis, in which the keys are
are the paths to the original images, and the values are the values estimated by all YOLO poses
keypoint coordinates for the given image, but with the corrected foot coordinates. The method
The aim is not to recreate the entire keypoint system, but to create a targeted, small-scale

Machine Translated by Google



correction that helps achieve more accurate results on the toes, thereby supporting the later motion analysis stages.

The full code for the component can be viewed in the skeleton.py script.

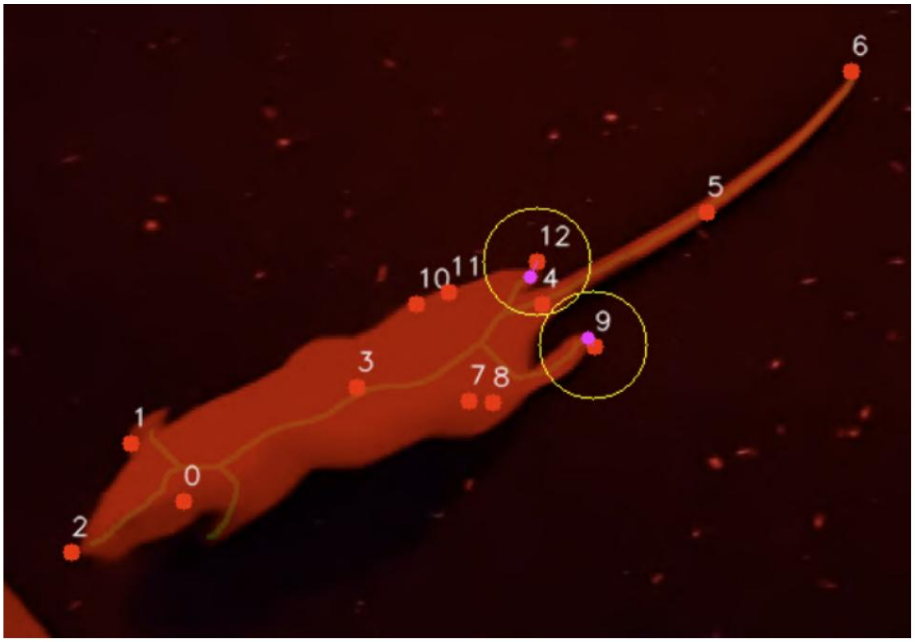



Figure 3.12. Skeleton and keypoint derived from the binary mask of the rat Visualization of repair test.

Algorithm 3: Determine the search radius based on the bounding box.

Require: Presult (prediction result object), Rdefault (default radius), Sfactor (scaling factor)

bbox ÿ Presult.boxes.xywh[0] ÿ Get bounding box [cx, cy, w, h]
if Presult contains bounding box and bbox is not empty then
  w ÿ bbox[2] h ÿ ÿ Width
  bbox[3] ÿ Height
  diagonal ÿ ÿ w2 + h 2
  calculated_radius ÿ round(diagonal · Sfactor)
  final_radius ÿ ensuring max(5, calculated_radius) ÿ Minimum 5 pixel radius
  return final_radius
else
  return Rdefault
end if

Machine Translated by Google



Algorithm 4 Toe correction based on nearby skeletal endpoints.

```
Require: T oeCoord = (tx, ty) (current toe coordinate), SearchRadius = R,
    Endpoints = {e1, e2, ..., em} (list of skeleton endpoints)
 1: if tx = 0 and ty = 0 then return                    ÿ Invalid input toe coordinate
 2:     None, [] 3: end if                              ÿ No correction, empty list

 4: if Endpoints is empty then                          ÿ No skeletal endpoints
 5:     return None, [] 6:                              ÿ No correction, empty list
end if
 7: NearbyPoints ÿ [] ÿ List for storing nearby points
 8: for all endpoints e = (ex, ey) in Endpoints do
 9:     distance ÿ (ex ÿ tx) 2 + (ey ÿ ty) 2 if          ÿ Euclidean distance
10:     distance ÿ R then
11:         Add to NearbyPoints list
12: end if
13: end for
14: ChosenPoint ÿ None
15: if NearbyP oints is empty then                      ÿ No nearby endpoints found
16:     return None, NearbyP oints 17:                  ÿ NearbyPoints is empty here
else if SizeOf(NearbyP oints) = 1 then                  ÿ Exactly one nearby endpoint
18: ChosenP oint ÿ NearbyP oints[0] 19: else 20:        ÿ Selecting the single point
sum_x ÿ                                                 ÿ Multiple nearby endpoints
0, sum_y ÿ 0
21:     for all point p = (px, py) in NearbyP oints do
22:         sum_x ÿ sum_x + px
23:         sum_y ÿ sum_y + py
24: end for
25: avg_x ÿ round(sum_x/SizeOf(NearbyP oints))
26: avg_y ÿ round(sum_y/SizeOf(NearbyP oints))
27: ChosenP oint ÿ (avg_x, avg_y) 28: end if            ÿ Selecting the average point

29: return ChosenP oint, NearbyP oints
```

### 3.6.6. Data analysis

Since this is a relatively more complex module, its operation is divided into subsections. I will illustrate this. In the usual order, the output of the previous program section receives as input (a dictionary containing corrected coordinates). After parsing Categorize the video on the scale described in the introduction to the paper.

#### Corrected labels

The key points contained in the input dictionary are saved by the program in an intermediate state, which is a useful starting point for further analyses during development. The

Machine Translated by Google



The saving process is done by iterating through the dictionary and using the image name.
va creates a text file with the same name, containing the YOLO format
writes the data, paying attention to the fact that the data is not normalized based on anything,
xel values. It only loads keypoint data from text files
into a list whose element is a pixel coordinate in (x, y) format
sublist.

Time series

A time series is a data sequence in which data is presented sequentially in chronological order.
appear. The purpose of time series data analysis is to reveal changes over time.
a lot of patterns.

I read the data of both legs from the list containing the coordinates, then each
I create two time series for each leg. In my thesis, the time series are based on the hind legs of the rat.
I define it based on the key points of the frames. For each frame I calculate the
the distance between the foot and the thigh, so each element of the time series is a change of this distance
This metric is a good indicator of the movement of the legs,
and its amplitude, and also serves as a basis for categorizing behavior. A
So for a given leg I create a time series illustrating this distance in pixel values,
and also a time series illustrating the same distance in a normalized state. The behavior
To analyze the evolution, we use the normalized time series, the normalization process is described in
I will explain in a later subsection.

I generate the time series using the ndarray type, they contain double values.
Similar to the corrected labels, I went down here too and then read them back to the me-
The saved file has the extension .npy, which is the standard NumPy binary file.
format and allows for quick reloading later.

Normalization

The raw (pixel) time series are Euclidean distances measured between the thighs and toes of the legs.
distances for each frame. Since these values are absolute,
They represent a fixed distance in the coordinate system within the image, therefore they affect
the distance is, for example, if the rat is seen closer in the recording (the distance
(clearly larger). In order to filter out the phenomenon, it is necessary to analyze the data

Machine Translated by Google



normalization based on size to make different
also rats taken from a distance.

Normalization uses the distance between the spine and the base of the tail as a reference length. This body length is a relatively stable value, proportional to body size. The
We divide the foot distances measured in each frame by this value, and we get
and the proportional distances.

This method ensures that the analysis is independent of the camera zoom-
or the body size of the rat, and the resulting time series describes the movement pattern represents, not absolute dimensions.

Z-score normalization and DTW

Z-score normalization is a scaling procedure that converts raw values into
from the mean and standard deviation of their own distribution into vertical ratios. All
We subtract the average of the given variable from each data point, then divide it by its mean. with , so the result will be a distribution with zero mean and unit variance. This means that some characteristics become comparable regardless of their original units of measurement or from their magnitude.

DTW (Dynamic Time Warping) is an algorithm based on dynamic programming, which is used to create a time series between two time functions (in this case, leg movement time series). DTW is flexible in handling time differences and
speed differences: it measures the similarity of curves by locally stretching or compresses rows to achieve the lowest possible distortion cost.

In my context, however, the characteristics of rodent leg movements are beyond showed a fine and noisy pattern: the amplitude of the movement in the different recordings and its frequency is highly varied, and noisy data points during DTW path matching did not provide a large enough difference between rats of different health status
Even Z-score normalization did not reduce the internal standard deviation sufficiently, so the The distance matrix obtained based on DTW did not lead to reliable clustering or for classification.

Furthermore, the internal workings of pre-made, “black-box” packages are not I considered it quite transparent, so I designed and implemented a processing procedure that I built it, every step of which I built with my own hands. This allowed
made sure that every computational phase corresponded to the experimental requirements.

Machine Translated by Google



### Time series characteristics

To implement the classification, I need the observed data in the time series.
features that can be used to distinguish the movement of a healthy rat
generated time series from a paralyzed rat. Logical idea is amplitude-based
distinction, since analyzing the feet of a healthy rat is almost certain to
the change in distance between the thigh and the foot occurs over a larger range than
in the case of a lame rat.

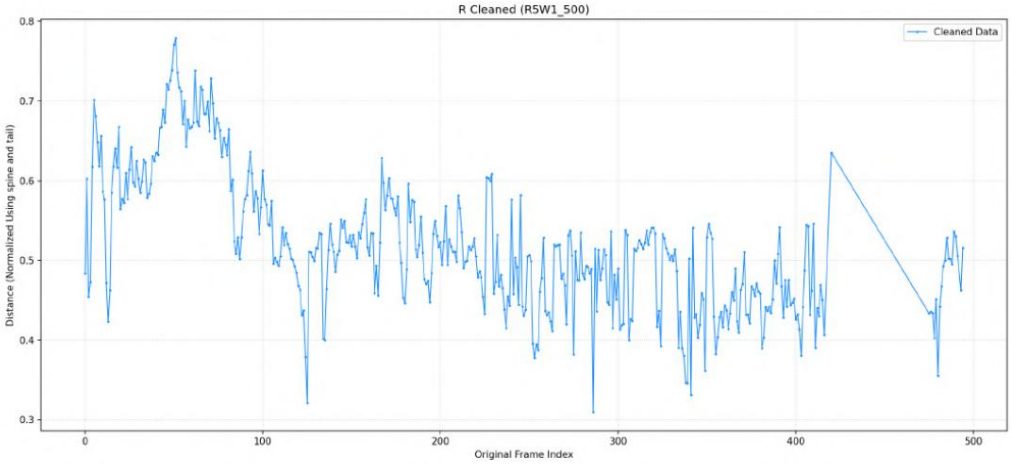
Before extracting features, the time series undergo a preprocessing. First step-
I discard invalid values (np.nan), which indicate that the norm
The calculated value was not calculated in the given frame for some reason,
for example, due to missing key points. As a second step, the dense values
I apply a mathematical transformation [27] to time series to make them smoother and more
time series. As a third step, I look for the factors that play a role in the smoothed time series.
indices of peaks and valleys, which indicate local maximum and minimum values
As a fourth step, I calculate the average of the time series using the aforementioned indices.
amplitude and their frequencies. As a final step, the two signals belonging to one video
I average the features of each foot, so a video is always described by two features, not four.

### Charts

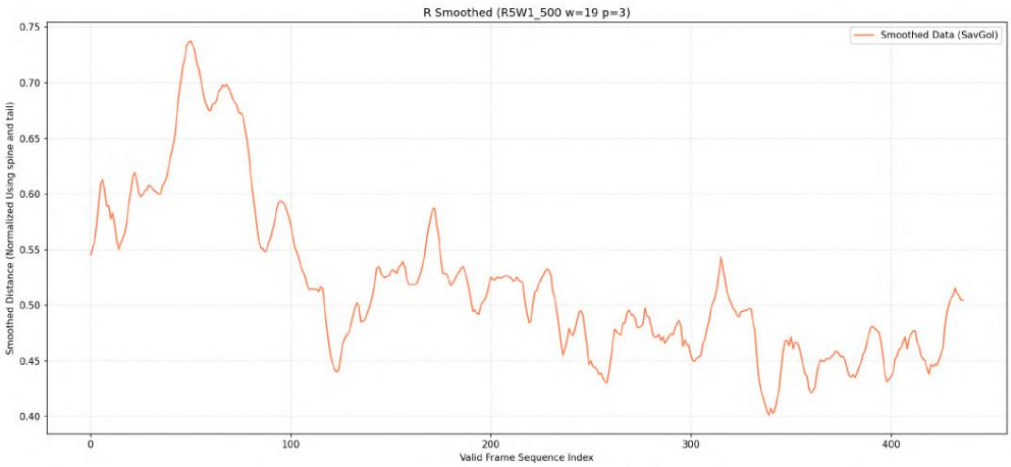
The time series in the previously listed compressed, smoothed and extreme value-marked states
can be saved at will, if the user selects the item after the program has run.
would be the process. In addition to the final classification, the user interface also provides a drawing
The smooth time series generated by the input video is applied to both legs, and the same
these are also for the sample video of the category chosen by the program. For illustration purposes,
Figures 3.13a, 3.13b and 3.13c show a time series in the various states.

Machine Translated by Google

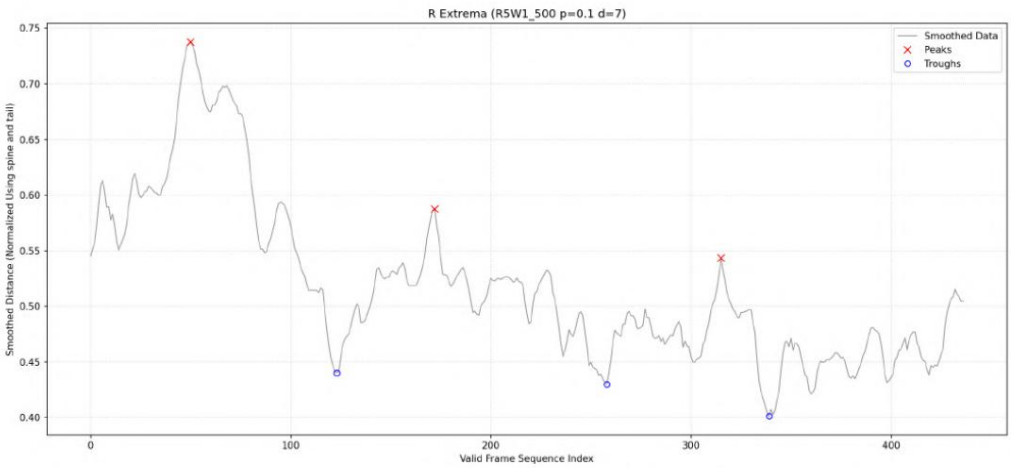





(a) Filtered time series.




(b) Smoothed time series.




(c) With local extreme values.

Figure 3.13. Time series of the right hind leg movement of a sample video after different processing steps.

Machine Translated by Google



Comparison

The production application always calculates time series only for the input video and the figures associated with them. The data of the reference classes is saved from the reloaded by the application when it runs. These are in the file structure shown in Figure 3.14. We store the main features of a given class in Json files, their content illustrated in source code 3.7. The saved states must be the latest contain time series data according to the method, which may have changed during development, or may change again during further development.

```
1  "crippled": {
2      "category": "crippled",
3      "amplitude": 0.26510135904167714,
4      "frequency": 0.007281553398058253
5  },
6  "barely moving": {
7      "category": "barely moving",
8      "amplitude": 0.22515096615898295,
9      "frequency": 0.010095389507154214
10 },
11 "intermediate": {
12     "category": "intermediate",
13     "amplitude": 0.2697913547147352,
14     "frequency": 0.009966777408637873
15 },
16 "almost healthy": {
17     "category": "almost healthy",
18     "amplitude": 0.24277992233672452,
19     "frequency": 0.011629546095291796
20 },
21 "healthy": {
22     "category": "healthy",
23     "amplitude": 0.24066798243544263,
24     "frequency": 0.0066566794710244195
25 }
```

Source code 3.7. JSON of amplitude and frequency values of reference categories its structure.

Machine Translated by Google



Exchange reference data

If the method of data analysis changes, it is mandatory to generate the
reference data according to the new procedure for fair comparison. This
the developer can do it in the following way:
The contents of the plots, predictions, and timeseries libraries shown in Figure 3.14 and
Delete the features_norm.json file after making a backup. If the reference
you would also replace the videos taken as a basis, you can do this in the videos folder, but video files
names should continue to comply with the system set out in subsection 3.3.2.
(videos related to a given category should have the category name). After that, run-
sa the execute_reference() function in the scripts/pages/pipeline.py script. This
possible from the console using interactive python, or by rewriting the script. This
creates the necessary graph data and '.json' file.
The full code of the component can be viewed in the evaluate.py and compare.py scripts.
in a bowl.

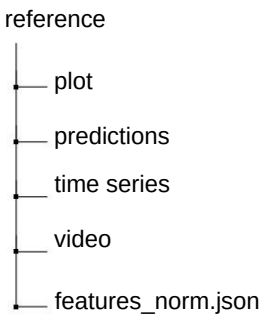



Figure 3.14. Structure of reference data.

## 3.6.7. Output Summary

The actual output returned by the program is a number that takes the value
two of the class to which we classified the rat in the input video
(1-5). I saved the video classification in text format for future retrieval.
This number is returned in section 3.6.6, but the program
Depending on the settings you specify when running the
presented data can also be saved in addition to all that the program
I explained earlier in the description:

Machine Translated by Google



Drawings

The estimated key points can be illustrated in the images in a unique style, such as
The accuracy of the model can also be checked; the cause of a detected anomaly can be investigated. The
The drawing style can be seen in the attached images in Figure 3.7, but can also be summarized verbally:
The original or corrected coordinates of the estimated key points are added to the rat.
with (it is possible to draw both versions), their serial number accompanies them above them
shifted to the right. I connect the adjacent keypoints based on my own choice, so
the change in the distance between the key points is more spectacular. The skeleton (not mixed-
(in accordance with the centerline axis calculated in section 3.6.5) its original color is black,
and the key points are green, but this can be influenced by which part of the body they belong to.
The points on the right foot and the lines connecting them are clear.
blue and the left leg is colored yellow to highlight these body parts as the project
the subjects of its investigation.

Video generation

The images described above can also be combined into a video format, where
You can give your own FPS (frames per second) measurement.
However, this requires that images of the appropriate category are also saved.
I designed my program this way because the facts observed in the video require deeper analysis.
I assumed that the images that make up it would also be necessary.

Charts

The time series mentioned in section 3.6.6 and their refined states are saved.
are available in order to understand and examine the program's decision.
The full code for the plots can also be viewed in the evaluate.py script.

# 3.7. Run options

## 3.7.1. The user interface

To implement the user interface of the project, a Python framework called Streamlit was used.
I used a web framework based on .NET. My choice was primarily due to the fact that
that Streamlit enables the development of simple, fast and interactive user interfaces
with minimal frontend knowledge. Everything can be controlled from Python,

Machine Translated by Google



so I didn't have to write any HTML or JavaScript. There are also a lot of built-in
offers elements such as sliders, text input fields, buttons, pop-up text boxes,
windows that can easily make your application interactive with a few lines of code
Matplotlib, OpenCV and PIL can be used to display the visualizations.
libraries, so I could display images and graphs directly on the page
to do.

Since the focus of the research is on image processing and data visualization algorithms,
was in many developments, it was important to create the UI (user interface) without
take a disproportionate amount of time out of the project. However, simplicity also comes at a price
There is: the interface customization is more limited than, for example, a full-fledged front-
end framework. So there is a compromise in terms of design and layout.
I had to knit. I accepted that I couldn't shape everything completely freely, but
But in return I got a very fast working application.

During the project, I worked on a remote server, using Streamlit to
user interface worked as a web application. So I didn't need
I accessed the program from a separate graphical interface on the server, via a browser. Important
I thought this would be possible, since my program requires a lot of computer power. I
thought it was rare to find a desktop computer that
has enough virtual memory to run my program, so the application
its users will probably run it from a similar environment to mine.

I made the user interface navigable with a side-lined menu, where
I have grouped the two main components of the program (sections 3.5 and 3.6) based on the
pages. When the program is run, Streamlit runs the index.py file, which
will immediately redirect you to home.py, from where you can explore further if you wish.
application.

### Protection against user errors

During development, I experimented a lot with methods to protect the app.
the intended operation of my application against the user's curiosity. The former is
The price of simplicity discussed in the past was also felt here, as it was only in a roundabout way.
you can restrict user access to controls. The final, and I think
The most common solution is that if any task is currently running on the page, then the
I make all visible controls inactive so that their values cannot be changed.
In addition, I will ask the user to wait with a warning.

Machine Translated by Google



the end of the run. Against the user navigating away from the current page
Meanwhile, unfortunately I couldn't take action within the framework. In such cases, in the background
my program continues to run with the parameters started from the interface, but in the interface this
it can no longer be tracked, for example when a page is reloaded.

### 3.7.2. Command line execution

Video analysis

In addition to the Streamlit-based graphical user interface, the program can be run from the command line
can also be run. This is intended to allow more advanced users to use it comfortably
the application, without UI.

The command line version is implemented using the argparse library, which
allows the user to control what is saved with various switches
during processing. The required parameters are the path to the input video, some
as the output folder where the results will be placed. In addition, there are several optional
switch is also available in the previously mentioned save options rules in the thesis
The full code can be viewed in the nogui_pipe.py script.

Data preparation and model training demo

In order to cover all basic functions without a user interface
executable, I created the nogui_prepare.py and nogui_train.py scripts-
also, which can be used to similarly generate a dataset or trainable model,
as from the user interface. The implementation is also done with the argparse library.
did.

## 3.8. Testing

### 3.8.1. Backend

The operation of the various components is due to the nature of the professional thesis.
basically, you can only approve or criticize it with your eyes. The actual accuracy
To measure it, it would be necessary to create labels for the input data that
my application is trying to avoid it. The performance presented in section 3.5.2.

Machine Translated by Google



Comparative tables, for example, are objective tests, since the data for these
I had the labeled measuring base.

The constraints in the program (such as the length of the input video) are tested
There was no possibility of further testing beyond baling.

### 3.8.2. Frontend

The user interface is based on the user documentation described in Chapter 2,
or can be tested based on what is presented in section 3.7.1. Basic tests:

- The predict button appears in the pipeline menu only if you have specified
  correct input video and output directory path.
- After pressing any button, it is not possible to perform the given task until the task is completed.
  to access the controls on the page. A warning message appears,
  which discourages the user from messing with the program, but this is not
  possible to disable.
- After the task is completed, a summary of the task results is displayed, indicating whether the task was successful.
  about.

## 3.9. Results

To train the models, I used 6000 annotated images, which were collected by the Semmelweis
Prepared by the staff of the Institute of Biophysics and Radiation Biology of the University of Medical Sciences
tek using DeepLabCut. I used this as the measurement basis (hereinafter GT,
as ground truth). I converted the annotated data to YOLO format (3.2.
source code) to formally match the data that a model predicts
created when serving.

To determine the accuracy, I evaluated the mean square error (MSE),
as follows. Only the relevant key points, i.e. [7, 9, 10, 12] (2.2.
(Table 1) I calculate the four-point difference between the GT and the model-estimated
its Euclidean distance, but only if the point in GT is not originally zero
coordinate. For a given image, the final MSE is the average of these squared errors for the given image.
for valid keypoints found on the map. Lower MSE means better accuracy
indicates. For a model, the final MSE is the average of the MSEs calculated for all test images.

Machine Translated by Google



This script (mse.py) is located in the scripts/benchmarks folder within the project directory.

The results are presented in Tables 3.4 and 3.5. These are

legend can be viewed here:

- full - the model was trained on full-size images
- cropped - the model was trained on cropped images
- dark - the entire image is dark except for the rat mask
- v8 - the model is based on YOLOv8l-pose
- v11 - the model is based on YOLOv11l-pose

Table 3.4. MSE based on all key points

| Model name # entries | Total MSE | Average MSE | |
|---|---|---|---|
| full_v8 | 4521 | 19.397779 | 0.004291 |
| full_v11 | 4522 | 24.590980 | 0.005438 |
| darkfull_v8 | 4510 | 24.532054 | 0.005439 |
| darkfull_v11 | 4518 | 28.378929 | 0.006281 |
| cropped_v8 | 4521 | 109.731735 | 0.024272 |
| cropped_v11 | 4521 | 132.152753 | 0.029231 |
| darkcropped_v8 | 4521 | 107.613112 | 0.023803 |
| darkcropped_v11 | 4521 | 127.968257 | 0.028305 |

Table 3.5. MSE based on thigh and foot

| Model name # entries | Total MSE | Average MSE | |
|---|---|---|---|
| full_v8 | 4500 | 30.386261 | 0.006753 |
| full_v11 | 4500 | 54.308615 | 0.012069 |
| darkfull_v8 | 4492 | 45.833419 | 0.010203 |
| darkfull_v11 | 4496 | 57.527402 | 0.012795 |
| cropped_v8 | 4499 | 147.725306 | 0.032835 |
| cropped_v11 | 4500 | 160.793024 | 0.035732 |
| darkcropped_v8 | 4500 | 130.749963 | 0.029056 |
| darkcropped_v11 | 4500 | 130.684783 | 0.029041 |

I also measured the variants in terms of sensitivity (recall), which measures whether

how well the model is able to find all the key points. This can be calculated using the following formula

I calculated:

$$\text{Recall} = \frac{\text{Correctly found key points}}{\text{Correctly found key points} + \text{Key points omitted but actually present}}$$

Machine Translated by Google



I calculated two types of acceptance methods: in the first measurement, I accepted if the input
The keypoint is closer to the point in the GT than 5% of the image height,
and for the second, 10% of the largest dimension of the building box was the criterion.
um. The final recall value was calculated by averaging all inputs. The
The entire process can also be tested with the scripts/benchmarks/recall.py script.
The results can be seen in Tables 3.6 and 3.7:

Table 3.6. Recall (image height 5%)

| Model name # entries Average Recall % | | |
|---|---|---|
| full_v8 4500 97.97 | | |
| full_v11 4500 96.78 | | |
| darkfull_v8 4492 96.75 | | |
| darkfull_v11 4496 96.24 | | |
| cropped_v8 4500 56.74 | | |
| cropped_v11 4500 54.91 | | |
| darkcropped_v8 4500 56.05 | | |
| darkcropped_v11 | 4500 | 55.56 |

Table 3.7. Recall (bounding box 10%)

| Model name # entries Average Recall % | | |
|---|---|---|
| full_v8 4500 97.69 | | |
| full_v11 4500 96.27 | | |
| darkfull_v8 4492 96.09 | | |
| darkfull_v11 4496 95.57 | | |
| cropped_v8 4500 56.25 | | |
| cropped_v11 4500 54.44 | | |
| darkcropped_v8 4500 55.54 | | |
| darkcropped_v11 4500 55.16 | | |

To run the scripts of the procedures, it is necessary to collect them into a library
the predictions of the models you want to compare, and the GT in YOLO format.
The scripts iterate through this library and summarize the results of the models.

The best performing model is the YOLOv8l pose, which was trained on the entire image,
without background masking.

Machine Translated by Google

# Chapter 4

# Summary

The application presented in the thesis is a complex and modularly structured
implements a system that serves video-based behavioral analysis of rats
through the examination of its movement patterns. During the development, the Python programming language was used
language and several modern libraries. I designed it with a user interface
I developed a new application that has two main parts. The pose estimation model
training and using the model for prediction. The application can be run locally
or with remote access on a server. Organizing and uploading your own data in the program's book-
before use. To generate annotated data, DeepLabCut
annotation software can be used.

Among the tools for machine vision and deep learning, YOLOv8-pose and Segment
I combined Anything Model (SAM) models. The model with the training component
own yolov8l-pose model can be taught, so the program can be reused, even in other
also for data.

The system processes the input video, key points and masks step by step.
produces parts, repairs the faulty key points with a synthetic skeleton, and then the resulting
performs the classification based on the extracted time series features into a five-level motion category
on a scale.

During the implementation, I paid special attention to reliability and flexibility.
solutions that ensure quality, so each main component of the program is separated
It is easily runnable, configurable, and has a graphical user interface.
One of the greatest strengths of the development is that it can filter out various noise sources.
Data refinement occurs at several levels. The software responsible for processing the video data is
detail for an input video can tell you on a scale of 1 to 5

Machine Translated by Google



how healthy the observed rat is moving. From now on, the processing steps are
It is possible to save the data generated in intermediate states using switches.

Through the graphical user interface, users with less IT skills can
users can also perform video analysis without creating annotations. The main to-
Further development opportunity is the accuracy of classification.

The full program can be downloaded via the OneDrive link below.
After logging in with your ELTE Faculty of Informatics organizational account:
https://ikelte-my.sharepoint.com/:f:/g/personal/xikvif_inf_elte_
hu/EpSsI66NMklFtpFJMDKVeS4BcsI9F2Dc7kjZR9uZmVZYHw?e=6dMjkA

## 4.1. Difficulties

My program is entirely based on NIPG (Neural Information Processing
Group) remote server, which provided insight into ssh encryption,
The concept of creating Apptainer environments and Jupyter Notebooks. For this reason, we
I had to use some special solutions, such as viewing pop-up windows.
was not possible, so IO operations were used to view the data.
This circumstance inspired the use of Streamlit. The features provided by Streamlit
not fully customizable, I held myself back by trying to customize it more
I wanted to customize it as much as possible.

It was a challenge to keep the versions of the packages I used consistent,
because often one package affected the version number of another during installation. The
I had to modify the local files of the ultralytics library I was using: the theme-
with the guidance of my manager, I would supplement the code with additional image distortion options,
which can increase the generalization ability of the pose estimation model.

Video recordings intended for behavioral observation are not a stable top or bottom
were not taken with a point-and-shoot camera, but rather handheld. For this reason, the video recording
There may be a basic tremor, or the person taking the recording is following the rat
while moving, sometimes zooming into the recording. The aforementioned facts are all
introduce noise into the system for keypoint analysis. Accurate classifiers
The aim of this research was to eliminate these. To filter out different types of noise
I used methods such as optical flow in the case of a moving camera,
and body normalization to ensure that a given distance is pixel-independent
be.

Machine Translated by Google



## 4.2. Development opportunities

Taught and operated on cut-out images included in the developer documentation
The potential of models can be realized primarily in the treatment of special cases, however
fine-tuning their operation requires further investigation. In extreme cases
diversity suggests that the application of such models is more robust to data
can become more effective with working methods. Models learned on full images-
revealing the reason for the lower performance compared to the system can contribute to
to improve its reliability and generalizability.

Comparing the performance of different segmentation models is a challenge for developers.
has been done to a limited extent due to time constraints. The documented
Based on the results (Table 4.1), there are alternatives such as FastSAM,
SAM2 or YOLO segmentation-optimized models that potentially
they offer faster operation, although the complexity of their integration and occasional new features
their training needs should be considered. In future developments, it may be worth considering these
comparison with the currently used solution.

| Model | Size (MB) | Parameters (M) | Speed (CPU) (ms/frame) |
|---|---|---|---|
| Meta SAM-b | 375 | 93.7 | 49401 |
| Meta SAM2-b | 162 | 80.8 | 31901 |
| Meta SAM2 | 78.1 | 38.9 | 25997 |
| MobileSAM | 40.7 | 10.1 | 25381 |
| FastSAM one | 23.7 | 11.8 | 55.9 |
| YOLOv8 with backbone | | | |
| Ultralytics YOLOv8n-seg | 6.7 (11.7x smaller) | 3.4 (11.4x less) | 24.5 (1061x faster) |
| Ultralytics YOLO11n-seg | 5.9 (13.2x smaller) | 2.9 (13.4x less) | 30.1 (864x faster) |

Table 4.1. Comparison of models by size, number of parameters and CPU speed based on.

Ultralytics just released its latest generation while I was writing this thesis.
rational YOLOv12 models, which did not yet support pose estimation. The
with architecture teaching and with training started from the weights of the pre-trained model
further improvements in accuracy may become available, so that subsequent corrections at runtime
could be accelerated by reducing steps.

Highest priority improvement opportunity from input videos
classification logic of extracted data. Biological research questions and related
The definition of statistical patterns is beyond the scope of this thesis, but the
necessary for research results.

Machine Translated by Google

# Acknowledgement

I would like to express my gratitude to my scientific supervisor, Dr. Gelencsér-
Anna Horváth for her continuous support, professional ideas and flexibility
which has been provided during the entire research so far, the creation of the professional thesis
before, not just under.

I am grateful to my father, Dallos Lorand, for occasionally doing things for me.
trusted me and also helped me brainstorm ideas; and my whole family
for their support and understanding.

I would like to thank Mrs. Róbertné Molnár (maiden name: Erzsébet Baranyai) for constantly making sure that I did not do this on an empty stomach while I was writing this thesis. Thanks to her, I was able to complete it significantly more efficiently.
my studies.

Machine Translated by Google

# Bibliography

[1] CS Hall. "Emotional behavior in the rat. I. Defecation and urination as measures of individual differences in emotionality". Journal of Comparative Psychology 18.3 (1934), 385–403. page doi: 10.1037/h0071444.

[2] RN Walsh and RA Cummins. "The open-field test: a critical review". Psychological Bulletin 83.3 (1976), 482–504. page doi: 10.1037/0033-2909. 83.3.482.

[3] Aptainer Contributors. Installation on Linux — Apptainer Documentation. Accessed: 2025-04-16. 2024. url: https : / / apptainer . org / docs / admin / main/installation.html#installation-on-linux.

[4] Alexander Mathis et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". Nature neuroscience 21.9 (2018), 1281–Page 1289.

[5] Guido Van Rossum and Fred L Drake Jr. Python tutorial. Vol. 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[6] Talmo D Pereira et al. "SLEAP: A deep learning system for multi-animal pose tracking". Nature methods 19.4 (2022), pp. 486-495.

[7] Arindam Sengupta and Siyang Cao. "mmpose-nlp: A natural language process sing approach to precise skeletal pose estimation using mmwave radars". IEEE Transactions on Neural Networks and Learning Systems 34.11 (2022), 8418–8429. page.

[8] Ultralytics. Ultralytics YOLO Documentation. 2025. url: https : / / docs . ultralytics.com (accessed 2025. 04. 16.).

[9] Yuxiang Yang et al. “An Effective Yak Behavior Classification Model with Improved YOLO-Pose Network Using Yak Skeleton Key Points Images.” Agriculture; Basel 14.10 (2024).

Machine Translated by Google



[10] Zehra. Neural network components. 2019. url: https : / / zerzavot . medium.com/neural-networks- components- a28c03d9dec (access date 2025. 04. 19.).

[11] Facundo Bre, Juan Gimenez and Víctor Fachinotti. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". Energy and Buildings 158 (Nov. 2017). doi: 10.1016/j.enbuild.2017.11.045.

[12] Benedek Gergÿ. Computer Vision / Machine Vision: what does Machine Vision mean, and what can it be used for? 2020. url: https : / / lexunit . hu / blog / mit - means - a - machine - view - and - what - can - it - be - used for/ (access date) 2025. 04. 13.).

[13] Xia Zhao et al. "A review of convolutional neural networks in computer vision". Artificial Intelligence Review 57.4 (2024), p. 99.

[14] Charles R Harris et al. "Array programming with NumPy". Nature 585.7825 (2020), pp. 357–362.

[15] Nikhil Ketkar et al. "Introduction to pytorch". Deep learning with python: learn best practices of deep learning models with PyTorch (2021), 27–91. page

[16] Gary Bradski. "The opencv library." Dr. Dobb's Journal: Software Tools for the Professional Programmer 25.11 (2000), 120–123. page

[17] Alex Clark et al. “Pillow (pil fork) documentation”. readthedocs (2015).

[18] Sandro Tosi. Matplotlib for Python developers. Packt Publishing Ltd, 2009.

[19] Jeff Reback et al. "pandas-dev/pandas: Pandas 1.0.5". Zenodo (2020).

[20] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". Nature methods 17.3 (2020), 261–272. page

[21] Stefan Van der Walt et al. “scikit-image: image processing in Python”. PeerJ 2 (2014), e453.

[22] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLOv8. Version 8.0.0. 2023. url: https://github.com/ultralytics/ultralytics.

[23] Alexander Kirillov et al. Segment Anything. 2023. arXiv: 2304 . 02643 [cs.CV]. url: https://arxiv.org/abs/2304.02643.

[24] Anaconda, Inc. Anaconda Documentation. https://www. anaconda. com / docs/main. Accessed: 2025-04-17. 2024.

Machine Translated by Google



[25] Mathis Lab. DeepLabCut Standard User Guide. https : / / deeplabcut .
github. io / DeepLabCut / docs / standardDeepLabCut _ UserGuide.html.
Accessed: 2025-04-17. 2024.

[26] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial
al expansion". Image Analysis: 13th Scandinavian Conference, SCIA 2003
Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13. Springer. 2003, 363–
Page 370.

[27] Ronald W Schafer. "What is a Savitzky-Golay filter? [lecture notes]". IEEE
Signal processing magazine 28.4 (2011), 111–117. page

Machine Translated by Google

# List of figures

Machine Translated by Google

Machine Translated by Google

# List of tables

Machine Translated by Google

# Algorithm list

Machine Translated by Google

# Source code list