Learning Object Structure from Keypoints

Hou Han

4th Year Project Report
Artificial Intelligence
School of Informatics
University of Edinburgh

2023

Abstract

Object structure learning is a task in computer vision and machine learning domains, aiming to empower machines with the ability to comprehend object representations in scenarios. In this study, we employ supervised learning to predict object structures from moving two-dimensional keypoints sequences of the object. Due to the limited availability of labelled datasets, we create and label several datasets of some novelty objects for experimental purposes. Our proposed model exhibits high performance across a diverse range of object datasets.

In addition, we explore the identification of underlying patterns in object keypoint movement within a constrained dataset, enabling the inference of object structures absent in the training set. we conduct a human experiment that highlights the challenges faced by supervised learning when inferring unseen object structures. The results demonstrate that our model outperforms human judgment, especially for those objects captured from extreme angles, but they are not excellent. Based on results, we also elucidate the difficulties of employing supervised learning for generalizability in this task.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 259168
Date when approval was obtained: 2023-04-10
The participants' information sheet and a consent form are included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hou Han*)

Acknowledgements

"As ephemeral as the morning dew, much precious time has regrettably slipped away."

First, I would like to thank my supervisor, Dr.Hakan, for his guidance, patience, and unwavering support on my project throughout the 4th year. I am eternally grateful to my loving parents, who support me emotionally and financially. I would also like to acknowledge my dear Miss Zhang, who is the early sun that illuminates the morning dew, bestowing upon each droplet an endless radiance and warmth.

In addition to the individuals mentioned above, I would like to express my gratitude to everyone who has supported me throughout my four years at university. This includes those who accompanied me during the challenging times in Edinburgh amidst the pandemic in the second half of my freshman year, those who were with me in my hometown when I was in poor spirits during the remote learning period of my sophomore year, as well as the companions who learn together during my junior and senior years in Edinburgh.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Learning the structure of objects is a task in computer vision that aims to infer the structure of objects in the real world from given images, videos, or other object representations. A widely known computer vision task is pose estimation. Traditional pose estimation involves using pre-annotated semantic keypoints and structures of a specific object to locate the keypoints' positions in images or videos. The semantic keypoints of an object include the center points of its main parts, such as the head and hand centers, and rigid and articulated connections between the main parts, such as human joints. However, traditional pose estimation focuses only on specific objects, such as human pose estimation, and it can only learn keypoints from the given structure, but not the other way around. The demand for studying structures of objects in different fields is gradually increasing, so one research direction is to explore whether it is possible to infer the structure of an object from its moving keypoint sequence. It can precisely locate the positions of object components by analyzing complex variations in keypoint features, thereby facilitating human comprehension of the object's structure and state for subsequent operations. In an industrial context, this would provide significant convenience for tasks such as robotic object manipulation with a mechanical arm. A challenge of the problem is the lack of such a large annotated dataset[10], and it is impossible to label all object structures and semantic keypoints, so current research has turned its attention to unsupervised learning[23], but one of drawbacks of unsupervised learning is that it cannot always provide a good accuracy on a task.

In this project, we aim to design a supervised learning method to learn the relationship between explored 2D object keypoint sequences and infer the structure of this object, which is also the connection relationship between keypoints. Additionally, we hope to evaluate whether supervised learning is suitable for this task, particularly with regard to its generalization ability, i.e., whether the model can attempt to learn the patterns of keypoint changes in small samples and use them for predicting other objects in the future.

1.2 Research Objectives

The main research problem in this project is:

Given a 2D coordinate sequence x of the keypoints of an object, with a size $F \times K \times 2$, where F means the number of frames, K denotes the number of keypoints in this object and 2 represents horizontal and vertical coordinates in 2D world, how can we use a supervised learning algorithm $f(x)$ to obtain the structure of the object, specifically, the connection relations among the keypoints?

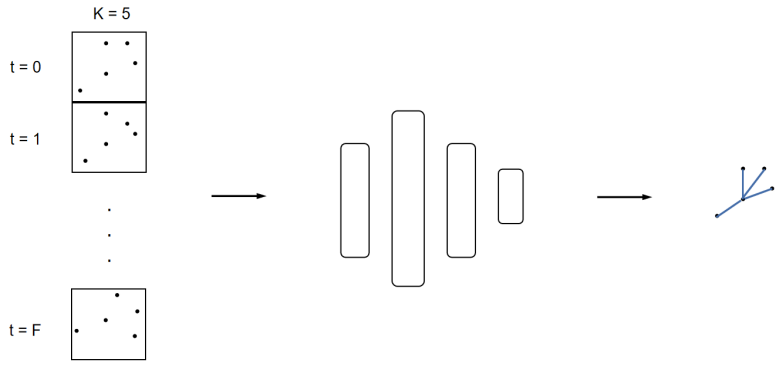


Figure 1.1: Inferring object structure from moving keypoint sequences through neural network

For this problem, we established four fundamental objectives and two addition objectives as milestones.

For **fundamental** objectives:

- Creating datasets for different kinds of novelty object.
- Designing a deep learning network to learn the structure of objects.
- Designing experiments to show the model is able to learn the structures of single object.
- Designing experiments to show the model is able to learn the structures of multiple objects.

For **additional** objectives:

- Designing experiments to investigate the generalization of the model, i.e. its ability to predict the structure of unseen objects.
- Improving the model to enable it to accept objects with varying numbers of keypoints as input.

1.3 Contribution

The main contributions of this research are outlined below:

- Generation of six novel object datasets with different numbers of keypoints. The objects move randomly in a 3D world and are projected from multiple viewpoints onto a 2D plane.
- Design and implementation of a deep learning network for predicting the structure of an object from keypoints by using supervised learning.
- Structural prediction experiments and analysis for single or multiple objects.
- Structural prediction experiments and analysis for seen and unseen objects in training set.
- A human experiment to demonstrate the ability of humans with rich prior knowledge in handling the problem of predicting object structure from a sequence of keypoints, and comparison with our model.

1.4 Outline

Chapter 1 Introduction is to show the motivation for this project, plan and milestones before starting the project, and contribution of the project after finishing it.

Chapter 2 Background provides an overview of previous and state-of-the-art methods of pose estimation and its downstream tasks, as well as research that can be used as a reference for designing the network, encompassing key concepts and classic techniques in neural networks and machine learning.

Chapter 3 Dataset shows the techniques of Mujoco and DeepMind Control Suite, strategy details for designing a dataset and data preprocessing.

Chapter 4 Methodology presents an overview of the designed neural network structures and explains the necessity of each component. Furthermore, it covers several machine learning techniques used in the experiment.

Chapter 5 Experiment includes several experiments to show the performance of our model and result analysis.

Chapter 6 Conclusion summarises the contents of this project, analyzes the reason why some goals are hard to handle and discusses the future work in keypoints-based structure prediction.

Chapter 2

Background

2.1 Pose Estimation and Its Downstream Tasks

Pose estimation is the process of determining the position of an object or a person's body parts from images or videos, often represented by keypoints. It can be divided into top-down and bottom-up methods in order to handle multi-objects estimation. The top-down method locates the object position using an object detector and then estimates the coordinates of the keypoints. The bottom-up method first estimates each joint, and then feeds the inferred joint to different downstream tasks, such as assigning joints to different bodies, action recognition, vision navigation and pedestrian tracking. Our project is similar to a downstream task in the bottom-up method, but the difference is that the input to our task comes from real keypoints that have not been labelled with skeletal components rather than recognized unreliable keypoints.

Although keypoint detection is not necessary in this project because all keypoints in our project should be guaranteed to be ground-truth, there is a huge lack of research into the utilization of supervised learning from keypoints to infer an object's skeletal structure, so some classic algorithms for pose estimation and its downstream tasks, particularly those catering to multiple categories, may be referenced. The part where they extract the relationships between keypoints from the known keypoints would be helpful.

2.1.1 Human Pose Estimation

The pictorial structure framework (PSF)[11] is one of the early methods to address Human Pose Estimation (HPE). It represents human body as a collection of keypoints and leverages non-linear regressors, such as two-layered Random Forest, to predict their locations.[33][3]

In recent years, with the widespread application and rapid development of deep learning, an increasing number of researchers have turned their attention to deep learning-based methods for Human Pose Estimation (HPE). The introduction of deep learning has led to an explosion of HPE methods, and their performance has improved significantly. Toshev

and Szegedy[43] initially used CNN to estimate human skeleton, and they named the estimator as DeepPose. Kaiming He et. al. proposed a flexible network structure Mask R-CNN[13] which is a way to solve object detection and instance segmentation tasks. The existing of higher hardware computing power allows a deeper network in deep learning. VGG19[36] was proposed by Simonyan an Zisserman. Based on VGG19, Zhe Cao et. al. proposed OpenPose[4] in 2019, which includes two network branches, one for predicting the confidence map for each body part, the other one for predicting a part affinity fields. ViTPose[46] employs plain vision transformers as backbones for feature, combined with a lightweight decoder for estimating poses extraction. The largest ViTPose model with more than one billion parameters achieves a new state-of-the-art performance in the MS COCO Keypoint Detection benchmark.

2.1.2 Robot Arm Pose Estimation

The robot arm is a typical, simple, and easy-to-learn articulated object. Each point of the robotic arm interacts with another point, and learning the keypoint features of the robotic arm is essential.

Rodrigues et al.[32] proposed a deep learning framework that is capable of detecting 3D robot arm keypoints and forecasting the future motion of the robotic arms according to detected keypoints. The presented approach amalgamates a self-calibrated convolutional method[28] and an Extreme Learning Machine (ELM)[17] neural network for pose estimation task, and a conventional encoder-decoder architecture consisting of RNN(LSTM and GRU) models is utilized for the prediction of future frames. Although we believe that there are potential drawbacks of the LSTM and GRU models in capturing spatial structures, they manifest superior performance in processing temporal features in this research endeavor.

2.1.3 Hand Pose Estimation and Hand Gesture Recognition

Hand Pose Estimation is tracking the joints of hands from the RGB or Depth images or frames of videos. Hand gesture recognition is an algorithm that classifies gestures based on known keypoints. Dynamic gesture recognition is similar to our task, which also uses keypoint sequences for downstream processing.

With the emergence of attention mechanisms, a commonly used method for keypoint-based dynamic gesture recognition is to leverage spatial-temporal attention to extract features of a keypoint graph[30][6]. Yuxiao Chen et al. proposed an algorithm called Dynamic Graph-Based Spatial-Temporal Attention (DG-STA)[6], which explores the relation between pairs of keypoints by their spatial and temporal structure. Spatial features of the graph are calculated by the multi-head self-attention of all points within the same time frame, while all points across different time frames are subject to attention calculations to capture temporal features of the graph. This method is also used in human action recognition [35]. Inspired by this method, we applied a similar information-extraction approach to our project.

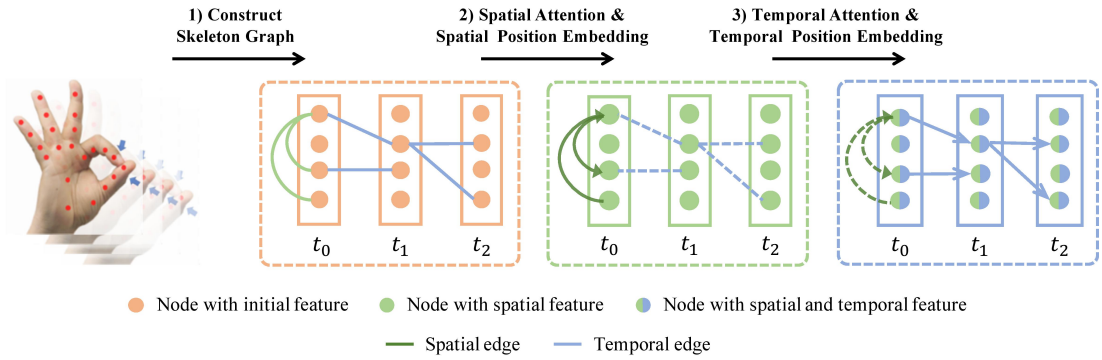


Figure 2.1: DG-STA

2.1.4 Pose For Everything

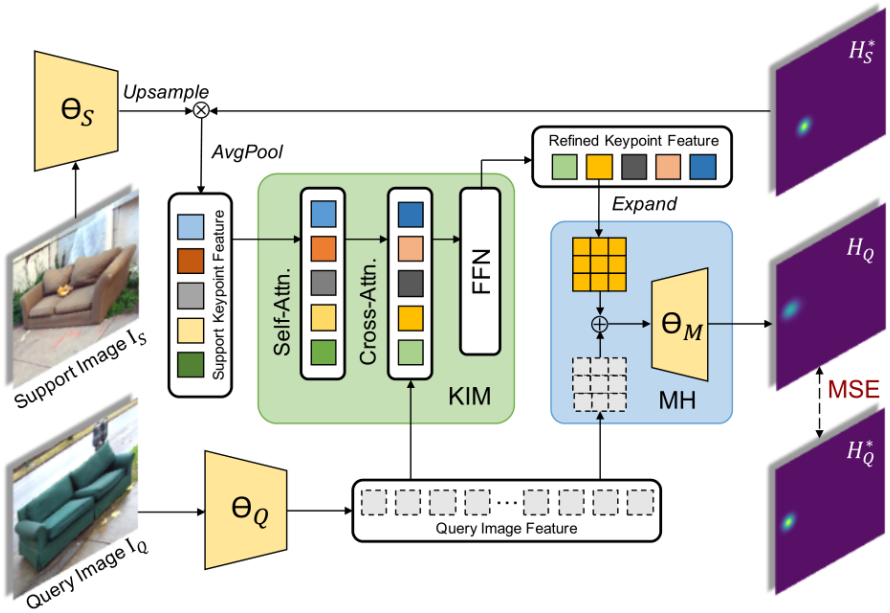


Figure 2.2: POMNet

In 2022, Lumin Xu et al. [45]. introduced a novel task for pose estimation, which aims to estimate the pose of everything using one model. There are three major challenges in this task. First, mapping keypoint locations from thousands of annotated images. Second, different objects have different numbers of keypoints with varying definitions, and connecting structural information takes work. Third, there is a lack of large-scale, multi-type datasets. For this new task, they also proposed a novel network named POMNet. The POMNet learns from the image and keypoint information of support images and predicts the keypoint locations and their structures of a query image. Two parallel feature extractors are used to extract keypoint features from support images and image features from query images. The Keypoint Interaction Module (KIM) refines the

keypoint features through three attention blocks, each of which includes a self-attention component and a cross-attention component. Finally, the Matching Head(MH) combines the refined keypoint features with the image features of query images to predict the keypoint locations and structures of the object.

Our task differs from POMNet in that POMNet requires one or more given support samples and the target sample to be predicted as input. In contrast, our study uses only a sequence of keypoints as input, which means the input is a dynamic form that lacks support samples, and the target keypoint positions are already calculated in advance.

2.2 Skeleton Representation

In pose estimation, researchers usually use bone maps to connect keypoints effectively. These bone maps or skeleton representations are presented as affinity fields [4] and explicitly expressed as offset values[31], and they are used into supervised pose estimation tasks for different object subjects.

We can view the skeleton as a topological structure or graph, in which the ground truth of the graph connectivity is pre-defined by humans. The manual annotation of ground truth skeletons for each image or video can be costly. We have to define the keypoints for each object category and label the coordinates of all keypoints in the image. Nevertheless, the reason why keypoint-based HPE models can still demonstrate outstanding performance is the existence of annotated human skeleton datasets such as MS COCO Keypoints[27], Human3.6M[18], MPII[2], and LSP[20], which are significantly extensive. When it comes to improving the generalization capability of these models, it is desirable to utilize diverse datasets from different objects, which would require manual annotation for obtaining their ground truth skeleton representation as supervised training data. Therefore, we have to create our own datasets to address the shortage of annotated dataset in supervised learning.

2.3 Unsupervised Object Structure Inference

The other way to handle the lack of annotated datasets is unsupervised learning. Recently, most researchers have been studying how to use unsupervised learning to extract the keypoints coordinates and link them together. One approach is to train models that learn to match object parts that are equivariant to geometric transformations[41][40][39], while another approach is to use structural representations that incorporate both keypoint locations and appearance to encode and reconstruct images[19][47].

The latest work is from Titas Anciukevičius et al.[1]. The work designed a model that aims to predict the skeleton structure of an articulated object from a single image. It includes two neural networks. The first network generates a foreground points cloud for a given image, while the second neural network takes the output of the first one and images as inputs to predict the structure of objects. The neural point transporter (NPT)

ultimately reconstructs the target image, enabling unsupervised learning.

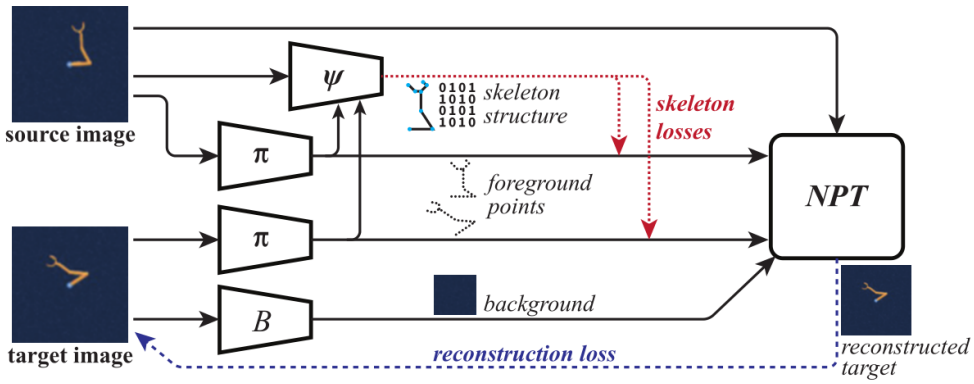


Figure 2.3: The network from Titas Anciukevičius et al.

In the real world, objects activate following the causal law. The ability to discover latent causal mechanisms from data is a critical technical issue in building intelligence. Yunzhu Li et al.[26] proposed Visual Casual Discovery Network(VCDN) to learn the structure of objects in a given video and predict their changes of the next time point. It includes a perception module for extracting keypoints, an inference module for inferring exogenous variables controlling the interactions between each pair of keypoints, and a dynamics module for predicting future motion.

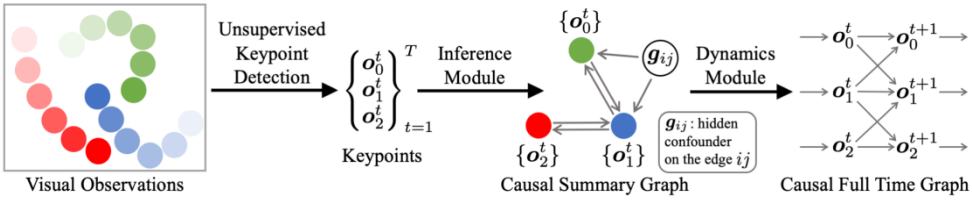


Figure 2.4: VCDN structure

2.4 Deep Learning Algorithm

Deep learning is currently one of the most popular and effective methods in the field of artificial intelligence. Unlike traditional methods, it does not require manual feature extraction. Instead, deep learning relies on a series of transformations and calculations of a neural network to obtain a certain feature representation of data samples. Deep learning is a type of machine learning, which includes both supervised learning and unsupervised learning. As used in this project, the goal of supervised learning is to extract useful information from labeled datasets. As we said before, there are too few labeled keypoint sequence datasets for different objects, so researchers used unsupervised learning, which learn features from unlabeled datasets.

Artificial Neural Networks(ANNs), also known as Neural Networks(NNs), are composed of a series of interconnected artificial neurons, which simulate the network structure of biological neural systems. The input of NNs always is the features from external data samples. A large number of neurons transmit information, and each neuron generates a single output that would be sent to the next layer of neurons based on the weighted sum of its inputs from the previous layers of neurons(or the initial data input), as well as a potential bias term. The formula of the process is shown as follows:

$$\begin{aligned} y &= f(\sum_{d=1}^{D} w_d x_d + b) \\ &= f(W^T x + b) \end{aligned} \tag{2.1}$$

where D denotes input size, w_d and x_d is dth input and weight, f denotes an activate function, y denotes the output, W denotes weights and b denotes a bias term.

The behavior of neurons transiting information in a neural network is named as forward propagation, which is the process of network inference. Meanwhile, we would like to seek the optimal parameter in the neural network, such as weight coefficients W and bias term b.The difference between the computed result by NNs and the target value is referred to as the loss. The backpropagation algorithm[34] aims to find the best solution for those parameters, by computing the derivative of the loss function with respect to each coefficient, that is the direction of descent. As one of the most popular backpropagation algorithm, gradient descent are used in most deep learning models.

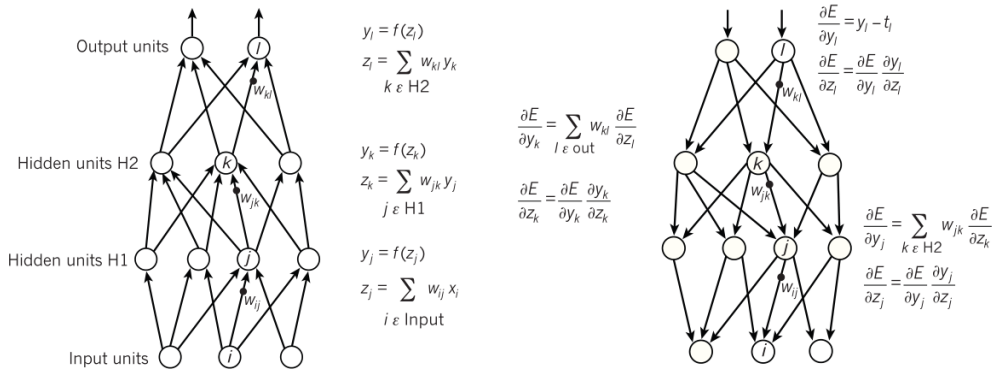


Figure 2.5: LeCun's backpropagation diagram[24]

2.4.1 Recurrent Neural Network

The ability of Recurrent Neural Network(RNN) to handle arbitrarily long sequences of temporal data is achieved through RNN's neurons with self-feedback connections. RNNs possess the ability to store memories, although these memories may fade over time due to the Vanishing Gradient Problem during backpropagation. However, the short-term memory capacity of RNNs remains powerful. The formula of the process is shown as follows:

$$
\begin{aligned}
h_t &= f(Uh_{t-1} + Wx_t + b) \\
o_t &= \sigma(Vh_t)
\end{aligned}
\tag{2.2}
$$

where x_t denotes the input of the neural network at time t, h_t denotes the neuron activity value of the hidden layer at time t, o_t denotes the output at time t, b denotes the bias terms, W, V, and U are the weights from different status.

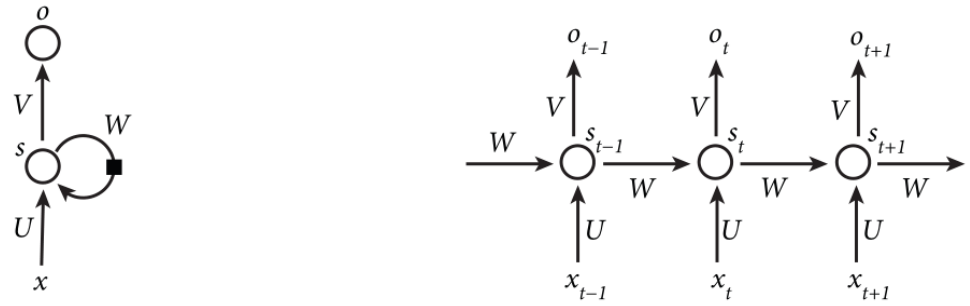


Figure 2.6: LeCun's RNN diagram[24]

2.4.2 Long Short-term Memory and Gated Recurrent Unit

In RNN, one weakness is that it cannot deal with long-dependency relations. For example, in English, the form of the predicate verb is often influenced by the subject, but sometimes they are far apart from each other, so Long Short-term Memory (LSTM)[16] always is used to solve this problem. LSTM is a gated mechanism that includes three gates controlling the degree of information retention. The input gate controls the degree of input information retention to be passed, the output gate controls the degree of information retention for the activation of the current memory cell to be passed to the output layer, and forget gate controls the proportion of previous information passed to the memory unit. LSTM uses memory cells to record temporal data and effectively addresses the Vanishing Gradient Problem in RNN through gated mechanisms that handle long-term dependencies.

Gated Recurrent Unit(GRU)[7] is a variant of LSTM, which includes a reset gate and update gate. The Reset gate decides how much the unit updates its activation, and the update gate is similar to the forget gate in an LSTM unit. In most tasks apart from machine translation, LSTM and GRU perform comparably. However, GRU has the advantages of faster computational and fewer parameters due to the lack of a gate and memory cells. Therefore, in our task, GRU is a good method to process temporal features of the keypoints sequence.

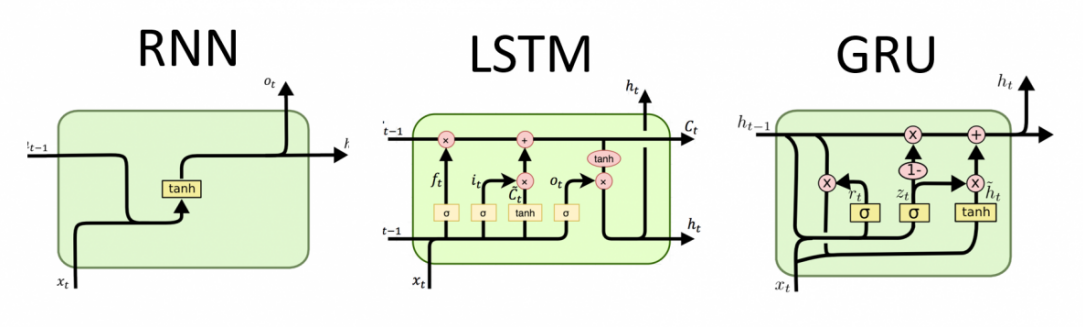


Figure 2.7: RNN, LSTM and GRU diagram

2.4.3 Convolutional Operation

Convolution is a mathematical operation frequently employed in the field of image processing and signal processing. One of the most prominent methods constructed using convolution operations in deep learning is the Convolutional Neural Network (CNN)[25]. A convolutional kernel moves over the input matrix throughout the convolution process, calculating the dot product between the local region and the kernel elements at each step. This enables the convolutional kernel to capture local features in the input data.

Channels refer to a set of features at a specific depth dimension in the data. In CNN, channels describe different types of information and features. For example, in the field of image processing, RGB images have three channels, including the intensity of red, green, and blue. However, channels could be varied through computation in CNN, so the network could simultaneously expand more channels information or fuse different channels' information, thereby improving the model's performance and generalization capabilities. In some classic models, channel transformations are often used to expand the limited available features, such as in LeNet[25], VGG16[36].

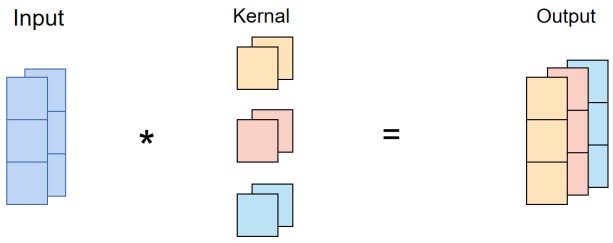


Figure 2.8: CONV1D computation process.

One-dimensional convolutional layers are a popular choice for expanding or compressing features in deep learning. They do not recognize spatial patterns of matrices but rather fuse or expand channels. The figure shows a method of expanding a two-channel one-dimensional vector into a three-channel one-dimensional vector using convolutional kernels from three different channels. It can actually be regarded as a linear layer, but

with fewer parameters. We will use this approach in our model's Keypoint Embedding Generator.

2.4.4 Self-Attention

The core target of the attention mechanism is to shift focus from the whole to those important points. When humans observe and perceive an object, they perform a global scan of the object and selectively identify the areas of focus, to get more detailed feature information related to the focal target while ignoring other irrelevant information. The concept of attention mechanisms has been transferred in the field of deep learning to extract key information from data.

Self-attention is a variant of the attention mechanism, whose basic idea is to use the mutual correlation information between the input to automatically determine the allocation of input weights. Unlike attention mechanisms, it reduces the dependence on external information input and is suitable for capturing correlations within data. The processing of self-attention is as follows:

- **Initialize the Query, Key and Value**

  $$Q = W_q(X), \qquad K = W_k(X), \qquad V = W_v(X)$$

  where Q, K, V is the query, key, and value, W_q, W_k, and W_v are different transformation matrices. Query is a representation of the current input element which compares with other vectors to compute attention weights for its own output. Key represents all input elements for compatibility calculation. Value is used to compute every output vector based on these weights for generating the output.

- **Calculate the self attention scores**

  $$S = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

  where S refers to the distribution of correlation or similarity between each sample and each piece of information, represented by attention scores. d_k denotes the dimension of the key vectors.

- **Calculate the attention value**

  $$\text{Attention} = SV$$

  where Attention denotes the final result of the self-attention.

- **Multi-Head Attention** In some self-attention models, they always use a technology named multi-head attention based on an attention mechanism. Multi-Head attention uses multiple queries to calculate the attention in parallel. Each attention head focuses on a different part of the input information and will be concatenated at last. The step of multi-head attention is that:

  $$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$
  $$\text{Multihead} = \text{Concat}(\text{Head}_1, \text{Head}_2, ...\text{Head}_n)W^o$$

Here, Head represent the attention values from different attention heads, and W_o is the output transformation matrix.

- **Padding Mask** In Machine Translation, because the sentences always have different lengths, the input size would be different. To mitigate this issue, sentences are either truncated or padded with meaningless tokens to maintain the same length. In our additional task, which involves processing inputs with varying numbers of keypoints, masks are a suitable solution. To apply the padding mask, we first create a binary mask matrix M representing whether the same position of score matrix S is a padding token. The value of these paddings will be set to negative infinity in the score matrix S, whose equation is:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} \odot M\right)V,$$

We execute the self-attention operation on the input layer using the previously described steps to obtain the attention-based representation of the final input vector. Each element of these representations encapsulates the attention relationships among all elements within the entire vector. In the Transformer architecture proposed by Vaswani et al.[44], self-attention was employed as the core component in both the encoder and decoder layers for the first time.

Chapter 3

Dataset

3.1 MuJoCo

Multi-Joint dynamics with Contact (MuJoCo)[42] is a physical engine used to simulate the motion of multi-joint robots in the real world. It is based on fundamental physics, and dynamics, including multi-joint dynamics, contact dynamics, inverse dynamics, etc., for modeling, and it provides users with a platform to simulate joint-based robots. Users can design and create multi-joint robots in MuJoCo through XML format or C++ API. It can record the robot's motion state, trajectory, and visualized operations through its provided tool interface. In this project, we focus on modeling and simulating the motion of interesting articulated objects beyond typical human figures, so MuJoCo is a great modeling platform to build our dataset.

3.2 DeepMind Control Suite

DeepMind Control Suite[38] is a set of continuous control tasks with a standardized structure developed by DeepMind in order to provide performance benchmarks for reinforcement learning models. Over 20 physics simulation environments provided by DM Control Suite were designed based on the MuJoCo engine to simulate real-world environments as realistically as possible. Although their main task is for RL, their models include either classic articulated objects like robot arms and pendulum clocks and some novel articulated objects. We selected 6 of the most meaningful articulated objects for our task and generated the datasets based on them, including Hopper, Manipulator, Cheetah, Walker, Swimmer, and Quadruped.

3.2.1 Hopper

The planar one-legged hopper consists of 9 keypoints and 10 connections. Its motion has a large amplitude, and there is a more relaxed movement limitation. All other joints are articulated. The stable triangle has fixed the relative position of its base and apex.

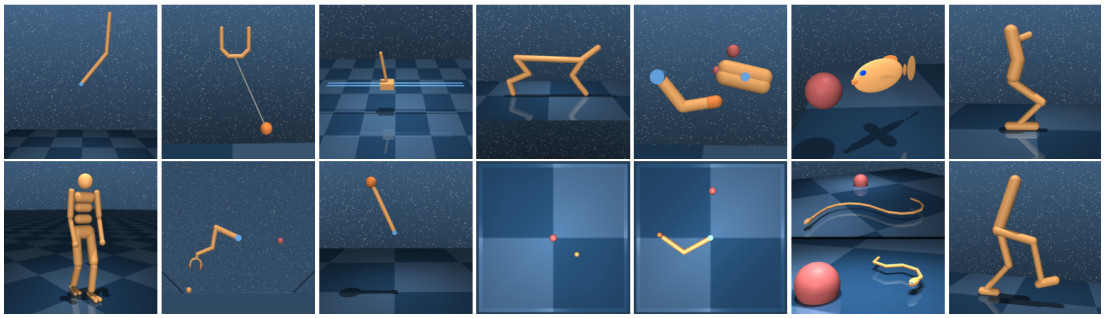
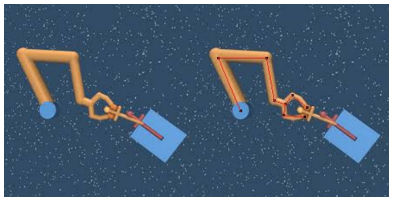


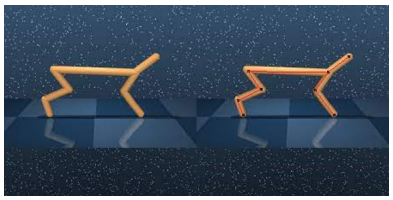
Figure 3.1: DeepMind Control Suite task

3.2.2 Manipulator



The planar manipulator consists of 9 keypoints and 8 connections. Its motion has a normal amplitude, and there is a relaxed movement limitation(the maximum angular separation between points) in the arm part. All connections are articulated.

3.2.3 Cheetah

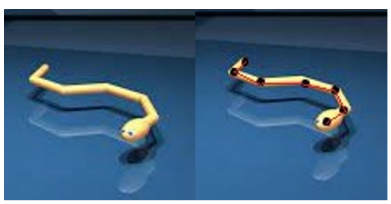


The running planar biped consists of 9 keypoints and 7 connections. Its motion has a small amplitude, and there is a strict movement limitation, that is, the maximum angular separation between points. All connections are articulated, and all points are distributed uniformly.

3.2.4 Walker

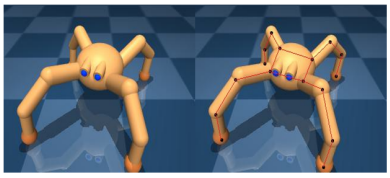
The improved planar walker consists of 10 keypoints and 11 connections. Its motion has a large amplitude, and there is a more relaxed movement limitation. All connections are articulated, and all points are distributed uniformly.

3.2.5 K-Swimmer



The K-link planar swimmer consists of $K(K > 1)$ keypoints and $K - 1$ connections. Its motion has a very small amplitude, the joint configuration is streamlined, and there is a relaxed movement limitation. All connections are articulated, and all points are centralized.

3.2.6 Quadruped



The running planar quadruped consists of 16 keypoints and 16 connections. Its motion has a very small amplitude, and there is a strict movement limitation. All connections are articulated. The distribution of keypoints is messy.

3.3 Data Generation

In the processing of data generation, we performed motion simulations for each model in the MoJoCo 3D world. It is noteworthy that each simulation is random, while maintaining the model's origin movement limitation and speed. We generated videos of these simulations and recorded the keypoint positions of objects in the videos. In DeepMind Control Suite, there are two cameras with fixed perspectives so that they can follow the movement of object, named camera 0, and camera 1, and a fixed position camera, named camera 2. For each camera of each model, we generated 10000 videos and keypoints sequences to create the dataset. However, because some perspectives are not satisfactory, especially those cameras which have a too long or short distance to object, we have to abnegate those datasets.

3.3.1 Keypoint Extraction

One challenge of this task is that the DM Control Suite does not provide the coordinate information of the keypoint positions in their code. Instead, the code provides the 3D coordinates of the corners of the components of objects. For example, the Swimmer consists of a spherical head, cylindrical body segments, and a world floor. Eight corners of the cylindrical body segment are provided. We projected these corner 3D coordinates onto a 2D plane and then calculated the average of the component possible contact surfaces to obtain the 2D coordinates of the component contact points. We labelled the keypoint positions of the objects based on the possible 2D coordinates of the component contact points. Since DM Control Suite randomly arranges these points, and there may be dozens of possible contact points in a single model, so labelling is a time-consuming task, and we have to find the one-to-one correspondence between the points in the picture and the coordinates in the array. Figure 3.2 illustrates the intuitive method for extracting potential keypoints from each object part, where the black dots represent the recorded points in DM Control Suite, and the blue ones are the computed keypoint candidates.

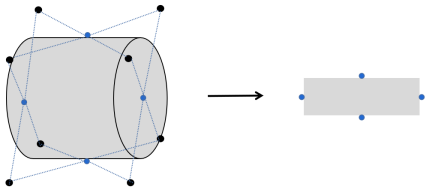


Figure 3.2: Keypoint Extractor

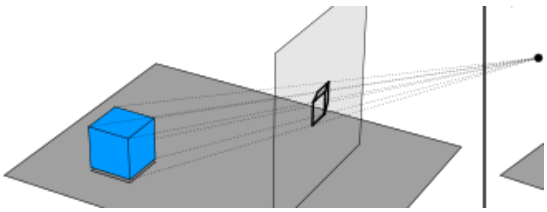


Figure 3.3: Perspective projection

3.3.2 Coordinate Projection

In section 3.3.1, we mentioned that the 3D object requires to be projected onto a 2D plane. In our task, we need to use perspective projection, which is a non-linear projection. When an object in the 3D world is projected onto an image, the perspective projection should have the effects that distant objects appear smaller than nearer objects from the observation point.

Suppose $\mathbf{a}_{\mathbf{x},\mathbf{y},\mathbf{z}}$ denotes the 3D position of a point A that is to be projected, $\mathbf{c}_{\mathbf{x},\mathbf{y},\mathbf{z}}$ denotes the 3D position of a camera point C, $\theta_{x,y,z}$ denotes the orientation of the camera, $\mathbf{e}_{\mathbf{x},\mathbf{y},\mathbf{z}}$ denotes the display surface's position relative to the camera pinhole C[5], and $\mathbf{b}_{\mathbf{x},\mathbf{y}}$ denotes the 2D projection result of $\mathbf{a}$.

Firstly, we defined a vector $\mathbf{d}_{\mathbf{x},\mathbf{y},\mathbf{z}}$ representing the position of point A with respect to a coordinate system defined by camera position C, called camera matrix.

$$\mathbf{d}_{\mathbf{x},\mathbf{y},\mathbf{z}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin(\theta_x) & \cos\theta_x \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} (\mathbf{a}_{x,y,z} - \mathbf{c}_{\mathbf{x},\mathbf{y},\mathbf{z}})$$

Then we apply camera matrix to multiply homogeneous matrix:

$$\mathbf{f}_{\mathbf{x},\mathbf{y},\mathbf{z}} = \begin{pmatrix} 1 & 0 & \frac{e_x}{e_z} \\ 0 & 1 & \frac{e_y}{e_z} \\ 0 & 0 & \frac{1}{e_z} \end{pmatrix} \mathbf{d}_{\mathbf{x},\mathbf{y},\mathbf{z}}$$

Finally, Using similar triangles to calculate the 2D coordinates $\mathbf{b}_{\mathbf{x},\mathbf{y}}$ of the point $\mathbf{a}$:

$$\mathbf{b}_{\mathbf{x},\mathbf{y}} = \frac{\mathbf{f}_{\mathbf{x},\mathbf{y}}}{\mathbf{f}_{\mathbf{z}}}$$

Figure 3.3 shows the macro process of perspective projection, whereby the eight points of a cube are projected onto a 2D plane. Once we obtain these two-dimensional points, the center points of each contact surface, which are potential candidates for keypoints, can be easily obtained.

3.3.3 Link labeling

In human pose estimation, the MS COCO keypoint dataset[27] divides the human body into 17 keypoints and connects keypoint pairs to demonstrate the structure of the object. In our new dataset, we imitate them by connecting keypoint pairs, annotating the object

structure as output, but we do not annotate the type of a single keypoint. The final representation of the object structure is a vector consisting of 0 and 1, and the length of the vector is K*(K-1)/2, which is the number of all possible keypoint pairs. In this output vector, 0 represents no link, and 1 represents a link.

The annotated structure for all objects has already been presented in section 3.2. It needs to be clarified that the number of keypoints and links of the K-swimmer are variable depending on our requirements. In the experiments conducted in this project, we used a 9-swimmer as the experimental object.

3.4 Data Preprocessing

3.4.1 Shuffling

For a machine learning task, the generated keypoint sequence has significant drawbacks. Firstly, our goal is to learn the structure of the known keypoint sequence, so in each video, the order of these keypoints should be completely random to provide a prerequisite for the machine learning model to learn their relationship. Therefore, shuffling the keypoint sequence for each video is essential. After shuffling the annotated keypoints, the connection vectors between the keypoints also need to be updated based on the shuffled keypoint sequence. However, we did not shuffle the keypoints order among frames in each video to keep points sequence consistent.

3.4.2 Frame Selection

Frame selection is a vital step during data preprocessing for our dataset. Due to the different movement speeds of objects in DM Control Suite, there is a large difference in the amplitude of object movement in videos of the same length. This biased movement amplitude greatly affects the extraction of temporal information from object keypoints. If the object amplitude is too large, it is difficult for the machine to learn the keypoint relations between large-scale movement processes, because it is difficult to find the causality, while if the object amplitude is too small, the variation in the keypoints along the time axis may be too slight, resulting in a lack of temporal information. Hence, for objects with the varying speed in DM Control Suite, we extracted video segments of different lengths and assigned an inter-frame interval λ for each object. We selected one frame to be retained in the dataset for every λ frames. Objects with higher speeds are distributed to a designated inter-frame interval λ, which is shorter than that of objects with lower speeds. Specific inter-frame intervals and segment duration parameters are presented in table 3.1. The final frame number for each object in our new dataset is set to 101. Since the frame number was set to more than 101 at the beginning of this project, and recreating the dataset is a costly thing, these keypoint sequences datasets was trimmed to the first 101 selected frames in our code before being inputted into the neural network.

	Hopper	Manipulator	Cheetah	9-Swimmer	Walker	Quadruped
Video Duration(/s)	2	8.2	8	24	2	8
inter-frame interval	4	4	8	8	4	8

Table 3.1: Video duration and inter-frame interval settings for each object. Note that Different models in DM Control Suite are set to different frame rate when creating a video. We used the default frame rate when we created videos.

3.4.3 Data Augmentation

Due to the free-viewpoint camera's propensity to focus on the object's center, keypoints tend to cluster at the very center of the image if the object is not so big in the image. To enhance the model's generalization capabilities, we performed data augmentation on a portion of keypoint sequences from the free-viewpoint camera, ensuring the flexibility of keypoint position.

In the datasets that employ camera center focusing, we incorporated 2,000 keypoint sequences after translation transformation. These transformations were processed after the 2D projection to ensure correction in the two-dimensional space. These translation transformations have certain restrictions, which guaranteed that the object points would not exceed the camera's field of view. Other object datasets did not receive excessive data augmentation, as we believe that the 10,000 randomly generated samples are sufficient to help us extract valuable information effectively.

Chapter 4

Methodology

4.1 Network Architecture

We propose a deep learning model based on the self-attention mechanism and Gated Recurrent Units (GRUs) to address the object structure prediction problem introduced in the Section 1.2. The architecture of the network is illustrated in Figure 4.1.

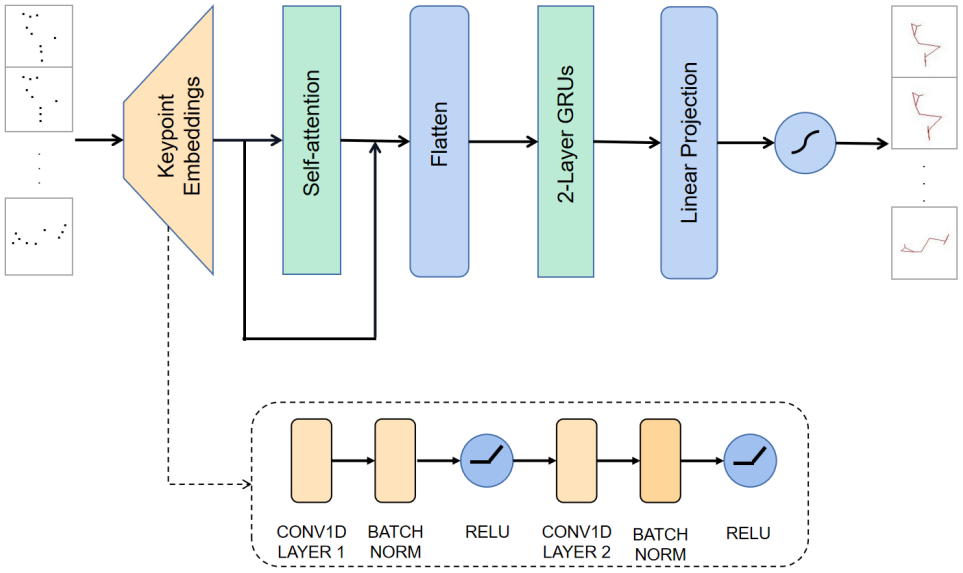


Figure 4.1: Network Architecture

Similar to the spatial-temporal attention graph network proposed for the hand pose recognition task mentioned in the section 2.1.3[6], we also construct our network by spatial and temporal modules for processing spatio-temporal information. The spatial module consists of a single-head self-attention layer and a residual connection, facilitating interactions between each keypoint in a frame and all other keypoints within the same frame. This module calculates attention weights and extracting spatial

features. The temporal module is composed of a two-layer GRU designed to explore the relationships between frames over time.

4.1.1 Keypoint Embedding Generator

Assume that the input keypoint sequence has a size of $[B,N,K,2]$, where B denotes the batch size, N denotes the number of frames, K denotes the number of keypoints, and 2 represents 2D coordinates. Firstly, the input sequence is reshaped into a shape of $[B*N,K,2]$ and fed into a keypoint embedding generator containing two 1D convolutional layers with a kernel size of 1. The first convolutional layer has an input channel of 2 and an output channel of 128, while the second convolutional layer has both input and output channels of 128. A batch normalization and a ReLU activation function follow each 1D convolutional layer in the generator to help the model converge faster, prevent gradient vanishing, and provide non-linear relationships.

The primary purpose of the keypoint embedding generator is to expand the feature representation of each keypoint. Considering that the features represented by 2D keypoints are limited, expanding the keypoint dimension is a good method to acquire more features. Since one-dimensional convolutions are typically used for dimension expansion and linear transformations are generally employed for dimension compression, we utilize two consecutive one-dimensional convolutional layers as the keypoint embedding generator.

4.1.2 Spatial Module

The size of obtained keypoint representation sequence matrix is $[B*N,K,128]$. The matrix is subsequently fed into the self-attention module within the model responsible for spatial information processing. As we need to extract the attention relationship between the keypoints in each frame, we can multiply the frame dimension with the batch size together, similar to the attention heads, which does not affect our computation of attention relations. We have already discussed the specific attention calculation process in the background section, and we follow this method in our model. Notably, during the hyperparameter tuning process, we found that the performance of single-head attention in this model is superior to that of multi-head attention, so we set the number of heads to 1.

We added residual connections[14] after the self-attention module and introduced a learnable parameter γ. The weighting parameter γ is a value used to control the proportion of attention information added to the original information, with its value ranging between 0 and 1. By adjusting the weight of γ, the model can determine the extent to which the attention mechanism contributes to the final output. The calculation formula of the final result after the residual connection is as follows:

$$out = \gamma * ATT(x) + x \tag{4.1}$$

The purpose of adding residual connections is to facilitate the flow of information and control the gradient during backpropagation. They retain some information from the original keypoints and help protect against the Vanishing Gradient Problem[15].

4.1.3 Temporal Module

Thus far, we have described the process of the input keypoint sequence passing through the initial keypoint embedding layer and the attention-based spatial module, which results in an output with a size of $[B*N, K, 128]$. Subsequently, it will be fed into the temporal module. The initial design choices for the temporal module include LSTM, GRU, and temporal self-attention layers. Due to the limited hardware condition during the experiment, we tend to use less amount of memory while ensuring performance. Therefore, we chose the GRU as the framework for the temporal module, which is more simplified than LSTM and attention mechanisms. Simultaneously, we hypothesize that the temporal self-attention layer may help enhance information extraction capabilities, and our experiments will compare the effects of GRU and attention mechanisms on the temporal attention layer.

Since our focus lies solely on extracting temporal features, the spatial positions within each frame can be deemed inconsequential. We begin by flattening the information of all points within each frame, reshaping the matrix size to [B, N, K * 128], which signifies that a batch contains B videos, each with N frames and K * 128 features per frame. This matrix is channeled into a two-layer GRU, where hidden states are configured to 1024. The resulting output from the GRU is [B, N, 1024]. Unlike Natural Language Processing, we need to capture the feature information of the entire sequence, so we only require the matrix from the last processed frame of the GRU, resulting in a matrix with a size of $[B, 1, 1024]$. After the operation of squeezing, its size is $[B, 1024]$.

4.1.4 Linear Projection and Sigmoid Function

Because we are doing a binary classification task, we need to project the vector with 1024 dimensions into a 2D vector that contains only 0 and 1. We employ a Multi-layer Perceptron (MLP) comprising two linear layers. The first linear layer has 1024 input units and 512 output units, and the second linear layer consists of 512 input units and 2 output units.

The Sigmoid function is well-suited for binary classification problems as the last activation function. It helps us to map the input into an output ranging from 0 to 1, allowing for a convenient interpretation of the probability distribution between the two classes (0 and 1). After learning, those probabilities represent whether two keypoints are linked in the object's structure. In addition, it is smooth and differentiable, which is suitable for backpropagation processing. Therefore, Sigmoid is used as the last probability distribution function in our network.

4.1.5 Adam Optimizer

Adam optimizer[22] is an efficient stochastic gradient descent optimizer and is one of the most widely used optimizers in contemporary deep learning tasks. It combines the advantages of two prominent optimization technology, AdaGrad[9] and RMSProp. AdaGrad allocates a learning rate for each model parameter based on the sum of squared past gradients, which is subsequently adjusted in the learning process. On the other hand, RMSProp constrains the learning rate by the squared gradient within a specific

time window. The Adam optimizer adapts the learning rate by computing the first and second moments of the gradients, which respectively is the exponentially weighted moving averages of the gradients and their squares. Our model will employ the Adam optimizer in the learning process. The following equations show the parameter update process of Adam optimizer:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \end{aligned} \tag{4.2}$$

Where θ_t represents the parameters at time step t, m_t is the first-moment estimate (first-order momentum), v_t is the second-moment estimate (second-order momentum), $\hat{m}_t$ is the bias-corrected first-moment estimate, $\hat{v}_t$ is the bias-corrected second-moment estimate, α is the learning rate, β_1 and β_2 are the exponential decay rates for the first and second moments respectively, ε is a small number used to prevent division by zero, and g_t is the gradient at time step t.

4.2 Baseline Models

This section introduces the baseline models we used to compare with our model. The most significant problem in this section is that, as we are researching a novel task, there is a massive lack of universally accepted and standardized datasets within the industry, so we had to create our own dataset. Moreover, there are not related works of supervised learning being employed for object structure prediction tasks, as most tasks completed with unsupervised learning. Comparing models designed for unsupervised learning tasks with our supervised model would be considered unfair. Therefore, in the absence of previous work as a baseline, we have designed two basic structures similar to our model to serve as baseline tasks, which include:

- **GRU + MLP**
  After passing the Keypoint Embeddings Generator as same as our models, this baseline model first flattens the input in the temporal dimension and then adds a 2-layer GRU to extract temporal information from each frame. Subsequently, the GRU connects MLP, including two linear layers and dropout layers to project the features into a vector of size 2. Finally, a Sigmoid Function is utilized to compute the probability for each class. We use the same loss function, optimizer, and evaluation metrics as our model in this baseline model.

- **Spatial and Temporal Attention(STA)[37]**
  This baseline model adopts the same overall structure as our model, which consists of a spatial module and a temporal model. The difference is that both its temporal and spatial modules utilize attention mechanisms. It also includes

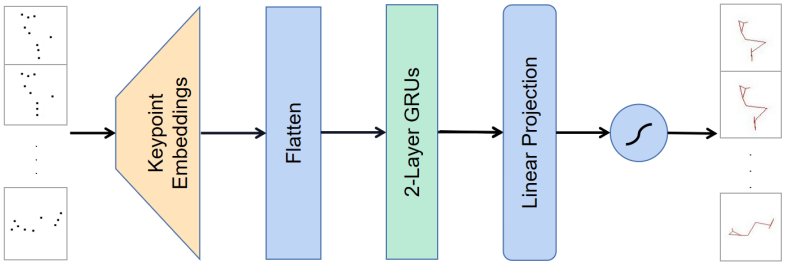


Figure 4.2: GRU+MLP Architecture

the same keypoint embedding module. After passing through the spatial module, the output matrix's size needs to be flattened on the spatial dimension followed by feeding to the temporal attention module. And then, the temporal attention module connects two linear layers to project it into a vector of size 2. Finally, a standard Sigmoid Function is used to compute the probability for each class. Furthermore, this model allows input with varying numbers of keypoints by incorporating padding masks in two self-attention layers. This is achieved by adding paddings to sequences with fewer keypoints in each batch, extending their length to match the object with the highest number of keypoints in the batch. We use the same loss function, optimizer, and evaluation metrics as our model in this baseline model.

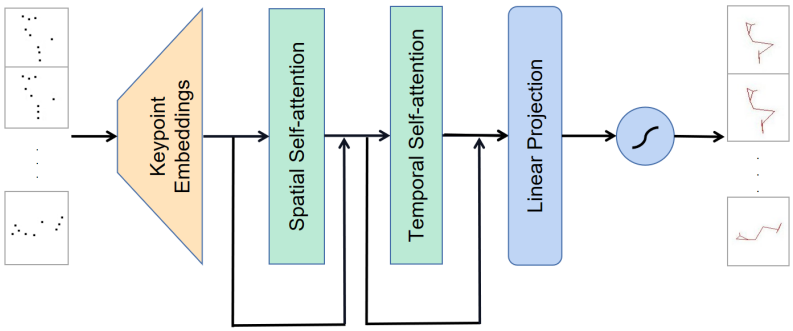


Figure 4.3: STA Architecture

4.3 Loss Function

In fact, the total number of all possible links between keypoints pairs greatly exceeds the actual number of existing connections in objects. For instance, in the dataset we mentioned, the hopper has 36 potential connections, of which 10 are genuine connections, labeled as 1, while the remaining 26 are labellled as 0. In more extreme cases, the Quadruped has 120 possible connections, with 16 being genuine connections labeled as 1, and the remaining 104 labeled as 0. As the number of keypoints increases, the situation of unequal distribution of labels may become more pronounced. The traditional binary cross-entropy loss function for binary classification tasks may perform

ineffectively on such imbalanced labeled datasets. It may lead to a situation that the predictions of the model are biased toward an all-zero outcome.

To avoid this problem, currently, there are several approaches for handling imbalanced labels. The first is resampling to adjust the proportion of different annotations in the dataset, including oversampling and undersampling. However, this method is not applicable to our task, because the number of keypoint connections for all objects is fixed, and we cannot adjust these fixed proportions through sampling and generation. The second approach is adding penalty terms into the loss function and re-weighting the loss, which is called Cost-Sensitive Learning[21].

In our work, we chose a Class-Balanced Sigmoid Cross-Entropy[8] as the model's loss function to address the imbalanced distribution of positive and negative samples. The class-balanced (CB) sigmoid cross-entropy loss is:

$$CB_{sigmoid}(\mathbf{z}, y) = \frac{(1-\beta)}{(1-\beta^{n_y})} \sum_{i=1}^{C} \log(\frac{1}{1+\exp(-z_i^t)}) \tag{4.3}$$

The underlying principle is to introduce a weighting factor α_i inversely proportional to the positive samples $E_{n_i} = (1-\beta_i^{n_i})/(1-\beta_i)$, where the hyperparameter $\beta_i = (N_i - 1)/N_i$, N_i is a set of hyperparameters, and the number of samples for class i is n_i so that $\alpha_i \propto \frac{1}{E_{n_i}}$, which penalizes results that do not conform to the positive-to-negative sample ratio. $\mathbf{z}$ is the predicted output, y is the actual output(0 or 1), and z_i^t is z_i if $y = i$, and $-z_i$ if not.

For the detection of a single object, the actual number of connections and the number of unconnected keypoints pairs corresponds to the sample size for each class. However, in the process of detecting multiple objects, this value is different. We sum the number of connections and unconnected keypoints pairs for all objects respectively in the dataset as the sample size for each class. In the case of using unseen objects as the test set, since we do not know the state of their keypoints' connections, we set reasonable values based on their number of keypoints. For instance, for an unknown object with nine keypoints, we empirically assume that the ratio of connected and unconnected keypoint pairs is approximately 9:27 (1:3). For the hyperparameter β in our model, we set it as 0.999 after tuning.

4.4 Cosine Annealing

Cosine Annealing[29] is a strategy for learning rate scheduling, which is used in the training process. The main idea of Cosine Annealing is adjusting the learning rate during the training process, starting from a fixed large high value and gradually decreasing to a lower but more suitable value in that state. The learning rate evolves in accordance with a cosine trajectory throughout the training process. This method enables the optimizer to learn broadly during the beginning phase of training, whereas reduced learning rates in subsequent stages could fine-tune the parameters in the vicinity of local optimal. The formula of the Cosine Annealing learning rate shows below:

$$\eta(t) = \eta_{\min} + 0.5 * (\eta_{\max} - \eta_{\min}) * (1 + \cos(\pi * \frac{t}{T})) \tag{4.4}$$

where $\eta(t)$ represents the current learning rate, $\eta_{\max}$ and $\eta_{\min}$ denotes the maximum and minimum learning rate in this algorithm, t is current epoch number, and T represents the number of iterations or epochs in a complete cosine annealing cycle.

In our model, the hyperparameter settings for the Cosine Annealing Learning Rate Scheduling include an initial learning rate of $5 * 10^{-4}$, a minimum learning rate of $1 * 10^{-9}$, and T_{max} is equal to the number of epochs.

4.5 Evaluation

In order to evaluate the performance of each model in the target task, a series of evaluation metrics will be employed in the experiments. Since all videos share a common keypoint connection vector, all evaluation metrics will be based on the total number of videos, and performance will be measured from the perspective of video dimensions. The primary evaluation metric needs to assess that every connection in the predicted results is correctly established, and connections that should not exist are not established. However, our datasets include some imbalanced datasets, whose negative classes may outnumber positive classes. Traditional accuracy cannot perfectly reflect the model's capabilities, and undetected connections may lead to hazardous consequences in practical applications. Taking these factors into account, we have adopted the following four evaluation metrics in this project, where TP denotes the number of True Positives, FN denotes the number of False Negatives, FP denotes the number of False Positives, and TN denotes the number of True Negatives:

- *Accuracy* is the most common metric for classification performance. It is the correct number of model identification divided by total number of samples, as follows:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.5}$$

- *Precision* indicates the proportion of true positive among the samples identified by the model as positive, as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

- *recall* indicates the ratio of the number of samples correctly identified by the model as positive samples to the total number of positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

- *F1 score* means a weighted average of the precision and recall. We regards the precision and recall are the same important in our project, so we choose F1 score, which is the harmonic average of them here, denoted by:

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{4.8}$$

Chapter 5

Experiment

This chapter primarily discusses several experiments that we designed for our model. These experiments include verifying the model's ability to learn the structure of a single object, the ability to learn the structure of multiple objects, the capability of handling single objects in extreme viewing angles, and the ability to learn unseen objects. In most of these experiments, we compare our model with the baseline model and use the evaluation metrics mentioned earlier. Furthermore, we will analyze the results regardless of whether they are positive or negative.

5.1 Experiment Environment

First of all, We need to introduce the experimental environment. Due to limited resources, we strive to complete the task on the local machine, which leads to some issues. These issues include frequent occurrences of insufficient GPU memory and the lengthy time required to generate datasets (over 12 hours for one dataset). Detailed operating environment configuration is listed in Table 5.1.

Operating System	WINDOWS10
CPU	Intel(R) Core(TM) i7-11370H 3.30GHz
GPU	NVIDIA GeForce GTX 3060 Laptop GPU
GPU Memory	6.0GB
Software Version	Python3.10.8, Pytorch1.13.0, CUDA11.0

Table 5.1: Software and hardware environment and operating system for training models

5.2 Hyper-parameter Tuning

To ensure that the model achieves optimal performance, hyper-parameter tuning is an indispensable task. We will aim to maximize the accuracy and F1 score mentioned in the evaluation section on the single Hopper dataset while ensuring that the tuning of these crucial hyperparameters stays within the maximum acceptable range for the

local GPU mentioned in the running environment. The final determined hyperparameter values are presented in Table 5.2.

Hyper-parameters	Optimal values
Batch size	64
Initial learning rate	0.0005
Embedding size	128
Adam weight decay coefficient	0.001
GRU Layer number	2
GRU hidden size	512
Attention head number	1
β in Class Balanced Loss	0.999
Projection Layers hidden unit size	1024,512,2

Table 5.2: Hyper-parameters Tuning Configuration

In these hyperparameters, the batch size is greatly affected by the memory capacity of our GRU. We set it to 64, which is the maximum acceptable batch size for our local GPU. From multiple experiments on the validation set, we have concluded that a smaller initial learning rate setting can effectively improve the model's performance. Therefore, we set the initial learning rate to a relatively small value of 0.0005. During the learning process, this value will gradually decrease due to the learning rate schedule. For some hidden layer sizes, our initial settings were relatively large. The original thought was to maximize information extraction from the data, but it posed a risk of overfitting. Fortunately, in our first experiment, overfitting was not a severe issue, so we removed some dropout layers from the original settings and increased the hidden layer parameters as much as possible. According to the original paper[8], the best value of β in CB Loss is obtained by trying values among 0.9, 0.99, 0.999, and 0.9999. In our tuning, we found the optimal value to be 0.999.

We were curious about why a single-attention head could achieve the best performance instead of multi-head attention. When setting breakpoints to explore the internal matrices, we discovered that we had multiplied the unused frame count with the batch size as the new batch size when calculating spatial attention, resulting in a very large value, which should have reached 6464 in our experiments. Therefore, when separating 2 or 4 attention heads (which we tried) from the 128 features, the impact was not very significant. The other explanation is that the underlying structure or patterns might be simple in our dataset enough that a single attention head can effectively capture the necessary information.

5.3 Single Object Structure Learning

To accomplish the primary goal outlined in Section 1.2, we will evaluate our model on six single datasets. These datasets utilize camera 0 in the DM Control Suite for capturing perspectives, which are fixed on the object. The six datasets encompass a range of complexities, including simple and intricate structures, varying numbers of keypoints,

and different action frequencies. All datasets are divided into training, validation, and test sets in a 6:2:2 ratio. The training runs for 100 epochs, and we will record all evaluation metrics values on the epoch with the best **F1 scores** on the testing set, since accuracy is not comprehensive enough. We also applied the Xavier initialization[12] to initialize the parameters before training. The result of the experiment is shown in Table 5.3.

	Hopper				Manipulator				Cheetah			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
GRU	0.738	0.617	0.806	0.699	0.645	0.387	0.695	0.497	0.834	0.607	0.844	0.706
ATT+ATT	0.925	0.855	0.85	0.853	0.788	0.672	0.750	0.709	**0.999**	0.960	**0.938**	0.949
GRU+ATT	**0.987**	**0.982**	**0.956**	**0.972**	**0.935**	**0.871**	**0.890**	**0.882**	**0.999**	**0.983**	0.930	**0.956**
	9-Swimmer				Walker				Quadruped			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
GRU	0.836	0.600	0.773	0.676	0.707	0.554	0.756	0.639	0.579	0.171	0.633	0.269
ATT+ATT	**0.981**	**0.852**	**0.945**	**0.896**	0.828	0.625	0.804	0.703	0.641	0.217	0.636	0.325
GRU+ATT	0.905	0.688	0.903	0.781	**0.972**	**0.928**	**0.949**	**0.938**	**0.717**	**0.266**	**0.656**	**0.378**

Table 5.3: Result of single object structure learning

From the overall result, our model demonstrates a significant improvement over the baseline models on five individual object datasets, and achieves high performance on the Hopper, Cheetah, and Walker datasets. However, compared with our models, the STA model gets excellent performance on the 9-Swimmer dataset. The structure of the 9-Swimmer is a single streamlined structure with simpler motion characteristics, and most of the motion in the dataset is forward creeping. We believe that STA can perform better when dealing with objects with simple motion characteristics and simple structures. Furthermore, the STA model outperforms the GRU-only model. The critical problem is the GRU-only model did not include a component to extract the spatial information.

However, all three models perform poorly when dealing with the complex Quadruped model with 16 keypoints, especially on the F1 score. We have made efforts to address and resolve the issues encountered in the Quadruped model, but unfortunately, the performance has not improved further. We propose several hypotheses regarding the problems that have arisen. First, during the process of checking the dataset for any bad data, we found that in a few cases, some legs of the Quadruped would ”walk out” of the camera’s view. However, this information was recorded as negative values in the dataset, and the impact should not be too significant. Additionally, there are instances where the leg joints of the Quadruped are significantly messy, making the assessment of these legs more complicated. Second, the issues may be caused by the highly imbalanced dataset. Although we have applied the Class Balanced Loss to correct the situation caused by the imbalanced dataset, a dataset with a positive-to-negative sample ratio of 104:16 could still potentially lead to some problems. Third, irregular movement and slower movement speeds may also make it difficult to capture relevant features when processing temporal data.

In those well-performing datasets, we can observe that the model tends to make incorrect connections instead of setting the true connections as negative when it faces an uncertain prediction, so the number of false positives is relatively high, resulting in the Precision

Figure 5.1: Our model's result examples during training on test set

values being noticeably lower than Recall in most datasets. We believe that this outcome is because of the Class Balanced Loss penalties those conservative predictions. Meanwhile, we checked the output during training in those well-performing datasets, shown in Figure 5.1.

Figures 5.2 and 5.3 show our model's training curves on Hopper and Walker. Unfortunately, due to limited GPU memory on our local machine, when we calculated the loss curves on the test sets, GPU always was out of memory, so we only recorded the training curves for accuracy and F1 score.

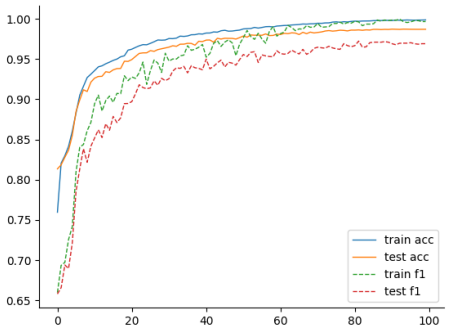


Figure 5.2: Our model's training curve for Hopper

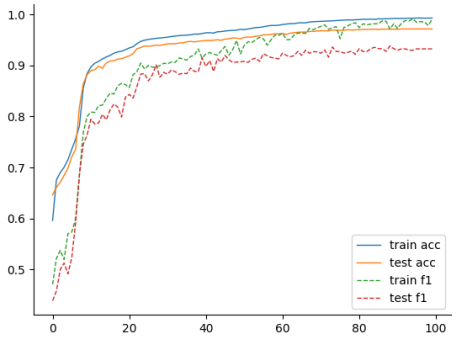


Figure 5.3: Our model's training curve for Walker

5.4 Multiple Object Structure Learning

In the fundamental requirements, we mentioned that our model should be able to learn from multiple objects and produce reliable results, so we conducted experiments on multi-object structure prediction based on our existing model. The experiment used four objects with the same number of keypoints to ensure that the output sizes were consistent. These objects included Hopper, Cheetah, Manipulator, and 9-Swimmer, all with 9 keypoints. We combined the four datasets and shuffled new datasets order,

	Multi-Object Dataset			
	ACC	PRE	REC	F1
GRU	0.831	0.630	0.888	0.737
ATT+ATT	0.866	0.752	0.728	0.736
ATT+GRU	**0.972**	**0.920**	**0.983**	**0.950**

Table 5.4: Result of multiple object structure learning

and then still divided them into training, validation, and test sets in a 6:2:2 ratio. The hyper-parameters on the test set used the optimal parameters obtained on the validation set as presented in Table 5.2. The results of the experiment are shown in Table 5.4.

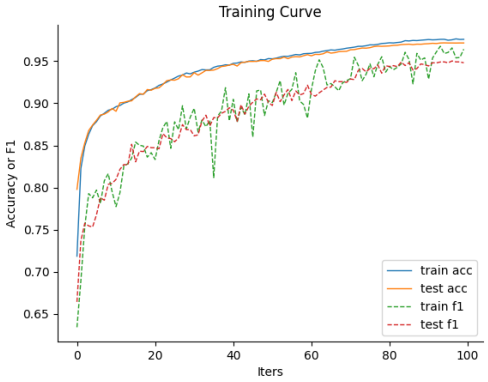


Figure 5.4: Our model's training curve for multiple-object dataset

We can see from the results of this experiment that our model significantly performs better than the two baseline models, demonstrating its ability to effectively learn multiple known object structures from the keypoint sequences. The training curve is depicted in Figure 5.4. As with the previous explanation, due to the lack of GPU memory, we were unable to record the magnitude of the testing loss. Instead, we present the performance curve as a substitute.

5.5 Unseen Object Structure Prediction

In this experiment, our objective is to examine whether our model processes the ability to construct the structures of objects that are not in the training set. It is quite a challenging thing to create a large number of object types, as this is constrained by our restricted spatial creativity and limited 3D modeling capabilities. Thus, we attempted to conduct experiments on the existing dataset. We employed the Leave-One-Out Cross-Validation(LOOCV) method, running experiments on four datasets. The experiment was carried out four times, with each time choosing three of the datasets as the training set and the remaining dataset as the validation set (test set). The final result is the mean of the four experiments, as shown in Table 5.5.

	Accuracy	Precision	Recall	F1 score
Hopper	0.750	0.565	0.594	0.579
Manipulator	0.732	0.325	0.430	0.370
Cheetah	0.756	0.444	0.773	0.564
Swimmer-9	0.808	0.522	0.634	0.573
Average	0.762	0.464	0.608	0.522

Table 5.5: The result of unseen object structure prediction

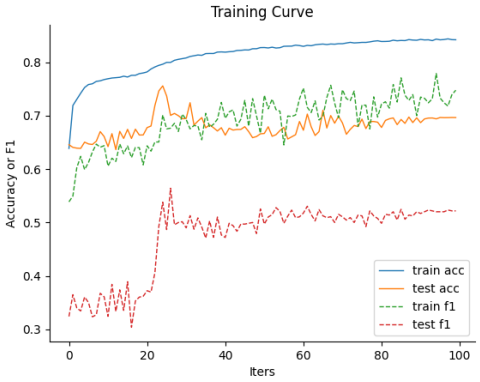


Figure 5.5: Our model's training curve for unseen Cheetah dataset

An unsatisfactory result can be seen from the table, with an average F1 score is 0.522, which means that our model demonstrates a weaker predictive ability for unseen objects. However, this result is not unexpected, considering that our model lacks a priori knowledge based on a large number of object models and only learns the structure of three novel objects to infer the structure of an unseen object. From the table, we observe that the model performs relatively better when we train on some more complex models to predict simpler objects, while predicting complex structures with a simpler training set is challenging. This suggests that our model is capable of learning some keypoint relationships to predict the structure of unknown objects. Figure 5.5 shows the training curve of this experiment on the Cheetah dataset, which has an increase on the test set in the beginning and becomes flatten quickly. Overfitting emerged prematurely. Although we attempted several approaches, such as incorporating dropout layers, but the effects were not significant. This indicates that the problem still originates from the lack of an adequate dataset. To validate the difficulty of performing this task under extremely limited prior acknowledgement, we conducted experiments with several human participants in the following section.

5.6 Human Experiment

Twenty-one participants with university education took part in a simple survey questionnaire, in which they were provided with a video of the Hopper keypoint sequence of 101 frames. This video included only the keypoints of the Hopper, and participants were

Figure 5.6: Hopper example captured by Camera 1 (extreme camera angle)

asked to infer the structural connections of the Hopper, as the machine did. The final results are displayed in Table 5.8. It is not difficult to see that even for humans, who possess enormous prior knowledge and innate awareness, it is challenging to perfectly predict the structure of these novel objects from a moving keypoint sequence only. Notably, participants with higher scores were predominantly university students or graduates majoring in art, design, mathematics, and physics. Humans got a similar f1 score with our model. This confirms that supervised models require an immense amount of data for machines to successfully predict the structure of unseen objects. Therefore, supervised learning may struggle to predict the structure of unseen objects from keypoints because of the absence of a large labelled dataset.

	Mean ACC	Max ACC	min ACC	Mean F1	Max F1	Min F1
Human	0.844	0.952	0.690	0.605	0.889	0.312
Our Model	0.750	/	/	0.579	/	/

Table 5.6: Results of 21 human participants predicting the unseen Hopper structure through keypoint sequences

In the interview with those participants, the majority of them indicated that their answers largely came from observing points with relatively stable positions. As a problem of projecting a 3D model onto a 2D plane, we also used cameras from different angles, such as the free-view camera 1(with extreme views) and fixed camera 2 in DM Control Suite, as previously mentioned. The 2D image of Hopper projected from Camera 1 is shown in Figure 5.6. We showed participants videos with extreme angles and cluttered keypoints as well. Their performance was very poor, even though they had prior knowledge from photos of the same object taken by Cameras 0 and 1. The experimental results are shown in Table 5.7. Since humans cannot effectively learn object structures from cluttered keypoints, we are curious about how our model would perform, which will be discussed in the following section.

	Mean ACC	Max ACC	min ACC	Mean F1	Max F1	Min F1
Human	0.720	0.929	0.690	0.509	0.738	0.364

Table 5.7: Results of 21 human participants predicting the Hopper structure captured by Camera 1 through keypoint sequences given the true structure of object by camera 0

5.7 Extreme Camera Angles

First, it is essential to reiterate the positions of the three cameras. Camera 0 is a tracking camera, which, for the Hopper, is situated at its side, providing an excellent viewpoint to observe the Hopper's structure. Camera 1, also a tracking camera, is positioned at an upward angle towards the Hopper, resulting in an extreme perspective where many points cluster in a small area. Camera 2 is a fixed-position camera, slightly off to the side, facing the Hopper.

As described in the previous section, we trained the datasets captured by various cameras at different angles, preventing the machine from learning the relative positions of keypoints based on their changes. In this experiment, we employed two representative datasets of Hopper, captured by Camera 1 and Camera 2. The splitting method and hyper-parameters remain the same as in the previous experiment. We conducted two experiments in this case. In the first experiment, the model was trained on the dataset captured by Camera 1 and Camera 2, and tested on the same dataset. In the second experiment, the model was trained on datasets captured by Camera 0, and tested on datasets captured by Camera 1 and Camera 2.

	Training Set	Testing Set	ACC	PRE	REC	F1
Our Model	Hopper_Camera_1	Hopper_Camera_1	0.982	0.980	0.941	0.960
Our Model	Hopper_Camera_2	Hopper_Camera_2	0.981	0.901	0.963	0.931
Our Model	Hopper_Camera_0	Hopper_Camera_1	0.802	0.626	0.669	0.647
Human	Two videos from Hopper_Camera_0 and Hopper_Camera_1	Hopper Camera_1	0.720	/	/	0.509

Table 5.8: Results of 20 human participants predicting the Hopper structure captured by Camera 1 through keypoint sequences given the true structure of object by camera 0

This experiment demonstrates that our model can learn object structures from concentrated and disorganized keypoint sequences, which is very hard for a human. Moreover, it is capable of learning the structure of an object from one perspective and making predictions from another extreme angle. Although the predicted F1 score is only 0.647, this value has already surpassed the predictive performance of human participants who previewed videos from both regular and extreme viewpoints.

Chapter 6

Conclusions

6.1 Summary

In this project, we proposed an object structure prediction network based on self-attention and GRU, which completes a downstream task of object pose estimation, specifically predicting the presence or absence of connections between keypoints in a given sequence of object keypoint transformations. Due to the lack of labeled datasets, most related research tends to use unsupervised learning for this task. In an attempt to use supervised learning for this task, we created our own dataset by DeepMind Control Suite, which provided motion videos and keypoint locations of various novel objects. Given the scarcity of related research in the field, we designed two simple baseline models. By using supervised learning, our network was able to learn the structure of the objects included in the training set more deeply compared with the two baseline models, regardless of whether the training set contained single or multiple types of objects. For structures of objects not seen in the training set, our network's predictions outperformed most human judgments, albeit imperfectly. We also analyzed the results of these experiments and provided some insights.

6.2 Result and Limitation

Reflecting on our initial fundamental goals, we successfully created a dataset of keypoint sequences for six novel objects and annotated their structures. Our proposed neural network effectively completes the task of learning object structures from keypoints. Compared to the other two baseline models, our model achieved significant improvements in accuracy and F1 score on four individual objects, while being comparable to the specifically designed Spatial and Temporal Attention (STA) baseline model. In experiments addressing multi-object tasks, our model achieved an increase of over 10.6% in accuracy and an F1 score improvement of nearly 22%.

Furthermore, we conducted research on the first additional objective. This study showed that predicting the structure of unseen objects through supervised learning is unreliable on our model in the absence of a large labeled dataset, with an average F1 score of only

0.522 across four test datasets. Although it can rival human judgment to some extent, its application in the industrial field is still inappropriate. Therefore, for a task that even humans with extensive prior knowledge find very challenging, a large-scale annotated dataset or unsupervised learning methods are necessary. Due to the constraints of research time and conditions, as well as the author's limited imagination and modeling capabilities, it is difficult to generate such a large dataset. However, we demonstrated that our model possesses a strong capability to learn keypoint sequences captured from angles that are difficult for humans to observe.

As for the research on the second additional objective, we successfully added this function on a baseline model, STA, to accept input with varying numbers of keypoints by employing masking operations to mask that useless attention in the matrices. However, incorporating this feature into the GRU-based model ended in failure, as we were unable to find a concise and efficient method that would not compromise the model structure while providing a masking-like functionality for those added paddings in the GRU.

6.3 Future Work

The future work of this project should primarily aim to establish a large and novel dataset of keypoints for some novel objects to enhance the generalization of the model. This is a complex task, as it requires substantial resources, including modeling, data collection, and annotation efforts. Considering the potentially low cost-effectiveness of this task in the industry field, we agree that unsupervised learning, which is widely used, would be a better direction.

In addition, the modeling of deep learning networks should be more diverse. During the network design process, we attempted to fuse kinetic components. However, due to limited devices and budget, which led to slow program execution and GPU memory overflow, the kinetic components had to be abandoned. Future work can integrate kinetic features with the deep learning network to improve network performance.

Bibliography

[1] Titas Anciukevičius, Paul Henderson, and Hakan Bilen. Learning to predict keypoints and structure of articulated objects without supervision. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3383–3390, 2022.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. Human pose estimation: New benchmark and state of the art analysis. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3686 – 3693. IEEE, June 2014.

[3] L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2016.

[5] Ingrid Carlbom and Joseph Paciorek. Planar geometric projections and viewing transformations. *ACM Computing Surveys (CSUR)*, 10(4):465–502, 1978.

[6] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris N. Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention, 2019.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[10] Aysegul Dundar, Kevin J. Shih, Animesh Garg, Robert Pottorf, Andrew Tao, and Bryan Catanzaro. Unsupervised disentanglement of pose, appearance and background from images and videos, 2020.

[11] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61:55–79, 2005.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[15] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[18] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[19] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. *Advances in neural information processing systems*, 31, 2018.

[20] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR 2011*, pages 1465–1472, 2011.

[21] Herman Kahn and Andy W Marshall. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control, 2019.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos, 2020.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[28] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Changhu Wang, and Jiashi Feng. Improving convolutional networks with self-calibrated convolutions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10096–10105, 2020.

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[30] Abu Saleh Musa Miah, Md. Al Mehedi Hasan, and Jungpil Shin. Dynamic hand gesture recognition using multi-branch attention based graph and general deep learning model. *IEEE Access*, 11:4703–4716, 2023.

[31] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model, 2018.

[32] Iago Richard Rodrigues, Marrone Dantas, Assis T de Oliveira Filho, Gibson Barbosa, Daniel Bezerra, Ricardo Souza, Maria Valéria Marquezini, Patricia Takako Endo, Judith Kelner, and Djamel Sadok. A framework for robotic arm pose estimation and movement prediction based on deep and extreme learning models. *The Journal of Supercomputing*, pages 1–30, 2022.

[33] Grégory Rogez, Jonathan Rihan, Srikumar Ramalingam, Carlos Orrite, and Philip Torr. Randomized trees for human pose detection. 06 2008.

[34] David E Rumelhart. Learning internal representations by error propagation, in parallel distributed processing. *Explorations in the Microstructure of Cognition*, pages 318–362, 1986.

[35] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Decoupled spatial-temporal attention network for skeleton-based action recognition, 2020.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[37] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[38] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[39] James Thewlis, Samuel Albanie, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of landmarks by descriptor vector exchange. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6361–6371, 2019.

[40] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. *Advances in neural information processing systems*, 30, 2017.

[41] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3229–3238, 2017.

[42] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[43] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Lumin Xu, Sheng Jin, Wang Zeng, Wentao Liu, Chen Qian, Wanli Ouyang, Ping Luo, and Xiaogang Wang. Pose for everything: Towards category-agnostic pose estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*, pages 398–416. Springer, 2022.

[46] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *arXiv preprint arXiv:2204.12484*, 2022.

[47] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2694–2703, 2018.

Appendix A

Dataset

A.1 Dataset Examples

This section primarily presents examples of all the datasets we have created.

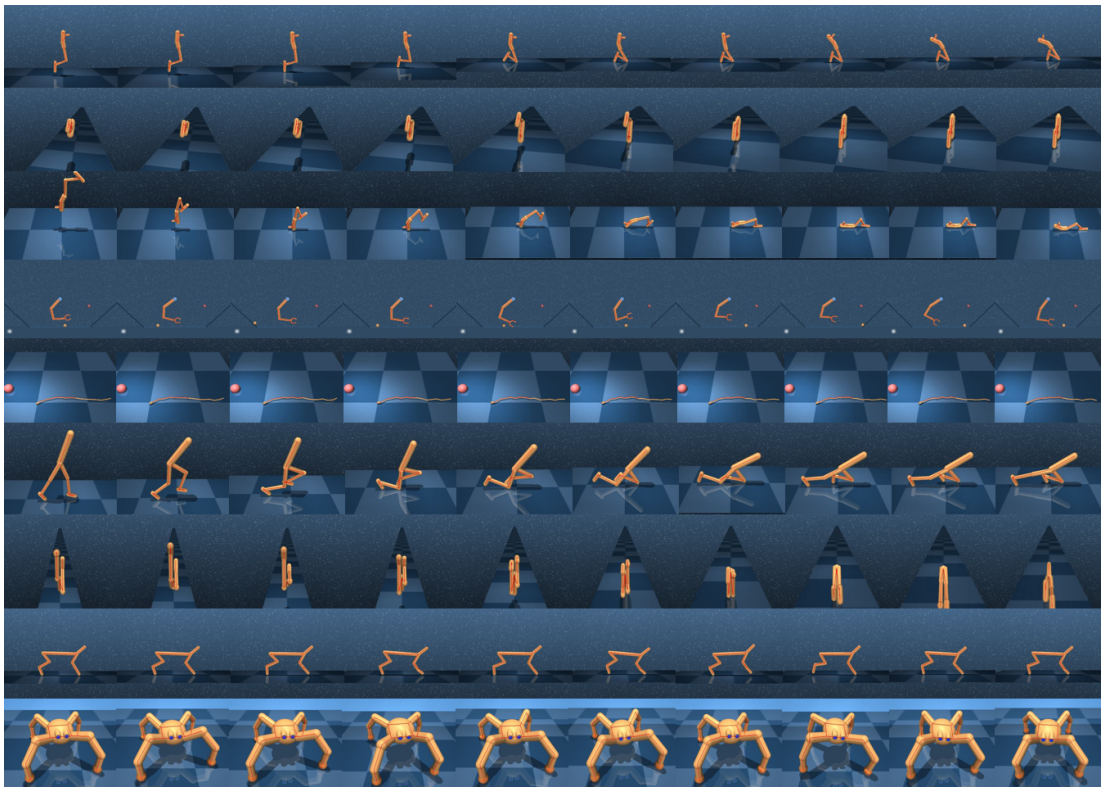


Figure A.1: Dataset

Appendix B

Participants' information sheet

Participant Information Sheet

Project title:	Learning Structure from Keypoints
Principal investigator:	Hakan Bilen
Researcher collecting data:	Hou Han
Funder (if applicable):	N/A

This study was certified according to the Informatics Research Ethics Process, reference number 259168. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

Hou Han, a UG4 student in the University of Ediburgh, will do the experiment for his undergraduate project. Hakan Bilen, a reader (associate professor) in the School of Informatics at the University of Edinburgh, is the supervisor of Hou.

What is the purpose of the study?

The project is to use supervised learning for learning keypoints sequence of a object. The model will be fed a keypoint sequence and output the structure of the object.

Why have I been asked to take part?

Because the model cannot learn well for unseen objects, we would like to show it is also hard for human, a entity with a lot of prior knowledge. We need to compare the result of my model and human result. Participants will be shown a video containing only a sequence of articulation points of an object, and then, through their observation and inference, as well as their own prior knowledge, they will deduce the connections between these key points, that is, the structure of the object.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time without giving a reason. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw,

contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

- Participant will be asked two questions about how to link the sturcture. You will be given two videos of the keypoints sequence. After watching the video, you will be given keypoints. You need to link them together.

- Means of collection is questionnaire

- Only 2 minutes required.

- No audio/video is being recorded

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

No

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 4 years. All potentially identifiable data will be deleted within this time frame if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team Hakan Bilen and Hou Han.

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted

cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Hou Han(s1919582@ed.ac.uk).

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information.

If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.

Alternative formats.

To request this document in an alternative format, such as large print or on coloured paper, please contact Hou Han(s1919582@ed.ac.uk).

General information.

For general information about how we use your data, go to: edin.ac/privacy-research

Appendix C

Participants' consent form

Participant number:_____

Participant Consent Form

Project title:	Learning Structure from Keypoints
Principal investigator (PI):	Hakan Bilen
Researcher:	Hou Han
PI contact details:	hbilen@ed.ac.uk

By participating in the study you agree that:

- I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1.	I agree to being audio recorded.	☐ **Yes**	☐ **No**
2.	I agree to being video recorded.	☐ **Yes**	☐ **No**
3.	I allow my data to be used in future ethically approved research.	☐ **Yes**	☐ **No**
4.	I agree to take part in this study.	☐ **Yes**	☐ **No**

Name of person giving consent	Date dd/mm/yy	Signature
Name of person taking consent	**Date dd/mm/yy**	**Signature**

THE UNIVERSITY of EDINBURGH informatics