Cascaded Deep Monocular 3D Pose Estimation with Evolutionary Synthetic Training Data

[Unsupervised Behaviour Recognition using Deep Reinforcement Learning]

Final-Term Project Report

Submitted by
Vihaan Akshaay Rajendiran
ME17B171

in partial fulfillment of requirements for the award of the dual degree of

BACHELOR OF TECHNOLOGY in
MECHANICAL ENGINEERING
and
MASTER OF TECHNOLOGY in
ROBOTICS

DEPARTMENT OF ENGINEERING DESIGN
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI 600 036

May 2022

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to those who provided me with the possibility to work on this project. A special gratitude I give to my project guide Prof. Ravindran, Computer Science Department, IIT Madras whose contribution in stimulating suggestions and encouragement helped me coordinate my project and drive it in the right direction.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of Prof. Vivek Kumar, JAX Laboratory, guided me consistently and provided me with several resources related to the project.

A special thanks to Prof. Asokan & IDDD Robotics Program of IIT Madras for giving me the right exposure and opportunity to work on my master's thesis.

ABSTRACT

KEYWORDS: Deep Reinforcement Learning, Behaviour Recognition, Synthetic Data Generation, Simulation, Evolutionary Operators, Data Augmentation, Deep Neural Networks, Option Discovery in Hierarchical Reinforcement Learning, Sim2Real, Computer Vision, Camera Calibration, GANs, Pose Estimation.

Modeling mice behavior to understand the neural circuits in its brain requires an understanding of the processes that generate the exhibited behaviors. This requires identifying behavior patterns and the variations in them from unstructured datasets (typical examples are camera captures and videos of mice in different environments). Following a typical reinforcement learning formulation, mice can be thought of as agents making decisions based on an inherent policy.

With the first step of identifying key points from ample raw video data, the project proposes using open-source three-dimensional synthetic animated mouse models to improve existing pose and estimation models. We developed a three-dimensional replica of the workplace at the JAX laboratory by building an environment similar to that of the dataset. The render engines on Blender allow the creation of more training data for existing 2D pose estimation models to improve their predictions for custom datasets. In addition, Since the outputs observed from 3D model-based pose estimation could yield behaviors of better quality than 2D, the project also uses the same simulations to build a model that lifts 2D poses to the corresponding 3D pose. Since this dataset generated could have a slight bias and be less in quantity, we also adopt various data augmentation techniques that are scalable for synthesizing massive amounts of training data for training 2D-to-3D networks and can reduce dataset bias.

** **

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview

With time, animals have evolved to have a complex behavioral repertoire to support their life system. Arguably, the most explicit representation of such complex behaviors would be the sequence of movements of their physical body. Animals typically exhibit common reproducible movements similar to the movements of hands of humans, which can be easily identified as reproducible across individuals, allowing us to understand hand gestures. These behavioral traits on a higher level can be described as being constructed from a set of primitive actions taken by the animal, like grooming for a mouse might include a particular sequence of movement of joints on the hands and legs. In summary, they are essentially temporally extended action sequences. In fact, It is shown in the behavior recognition literature (Berman et al. (2014)) that these sequences of actions are hierarchical in nature, meaning that each action is governed by a parent action.

The goal of the project is to develop unsupervised techniques to discover and track stereotyped behaviors from raw video data of mice (in our case, a fixed arena). We want to show that mice behavior is organized into a hierarchical structure as hypothesized. We propose to discover this structure using the options framework in Hierarchical Reinforcement Learning.

1.2 Motivation

The pattern of movements in animals (mice action sequences in our case) can be viewed as the combined result of the interworking of genes, metabolism, and neural signaling. As a result, predicting the behavior of mice and learning an equivalent hierarchical model can aid us in understanding the underlying biological mechanisms and the principles behind how an animal generates behavioral sequences.

In reinforcement learning literature, generalizing robot movement is one of the long-standing problems. On an intuitive level, it can be argued that we as humans perform complicated actions like picking up objects in a hierarchical manner. It is important to understand the inherent hierarchy underlying our method of performing tasks, to break down tasks into smaller actions, and to train machines that can generalize actions. This task of generalizing the robotic movements can be achieved to a great degree using option discovery in the field of hierarchical reinforcement learning. Results show that hierarchical models train faster than completely flat models, which in some way assures our assumption. This gives hope that such an attempt at behavior discovery in animals (mice) will enable us to replicate mice behavior and find options in the action trajectories.

Estimating 3D animal pose from RGB images is critical for applications such as action recognition and behavior analysis. Most current work in biology regarding mice by drawing conclusions and analyzing raw video data relies on 2-D pose estimators. This project's case, similar to that of most other experiments, would be benefitted by being able to extract 3-D key points. Since we analyze most physical properties and trajectories in the 3-D space, running algorithms that are based on intuitive models would highly benefit from state representations that have more meaningful dimensions.

1.3 Objective of the Work

Capturing a novel behavioral repertoire from unstructured datasets proves to be prohibitively expensive. Primary work focuses on a supervised learning approach supported by tedious human notations, which, while successful at providing insights into behavioral metrics, are limited by human perception and intuition as they are limited by human perception and intuition as they, directly and indirectly, require the human to identify behaviors and distinguish between them. In addition, unlike fruit flies and zebrafish, very little work has been done on mice. Unsupervised methods, on the other hand, can derive a behavioral repertoire from the data itself, which often results in the discovery of new behavior classes (Marques et al., 2018) or a deeper understanding of the construction of established stereotypical behaviors (Wiltschko et al., 2015). Driven by this motivation, a completely unsupervised approach is considered in this project.

Working on a new combined pose estimation pipeline with a supported 2D-3D lifting model helps in the unsupervised analysis of the raw video data of our case, as well as can be made useful by augmenting with other pose existing machine learning models to estimate more features and provide with a richer dataset for further analysis.

CHAPTER 2: PIPELINE & METHODOLOGY

2.1 Data Description

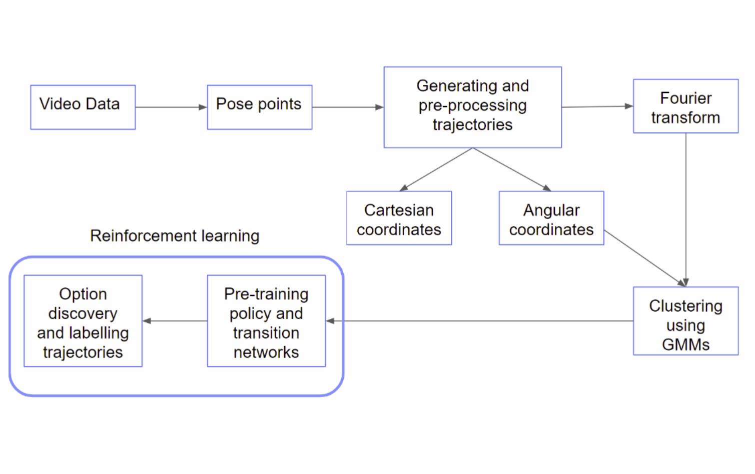
The Jackson Laboratory has several thousand hours of mice videos (belonging to different strains), along with pose estimation coordinates and confidence for these videos. These videos capture the motion of mice in a square arena. The data consists of the coordinates for the 12 key points on the mouse's body and confidence scores for each point. The 12 points are the nose, left ear, right ear, the base of the neck, center of the spine, left front paw, right front paw, left rear paw, right rear paw, the base of the tail, middle tail, and the tip of the tail. The data format is a video file along with a corresponding HDF5 file. Each HDF5 file contains two datasets:

• Key points: This is a dataset with size (#frames x #key points x 2) where the last dimension of size 2 is used to hold the pixel (x,y) position of the respective frame and keypoint

• Confidence: This dataset has size (#frames x #key points) and assigns a confidence value in the range 0-1 to each of the 12 points

2.2 Project Pipeline

The following flowchart describes the existing pipeline of the project.



Video Data
Pose points
Generating and pre-processing trajectories
Fourier transform
Cartesian coordinates
Angular coordinates
Reinforcement learning
Option discovery and labelling trajectories
Pre-training policy and transition networks
Clustering using GMMs

Fig 2.2: Visual Representation of Data Pipeline

The main focus of this report is to improve the overall 'Pose Estimation' sub-part.

CHAPTER 3: POSE ESTIMATION PIPELINE

3.1 Current Pose Estimation Framework

Pose estimation from video frames has been done using a network similar to the Deep High-Resolution architecture- HRNet (Sun et al. (2019)), developed and trained by Keith Sheppard at the Kumar Lab of the Jackson Laboratory, Maine. He used the smaller HRNet-W32 architecture and added two 5x5 convolution layers to the head of the network.

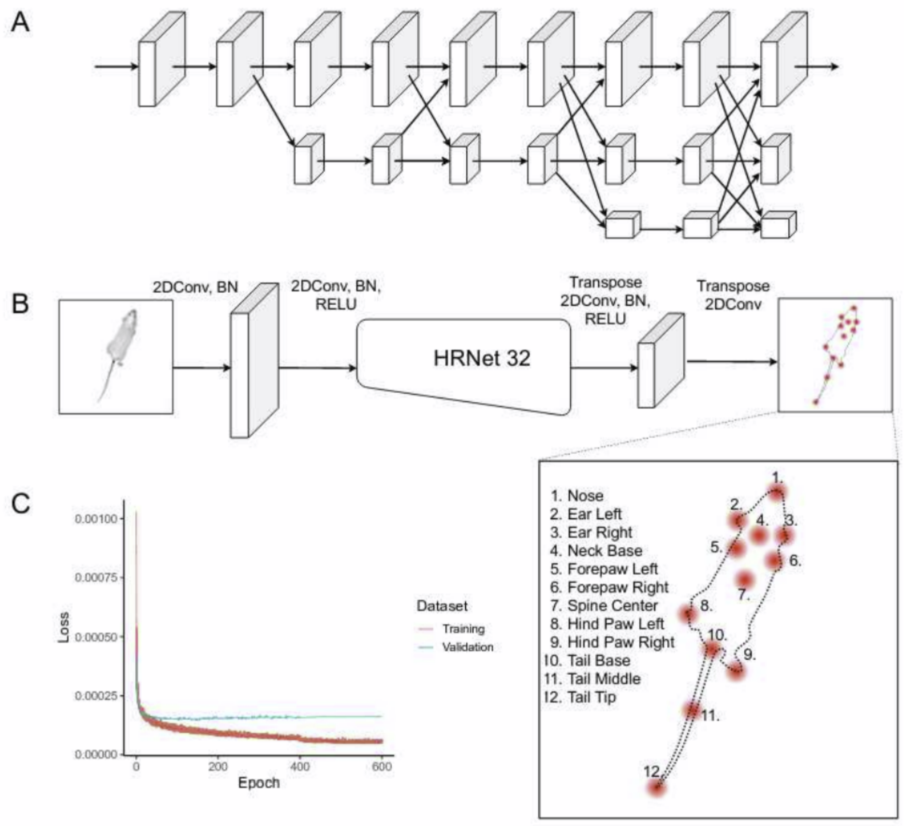


Figure 1: Deep convolutional neural network for pose estimation. (A) We chose the HRNet-W32 neural network architecture for performing pose estimation. The main differentiator of this architecture is that it maintains high resolution features throughout the network stack thereby spatial precision. (B) The inference pipeline which sends video frames into the HRNet and generates twelve keypoint heatmaps as output. (C) Training loss curves show network convergence without overfitting.

Fig 3.1: Original Pose Estimation Model

At the moment, with a few video frame processing, we utilize this model to obtain 12 keypoint locations in 2D (12x2 data points) for every feed-forward of the neural network using frames in the video. Since this will directly be used for the state representation in the reinforcement learning module, we wish to improve this situation.

3.2 Proposed Pose Estimation Stack

Exploring better state representations for the mouse agent in our experiment, one obvious and viable choice is to use 3D representation for each key point rather than just 2 dimensions for each key point. We believe this might help us obtain better results because the experiment tries to obtain behavior and hierarchies in decision making by the mouse for physical actions, and intuitively, this is highly dependent on the body structure. Naturally, any state representation based off of 3D points would be a better proxy for the true mouse body than 2D Points. To support this, we tried to add another layer to the pose estimation stack. Since there was already a model that gave us 2D Key points from video data, we wished to obtain 3D coordinates at the end of the stack. Let the first model that predicts 2D key points from video data be called 'Model A'. Now, a second model (Model B) is trained to lift 2D coordinates to 3D coordinates. The complete pipeline and training data plans are illustrated in the figure below.

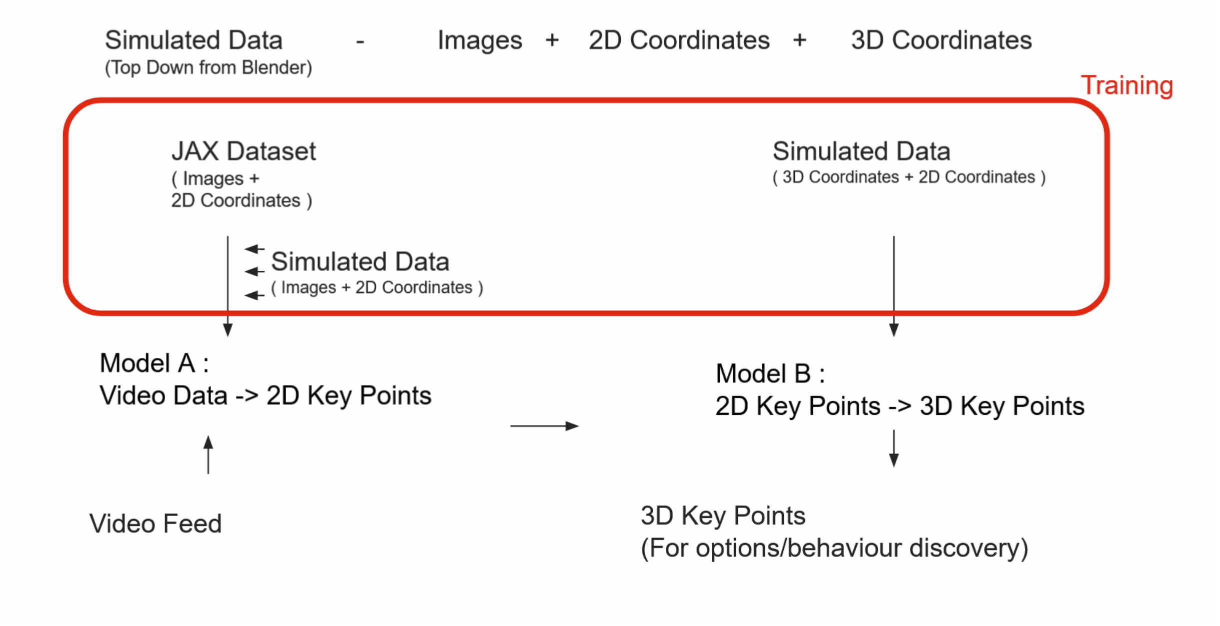

Simulated Data
(Top Down from Blender)
-
Images + 2D Coordinates + 3D Coordinates
Training
JAX Dataset
(Images +
2D Coordinates)
Simulated Data
(3D Coordinates + 2D Coordinates)
Simulated Data
(Images + 2D Coordinates)
Model A :
Video Data -> 2D Key Points
Model B :
2D Key Points -> 3D Key Points
Video Feed
3D Key Points
(For options/behaviour discovery)

Fig 3.2: Pose Estimation Stack & Data Pipeline

We shall first discuss about Model B first. Obtaining 3 dimensions from 2-dimensional key points isn't as straight forward and trivial as it is an inverse problem. Although, this should be determinable since the mice body has bone and skin constraints that restricts the 3rd dimension of key points from being undeterminable, building a direct mathematical model for obtaining these key points might be difficult. To tackle this problem, we wish to follow a data driven approach and use some function approximator to handle the mathematical constraints and give us an estimated height for the key points.

CHAPTER 4: DATASET GENERATION (Synthetic)

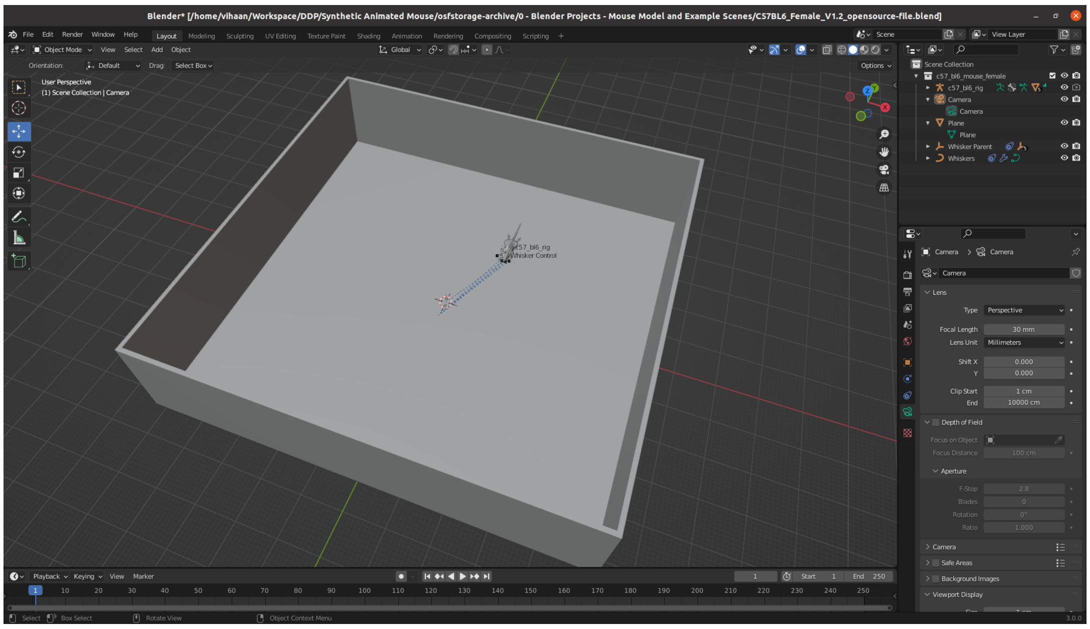
4.1 Simulated Environment

Since we don’t have 2D - 3D data available, we resort to obtaining them from simulations. We use a three-dimensional (3D) synthetic animated mouse that was built based on computed tomography scans that can be actuated using animation and semi-random joint-constrained movements to generate synthetic data with ground-truth label locations. This mouse model was built by an open-source mouse file (c57_bl6) and was imported into Blender, a 3-D rendering software. Blender has python integration, and it is possible to automate the rendering process through code. This makes it easy to generate abundant render data. This mouse model is in the right dimensions and comes with accurate control and measurements of body parts (including the whiskers). A new workspace similar to that of the arena was used to generate the JAX dataset [Geuther, B.Q *et al.*].

In brief, some parameters taken into consideration while reproducing the JAX environment are the following:

- The open-field arena measures 52cm [w] by 52cm [l] by 23cm [h]
- JAX data was acquired at 480 x 480 px resolution
- One camera and lens were mounted approximately 100cm centered above each arena
- Zoom and focus were set manually to achieve a zoom of 8px/cm
- Lighting is as described below:
    - Light contains a ring of ~100 conical LEDs.
    - Surface and lighting cause a slight gradient (corners dimmer than the center)

It is to be noted that the current pose estimation model developed at JAX applied contrast/brightness augmentation, so getting it exact shouldn’t be too necessary.


Blender* [/home/vihaan/Workspace/DDP/Synthetic Animated Mouse/osfstorage-archive/0 - Blender Projects - Mouse Model and Example Scenes/C57BL6_Female_V1.2_opensource-file.blend]
User Perspective
(1) Scene Collection | Camera
c57_bl6_rig
Whisker Control
Scene Collection
c57_bl6_mouse_female
c57_bl6_rig
Camera
Plane
Whisker Parent
Whiskers
3.0.0

Camera > Camera

Lens
Type: Perspective
Focal Length: 30 mm
Lens Unit: Millimeters
Shift X: 0.000
Y: 0.000
Clip Start: 1 cm
End: 10000 cm

Depth of Field
Focus Distance: 100 cm
Aperture
F-Stop: 2.8
Blades: 0
Rotation: 0°
Ratio: 1.000

Fig 4.1: JAX Environment simulated on Blender

4.2 2D - 3D Data Generation

We first spawn the mouse in various positions and orientations using python on blender. This enables us to control the mouse's location in the workspace and the orientation. For the sake of dataset generation, the mouse was spawned at a random location within the workspace. A random orientation along the Z-axis was also given to the mouse to ensure we obtained more generic data points.

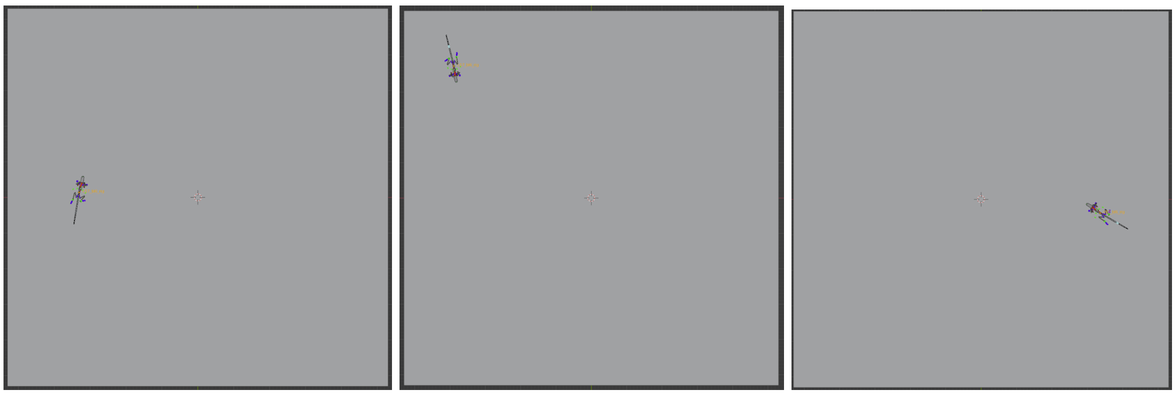


Fig 4.2.1: Mouse body spawned at different locations in the arena

The next step is to move each of the joints to different orientations to obtain joint-constrained poses that help us generate synthetic data. We first identify the key points that the original JAX model outputs so we can link them with the corresponding 3D points on the mouse model. This way, we can generate additional data for training the first model as well as obtain data for a new 2D-3D model that uses these same key points. The indices of points and their corresponding points on the mouse body as follows:

'snout' = 0

'ear_l' = 1

'ear_r' = 2

'cervical_vertebrae_l' = 3

'ulna_l' (tail) = 4

'ulna_r' (tail) = 5

'lumbar_vertebrae_2' = 6 (This is the frame of reference)

'tibia_l' (tail) = 7

'tibia_r' (tail) = 8

'tail_1' = 9

'tail_5' = 10

'tail_9' = 11

An annotated image of the armature obtained from blender is presented below.

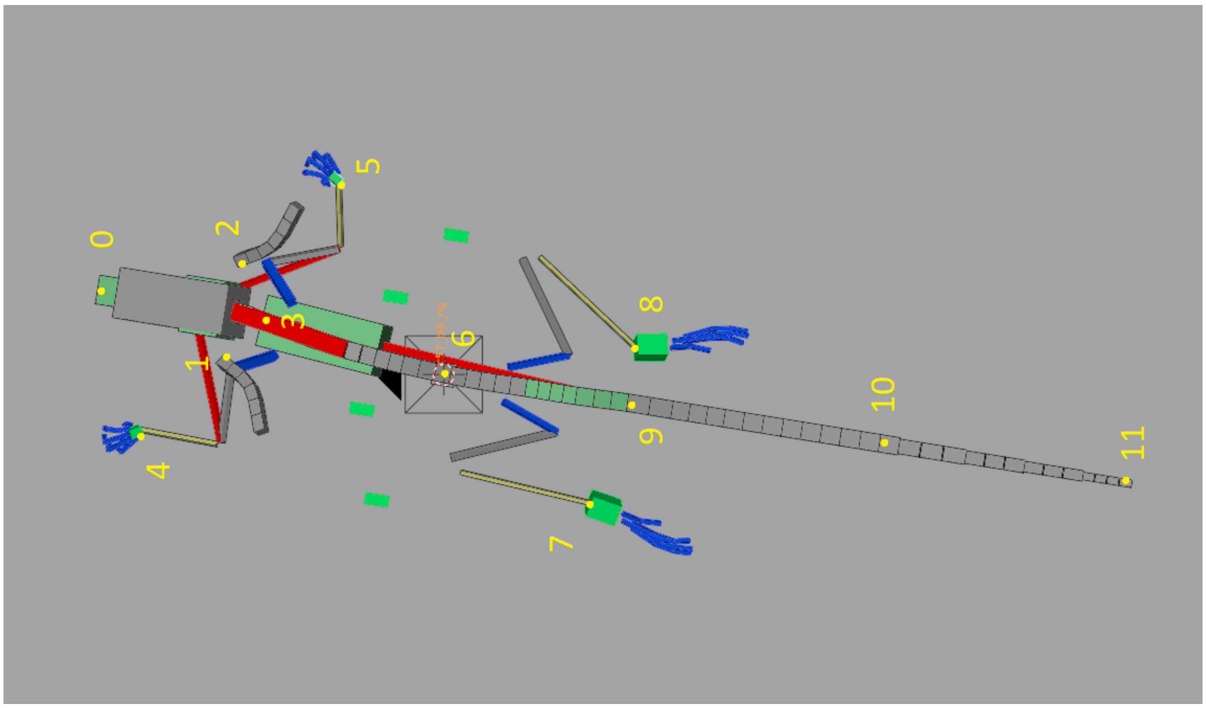


Fig 4.2.2: Desired Key Points marked and numbered on the blender mouse armature

4.3 Blender Control Elements

Now that we know each of these body parts, we move each of these parts randomly to obtain the mouse pose in different configurations (We shall use the term ‘configurations’ to talk about different spawns of the mouse with different joint angles). Although there is a catch here. In this model, there are only some elements we could control. By moving these elements of the mouse, the other generic parts end up in new positions satisfying the joint constraints of the mouse. A mathematical explanation of this can be described as follows.

Let $p_i \forall i \in$ joints be the 3d coordinates of the desired joints in the mouse (The 11 points we have mentioned above). We know that these p_i aren’t all independent. Let there be some other $c_j \forall j \in$ different set of control elements. These elements are the ones that we could freely move in the mouse and p_i’s are decided by these.

$$[p_0, p_1, \ldots, p_{11}] = f(c_1, c_2, .., c_{10})$$

The set of control elements for this open source blender mouse is as follows.

'head_ik'

'hind_paw_ik_r'

'hind_paw_ik_l'

'hindlimb_pole_r'

'hindlimb_pole_l'

'fore_paw_ik_r'

'fore_paw_ik_l'

'forelimb_pole_l'

'forelimb_pole_r'

'lumbar_vertebrae_1'

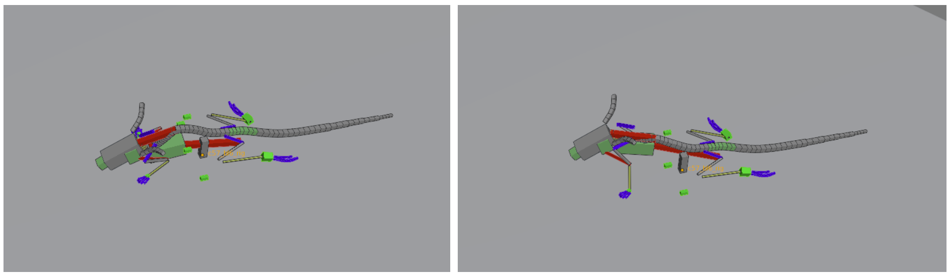


Fig 4.3: Left: Mouse armature in default orientation; Right: ‘head_ik’ control node lifted orientation

CHAPTER 5: DATASET - GENETIC EVOLUTION

5.1 Motivation

Training a function to map from c to j is a tedious task as the true function is defined based on implicit joint and bone constraints, is very non-linear, and has multiple possible orientations in the range for a single 2D project point on the domain. In addition, the number of control inputs (10) is lesser than the number of desired joints. Because of this reason, using random entries in the 'control inputs'**(c)** won't necessarily mean 'true' random values in the 'desired joint space'**(j).** This creates a bias in the dataset generated and might not account for all the extreme cases that the mouse could attain. For this reason, we adapt some data augmentation techniques to generate more valid data points for training Model B.

Let the mouse pose in 3D (considering all corresponding p_i) be denoted by P_k. Let the projection of this pose to the 2D plane be denoted by $\Phi(P_k)$. Ultimately, taking a discriminative approach, we seek a regression function F parameterised by Θ that outputs 3D pose as $\hat{P_k} = F(\Phi(P_k), \Theta)$. This regression function is implemented as a Deep Neural Network (DNN). The training process for this requires a dataset consisting of 2D poses and 3D poses $\{\Phi(P_k) , P_k\}_{k=1}^{N}$. This DNN can be trained by gradient descent based ona loss function defined over the training dataset $L = \Sigma_{i=1}^{N} E\left(P_i, \hat{P_i}\right)$ where E is the error measure between the ground truth P_k and prediction $\hat{P_k} = F(\Phi(P_k), \Theta)$.

Unfortunately, the way we obtain data, it is likely to have sampling bias and limitations on the variation of training data. A DNN can overfit to the dataset bias and become less robust to unseen $\Phi(P_k)$. We take a non-stationary view toward the training data to cope with this problem. While conventionally the collected training data is fixed and the trained DNN is not modified during its deployment, here we assume the data and the model can evolve during their life-time. Specifically, we synthesize novel 2D-3D pairs based on an initial training dataset and then add them into the original dataset to form the evolved dataset. We then retrain the model with the evolved dataset.

To explain this, we first propose a hierarchical representation of the mouse skeleton. The synthesis of novel 2D-3D pairs can be achieved by evolutionary operators.

5.2 Hierarchical Mouse Representation

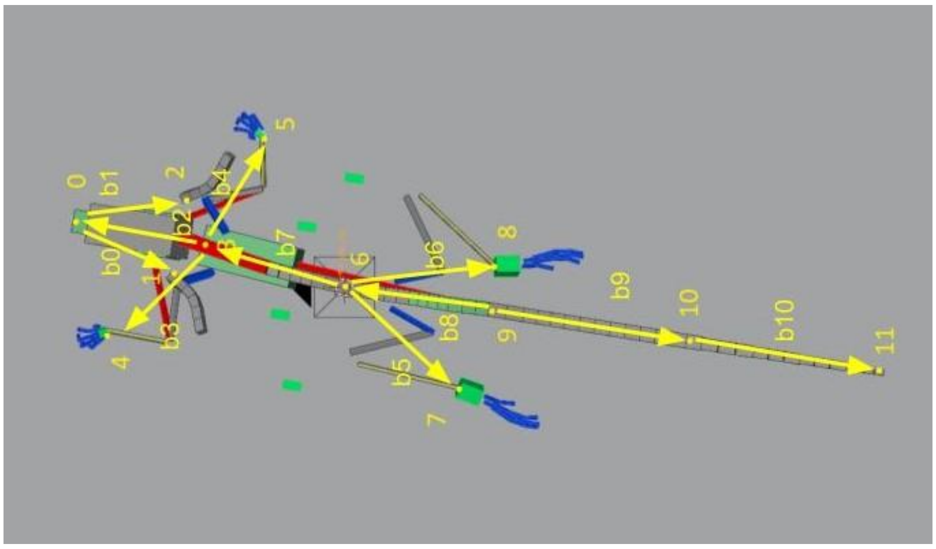


Fig 5.2: Mouse armature representation with bone vectors

We represent a 3D mouse skeleton by a set of bones organized hierarchically in a kinematic tree as shown in the picture above. This representation captures the dependence of adjacent joints with tree edges. Each 3D pose **P** corresponds to a set of bone vectors $\{\mathbf{b}^1, \mathbf{b}^2, \ldots, \mathbf{b}^{10}\}$ where each bone vector is defined as

$$\mathbf{b}^i = \mathbf{p}^{child(i)} - \mathbf{p}^{parent(i)}$$

where $\mathbf{P}^j$ is the jth joint in the 3D skeleton and parent(i) gives the joint index of the ith bone vector. This way, if we desire to move only a certain limb, tampering with each bone vector should do the trick. Once that is done, we can reconstruct the body structure from the bones. For convenience, these bone vectors can further be converted into spherical coordinates. For some bones (Like b7,b8,b9 and b10) whose lengths more or less remain the same, we can convert them to polar coordinates $b^i = (r_i, \theta_i, \phi_i)$. In this representation, the postuer of the skeleton is described by and change the pose by slightly varying the polar angles of these bones individually, and then we can reconstruct the body completely.

5.3 Synthesizing New 2D-3D Pairs

We first synthetize new 3D poses $D_{new} = \{\mathbf{p}_j\}_{j=1}^{M}$ from the original training dataset that was generated from blender $D_{old} = \{\mathbf{p}_j\}_{j=1}^{N}$ and project 3D nodes to 2D plane using the usual projection operator to form new 2D-3D pairs $\{(\Phi(P_j), P_j)\}_{j=1}^{M}$. Evolutionary operators[John Henry et al.] have constructive properties and can be used to synthesize new data given an initial set of poses. We shall look at a few operators below that were used in our experiment.

5.4 Crossover Operator

Given two parent 3D skeleton representation of mice pose, crossover is defined as a random exchange of bones. This definition is inspired by the observation that an unseen 3D pose might be obtained by assembling limbs from known poses. Mathematically, let the set of bone vectors for parent A and parent B be $S_A = \{\mathbf{b}_A^0, \mathbf{b}_A^1, \ldots\ldots, \mathbf{b}_A^{10}\}$ and $S_B = \{\mathbf{b}_B^0, \mathbf{b}_B^1, \ldots\ldots, \mathbf{b}_B^{10}\}$. Now we know the child has to be having the same number of bones and bone connectivity. Now for each bone of the child, we randomly pick from either of the parents, and once all bones have been decided, we reconstruct the body to obtain the child's body vector. Let the child bone vectors belong to the set $S_{child} = \{\mathbf{b}_{child}^0, \mathbf{b}_{child}^1, \ldots\ldots, \mathbf{b}_{child}^{10}\}$.

$$\mathbf{b}_{child}^i = random.choice(\mathbf{b}_A^i, \mathbf{b}_B^i)$$

An example of such an offspring is shown below in a picture. We can see that the offspring from these parents has taken bone structures at random from both these parents.

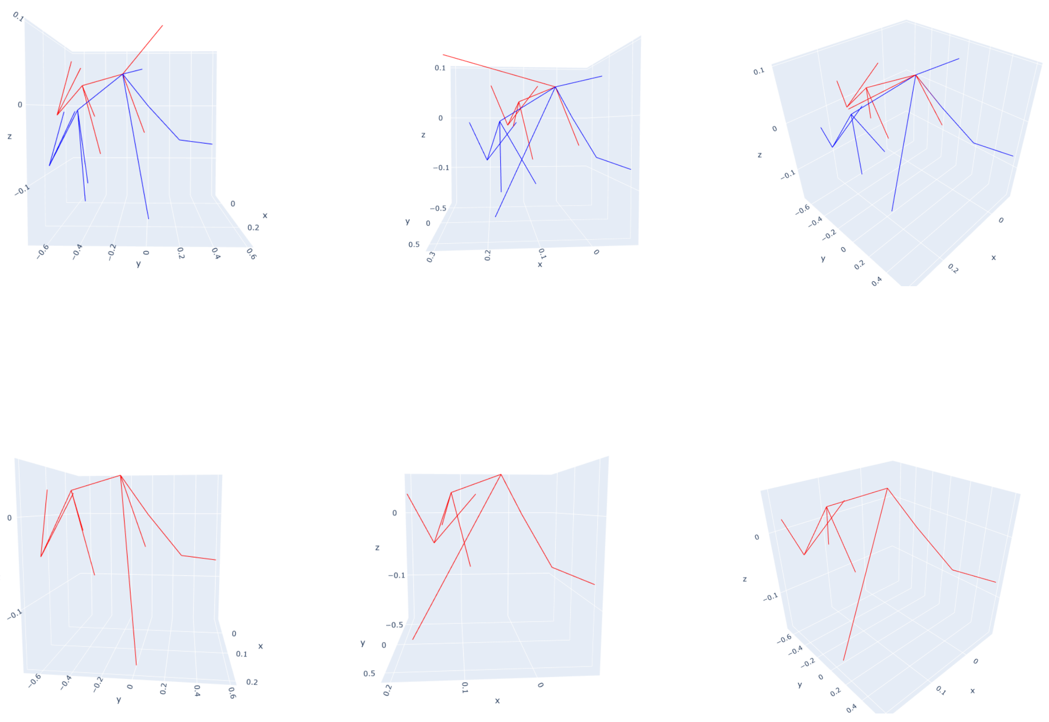


Fig 5.3: Top Row: Two Parent 3D poses (Red & Blue) Viewed from different angles
Bottom Row: Child - crossover of Parents from above from different angles

5.5 Mutation Operator

As motion of mice body is usually continuous, back-to-back motion can somewhat be inferred as slight movement in the bones. Since we know these bones cannot change in length but only in orientation, we can use this exploit this fact by taking existing pose from the original dataset and altering the orientation of bones to generate new poses.

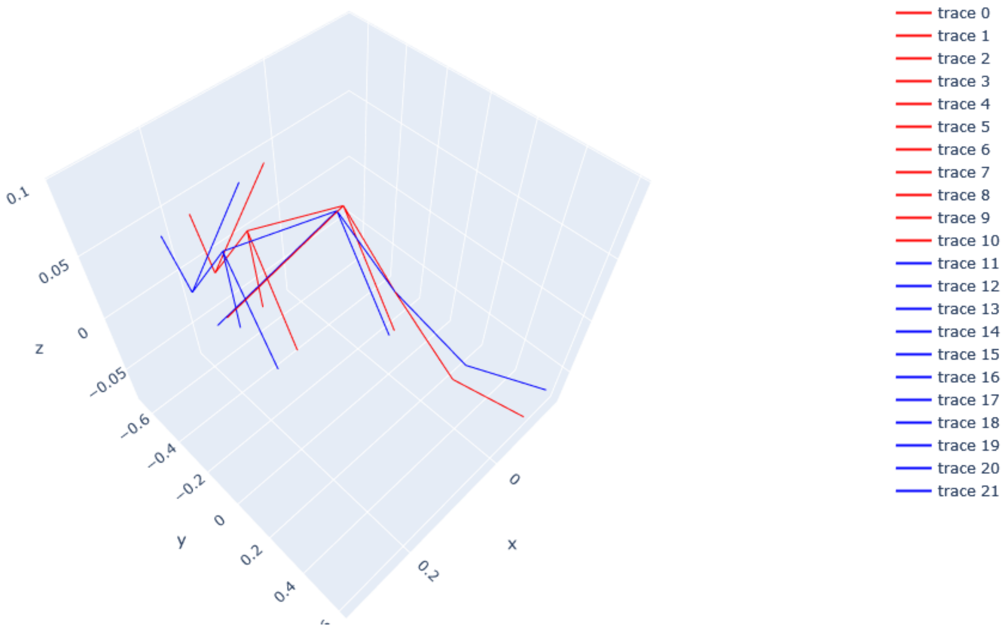
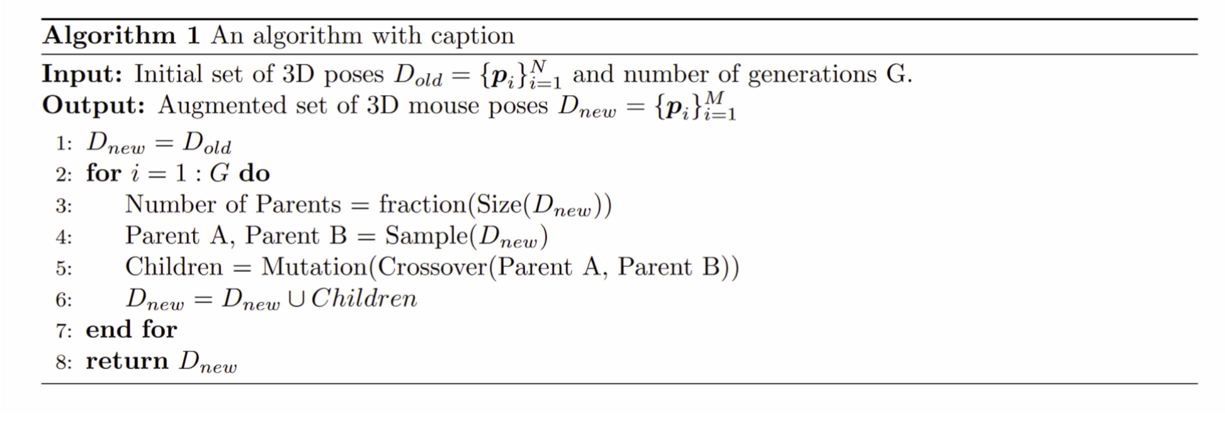


Fig 5.4: Mouse A(Red) and Mouse B(Blue) obtain from mutation of A.

For our case, we know the limb bone vectors can change length. This is because the key points are on the main body and the end of the limb, which have 3 bones in-between them in the actual body. Since there orientations can change individually, the final length of the bone vector in this model won't be constant. So the only parts of the body we mutate in this case are all the bone vectors along the spine. We know these are like threads and the lengths in some way remain the same, this approach might help us generate valid 3D poses.

5.6 Evolution Process

The above operators are applied repeatedly to D_{old} to obtain a new generation by D_{new} synthesizing new poses and merge with old poses. This way, we obtain more valid training data points. This evolution process repeats for G generations and is depicted below.



Algorithm 1 An algorithm with caption

Input: Initial set of 3D poses $D_{old} = \{\boldsymbol{p}_i\}_{i=1}^{N}$ and number of generations G.

Output: Augmented set of 3D mouse poses $D_{new} = \{\boldsymbol{p}_i\}_{i=1}^{M}$

```
1: $D_{new} = D_{old}$
2: for $i = 1 : G$ do
3:     Number of Parents = fraction(Size($D_{new}$))
4:     Parent A, Parent B = Sample($D_{new}$)
5:     Children = Mutation(Crossover(Parent A, Parent B))
6:     $D_{new} = D_{new} \cup Children$
7: end for
8: return $D_{new}$
```

Fig 5.4: Data Evolution Algorithm

CHAPTER 6: EXPERIMENTS

6.1 Cascaded Deep 3D Coordinate Regression

Since there hasn't been much work on this line for mice, we relied on 2D-3D lifter experiments on human data. Going through various papers, cascaded models and graph models were common and seemed to yield good results. We decided to experiment with a cascaded model for this project phase and saved graph models for policy networks. Since the mapping from 2D coordinates to 3D joints can be highly nonlinear and difficult to learn, we adapt a cascaded z-coordinate(only the height) regression model[11]

$$\hat{\mathbf{p}} = \Sigma_{t=1}^{T} D_t(i_t, \Theta_t)$$

where D_t is the t'th deep learner in the cascade parametrized by Θ_t and takes input $\mathbf{i}_t$. The architecture is as depicted in the figure below. Each learner in the cascade is a feed-forward neural network whose capacity can be adjusted by the number of residual blocks. To fit an evolved dataset with plenty 2D-3D pairs, we use 5 learner blocks.

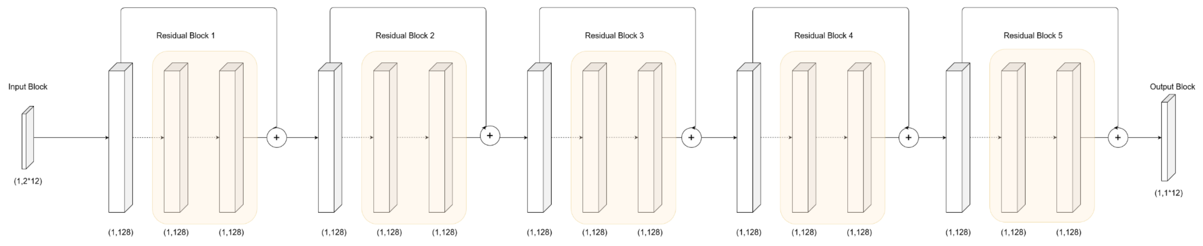


Fig 6.1: Cascaded NN Architecture used for regression analysis

The input block of the neural network has 24 nodes (12x2) for all the 2D projections of 12 key points of the mouse. The output layer has 12 nodes (12x1) which correspond to the heights of each of these key points. Once we have the predictions, it is possible to put the data together (input layer and output predictions) to reconstruct the mouse body. The network parameters were updated using the 'adam' optimizer with a learning rate of 0.001 for 1000 epochs while constantly checking the training loss and the validation loss following a mean squared error loss function.

6.2 Data Augmentation & Results

We began collecting data from blender, which stores the 3D coordinates as mentioned above for 10000 different poses as rows in a CSV file. For preparing the dataset, we first segregate these 10000 rows into 12 3D points from each row and separate them into the 2D components and corresponding z-component (height).

Experiment 1: With a 75%-25% split for training data and validation data, we separate the data and prepare it for training the neural network with it. We train the above neural network in mini_batches of size 64 pulled from the training dataset as part of experiment 1.

Experiment 2: For later experiments with the evolution idea in practice, we choose a tenth of the number of data points in our dataset for the number of crossovers. So when we start with 10000 data points and we follow it up with 5 generations, we obtain 999, 1099,1209, 1330, and 1463 new data points for each generation. At the end, we have a total of 16099 data points (before splitting into train and validate) for experiment 2.

Experiment 3: Similar to the above setting, we run a similar training session but with 10 generations instead of 5. Following this procedure on 10000 data points to start with, we obtain 999,1099,1209,1330,1463,1609,1770,1947,2142,2356 new data points for each generation. At the end, we have a total of 25923 data points (before splitting into train and validate) for experiment 3

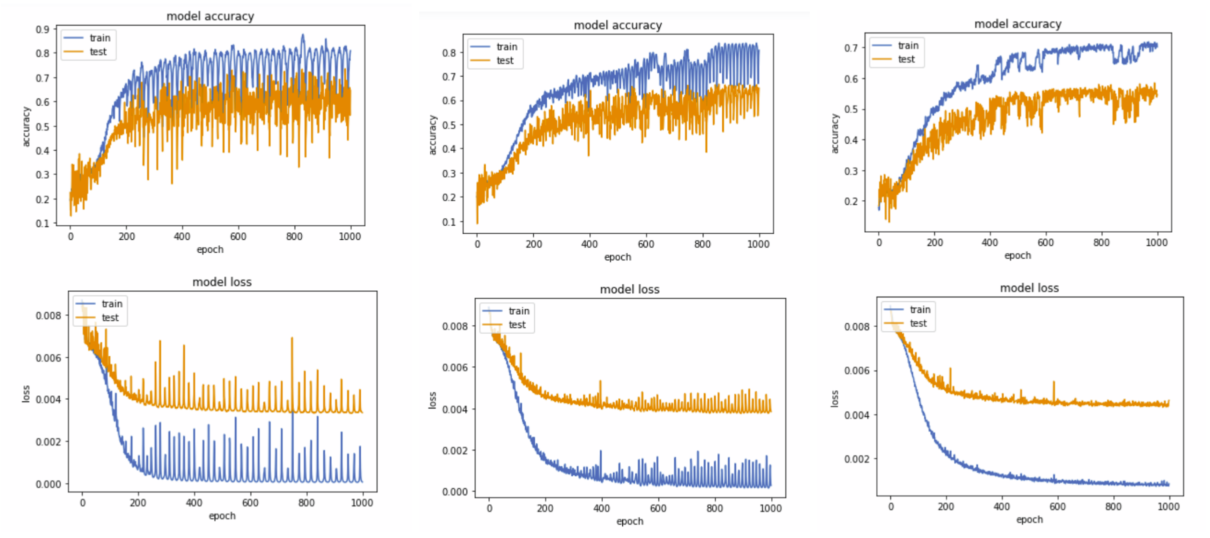


Fig 6.2.1: Accuracy and Loss Measures Visualization

From a few observations purely from the training perspective, we could see that the model slightly kept getting better accuracy and slightly lower validation and training loss as we kept going for experiments with more generations. But it is hard to conclude that the cause for the same was better data samples. Simply put, this could also be because of more interpolated samples in training and validation sets. To tackle this problem, we can use other metrics but unfortunately, we do not have other methods or datasets similar to ours to compare our performance with. Here are some predictions from these models compared and displayed along with the truth models.

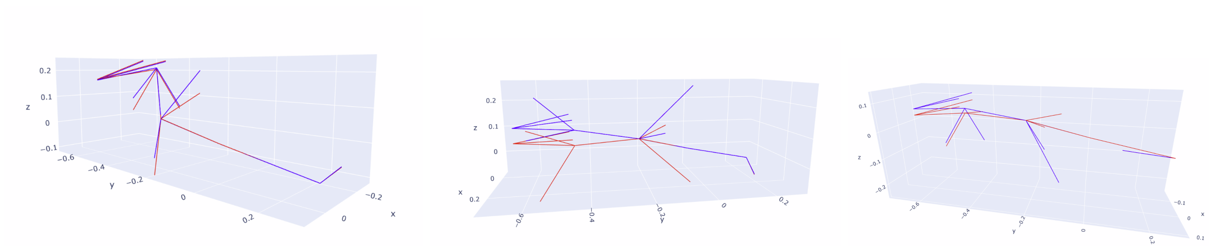


Fig 6.2.2: 3D Keypoints (Red:Predicted;   Blue:Truth)

Here are some poorly predicted examples.

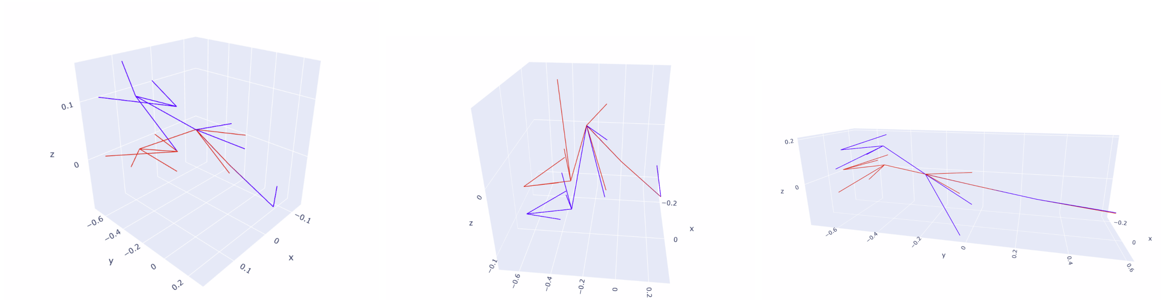


Fig 6.2.3: 3D Keypoints (Red:Predicted;   Blue:Truth)

A visual inspection of cases picked up at random helped us gain better insight into the model's performance. It appears that the model has done decent with models and poses of mice that look 'natural' to the human eye. It is mostly for those weird poses (An example would be where the mouse stretches all its limbs out) that the model is performing very poorly. Obviously, it's not like the function has some sort of intuition to how a mouse in real life moves. This pattern of prediction shows us how crucial the dataset generation is. At the moment, we have done it by randomizing the blender control elements, and the model has picked up the most occurring poses quite well.

Mathematically analyzing, for each 2D point, there can be an infinite number of solutions for the third coordinate, but the number of constraints (the number of bones) is limited, and the set of inverse solutions that we can obtain is not unique. For any mapping that isn't one-one, it is crucial that the solution points we take all fall in one set (In our case, realistic poses), so the one-many type dataset is more skewed to that one class in the range that is more realistic. This way, our function approximator also learns to predict more realistic poses.

CHAPTER 7: THE GAN EXPERIMENT
(Previous Work)

7.1 Premise

Placing the camera on a similar spot as that of the JAX workspace, it is possible to create renders of the mouse. A default mouse pose renders is shown below side by side with a JAX image. There is a gap between the simulation and the original image from the dataset.

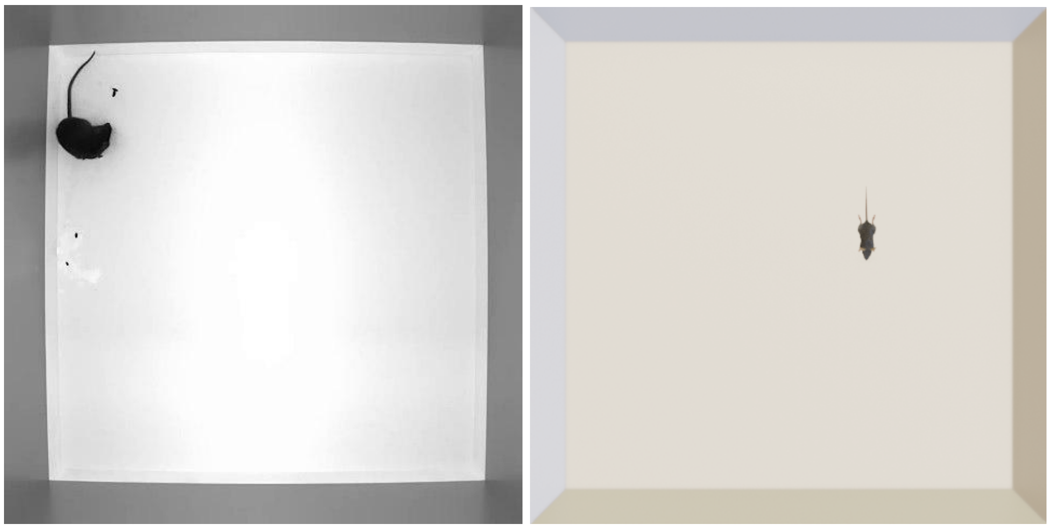


Fig 3.2.2: JAX Dataset(left) & Blender Environment Render(right)

7.2 Real2Sim Gap

To use the rendered image, we need it to look as close to the real image with some added noise. To achieve this, an encoder-decoder framework can be used. If trained completely, this encoder-decoder will take the rendered image from the blender and produce the corresponding real-like (we shall call this synthetic data) image. This can be achieved using a conditional generative adversarial network similar to pix2pix.

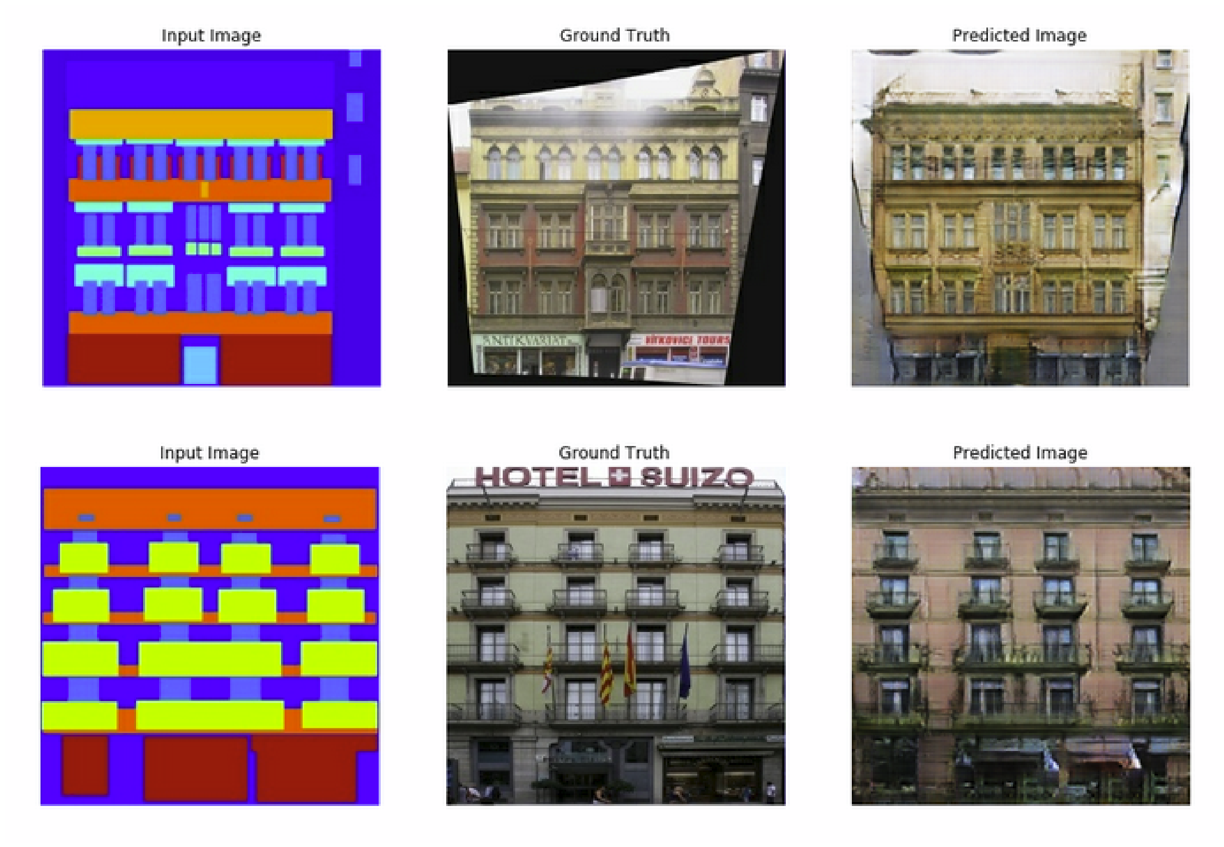


Fig 3.3: Image-to-Image Translation with Conditional Adversarial Networks

With a few images of Set A (Blender Renders in our case) and Set B (JAX dataset in our case), pix2pix trains an encoder-decoder that transforms Set A images to look as close to Set B (as illustrated in the image).

There is a catch to this though. While generating Set A(Blender), we need to make sure that the 2D points of Set A match that of 2D points in Set B. But unfortunately, for the Set A case, that we obtain from blender, we don't really have control over the $p_i \forall i \in$joints. We only have control over the blender elements (as discussed before) specifically $c_j \forall j \in$ set of control elements. These elements are the ones that we could freely move in the mouse and p_i's are decided by these.

$$[p_0, p_1, \ldots, p_{11}] = f(c_1, c_2, .., c_{10})$$

We tried building the inverse function $f^{-1}(p_0, p_1, \ldots, p_{11}) = [c_1, c_2, .., c_{10}]$ trained with data from blender again but it didn't yield good results (again, it is an inverse problem with multiple solutions and random data generation hurt the process more than help).

CHAPTER 8: HIERARCHICAL REINFORCEMENT LEARNING (Previous Work)

Model B, once trained, can also be used in parallel with other existing pose estimation models that return 2D poses for a more feature-rich dataset. After a few post-processing steps, this data is converted to state representations and is pushed into the Reinforcement Learning module for option discovery.

8.1 Setting up Options Framework

The mice's behavior is studied in an imitation learning setting for this project. Imitation learning involves inferring the control policy using a set of demonstrations of the behaviour of a human or animal. The system is modeled as discrete-time, having at time t a state st in some state space S, such that taking action at from action space A induces a stochastic state transition st+1 ~ p(st+1|st, at). The initial state is distributed s0 ~ p(s0). The demonstrations are trajectories of mouse behavior, consisting of a sequence of states and actions ξ = (s0, a0, s1, a1, ...sT, aT) of a given length T

In the options framework as described in 2.2, an option for this application will be a low-level behavior that can be invoked by the high-level behavior trait. Formally, an option h in an options set H is specified by a triplet < Ih, πh, ψh >. Initiation set Ih is the set of states where the option can be invoked if the previous option terminates. Control policy πh is a probability distribution over actions given the current state. The stochastic termination condition ψhϙ[0, 1] is the probability that the option will terminate. The high-level policy ηh defines the distribution of options given to the state.

We assume that the mice behavior trajectories ξ = (s0, a0, s1, a1, ...aT −1) are generated by a hierarchical policy. We have the gradient update rule, which will be used for updating the neural network parameters[11] corresponding to each option. There are 2 sets of neural networks for every option. It is assumed that the initiation set for all options is the state space.

8.2 Addition of More Layers

An implementation of the Deep Discover of Continuous Options was implemented earlier, and now additional layers of hierarchy can be added in such a way that it utilizes the new addition of dimensions in the state representation well.

CHAPTER 9: FUTURE WORK & REFERENCES

This is a very impactful project with a good pipeline. There are many sub-problems we handle separately to obtain final results. Each of these sub-problems can be worked on separately and solutions can be improved. Here are some modules involved and few suggestions:

1. **2D Pose Estimation:**

The accuracy of the model can be improved. Currently, we are using a model that was trained with hand-labelled data. Using the GAN idea proposed, if we are able to obtain a good inverse function (either mathematically or data driven) we could generate more training data for the model.

2. **2D-3D Model:**

Currently, data was generated by randomizing blender elements. Obtaining more realistic data and data from various sources, we can project it to a 2D-3D setting and training the original model could help. We could also use 'Graph Neural Networks' as our prediction model. Since most physical systems are better represented as graphs, this has a lot of potential. Many GNN-based 2D-3D lifters have been trained for humans and have worked well.

3. **Option Discovery:**

The current option discovery algorithm with results only used one meta-policy (ie. one layer). We could introduce more layers and test them out. Along with this, we could also use newer option discovery algorithms. There also seems to be good results with using GNNs again for policy networks instead of traditional feed-forward neural networks.

References

1. **Berman, G. J., D. M. Choi, W. Bialek,** and **J. W. Shaevitz** (2014). Mapping the stereotyped behaviour of freely moving fruit flies. Journal of The Royal Society Inter-face, 11(99), 20140672.

2. **Botvinick, M. M., Y. Niv, and A. G. Barto** (2009). Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. Cognition, 113(3), 262–280.

3. **Geuther, B. Q., Peer, A., He, H., Sabnis, G., Philip, V. M.**, & **Kumar, V.** (2020). Action detection using a neural network elucidates the genetics of mouse grooming behavior. bioRxiv.

4. **Marques, J. C., Lackner, S., Felix, R., & Orger, M. B.** (2018). Structure of the ze- ´ brafish locomotor repertoire revealed with unsupervised behavioral clustering. Current Biology, 28(2), 181–195.

5. **Wiltschko, A. B., M. J. Johnson, G. Iurilli, R. E. Peterson, J. M. Katon, S. L. Pashkovski, V. E. Abraira, R. P. Adams,** and **S. R. Datta** (2015). Mapping subsecond structure in mouse behavior. Neuron, 88(6), 1121–1135

6. **Sheppard, K., Gardin, J., Sabnis, G., Peer, A., Darrell, M., Deats, S., Geuther, B., Lutz, C. M.**, & **Kumar, V.** (2020). Gait-level analysis of mouse open field behavior using deep learning-based pose estimation. bioRxiv.

7. **Sun, K., Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu**, and **J. Wang** (2019). High-resolution representations for labeling pixels and regions. arXiv preprint arXiv:1904.04514.

8. **Bolaños, L.A., Xiao, D., Ford, N.L. *et al.*** A three-dimensional virtual mouse generates synthetic training data for behavioral analysis. *Nat Methods* 18**,** 378–381 (2021). https://doi.org/10.1038/s41592-021-01103-9

9.**Geuther, B.Q., Deats, S.P., Fox, K.J. *et al.*** Robust mouse tracking in complex environments using neural networks. *Commun Biol* **2,** 124 (2019). https://doi.org/10.1038

10. **John Henry Holland *et al.*** Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence. MIT Press,1992.

11. **S. Li, L. Ke, K. Pratama, Y. -W. Tai, C. -K. Tang and K. -T. Cheng,** "Cascaded Deep Monocular 3D Human Pose Estimation With Evolutionary Training Data," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6172-6182, doi: 10.1109/CVPR42600.2020.00621.

11. **Deepak Mittal∗, Avinash Reddy†, Gitakrishnan Ramadurai‡, Kaushik Mitra†, Balaraman Ravindran**∗ (2013). Training a Deep Learning Architecture for Vehicle Detection Using Limited Heterogeneous Traffic Data.

12**. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros.** (2018) Image-to-Image Translation with Conditional Adversarial Networks. https://arxiv.org/abs/1611.07004

13. **Krishnan, S., R. Fox, I. Stoica,** and **K. Goldberg** (2017). Ddco: Discovery of deep continuous options for robot learning from demonstrations. arXiv preprint arXiv:1710.05421.

14. **Libby Zhang, Tim Dunn, Jesse Marshall, Bence Olveczky, Scott Linderman "**Animal pose estimation from video data with a hierarchical von Mises-Fisher-Gaussian model" *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, PMLR 130:2800-2808, 2021.

15. **Bardia Doosti, Shujon Naha, Majid Mirbagheri** and **David J Crandall** (2020) "HOPE-Net: A Graph-based Model for Hand-Object Pose Estimation"**.** arXiv preprint arXiv:2004.00060.