Assessing Stem Cell Therapeutics in Murine Models

Giles Carlos, 54951701, gpcarlos@uci.edu
Nathan Gin, 67117388, nbgin@uci.edu
Alexander Nathaneal, 46645315, anathana@uci.edu
Vinh Nguyen, 50036019, vinhhn2@uci.edu
Owen Sitiabudi, 19975215, ositiabu@uci.edu

Github Repository: https://github.com/gilescarlos/Stats-170-Cap-Stone

1 Introduction and Problem Statement

Spinal cord injury (SCI) is a debilitating condition that impacts up to 300,000 individuals in the U.S. each year. As such, a significant focus of stem cell research is dedicated to developing therapeutics that could help bring affected individuals' livelihood back. Likewise, the tests and methods used to assess different therapeutics are incredibly important. The efficiency and rigor of these tests allow for discovery of novel therapies and help advance the research field dedicated to solving spinal cord injury. The ladder beam test is a behavioral method for assessing recovery following SCI in murine models. There are 3 types of mice: Knock-out, Wildtype, and Vehicle. The Knock-out mice have their CD44 gene removed to study its absent effect in recovery. The Wildtype are the control group while the Vehicle type have an additional injection. During the test, mice must walk across a ladder with fifty rungs. Injured mice are expected to have more missteps while recovering mice should have more plantar or sufficient steps. However, to properly utilize this test our sponsors must spend several hours analyzing hundreds of videos of mice completing the task in order to individually score each mouse's performance. Consequently, this tedious process tends to slow down the lab's workflow and some automation would provide a significant increase in efficiency. Streamlining the ladder beam task and its data analysis would allow for significant efficiency in retrieving data necessary for the advancement of SCI research and therapeutics. Our primary goal is to use appropriate machine learning algorithms and build off existing research to aid in the lab's data collection process for the ladder beam task. Secondly, we will assess the effectiveness of different treatments through appropriate statistical models and hypothesis tests.

2 Related Work

Several other labs have encountered this bottleneck in efficiency and have aimed to address this problem as well. One such solution is DeepLabCut[1], which is an open source software developed in Python that utilizes a deep neural network to track various body parts in multiple species across a broad collection of behaviors or physical assessments. Another study was able to utilize the software to conduct a comprehensive 3D gait analysis of mice after focal cerebral ischemia[2]. The authors concluded that using a previously trained data set for tracking mice movement, their own recordings of mice completing the ladder beam task, and the DeepLabCut neural network provides accurate and sensitive data to describe the complex recovery of rodents following a stroke. Other researchers have also been successful in building upon DeepLabCut

[1]Mathis, A., Mamidanna, P., Cury, K.M. et al. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci 21, 1281–1289 (2018). https://doi.org/10.1038/s41593-018-0209-y

[2]Weber, R.Z., Mulders, G., Kaiser, J. et al. Deep learning-based behavioral profiling of rodent stroke recovery. BMC Biol 20, 232 (2022). https://doi.org/10.1186/s12915-022-01434-9

and creating their own toolbox utilizing a convolutional neural network appropriate for their recording equipment[3]. However, these other labs had access to several camera angles and performed analysis on other behavioral tasks besides the ladder beam task. For example, in one study a lab obtained training data from three camera angles that could be utilized by DeepLabCut, two side-views and one bottom-up view. In our case, we only have a bottom-up camera view where the obstruction of the ladder rungs can be a source of inaccuracy.

3 Data Sets

We obtained our data from our sponsors at the UCI Stem Cell Research Center. Specifically, we received about a hundred videos of mice completing the ladder beam task and the corresponding manually labeled excel sheet corresponding to the videos.

Table 1: Example of First 10 Rows of Manually Input Data Set Obtained from Ladder Beam Task

Animal ID	**Total Good Steps**	**Total Bad Steps**	**Avg Good Steps**	**Avg Bad Steps**	**LB Score**	**Type**
1	276	24	46.0	4.00	92.0	w
2	280	20	46.7	3.33	93.3	k
3	255	45	42.5	7.50	85.0	w
4	228	71	38.0	11.83	76.3	k
5	267	33	44.5	5.50	89.0	v
6	279	21	46.5	3.50	93.0	v
7	276	24	46.0	4.00	92.0	k
8	226	74	37.7	12.33	75.3	w
9	227	73	37.8	12.17	75.7	k
10(11)	222	78	37.0	13.00	74.0	v

The post_injury data set has 30 rows and 7 columns where each column represents a single variable. The animal and type variables are categorical (nominal) and the rest of the variables are numerical (discrete). Total_good and Total_bad measures the quality of the steps of the mouse as it traverse through the ladder beam. Type variable indicate 1 of the 3 mouse types: wild (w), knock-out (k), and vehicle (v). Lb_score shows the overall performance of each mouse On average, each mouse has 42.83 good steps and 7.15 bad steps. The mean lb score of the mice is 85.69, with a lowest of 72.33 and a highest of 97.00. The highest total good steps the mouse has is 291 and the lowest is 217 good steps. While for the bad steps, the highest is 83 and the lowest is 9 bad steps.

Table 2 corresponds to a single training data set from one video of the ladder beam task being completed. From this particular video, nineteen frames were extracted which produced a data set with 288 rows and 5 columns. Each row represents a the location of one of the mouse's paws during a particular frame from a specific video. The video column indicates what video the frames were extracted from. Likewise, the frame column indicates which particular frame the body part was being tracked. Lastly, the body part, x, and y column correspond to which paw or part of the body, the x coordinate, and the y coordinate, respectively. In total, we obtained training data sets from three separate videos. It should be noted that in each data set there are several missing values in the x and y columns indicating that a particular part of the body was blocked or obstructed by a ladder rung. Therefore, that specific body part was not able to be labeled.

[3]Aljovic, A., Zhao, S., Chahin, M. et al. A deep learning-based toolbox for Automated Limb Motion Analysis (ALMA) in murine models of neurological disorders. Commun Biol 5, 131 (2022). https://doi.org/10.1038/s42003-022-03077-6

Table 2: Example of First 10 Rows of Training Data from a Single Video

Video	**Frame**	**Bodypart**	**X**	**Y**
20230112_145948	img0528.png	front_left_paw	1512	173
20230112_145948	img0528.png	front_right_paw	NA	NA
20230112_145948	img0528.png	front_mid	1525	117
20230112_145948	img0528.png	back_left_paw	NA	NA
20230112_145948	img0528.png	back_right_paw	NA	NA
20230112_145948	img0528.png	back_mid	NA	NA
20230112_145948	img0528.png	nose	1525	257
20230112_145948	img0528.png	tail	NA	NA
20230112_145948	img0614.png	front_left_paw	1521	320
20230112_145948	img0614.png	front_right_paw	1580	166

4 Overall Technical Approach

4.1 Data Wrangling

In order to utilize the videos provided by us, we needed an efficient tool for extracting frames in the video and labeling the location of mouse paws. We utilized DeepLabCut (DLC), an open source Python tool for developing models capable of tracking animal behavior and movement. The process for extracting our data from this tool goes as follows. First, we input a video or list of videos that we wish to obtain training data from into DLC. DLC then extracts a particular set of frames from each video according to the options we specify. Next, we individually label each frame by marking the location of each of the four mouse paws. Lastly, from those labeled frames DLC produces a training data set.

Table 3: Example of First 10 Rows and 5 Columns of Uncleaned Training Data from a Single Video

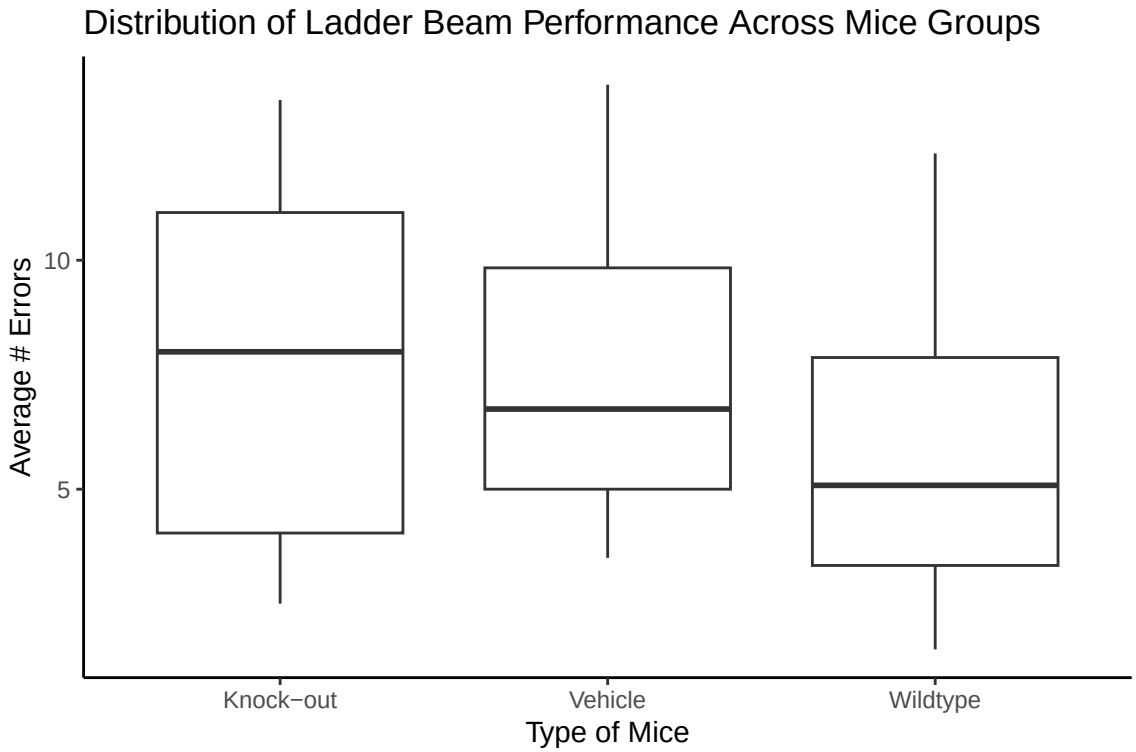
scorer	**...2**	**...3**	**NG...4**	**NG...5**
bodyparts	NA	NA	FrontLeft	FrontLeft
coords	NA	NA	x	y
labeled-data	20230112_145948	img0528.png	1511.8120777168124	173.10722403026364
labeled-data	20230112_145948	img0614.png	1520.60058048641	320.14297883682275
labeled-data	20230112_145948	img0924.png	1529.8501764402251	814.0021748327481
labeled-data	20230112_145948	img0990.png	1547.4319112269511	888.7113003469041
labeled-data	20230112_145948	img1102.png	1548.6915056663374	1143.1493771030198
labeled-data	20230112_145948	img1198.png	1547.5668923039445	1265.4752907327857
labeled-data	20230112_145948	img1209.png	1546.0140946710305	1264.6988919163284
labeled-data	20230112_145948	img1344.png	1564.9997563054462	1495.0249174673277

However, the training data must then be pre-processed since the outputted format is not efficient to work with. As shown in the table above, each frame corresponds to a single row but the names for the columns need to be cleaned. In order to clean the training dataset initially produced by DLC into the table described in the Dataset section, the column names had to be changed and rows could be broken down to represent multiple body parts. First, we removed the scorer's name since it is irrelevant to the data. The columns containing the video file and the frame were given appropriate names. Then, each body part was given a correct column name. For example, FrontLeft corresponded to the front left paw of the mouse. Lastly, each individual row was transformed into four rows, one corresponding to each paw of the mouse at that specific frame.

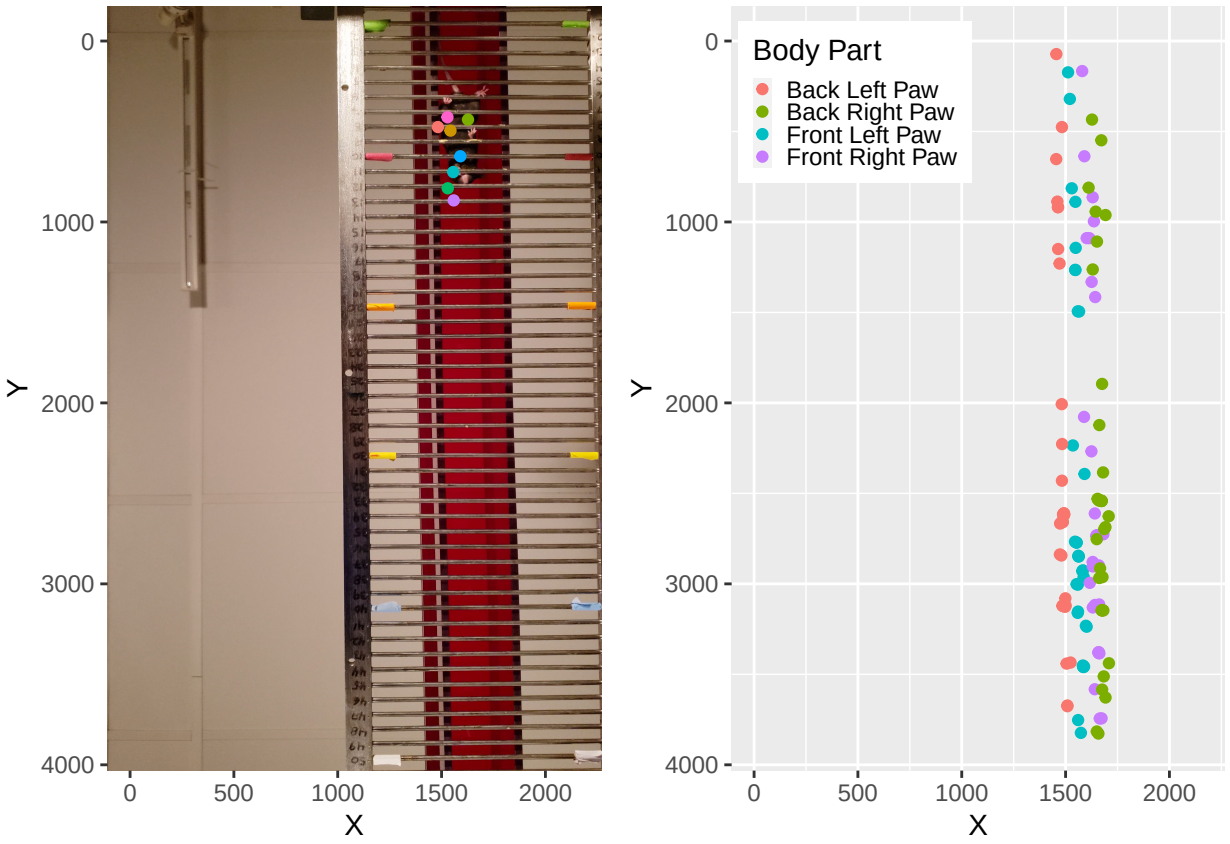
4.2 Exploratory Data Analysis

Table 4: Summary statistics of bad steps for each type of mouse

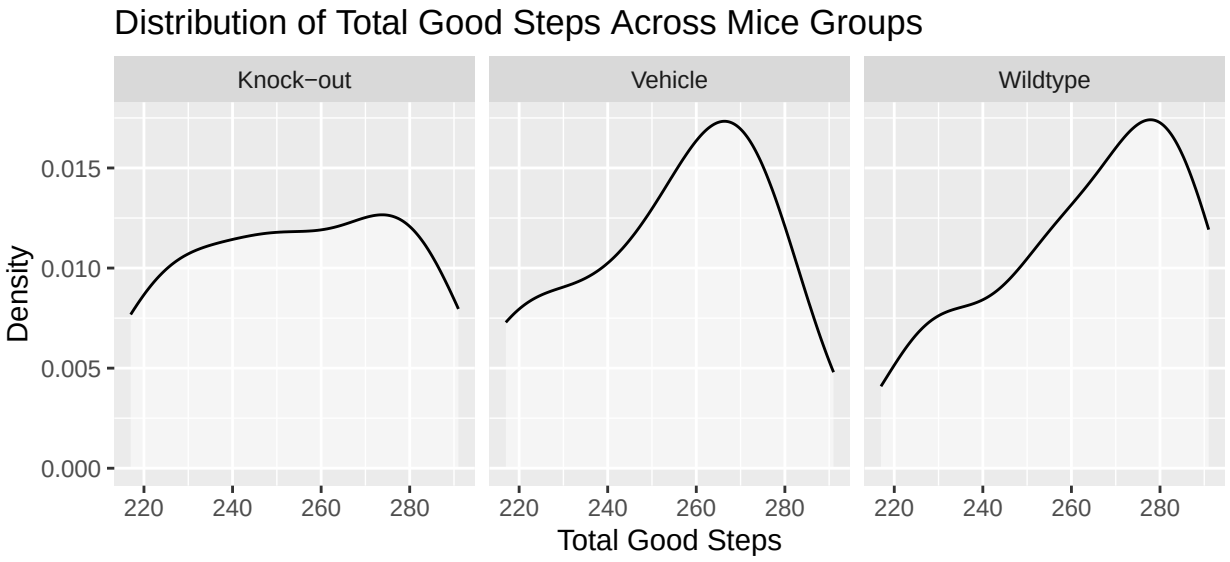
Type	Mean Bad Steps	Variance Bad Steps
Knockout	7.62	16.0
Vehicle	7.77	13.0
Wildtype	6.07	13.5

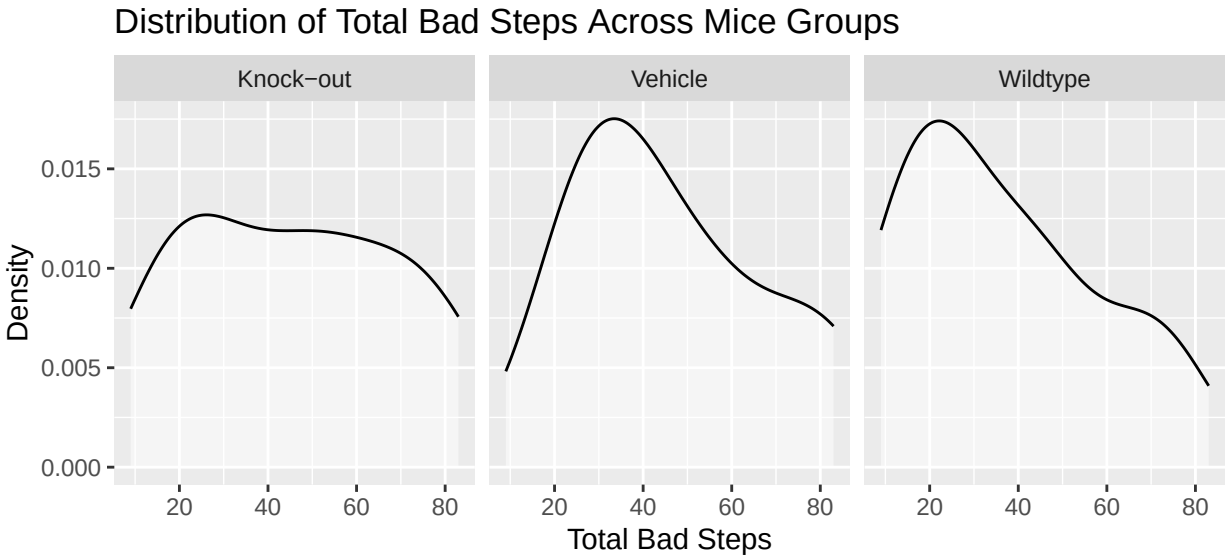


When comparing the three groups of mice, we can see that on average the knock-out mice performed the worst while the wild type performed the best. The vehicle group and wildtype group appear to have similar variance while the knock-out group's variance is slightly higher. Based on the plot and table above, there may be some evidence that the stem cell therapeutic helps certain groups of mice in their recovery a little more.



The two plots above, describe how the paws are tracked in a video and the general behavior of each paw. The first plot, indicates that we are tracking each paw by emphasizing the location of the toes. Although there are only four paws, four additional points were labeled on the mouse to better track the skeletal structure. Specifically, the nose, tail, middle point between the front paws, and middle point between the back paws were labeled. The second plot describes the natural behavior one would expect for the paws of a mouse. That is, the front paws remain in front of the back paws for majority of the video. It must be noted that the recordings were done with a stationary camera so the mouse was moving forward for the most part, but there may be instances in the video where the mouse backtracks.





The above two graphs show the distribution of the total good and bad steps across the mice groups. According to the graphs, almost every mouse having different density of total good and bad steps. The Vehicle type has less good steps compared to the Wildtype, and more bad steps compared to the Wildtype. Overall, the total good steps and bad steps across mice groups are differently distributed.

4.3 Data Modeling

Our main interest is to build a model that can accurately track a mouse paw as they complete the ladder beam task and capable of classifying mouse steps as good ones or bad ones. As a result, we utilize DeepLabCut for the object detection model, which is a software package for animal pose estimation that allows the training of a deep neural network using trained data.

We start by uploading the videos and then extracting the frames to be manually labeled as the training data. We design the skeleton of the mouse and manually label the mouse's body parts and rungs throughout these frames for the neural network to be trained on. Through this process we then can create the training data sets. After obtaining the training data sets, we train the network, while saving for every 100 iterations with a total of 166500 iterations. We then evaluate the trained network by computing the mean average Euclidean error between the manual labels and the ones predicted by the network. Then we can use the trained network to analyze new videos and plot the mouse steps in x and y coordinate.

Table 5: ANOVA Model Summary

	DF	**Sum Squared Error**	**Mean Squared Error**	**F-Value**	**Pr(> F)**
Type	2	638	319	0.625	0.543
Residual	27	13779	510	NA	NA

We also built an ANOVA model comparing the total bad steps between the three types of mice. The Knock-out type has an average bad steps of 7.6, the Vehicle type has an average bad steps of 7.8, and the Wild type has an average bad steps of 6.1. The table above shows that at a significance level of 0.05 and with a p-value of 0.543, we can conclude that the average number of errors is not significantly different among the three types of mice after treatment. Diagnostics for the ANOVA model were performed which are discussed in the Experiments and Evaluation section below. We found that there is a possible violation to the normality assumption. Consequently, we ran a Kruskal-Wallis test to account for this. We concluded that the median number of errors is not significantly different among the three types of mice after treatment [p-value = 0.402].

5 Software

Self-written software used for this project, in order of use:

Table 6:

Scripts	**Descriptions**
cleaning_extracting.Rmd	A script to clean and transform data for appropriate analysis
eda.Rmd	An exploratory data analysis (EDA) script for visualizing ladder beam performance and tracking individual paws throughout a single video
anova_model.Rmd	A script for creating visualizations and statistical analyses (ANOVA) on two datasets ('post_injury' and 'raw_post_injury') to understand the differences in specific variables across different 'type' groups in the two datasets
dlc_model.ipynb	A data analysis notebook to train model using DeepLabCut to create labeled video

Third-party software and libraries used:

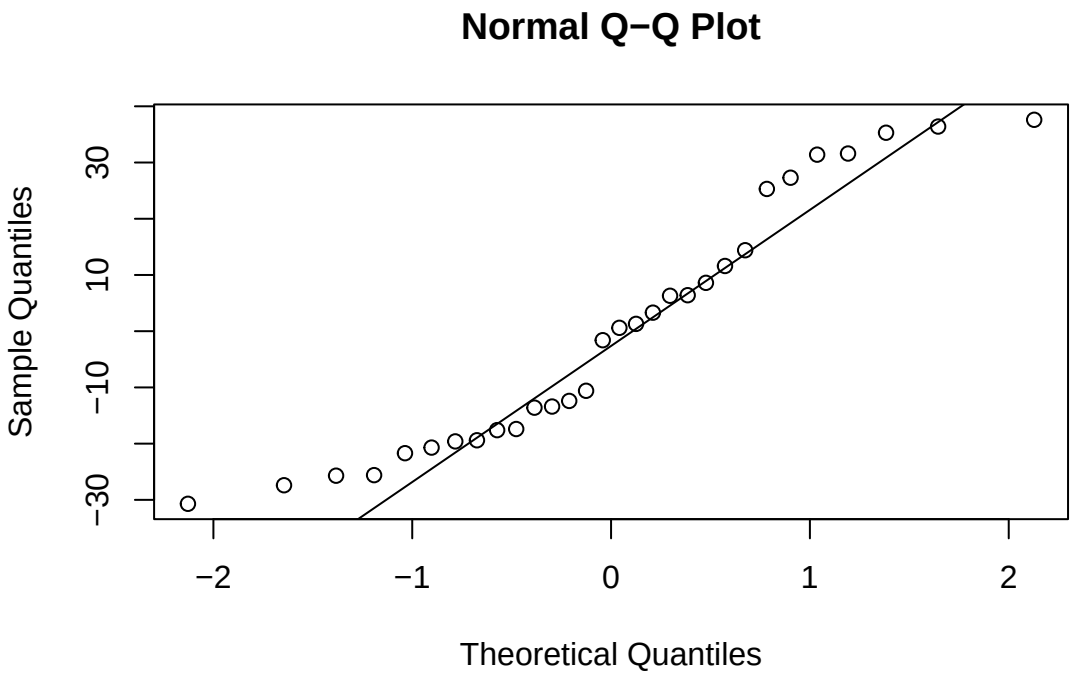
Table 7:

Scripts	**Descriptions**
DeepLabCut (DLC)	A software package for animal pose estimation using deep neural networks
TensorFlow	A Python library for machine learning, mainly on deep neural networks

6 Experiments and Evaluation

6.1 ANOVA Diagnostics

To check the validity for our ANOVA model we performed diagnostics to see if any of the following assumptions were violated: normality, constant variance, and independence. We know from the lab that each mouse was randomly assigned and tested independently so the independence assumption is not violated. Likewise, from our exploratory data analysis we saw that the variance across the three groups was fairly similar so constant variance can be assumed.

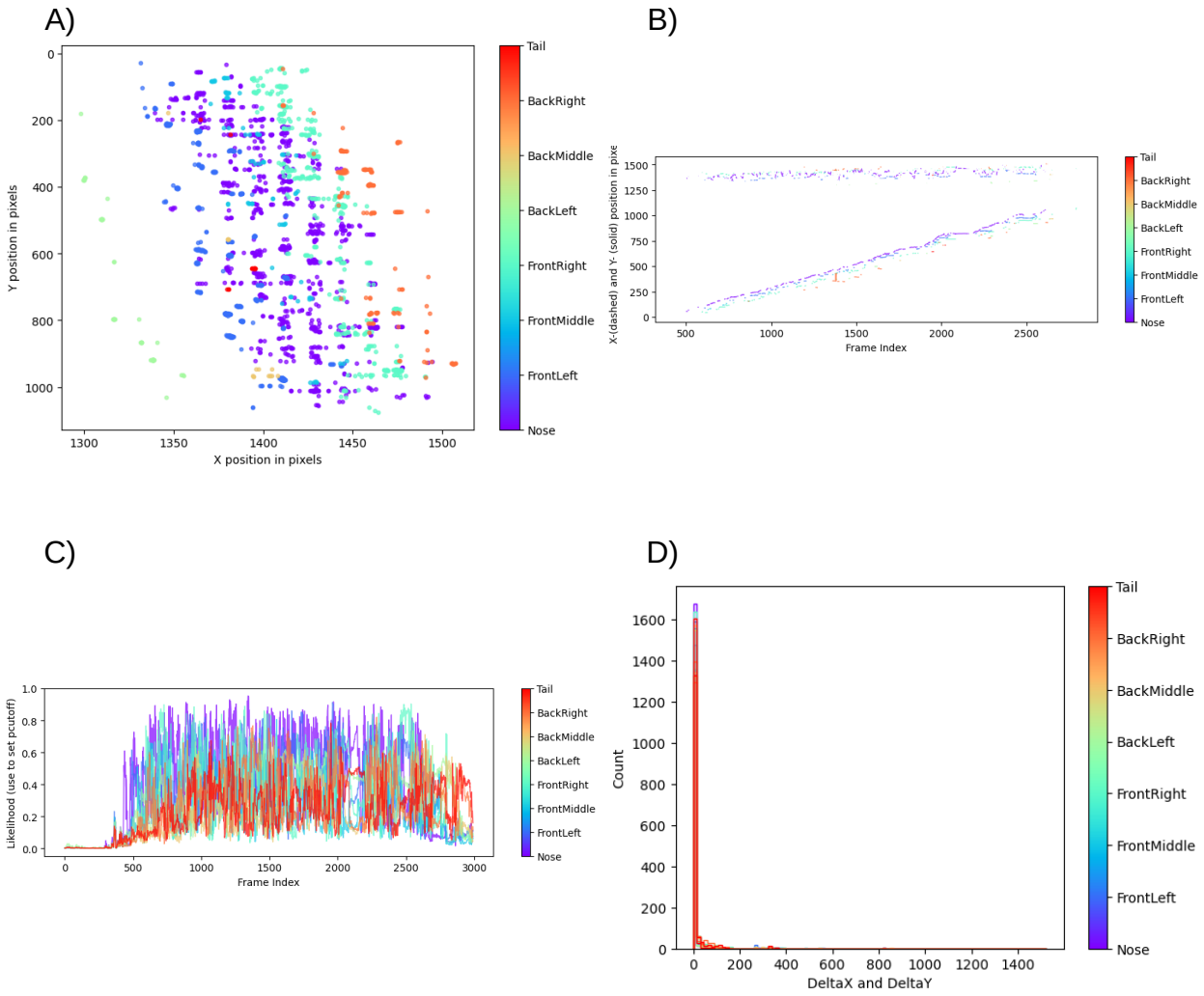


However, as show in the plot above, heavy tails in the QQ plot indicate there is a possible violation to the normality assumption. This may be due to the small sample size. In this case, a non-parametric test may be a more powerful test. Consequently, we chose to conduct a Kruskal-Wallis test to compare the median number of errors across the three types of mice.

6.2 Neural Network

Through the initial iterations of training the deep neural network, DeepLabCut reported 10 iterations with an error rate of 34.955% based on a p-value of 0.005 where it plateaued at around 0.3% error rate after 10,000 iterations. After running it for 166,570 iterations, it ended at 0.096% error rate with a p-value of 0.02. The run time took around 16 hours to train the neural network. The training error came out to be 105.05 pixels and the testing error came out to be 44.17 pixels. However, with a p-value of 0.6 both training and testing had an error rate of around 17 pixels. Performance is measured by computing the mean average Euclidean error between the manual labels and the ones predicted by the neural network. The video dimensions were 3840 by 2160 pixels.

We were then able to use DeepLabCut to create plots to analyze the accuracy of the trained neural network on the videos we created. From this it outputted CSV files and plots which showed the x versus y coordinates of the skeletal structure (A), skeletal structure coordinates each frame (B), likelihood of each body part in the skeletal structure each frame (C), and a histogram for the counts of delta x and delta y by body part (D).



The x and y coordinate plots versus time plot should be continuous, but because of the obstructions, it became dotted points which shows the difficulty the neural network had in identifying the body parts as it moved across the ladder beams.

The likelihood plot shows, the likelihood that the body part at each frame is predicted correctly. Ideally, the likelihood versus frames plot would keep a likelihood of one at the top of the graph if there was high confidence that the mouses body part would be there. However, in this situation, the likelihood of all the parts stays around the middle of the y axis. Another thing to notes is that the nose and front paws have the highest likelihood versus the tail and back of the mouse which have the lowest likelihood values.

The histogram of each body parts delta x and y shows the difference in pixel distance between labels in consecutive frames. This count should all have a low delta x and delta y because it would show a consistent tracking of each body part. Large jumps in these values over a frame would mean that it is losing track of that body parts position. This corroborates the likelihood plot and shows the front of the mouse is better tracked than the tail and middle of the mouse.

The videos were able to accurately track the body parts a majority of the time but the obstructions from the ladder beams made it impossible to track all parts of the mouse the entire time. Additionally, although the videos were taken in higher quality and from a mounted camera, the distance from the mouse made it difficult to manually label let alone for the neural network to process. This is why there are gaps in the data and rarely times where it detects the entire skeleton of the mouse at once. Looking frame by frame, you can see that it struggles to label all the parts with high confidence and often is unsure of whether the whiter parts of the body are the nose or a paw and where the middle, black parts of the body are due to these same

reasons. However, the video shows accuracy when it does detect the body part in frame.

7 Notebook Description

The models used for comparing the average number of missteps amongst the three types of mice were developed in the anova_model.Rmd file. The file contains all the major components we used to do the comparisons. First, we load in the cleaned summary sheet data produced by the R markdown file for cleaning. Then we built the ANOVA model and checked the diagnostics. Upon analyzing the diagnostics, we then conducted a Kruskal-Wallis test.

We also included dlc_model.ipynb because this Jupyter notebook contains all the code we used to create labeled data sets, train the network, and evaluate the network. The notebook starts by instructing how to download all the dependencies and set up an appropriate environment. From there, the subsequent code blocks show detailed steps in how the model was developed.

8 Members Participation

Table 8: Percentage of workload across group members

Task	**Giles**	**Nathan**	**Alex**	**Vinh**	**Owen**
Data Wrangling and Cleaning	50%	50%	0%	0%	0%
Exploratory Data Analysis	25%	25%	25%	0%	25%
ANOVA Modeling	20%	20%	30%	0%	30%
DLC Neural Net Modeling	50%	50%	0%	0%	0%
Report and Presentation Writing	25%	25%	20%	10%	20%

9 Discussion and Conclusion

What did you learn about the methods and algorithms you worked with? What did you learn about their strengths? And their limitations?

> In our project, we worked with DeepLabCut, a deep learning-based software for tracking body parts in videos. We found that DeepLabCut has significant strengths in accurately and efficiently tracking the movement of mice during the ladder beam task even with a relatively small training data set. However, collecting and annotating a large dataset can be time-consuming and labor-intensive. Furthermore, we found that the obstructions in the video made the model lose track of certain bodyparts of each mouse. Lastly, training such a large neural network requires a lot of computational power.

What ended up being harder than you expected in your project? What was surprising about your project?

> A surprising aspect of the project was the complexity of the mouse movements captured in the ladder beam task videos. The variability in step patterns, paw placements, and subtle differences in behavior among mice posed challenges in accurately tracking and analyzing the data. Some videos had the ladder on the left side rather than the right side of the frame. In some cases mice would get stuck in the middle and not complete the whole task. These all ended up being difficulties for the model.

What other lessons did you learn, expected or unexpected (e.g., perhaps about the tools you used, if you used anything out of the ordinary?

One unexpected lesson we learned is the amount of time it requires to set up appropriate environments and download necessary dependencies to get software to run. In many cases throughout the project, what seemed to be simple tasks or installations ended up requiring several hours of troubleshooting and debugging. Also, it was interesting to see how small obstructions greatly hinder the performance of our CNN.

If you were in charge of a research lab, what ideas and directions might you invest in over the next year or two to try to make major progress on this problem? Feel free to be speculative in discussing possible future directions.

In the future, we would film the mouse doing the ladder beam test in a more standarized way. Specifically, we would try to obtain videos with side angles as well so we have more points to track the structure of each mouse. Ultimately this would set up a better working environment for pose estimation in the videos. We could also leverage RCIC's GPUs to increase training speed of our CNN. After building an accurate tracking model then we could transition into building a model capable of classifying bad steps and good steps. Lastly, we could use longitudinal data to analyze the trend in missteps over time. This would allow us to compare the rate of recovery between the different types of mice rather than analyzing data from a single week post-injury.