Learning Object Structures From Keypoints

Saumya Shankar

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2022

Abstract

Artificial Intelligence (AI) algorithms are doing wonders in the standard computer vision tasks like image classification, video analysis, object detection or image segmentation by learning a task-specific representation. But such representations usually fail in the domain of Reinforcement Learning for motion planning and control tasks where AI algorithms are expected to reuse the learned representation to handle future novel scenarios. Such conditions demand learning a task agnostic, generic and interpretable representation for an object skeleton. One of the standard possible ways to learn such representation for better object understanding is to learn the reduced latent representation i.e. keypoints for an object. Recognising the connections between this set of keypoints of an object is an important task as it can help AI algorithms to have the ability to interact with or handle the object in any novel scenario. This project focuses on building an algorithm that can recognise the entire structure of an object given the keypoints x-y coordinates as a time series data while it performs a random action in a video. Three different methods are proposed in this project in addition to the LSTM baseline model which can be grouped into two categories of Link Level or Object Level models. The first method which comes under Link Level category calculates and uses the frequency and jerk signals for a better understanding of the causal relationship between a pair of keypoints along with given time series data as an input to the LSTM model. In the second method, information from neighbouring keypoints as contextual information is also included in addition to the frequency, jerk and time domain signals of the target pair of keypoints as an input to the LSTM model. This model architecture turns out to be the best performing model in all of the different experiments conducted in this project. Lastly, a Transformer based architecture with a unique masking technique and an inner product layer has also been proposed. This architecture has an added advantage of predicting the entire structure at once, instead of pairwise predictions, which makes it more suitable for real time object structure prediction in comparison to other proposed methods. The effectiveness of proposed architectures has also been explored in different simulated environments like seen or unseen topologies at the test time or with different kinematics structures such as fixed length or dynamic length robotic arms or multiple objects in a video.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

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.

(*Saumya Shankar*)

Acknowledgements

This project was a highly enriching experience and an opportunity to strengthen my knowledge in various technical aspects of Artificial Intelligence and other associated fields. I would like to express my sincere gratitude to my supervisor, Dr. Hakan Bilen, for providing his valuable support and guidance throughout this project. Lastly, I would like to put forth my gratitude to my family for their constant support, encouragement, and love.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

What actually an object is? Cognitive science experts have performed several research studies to understand this problem and it suggests that humans, even newborn infants, perceive an object as the collection of several different small parts arranged in a specific structure moving together [3, 25]. Humans have an amazing ability to recognise the structure of an even unseen object based on its movement; they can even predict the future motion of the object in an unseen environment. [7] suggests that this unique understanding of the object is performed in three different steps; first, humans build a reduced latent representation of an object in terms of keypoints then understand the structural relationship between the keypoints and finally deduce the dynamics controlling the movements of these keypoints based on the detected structure. This ability is also responsible for predicting the counterfactuals like ”what happens if?” [25]. This general understanding of an object skeleton without being dependent on any specific task is what gives humans the amazing generalisation ability for manipulating objects in a novel environment and is the core of human intelligence.

Unlike humans, Artificial Intelligence (AI) machines create task-specific representations of any object. Such representations have helped AI to achieve great results in the field of image classification [11], deep generative models [5], object detection [22] and similar domain-specific tasks. However, these representations have made it difficult for AI machines to perform in novel environments, specifically in the domain of Reinforcement Learning, where there is a need to reconfigure the learned representations for future tasks to precisely handle the objects and tools in new environments. AI machines have been in pursuit of such human intelligence for a very long time [33].

A task-agnostic and generic understanding of an object skeleton in terms of its keypoints and structural relations between them can help AI to excel in the field of object tracking[22], action recognition, trajectory prediction, video synthesis [28], few-shot Learning [30] and robots motion planning and object grasping. The most important and comparatively unexplored part of this three-step process in object understanding is structure recognition given the reduced latent representation i.e, keypoints of an object based on its movement. Keypoints of an object combined with its complete hierarchical structure i.e, the way in which keypoints are related to others can bridge the gap between human and AI intelligence in the domain of object understanding up to great extent.

Extracting the entire interactions set between each keypoints based on their trajectories data while performing a specific motion is not an easy task because of the complex hidden dynamics controlling the motion of each keypoints and the entire object. Another major challenge for any learning algorithm would be to distinguish the interaction between a pair of keypoints with simply a correlation between their trajectories. Additionally, it could also happen that a keypoint trajectories is controlled by several other keypoints but only one of them has a direct link with it. Thus, the learning algorithm is supposed to be able to distinguish between actual interaction and correlation and then also between direct and indirect interactions.

1.2 Problem Statement

Conventional Machine Learning or AI algorithms have proved their mettle in the field of image classification, object detection, image segmentation or deep generative models with the help of a learning algorithm that learns the representation specifically for the task chosen. But due to this task-specific representation learned by AI machines, they find it difficult to perform several complex sensory-motor behaviours [12] like object grasping - where AI algorithms are expected to re-use the learned knowledge in unseen future environments. On the other hand, unlike AI machines, humans build a task agnostic representation of an unknown object skeleton based on its movements using which they can understand - how to interact or handle it.

There are several recent developments where AI machines try to imitate such human-like intelligence by predicting keypoints or by modelling its dynamics but predicting the hierarchical structures of keypoints coordinates are still not well explored. The main problem this project focuses on is predicting the object structure or links between keypoints given sequential x-y coordinates (time domain signals) of keypoints positions

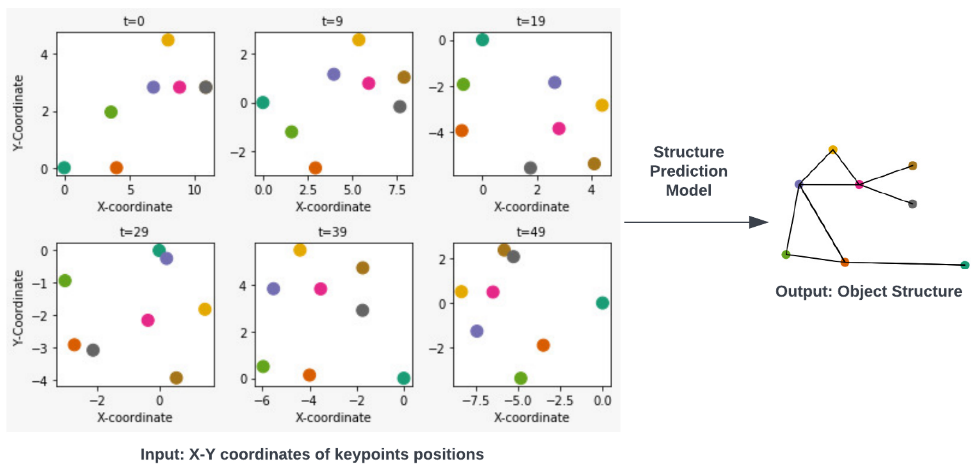

t=0
t=9
t=19
t=29
t=39
t=49
Y-Coordinate
X-coordinate
Structure Prediction Model
Output: Object Structure
Input: X-Y coordinates of keypoints positions

Figure 1.1: Structural Inference Model workflow. *left:* time-series data of keypoints positions of an object as input $O_n^{1:T}$. In the figure few samples are mentioned from t=0 to t=T *right:* inferred edges between each keypoints $\tilde{\varepsilon}_n$

while the object performs specific motion for a given time period in a video (Fig. 1.1). Mathematically formulating, this project attempts to build a structure prediction model, f_θ^ε, parameterised by θ that takes a sequence of x-y coordinates of each keypoints positions $O_n^{1:T}$ as input and predicts all possible links $\tilde{\varepsilon}_n$ (ε_n=ground truth link sets) between keypoints where n refers to any n_{th} data point or simulation and T refers to T frames in the simulation video or T time-steps in input sequential data.

$$\tilde{\varepsilon}_n = f_\theta^\varepsilon(O_n^{1:T})$$

1.3 Research Hypothesis and Objectives

The aim of this project is to build a Deep Learning architecture that can take x-y coordinates of keypoints of an object as time series data input and predict the structure of that object or in other words predict each possible link between these keypoints as an attempt to:

1. Bridge the gap between AI and human intelligence in terms of object understanding.
2. Make AI capable of learning a task agnostic and generic representation of an object skeleton in terms of keypoints (given in the current project scenario) and

connections between the links (end solution of the project using keypoints as input). This might also eventually help AI in other downstream tasks such as activity recognition, Few-Shot learning or trajectory prediction.

3. Make AI capable of performing well in the Reinforcement Learning domain where it is necessary to understand the complete structure of an object and its dynamics so that it can precisely manipulate the objects and tools in an unseen environment as well.

The structure prediction problem that is dealt with in this project can be strongly connected with the concept of causal discovery as well where the algorithm tries to learn the cause and effect relationship between different keypoints of an object [10, 16]. Given the huge success of Deep Learning in sequential data and its recent development in Causal Discovery, we hypothesise that a Deep Learning structure prediction tool using the keypoints representation of the object as an input can make the AI machines capable of recognising the structure based on observing the motion of any object (even unseen object). This detected structural relationship between the keypoints of an object might also help in improving the performance of other downstream applications in the field of Computer vision and Reinforcement learning.

The objectives of this project are as follows:-

- Create simulated data in three different settings consisting of 3,4 and 5-link robotic arms with different structures and end effector shapes performing different motions in each episode simulation (video). Three different settings are:-
    1. A robotic arm with fixed arm length
    2. A robotic arm with changing arm length for a single simulation
    3. Predict the object structure if multiple objects are performing different motions in the same episode.
- Create Deep Learning architecture at two level
    1. Link Level:- link prediction between each pair of keypoints - a binary classification problem
    2. Object Level:- predict the entire structure at once considering all keypoints together.
- Analyse the model performance in two different settings

1. Predict the object structure at test time which has already been seen at training time
2. Predict the object structure at test time which is not seen at the training time.

- Analyse the proposed models performances for different simulation settings.
- Analyse the effect of the length of links on the model performances.

1.4 Results Achieved

In this project, three different methods have been proposed in addition to an LSTM baseline to solve the object structure recognition problem based on time series data of x-y coordinates of keypoints. It can be broadly classified into two categories:- Link-level methods to predict the links for each pair of keypoints and Object-level methods to predict the entire object structures at once. The first method comes under the link-level category where the usage of frequency and jerk signals has been proposed along with the given x-y coordinates as an input to the LSTM. This method has led to a significant increase in the performance when compared to the LSTM baseline using only x-y coordinates of keypoints. It particularly helped in the prediction of several links which were missed earlier by the baseline method but also led to several false predictions as well. As an improvement over this with the same LSTM backbone, another link-level method has been proposed for using frequency and time-domain signals of all of the neighbouring keypoints along with the target input pair for which link prediction has to be done. This specific architecture is the best-performing method across all of the experiments and it particularly helped in removing the false predictions which were made by the previous methods using contextual information from keypoints other than the target input pair. But being a link-level model, it's still not an apt method for a larger number of keypoints and thus an object-level method has also been proposed using a multi-headed self-attention transformer with a specific masking technique and an inner product layer at the end to predict the entire object structure. It's the second best performing model, but still more preferred because of its real-time advantage of making a single-stage prediction of the entire object structure irrespective of the number of keypoints in an object or multiple objects. To demonstrate the effectiveness of the proposed methods, they have been tried in different environments:- seen or unseen topologies in the test set and with different kinematics structures:- fixed or dynamic length robotic arms or multiple robotic arms in a scene. Different possible structures of robotic arms have also been tried which include 3,4, or 5-link robotic arms with different link arrangements or end effector shapes. Additionally, it has been observed that larger links have higher chances of not getting predicted but this effect was more prominent for the baseline method and it has been solved up to the great extent by three other proposed methods.

1.5 Thesis Structure

The thesis contents are organised as follows: Chapter 2 will introduce the LSTM and Transformers in Deep Learning, followed by an overview of the past works in object structure recognition model. Chapter 3 will mention how the artificial datasets used in this project are created, followed by a discussion over data distribution, label descriptions and Data preprocessing steps. Chapter 4 will mention the details of the methods and the deep learning architectures proposed in this project. It will also introduce the evaluation metrics used for all of the experiments. Chapter 5 will discuss different experiments conducted in this project, comparison of model performances and analysis of their results. It will also discuss the effect of the length of rigid connections on the model performances. In Chapter 6, proposed architectures and results obtained in this project will be analysed and compared with the previous works. Finally, Chapter 7 will give conclusive remarks and suggests future works for this project.

Chapter 2

Background and Related Work

2.1 LSTM

LSTM stands for Long Short-Term Memory Networks and is a deep neural network derived from RNN (Recurrent Neural Networks) family. RNNs and LSTMs both are specifically designed for sequential data like text, monthly sales or time-series sensor data. Conventional feed-forward neural networks work well for tabular data but don't work for sequential data because of their inability to learn from past data and current input data collectively. Convolutional Neural Networks (CNNs) can capture temporal information up to some extent but are limited to learning only local patterns and usually fail to learn long-range dependencies. RNNs are able to learn from small sequential data very well but are unable to learn long-range dependencies because of vanishing gradient descent problem [26]. LSTM was specifically designed to learn long-range dependencies by solving the problem of vanishing gradient descent with the help of a series of gates; forget gate, input gate and output gate Fig. 2.1.

LSTM output at any timestep t depends on c_t =cell state i.e. present long-term memory of the network, h_{t-1} = previous hidden state i.e. output at previous timestep and x_t =current input data. The core working of LSTM can be mentioned as a 3-step process. At first step, forget gate decides which part of the c_{t-1} should be given less importance based on h_{t-1} and x_t. The second step consists of two parts; a tanh-activated neural network and an input gate. The first part decides how much to update c_{t-1} based on x_t and h_{t-1} and then the input gate decides if the current input data is worth remembering or not. The outputs of these two parts are then multiplied with each other and then finally added to c_{t-1} to get a new cell state c_t. Lastly, the output gate produces a new vector based on h_{t-1} and x_t which is then further multiplied with c_t elementwise

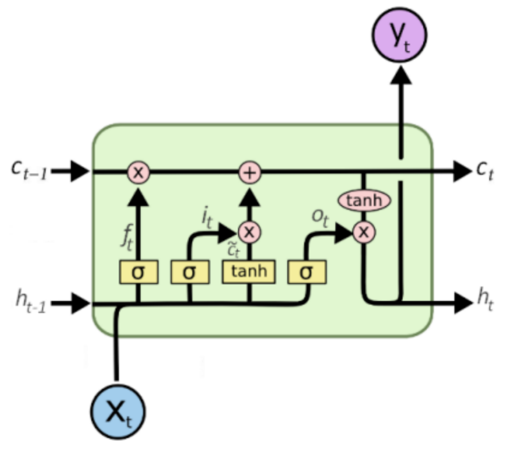


Figure 2.1: The LSTM unit [26]

to make sure that only necessary information from c_t is produced as the output of the current LSTM cell as h_t. Finally these steps will be repeated several times depending on the number of timesteps being used by LSTM architecture.

2.2 Transformers

Transformers have been producing state of art results in variety of tasks in NLP and even in computer vision domain [18]. The main principle behind the working of transformers, specifically the transformer encoder layer, is that it takes in a set of input vectors apply a mechanism called self-attention which helps the transformer to learn the relationship between the input vectors and then the output is passed through a series of residual layers, Layer normalization and blocks of Multi-Layer Perceptrons to generate a final set of contextualised embedding vectors as shown in Fig. 2.2.

The self-attention is the core principle behind the working of transformers. It basically takes in a sequence of vectors and generates another sequence of vectors of the same length but now each vector will also have information on how it's related to other input vectors. Let's say the input sequence is denoted by $\mathbf{x}_1,..,\mathbf{x}_i,...\mathbf{x}_t$ and the output sequence is denoted by $\mathbf{y}_1,..,\mathbf{y}_i,...\mathbf{y}_t$. It, first, transforms each element of the input sequence of vectors $\mathbf{x}_i$ into 3 different vectors called query $\mathbf{q}_i$, key $\mathbf{k}_i$ and values $\mathbf{v}_i$ to serve 3 different purposes by multiplying the input vectors sequence with a set of 3 trainable kXk weights matrices W_q, W_k, W_v. Then each element of the output sequence

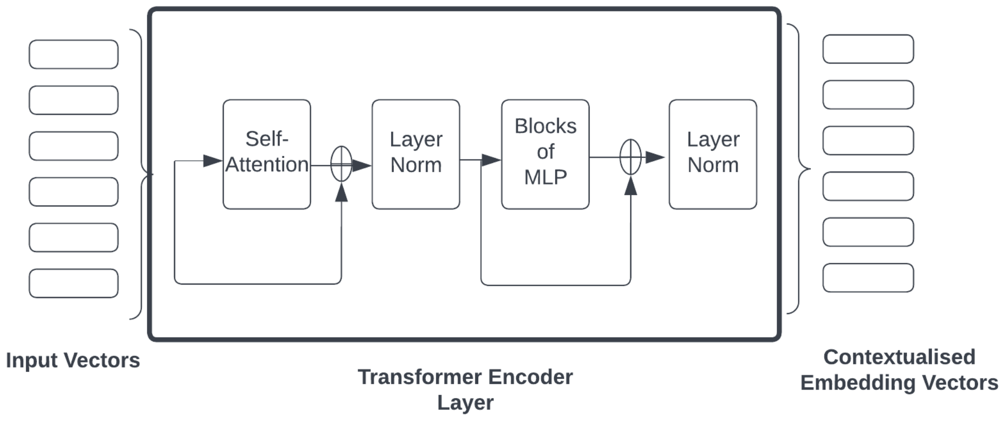


Figure 2.2: Transformer Encoder Layer

of vectors $\mathbf{y}_i$ is calculated by a weighted sum of values vector $\mathbf{v}_i$. These weights are called attention weights $\mathbf{w}_i$. Mathematically, the entire operation can be formulated as follows:-

$$\mathbf{q}_i = W_q\mathbf{x}_i; \mathbf{k}_i = W_k\mathbf{x}_i; \mathbf{v}_i = W_v\mathbf{x}_i \tag{2.1}$$

$$w'_{ij} = \frac{\mathbf{q}_i.\mathbf{k}_j}{\sqrt{k}} \tag{2.2}$$

$$\mathbf{w}_i = softmax(\mathbf{w'}_i) \tag{2.3}$$

$$y_i = \sum_j w_{ij}\mathbf{v}_j \tag{2.4}$$

A single self-attention layer captures a specific semantic meaning only, thus it's usually a better practice to use another variant of self-attention called multi-headed self-attention where several attention heads are calculated with a different key, query and values matrices in an attempt to learn more than one semantic meaning from the given input sequence of vectors and then each of these heads outputs is concatenated and multiplied by a weight matrix to get the final contextualised embedding for vector in the input sequence.

2.3 Previous Works

2.3.1 Implicit object structure recognition model

As mentioned in section 1.1, object understanding can be considered as a 3 step process of predicting keypoints, identifying interaction between them and finally modelling the dynamics controlling the keypoints positions. There exist several previous works which have tried solving the problem of detecting the keypoints in both supervised and unsupervised settings. In supervised settings, it's just a simple regression problem with a CNN-based architecture used as a feature extractor. Detecting keypoints in an unsupervised way is a bit tricky where most of the previous works revolve around the concept of having an autoencoder architecture with keypoints-based bottleneck to reconstruct the images sampled either from the video or some random perturbation of the source image [13, 6]. Several attempts have also been made to learn the dynamics module to predict the future keypoints positions [19, 29, 31]. Even though both of these categories of models are not explicitly modelling the structure but they are implicitly using some complex representations of the structure which are hard to decode and are not interpretable in terms of the keypoints of the object.

2.3.2 Explicit object structure recognition model

In [16], author mentioned the importance of learning the structure explicitly by demonstrating an improvement in the performance of a dynamics model to predict the future keypoints. Learning the structure prediction model explicitly will help us in building a more interpretable task agnostic representation and thus a better and more generalised understanding of an object. Recently few attempts have been made in order to explicitly learns the structural model as well to infer the structure of the object. [39] proposed a multi-modal architecture for simultaneously learning the kinematics and structure of an articulated robotic structure. It expects both point cloud data and joint angle sensor data as input to the model. But in real life scenario, an architecture which can work directly on RGB images captured from a camera or keypoints extracted from such images would be more ideal. Also, the author proposed this architecture only for chain-like straight simple robotic structures. The author proposed a 3 stage process to finally predict the kinematic structure. It first segments the point cloud data corresponding to different links based on a minimum matching error criterion followed by learning the rigid transformations associated with each of the predicted parts and then finally using these two results to infer the kinematic structure of the robot. [17] also attempted to solve the problem of identifying robot kinematic structure but they actually used the fiducial markers on each of the links, so essentially they are just predicting the correct sequence of links from already given prior links information between the keypoints. Additionally, the proposed method not only depends on the time-series data of the position but also expects the orientation details as well and it also works only for serial kinematics [27]. The proposed architecture checks the feasibility of a system of kinematic equations which tell us whether two links are consecutive or not i.e. identifying the correct order of links and also the type of joints that exists between the two links.

There have been few recent works which attempted to build a complete system for object understanding and in an unsupervised way as well to determine the keypoints, and interactions between them and finally to predict the future motion of the objects in terms of either predicting the keypoints positions or producing a series of future frames as a video. [36] proposed a novel architecture that simultaneously learns the hierarchical object representation and dynamics model as well for future motion prediction. But this architecture doesn't produce a reduced set of latent representations in terms of key points, instead, it's much similar to the task of object segmentation and then outputs a hierarchical relation between the segmented object parts. [10, 16] are the most similar works when compared with the objectives of this project where the author proposed to build one complete system of object understanding in terms of determining keypoints, interactions between them and also learning the dynamics model for future motion prediction.

The structure prediction model is the most important part of this 3 stage process of object understanding as shown in one of the results in [16] where the author has shown the importance of causal discovery (structure prediction model) in the overall model performance for object understanding. Thus, unlike most of the previous work, this project focuses on building an explicit structure prediction model and validating it in different test set conditions and with different kinematics structures.

2.3.3 Similar structure recognition problem in other domains

The object structure recognition problem can also be closely related to standard graph or structure prediction problems of other domains. It can most closely be related to the static link prediction problem [14] used widely in the domain of Recommender systems or Social network analysis or Question-Answering systems in Natural Language

Processing. Most of the recent works have solve these problems with the state of art performances using Graph Neural Networks (GNNs) [37] to learn from the neighbouring node or keypoints information [1, 38, 35]. One of the major differences between these methods and our current problem is that GNNs expect some prior information about the whole structure to be already given in terms of partial sets of links and then they can predict the other links. But in the case of this project, only keypoints positions sequential data are given and the entire link sets have to be predicted from it. As mentioned in [10, 16], the current problem can also be closely related to the works done in the field of causal discovery. The major dissimilarity though exists in the fact that such causal inference-based solutions either assume complete knowledge of the intervention or have scalability issues with the large number of keypoints.

Chapter 3

Datasets

To build an explicit structure recognition model in a supervised setting, an annotated dataset of a variety of structures as ground truths are expected with the keypoints time series data of each object structure performing some random motion. But most of the past works are building structure prediction models implicitly only, a few previous works that are explicitly building a structure recognition model are doing it in an unsupervised way [10, 16]. Instead of the actual structure, they are using keypoints as the ground truth based on a specific workflow. In this workflow, Keypoints time series data for the first half of the episode are used to predict the structure. Then a dynamics model is also learned to use both keypoints and predicted structures to predict the keypoints for the second half of the episode. Predicted keypoints positions are then used for calculating mean squared error as the loss function. Thus, as far as the current project's objectives are concerned and in the lack of required annotated datasets, synthetic datasets with several structures as ground truth are created to focus explicitly on the structure recognition model only and test it in different simulated environments.

3.1 How the datasets are created

To solve this problem of object structure recognition, three artificial datasets have been created with Fixed length robotic arms, Dynamic length robotic arms and Multiple objects in a video. These artificial datasets contain episodes generated from random different object structures focused around 3, 4 and 5 link robotic arms. It has been made sure for these datasets to have closed loops, changing angles, multiple connections to a keypoint, different edge lengths, different random trajectories for end effectors to be followed in each episode and different end effectors shapes. To have smooth transitions (more real-time simulation) between each time-steps, episodes are created with a combination of forward and inverse kinematics.

3.1.1 Fixed and Dynamic length robotic arm

Both Fixed and Dynamic length robotic arms datasets use the same backend process to create the artificial episodes of robotic arms following a random trajectory for each episode with the only difference that for dynamic length robotic arms, each arm length changes in a periodic manner with a condition that overall length of all robotic arms should remain same to ensure the working of kinematics equations and thus the smooth transitions between each frame of the video in an episode. The main backend process behind the generation of each episode can be described as follows:-

1. For each episode, create a random trajectory and a random robotic arm structure.
2. Now based on the number of frames to be generated in each episode, find a change in the location of the end effectors in coordinate space for each timestep or frame of that episode.
3. Based on this change in location of the end effector in coordinate space, all the other joint angles are determined using inverse kinematics.
4. Once all the joint angles are determined, use them along with the given arm length to implement the forward kinematics and find all of the joint new positions.
5. Repeat step 2-4, until the required number of frames are generated for that episode.

Sample frames of two episodes of fixed length and dynamic length robotic arms are as shown in Fig 3.1 and Fig 3.2 respectively

3.1.2 Multiple Objects in a video

While building an episode with multiple objects in it and following some random trajectories, there are two assumptions in which this dataset can be created. First, multiple objects can interact with each other as well and second, each of the multiple objects are following their own random trajectories completely independent of each other. For the scope of this project, these episodes are created with the second assumption that in each episode or video a pair of random robotic arms are following their own random

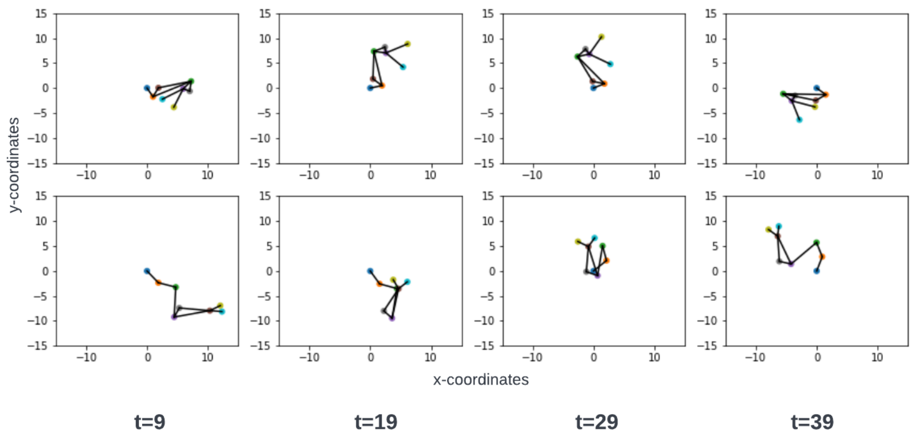


Figure 3.1: 4 samples from two episodes of a fixed length robotic arms.

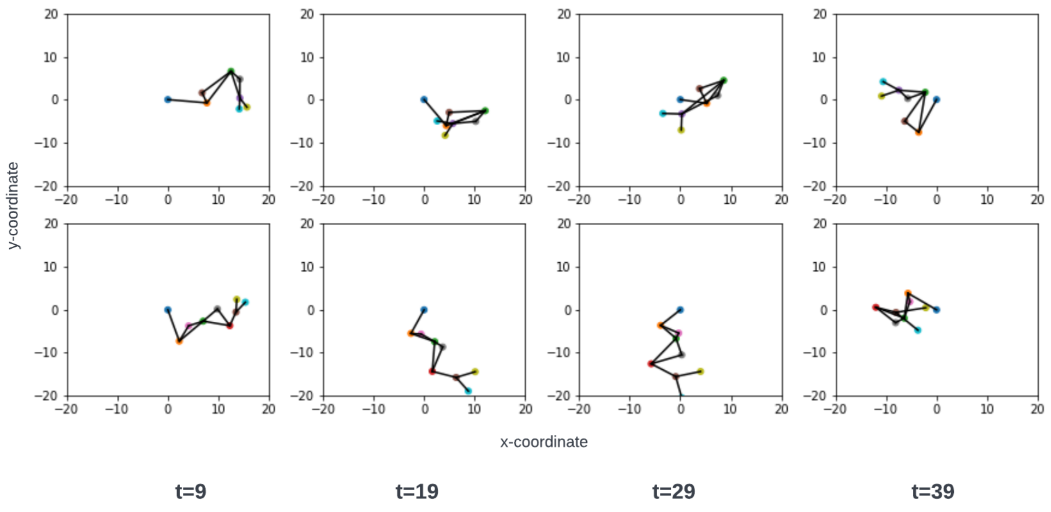


Figure 3.2: 4 samples from two episodes of a dynamic length robotic arms.

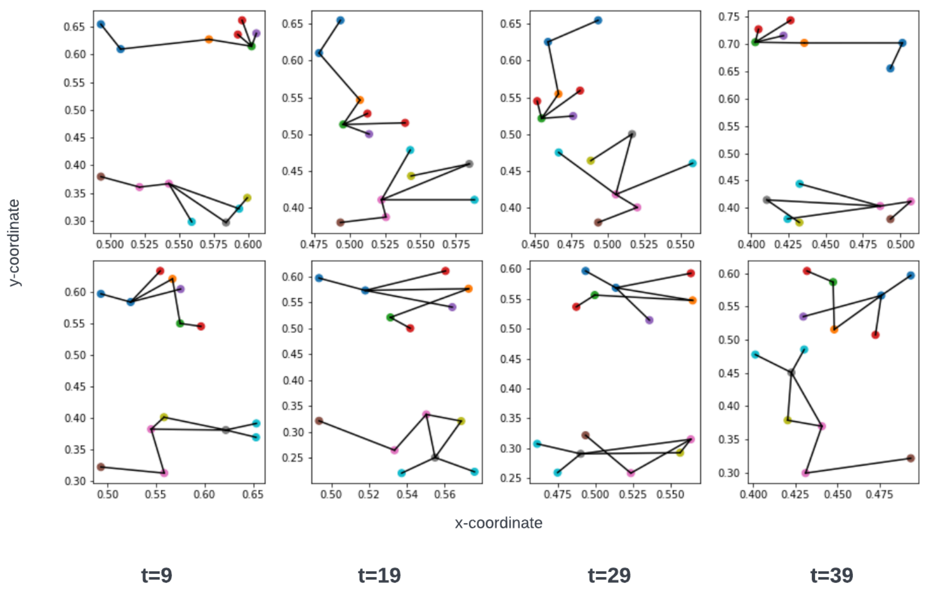


Figure 3.3: 4 samples from two episodes with each having pair of fixed-length robotic arms.

trajectories with no interactions between them. Two versions of this dataset have been created; one where pair of objects are fixed-length robotic arms and another where pair of objects are dynamic-length robotic arms. These episodes are basically created by a random combination of any two episodes of fixed length or dynamic length robotic arms. For example, two episodes of fixed-length robotic arms are chosen at random and then they are arranged frame by frame horizontally or vertically or diagonally to create a new episode where in each frame there will be two random fixed-length robotic arms following their own independent randomly sampled trajectories. Sample frames of two of the episodes are shown in in Fig. 3.3

3.2 Data distribution and Label Descriptions

Two categories of methods are proposed in this project i.e, Link Level and Object Level. The Link Level method formulates this object structure recognition model as a binary classification problem and makes a link prediction for each possible pair of keypoints but the Object Level method formulates this problem as a varying size adjacency matrix prediction problem. In total 1000 videos or episodes of artificial datasets are created in

Kinematic Structure	**Testing condition**	**Training**	**Testing**
Fixed-length robotic arm	Seen topologies	880	120
Fixed-length robotic arm	Unseen topologies	875	125
Dynamic-length robotic arm	Seen topologies	880	120
Dynamic-length robotic arm	Unseen topologies	875	125

Table 3.1: Train test distribution (number of objects) for Object Level methods

each of the settings, a total of 8 different structures (these 8 also have several different permutations of links which amount to a total of 32 different sub-structures) each having 125 unique episodes. Both fixed and dynamic length robotic arms have similar train-test split strategies as discussed below.

For Object Level methods and unseen topologies at the test time, 7 different structures of all 3 and 4-link robotic arms are used as training dataset i.e. $7 * 125 = 875$ training instances and 5 link robotic arms are used as test dataset i.e. $1 * 125 = 125$ testing instances and thus, train test split of 875:125 or 7:1. Object Level methods with seen topologies at the test time also have a similar train test split ratio but now these 125 test instances have equal proportions of all different 8 structures. In fact, 15 instances of all 8 different structures are considered in the test set and thus a test split of 880:120, approximately 7:1.

For the link-level methods, a train-test split is not performed again. Instead, for each of the training instances in the object-level methods training dataset, all possible pairs of key points are considered as training instances for the link-level methods and similarly, the test dataset is also created for link-level methods in both seen and unseen topologies experiment settings. The final number of training and test instances are shown in Table 3.1 and 3.2. The difference in train and test distribution in seen and unseen topologies in link-based methods is because of the fact that in unseen topologies all test instances are 5 link robotic arms and thus the number of possible pairs of links is larger than in the case of seen topologies which has mix of 3,4 and 5 link robotic arms in the test set.

Both of these categories of methods link-level and Object Level have different types of labels. For the link-level methods, labels are just binary numbers 1 or 0 indicating whether there exists a link or not. But for object-level methods, ground truth and model output is a symmetric square matrix called adjacency matrix. In graph theory, a symmetric adjacency matrix is used to represent an undirected finite graph structure. Each cell (row, column) defines whether a connection exists between row and column

Kinematic Structure	**Testing condition**	**Training**	**Testing**
Fixed-length robotic arm	Seen topologies	21256	2798
Fixed-length robotic arm	Unseen topologies	18426	5628
Dynamic-length robotic arm	Seen topologies	21346	2851
Dynamic-length robotic arm	Unseen topologies	18424	5773

Table 3.2: Train test distribution (number of links) for Link Level methods

nodes or keypoints.

3.3 Data Preprocessing

Each of the time domain signals, jerk signals and frequency domain signals are normalized using the Min-Max scaler operation in the range of 0 and 1. The following operation is performed on each of the input signals

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{3.2}$$

Here, X_{scaled} represents the normalized output of the Min-Max scaler operation. X_{min} and X_{max} refer to the minimum and maximum of the input signal that have to be normalized. max and min define the range (min, max) within which the output signal X_{scaled} have to be normalized.

Chapter 4

Methodology

4.1 Different types of signals used

Determining the link which is similar to the problem of identifying the causal relationship between any two keypoints [10, 16] or determining how the change in one keypoint position is affecting the others just based on the time domain signals is a complex task as discussed in section 1.1. Thus, as a first attempt to solve this problem of object structure recognition, different types of input signals are explored that can be utilised to effectively identify the relationships between two keypoints.

4.1.1 Jerk Signals

Jerk signal refers to the gradient signal which captures how the signal values are changing with each timestep. And determining how the change in x-y coordinates of one keypoint position is affected by the change in other keypoints positions can help in identifying whether the link exists between any pair of keypoints or not. Jerk signals are calculated as second-order accurate central differences at points other than edges eq.4.1 and at the edges, it's calculated as first-order accurate one-side differences eq.4.2 and eq.4.3.

$$j[h] = \frac{j[h+1] - j[h-1]}{2} \tag{4.1}$$

$$j[h] = \frac{j[startpointofthesignal+1] - j[startpointofthesignal]}{1} \tag{4.2}$$

$$j[h] = \frac{j[endpointofthesignal] - j[endpointofthesignal-1]}{1} \tag{4.3}$$

Temporal deep neural networks like LSTM are expected to learn the jerk-based features already from the input sequential data but it can be observed from previous works [23, 15, 24] that still explicitly including jerk signals as well has helped in improving the performance of the conventional time series classification problems. In [10] also, the author has used the location and velocity of objects as input features vectors to infer the structural relationship between different objects. This velocity input can be considered similar to the jerk signal.

4.1.2 Frequency domain signals

The introduction of the frequency domain signals as an input to the LSTM is based on the hypothesis that connected links tend to move with similar frequency and thus it can better help in determining the cause and effect sort of relations between a pair of keypoints. The frequency domain signal represents the energy content of the time domain signal in any frequency bin. So, if two keypoints are connected and are moving with similar frequencies they are expected to have a peak in the spectral domain at frequencies close to each other and they are also expected to have similar energy content overall or either in a specific frequency bin. It has also been observed in several previous works of time series classification problems where the inclusion of frequency signals with LSTM models has helped in getting a significant increase in overall performance [15, 24, 21, 20]. Effect of frequency signals can be understood with the given example in Fig 4.1, if there are two time domain signals $x1$ and $x2$ such that $x1$ is a sine curve with a frequency of $500Hz$ and $x2$ is a cosine curve with a frequency of $500Hz$ and both of the signals are also added with some different random noise. As it can be observed even though it's hard to determine the relationship between them in the time domain, in the frequency domain relationship between them is clearly visible as both of the frequency signals are peaking up at the same frequency of $500Hz$.

To extract the frequency domain information from a time domain signal, power spectral density (PSD) is estimated using the welch method [32]. The Welch method is essentially a three-step process which starts by dividing the input time domain signal into overlapped segments followed by computing periodograms for each of them and then finally averaging them to get the PSD estimate.

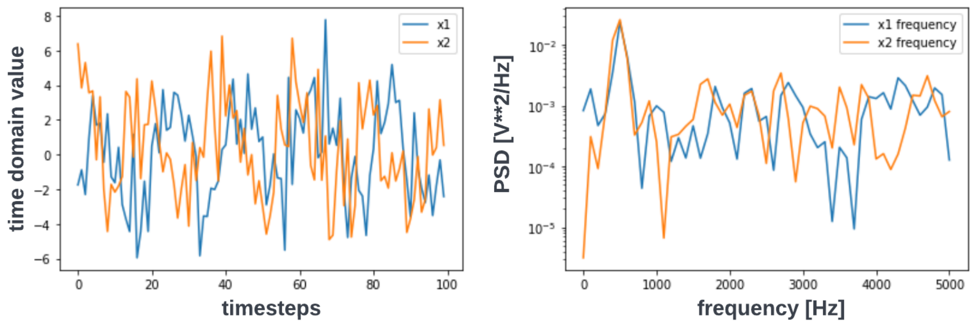


Figure 4.1: Time domain (left) to frequency domain (right) conversion.

4.2 Object structure prediction model architectures

To solve this object structure identification problem based on sequential input of key-points positions, all of the architectures proposed in this project can be categorised into two major categories:- Link Level and Object Level models.

Link Level models convert this object structure recognition problem to a binary classification problem where given signals for a pair of keypoints, the model has to predict whether there exists a link ($class = 1$) or not ($class = 0$). The advantage of using this type of architecture is that these methods are independent of the object structure, It just predicts 0 or 1 for each possible pair of keypoints. It is also easy to build such deep learning architectures where input sizes (input signals for pair of keypoints) and output sizes (0 or 1) will always be fixed. The major disadvantage of using such models in our case is that it's hard to use such models for real-time predictions of the structure which is represented by a lot of keypoints. For example, for 12 keypoints it has to make $12^2_c = 66$ predictions and this trend increases drastically.

Object Level models consider each episode or video as a single data point and predict a varying size symmetric adjacency matrix for each data point. The major advantage of using such models is that irrespective of the number of keypoints of an object it can predict the entire structure at once. It's more suited as a real-time approach for structure prediction problems. But there exist few engineering challenges while building such deep learning architecture. These methods should be able to handle varying input sizes and they also need to make sure that output size depends on the varying input sizes which can even change within a single batch of data. For example, for 15 keypoints input data output should be of the size $15X15$.

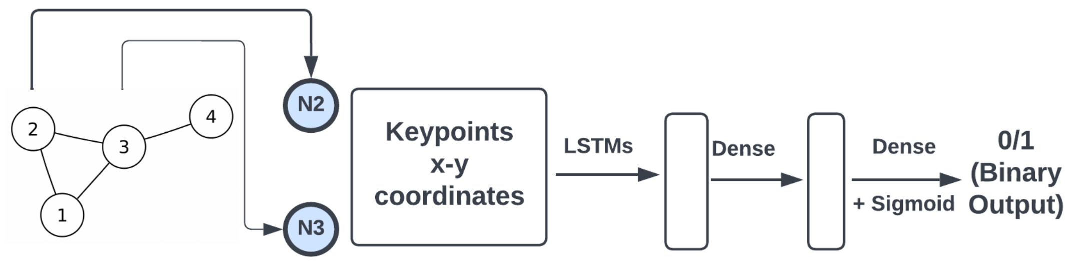


Figure 4.2: LSTM baseline.

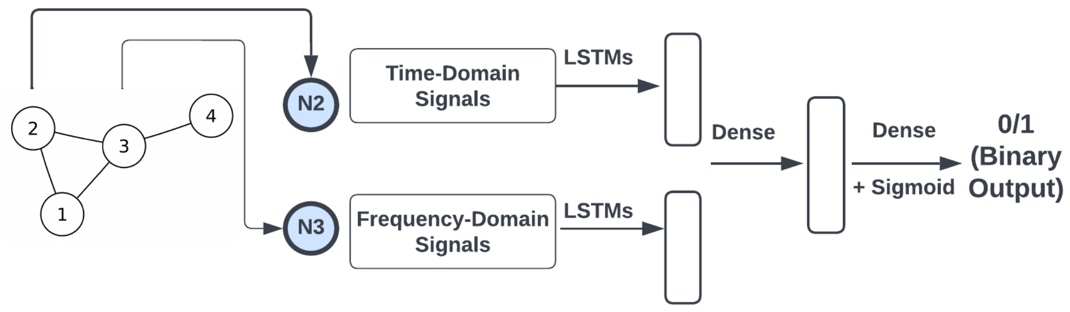


Figure 4.3: LSTM with frequency and jerk signals.

4.2.1 Link Level prediction models

4.2.1.1 LSTM Baseline

The baseline model (Fig.4.2) takes x and y coordinates signals of both keypoints in a pair and feeds them into a sequence of LSTM layers followed by a dense and sigmoid layer to give the link prediction probability between the input pair of keypoints.

4.2.1.2 LSTM with jerk and frequency signals

This method (Fig.4.3) first of all takes x and y coordinates signals of both keypoints in a pair and calculates the frequency and jerk signals for each pair coordinates. All of the time domain and frequency domain signals are passed through a separate series of LSTM layers; one for time domain signals and another for frequency domain signals. Outputs of these two LSTM blocks are then eventually concatenated and passed through a series of dense layers and a sigmoid layer to give the link prediction probability between the input pair of keypoints.

4.2.1.3 LSTM with jerk and frequency signals combined with neighbouring keypoints information

Two architectures discussed till now are using only inputs from the pair of keypoints between which link prediction is happening, no inputs from the neighbouring keypoints are considered. In order to include the neighbouring keypoints information as well, an LSTM-based architecture with required masking and padding is proposed which can learn to predict the link based on the signals from the input pair of keypoints as well as from the varying number of neighbouring keypoints. It's expected that because of the addition of contextual information from neighbouring keypoints, It can lead to a significant increase in the performance of the link prediction model.

This architecture (Fig.4.4) is also a link-level model but it learns from all of the object's keypoints to predict the link between any input pair. Consider the pair of keypoints between which the link is to be predicted as the target pair and all other keypoints inputs as the contextual or neighbouring input. This architecture can be visualised as a multiple inputs model where from one end it's learning the target keypoints representation and from the other end it's learning the neighbouring keypoints representation. First, it calculates the frequency and jerk signals for each of the keypoints. Then, all input signals for the target pair of keypoints are passed through two series of LSTM layers one for the time domain and another for the frequency domain signals and the output of these two series of LSTM layers is concatenated as the target pair representation. All of the neighbouring keypoints are also passed through a series of two LSTM layers with shared weights in an attempt to learn a unified LSTM embedding network for the time domain and frequency domain signals of all of the neighbouring keypoints. All of these LSTMs outputs are flattened and concatenated and then passed through a dense network to generate neighbouring keypoints representation. This neighbouring keypoints representation is then again concatenated with target pair representation and passed through a series of dense layers and a sigmoid layer for link prediction probability between the input pair of keypoints.

Additionally, because of the varying number of neighbouring keypoints, this architecture also needs the support of the Masking and padding. First of all, neighbouring keypoints information are to be padded to some maximum number of keypoints as the padding dimension, let's say N_{max}. Now a masking layer is created for neighbouring inputs to mask the pair of the keypoints already considered in the target pair and padded keypoints inputs. The same mask would also be propagated for subsequent layers as

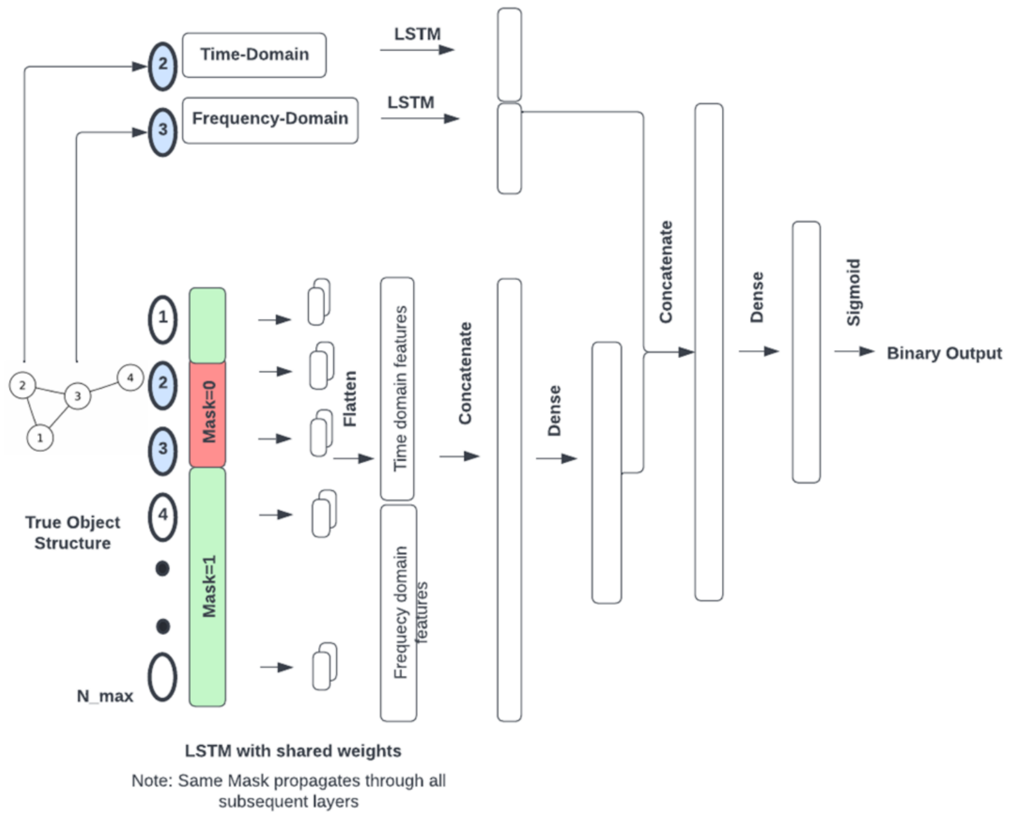


Figure 4.4: LSTM with frequency and jerk signals of target input pair + Neighbouring keypoints information.

well.

4.2.2 Object Level prediction models

4.2.2.1 Masked LSTM with Multi-Headed Self-Attention Transformers

LSTM-based architecture discussed in the previous section was able to include the contextual information from all of the neighbouring keypoints as well. But it was still making link predictions for each possible pairs and this approach becomes less and less ideal as the number of key points of an object increases. In order to avoid this problem and make the solution more feasible for real-time implementation, a Transformer based architecture is proposed that can take all of the keypoints as input and predict the entire structure at once. It also solves the problem of varying input sizes with the help of the padding mask. Additionally, it also uses the masking and an inner product layer to deal with the fact that this architecture is supposed to predict a symmetric matrix, the size of which depends upon the varying input size.

Transformers models with the multi-headed self-attention have proved their effectiveness in learning the contextual embeddings in several previous works in the field of NLP [18]. This architecture is also based on the similar hypothesis that the transformer layers might be able to learn to produce the contextualised embeddings of the keypoints in a way such that connected keypoints will have more similar embeddings compared with the keypoints which are not connected to each other. And then the link predictions can be done based on a similarity check between these updated embeddings.

From an architectural point of view (Fig.4.5), first of all, frequency and jerk signals are calculated for all of the keypoints to be used as input for this architecture. Then these inputs are padded up to specific padding dimensions to deal with the varying number of keypoints. There are no restrictions in terms of the value, the padding dimensions could take (in text-related tasks usually 512 or 1024 is used as a padding dimension). The first layer of this architecture is basically an LSTM embedding layer with the shared weights for all of the keypoints. This layer is actually responsible to take all of the signals as input for a keypoint and generate a keypoint embedding that can be used as an input for the subsequent transformer layers. The mask in the first layer would simply be True for all of the keypoints and False for all of the padded values and it would be propagated unchanged to the second layer. The second layer is a simple concatenation layer which will concatenate the LSTM outputs for time domain and frequency domain signals and it would also pass the mask unchanged to the next layer.

The third layer (actually it's a block of three layers of transformers) is the core of the architecture where multi-headed self-attention transformers layers are used to generate the contextualised embedding for each of the keypoints and masking would make sure that padded units are not affecting these embeddings. This layer would also simply pass the masking layer unchanged to the fourth and the last layer which is responsible for generating the symmetric adjacency matrix. It's expected that previous transformer layers have updated the embeddings for each of the keypoints in a way such that connected keypoints would have similar embeddings. Thus to predict the adjacency matrix, the fourth layer is implementing an inner product operation. Mask is also need to be changed at this layer so that loss can be calculated correctly. As mentioned in the Fig.4.5, if the padding dimension is N_{max}, then the output of this layer would be a matrix of the size of $N_{max}XN_{max}$ (because of inner-product operation). Assuming

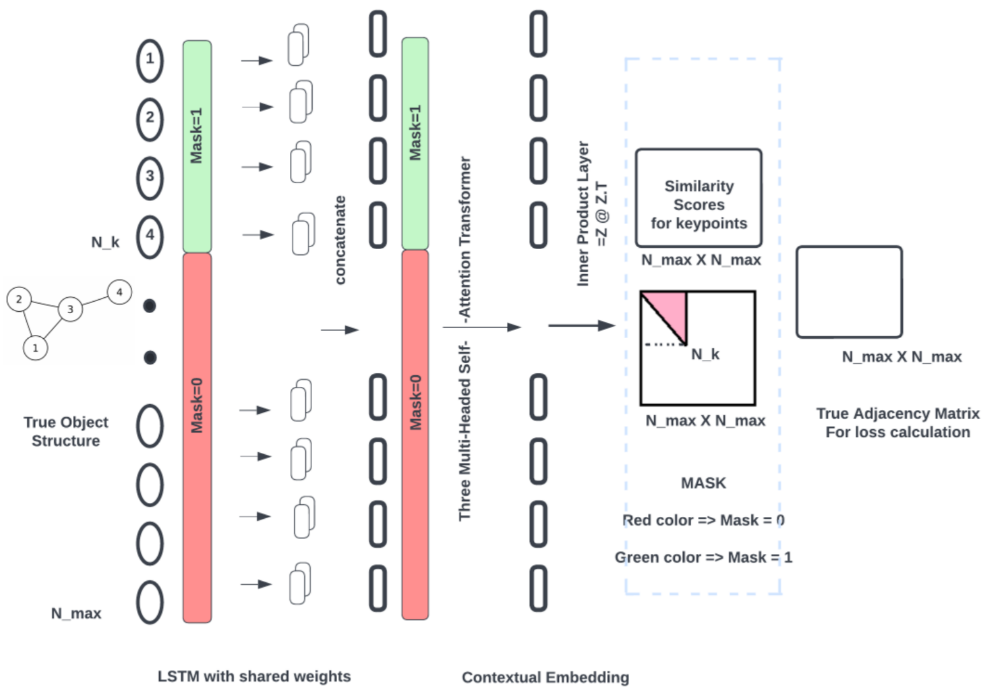


Figure 4.5: Masked Multi-Headed Self-Attention Transformer architecture.

the number of keypoints for a specific input or episode to be N_k, then the adjacency matrix should be of the size $N_k X N_k$ and thus loss should be calculated only using the first N_k rows and first N_k columns of the output matrix of this layer. Additionally, It is also expected to produce a symmetric adjacency matrix and thus the loss should actually be calculated only with the upper triangular part of $N_k X N_K$ submatrix in the bigger output matrix of the size $N_{max} X N_{max}$. Thus, the mask would change at this layer and becomes true only for the upper triangular part of $N_k X N_K$ matrix and then based on the conventional binary cross entropy loss with logits whole architecture can learn to correctly predict the symmetric adjacency matrix. This inner product layer at the end would force the transformer layers to generate similar embeddings for connected keypoints and dissimilar embeddings for the keypoints which are not connected.

4.3 Evaluation Metrics

For evaluating the performances of the proposed architectures, the entire problem is treated like a binary classification. So, even if a method predicts the entire structure i.e. symmetric varying size adjacency matrix at once, the performance metric is calculated by comparing the link predictions for each possible pair of keypoints. In this project, $class = 1$ refers to the class indicating link presence and $class = 0$ refers to the class indicating link absence. This problem is a classic example of an imbalance dataset where $class = 0$ would be a majority class, so instead of just looking directly at classification accuracy is not a good idea as it would be biased towards the link absence class. To avoid this problem, another performance metric called F1-Score has been used. This metric can be calculated separately for both classes, but in this project F1-Score for $class = 1$ i.e. link presence class is reported for all architectures. To calculate F1-Score, first, two other metrics Precision and Recall need to be calculated and then the harmonic mean of these two metrics are used to calculate the F1-Score as described in eq. 4.6. But before calculating these metrics, one first needs to calculate True Negatives (TN), True Positives (TP), False Positives (FP) and False Negatives (FN). FP refers to the number of instances where the model predicted a link between two keypoints which are not connected. FN refers to the number of instances where the model is not able to predict the link between two connected keypoints. TP and TN refers to the number of instances where the model correctly predicted the presence and absence of links respectively.

Results of the few LSTM models varied considerably, so F1-Scores presented in this entire project are average F1-score of 10 model iterations.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.4}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.5}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.6}$$

Chapter 5

Experiments and Results

5.1 Different data settings

5.1.1 Models comparison with similar topologies in train and test

In this setting, a stratified sampling has been done with respect to different structures of both rigid and dynamic length robotic arms. In other words, object structures for which predictions have to be done at test time have already been seen at training time. As expected models are performing well for already seen topologies at training time as shown in the Table 5.1. Even though all of the models are performing good, a performance increase can be observed between the models trained with just x-y coordinate signals and models trained with frequency signals in addition. There is also a significant increase in F1-Score in both the cases of rigid and dynamic length robotic arms when neighbouring keypoints information is considered. Object Level prediction using the proposed transformer-based architecture is not performing as good as LSTM with frequency and neighbouring keypoints information but it's still the second best performing model with an advantage of more real-time architecture as it can predict the entire structure at once instead of performing pairwise predictions for all possible pairs.

5.1.2 Models comparison with unseen topologies in test

In this experiment, train-test splitting is done in a way so that object structures in the test set have not been seen by the model at the training time. Particularly, to make the experiment more challenging, models have been trained on 3 and 4-link robotic arms with different structures in terms of arm length, end effector shapes or different links arrangements and then tested over a larger number of keypoints i.e, 5-link robotic

		F1-Score (Link Presence) in %	
Prediction Level	**Model Architecture**	**Fixed Length Robotic Arm**	**Dynamic Length Robotic Arm**
Link Level	**LSTM Baseline**	82.55	86.94
Link Level	**LSTM with jerk signals**	81.63	87.74
Link Level	**LSTM with frequency and jerk signals**	89.62	92.63
Link Level	**LSTM with frequency and jerk signals + Neighbouring keypoints**	95	93
Object Level	**Masked Multi-headed Self-attention Transformer**	87.24	92

Table 5.1: Model performances with similar topologies in train and test set

arms with different structures completely unseen at the training time. It can be clearly seen that the models are not performing well in these experiments when compared with the seen topologies experiments in the previous section. LSTM model with just x-y coordinates is the worst performing model but it can be observed model performance has increased significantly when frequency and jerk signals are also included. Similar to the unseen topologies experiments in the previous section, the LSTM model with frequency and jerk signals combined with neighbouring keypoints information has the highest F1-score among all architectures in both cases of rigid as well dynamic length robotic arms. Even though the proposed transformer-based architecture is not the best performing model when compared with the Link Level best performing model but it still has a significant improvement when compared with LSTM baselines and similar performance to LSTM with frequency signals Table 5.2. Sample predictions in Fig 5.1 also show the improvement in performance when LSTM models are used with neighbouring keypoints information and over-predictive nature for links in the case of transformer-based architecture.

Note: A direct comparison can't be done between the fixed length robotic arms and dynamic length robotic arms experiments as each of these data points are created at random performing random motions. But It can safely be concluded, that the proposed

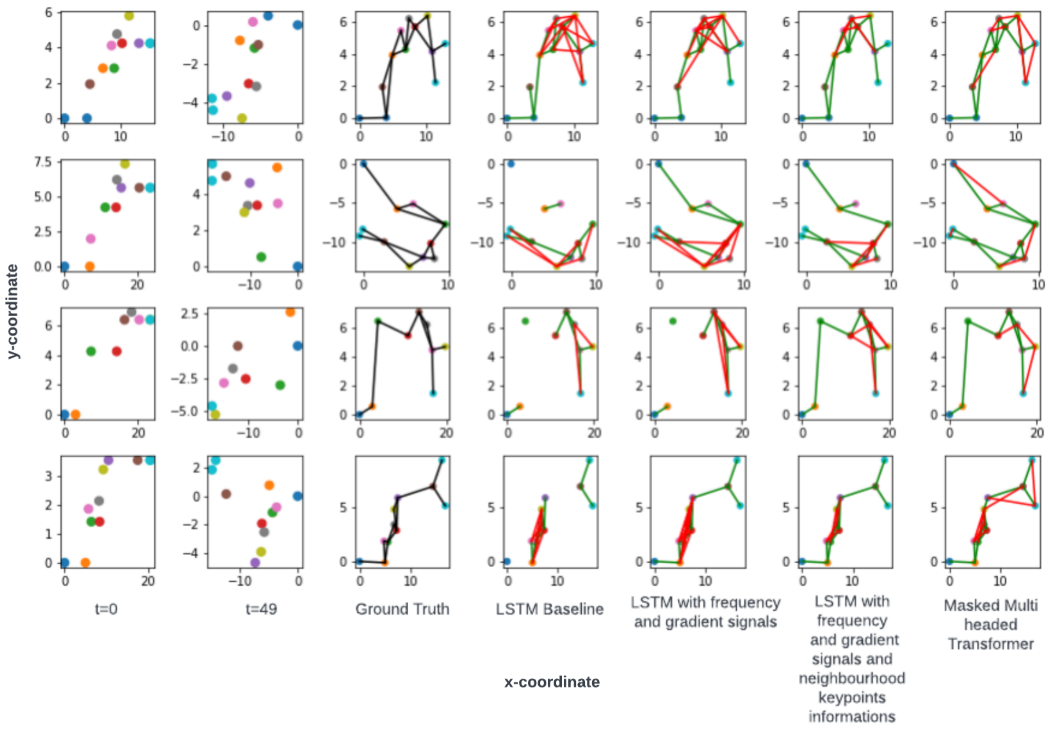


Figure 5.1: Sample predictions for each model architecture for Fixed length robotic arms and with unseen topologies at the test time.

model architectures are giving almost similar performances in both of the cases.

5.1.3 Multiple objects in a scene

This experiment is an attempt to train and test our model in the case of multiple objects in a single episode. To make the data simulation easy, objects in the video are not interacting with each other, each of the objects are performing their motion independently from each other. In this experiment, Masked Multi-headed Self-attention Transformer architecture is trained with a pair of 3 and 4-link robotic arms performing different actions simultaneously in a video and tested over a dataset where each data point has a pair of 5-link robotic arms in a single video. Transformer-based architecture gives the F1-Score of 81% and 79% in the case of fixed length and dynamic length robotic arms respectively as shown in Table 5.3. Additionally, a metric called inter-structure false positive (FP) rate has also been calculated which refers to the percentage

		F1-Score (Link Presence) in %	
Prediction Level	**Model Architecture**	**Fixed Length Robotic Arm**	**Dynamic Length Robotic Arm**
Link Level	**LSTM Baseline**	70.76	73.73
Link Level	**LSTM with jerk signals**	73.11	76.81
Link Level	**LSTM with frequency and jerk signals**	86.66	83.8
Link Level	**LSTM with frequency and jerk signals + Neighbouring keypoints**	89	87.82
Object Level	**Masked Multi-headed Self-attention Transformer**	83	85

Table 5.2: Model performances with unseen topologies in test set

		F1-Score (Link Presence) in %	
Prediction Setting	**Model Architecture**	**Fixed Length Robotic Arm**	**Dynamic Length Robotic Arm**
Object Level	**Masked Multi headed Self Attention Transformer**	81	79

Table 5.3: Masked Multi headed self-attention transformer performance for a pair of fixed-length robotic arms in a single episode simulation

of false link predictions between the keypoints of two different objects out of total false link predictions. The Transformer based architecture has the inter-structure FP rate of 0.5% and 2.5% in the case of fixed length and dynamic length robotic arms respectively. Sample predictions for the fixed-length robotic arm can be shown in Fig 5.2.

5.2 Results comparison

Though a similar trend has been observed in different experiments, the analysis of the following results has been focused on fixed length robotics arms with unseen topologies in the test set, so that we can compare different model architectures on a common

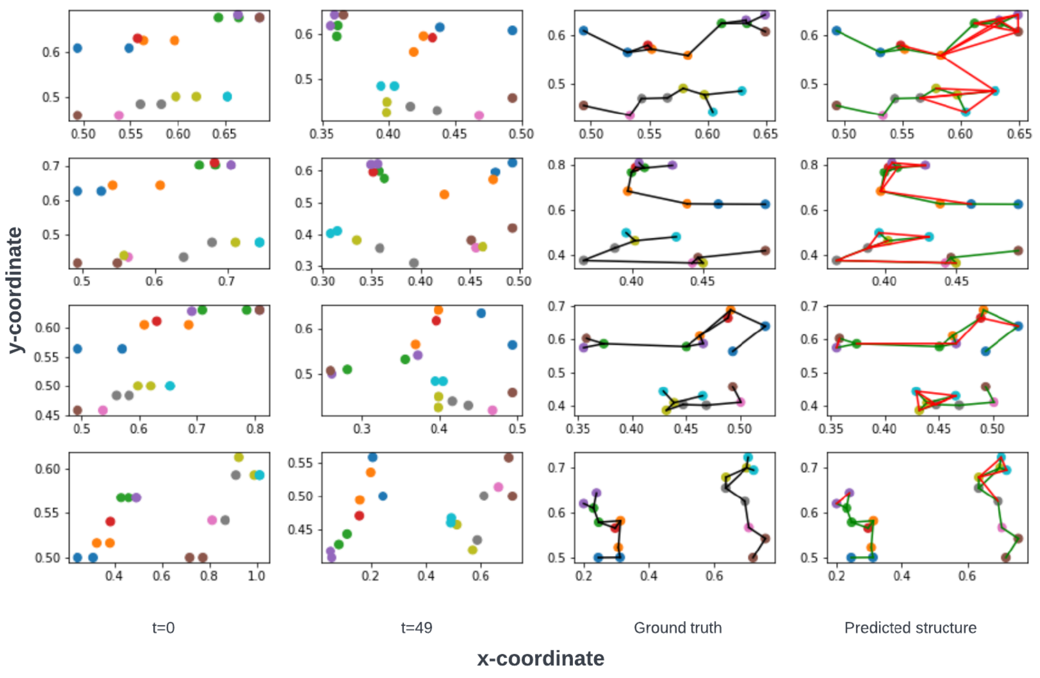


Figure 5.2: Sample predictions for each model architecture for a pair of Fixed length robotic arms in a simulation and with unseen topologies at the test time.

ground.

5.2.1 with vs without Frequency signals models

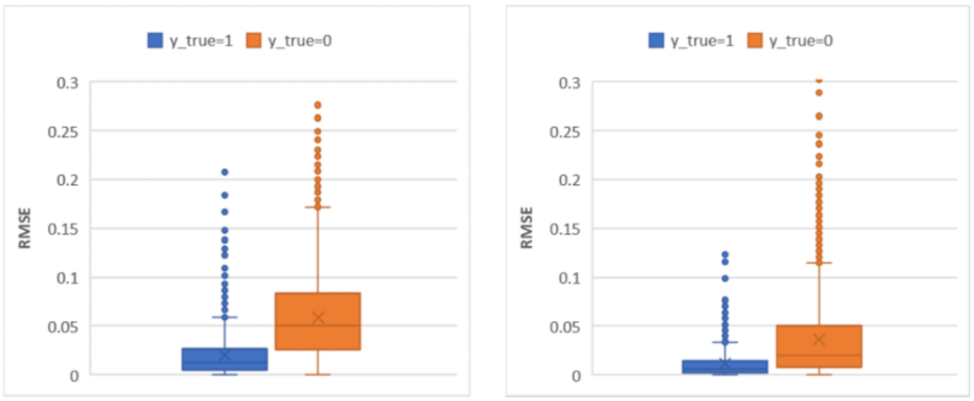
It has been observed that the addition of frequency signals has increased the performance in each of the experiments which suggests that frequency signals are able to capture the causal connection or the links between the keypoints better than conventional time domain or x-y coordinate signals. This hypothesis is further verified in Fig 5.3 where the distribution of the RMSE values between frequency signals of x-y coordinates for each pair of keypoints are plotted. It can clearly be observed from this figure that when there exists a link between a pair of keypoints RMSE is lesser compared to where a link doesn't exist between a pair of keypoints. In other words, connected keypoints have similar frequency signals thus lesser RMSE and keypoints which are not connected have dissimilar frequency signals and thus on average higher RMSE. It has actually also been observed that total energy content for a keypoint increases as its distance from the anchor point increases and thus giving a sense of relative positions of each keypoints in a simulation which is also helping in link prediction. It also helps in quickly identifying end effectors' keypoints as usually end effectors have the highest energy content among all other keypoints.

Comparing LSTM with frequency and jerk signals with the LSTM baseline model, It has also been observed that this model helps in improving the prediction of links more and thus ends up over predicting more False links as well which can be seen in Fig 5.1.

5.2.2 with vs without Neighbouring keypoints information models

Link Level prediction model using LSTM with the introduction of neighbouring keypoints information has proved to be the best performing model in all of the experiments. But it's not the most ideal option as well because of two reasons: i) This model is making predictions for each possible pair of keypoints and as the number of keypoints increases the number of predictions to be made increases drastically. ii) Also for each pair of key points, it needs all neighbouring keypoints information and thus making it hard to train with a larger number of keypoints.

Comparing LSTM using neighbouring keypoints inputs with LSTM using frequency and jerk signals but no neighbouring keypoints information, It has been observed that this model controls the over-predictions of False links which can also be seen in fig 5.1.



RMSE between X-coordinate frequency signals for a pair of keypoints

RMSE between Y-coordinate frequency signals for a pair of keypoints

Figure 5.3: RMSE between X-coordinate and Y-coordinate frequency signals for a pair of keypoints. y_{true} represents whether there exists a link between a pair of keypoints ($y_{true} = 1$) or not ($y_{true} = 0$).

5.2.3 Masked Multi-headed Self-attention Transformer

Even though the proposed transformer-based architecture is not the best performing model in terms of F1 scores but it's still performing better than that of the LSTM baseline and similar to that of LSTM with frequency and jerk signals. And the major advantage of using this architecture is that it is more suited for real-time prediction as this can predict the entire object structure at once and doesn't depends on the number of keypoints in the object structure.

It has also been observed this transformer architecture has a very high recall for the Link presence class i.e, this architecture is able to correctly predict all of the links but is also a victim of over-prediction of false links as the precision of the Link presence class is a bit on a lower side when compared with the other architectures. This dip in performance when compared with the link-level models 5.2 can also be attributed to the fact that the transformer is a data-hungry model and thus increasing the training data could help in improving its performance.

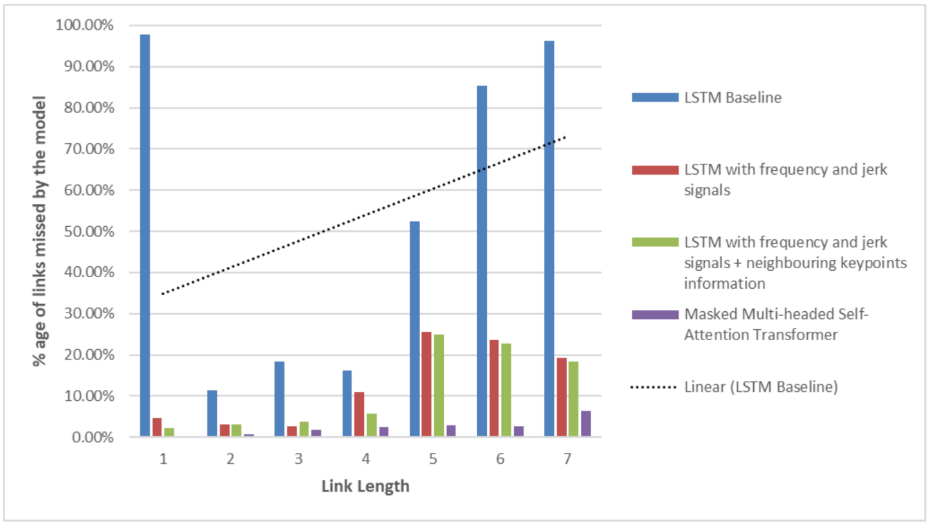


Figure 5.4: Effect of link length on model performance. The x-axis here refers to the link length and the y-axis refers to the percentage of actual links that are not predicted by the model. Other experiments have a similar trend as well but this graph refers to Fixed length robotic arms and unseen topologies at test time.

5.3 Effect of the length of a rigid connection between keypoints

In this section, an additional analysis has been done in order to identify the effect of link length on the model performances. As expected among all of the architectures that have been tried out an increase in link length increases the chances of it getting not predicted by the model as shown in Fig. 5.4. This effect is reduced by a lot with the addition of frequency signals and neighbouring keypoints information to the LSTM for link-level prediction models. It can also be observed that the transformer-based model has reduced this effect to almost negligible which suggests that transformer layers are able to build more effective contextualised embeddings for keypoints unbiased by the link length such that connected keypoints will have similar embeddings and thus inner product between these two keypoints will give the high link prediction probability. It can also be attributed to the fact that the transformer-based architecture has a recall of around 97% for link prediction. Fig 5.4 also suggests that very small length links are also tough to predict but this problem largely exists for the LSTM baseline model only and for all other models this effect has been reduced to almost negligible.

Chapter 6

Discussion

6.1 Comparison with the previous similar works

Unlike almost all previous works, this project focuses on building an explicit structure prediction model in a supervised setting with the objective of determining the best solution for the object structure detection problem. To further prove the effectiveness of proposed architectures, several experiments are conducted with different possible kinematic structures of objects and training settings like rigid and dynamically changing arm length, seen and unseen topologies in the test set and multiple objects in a scene.

Few of the previous works, [16, 10] who have modelled the structure prediction model explicitly, have used only the location or velocity of the keypoints as inputs. Unlike them, in this project, in addition to using the location and velocity of keypoints as time domain inputs, the use of frequency signals for each of the keypoints as additional input signals has been proposed to determine the relationships between the keypoints for the link prediction task and it has also shown significant improvement in all of the experiments as mentioned in the section 5.1. The use of frequency signals was based on the hypothesis that connected keypoints tend to have more similar frequency signals than the keypoints pair which are not connected as described in more details in the section 5.2.1.

The object structure prediction problem in this project can be closely related to standard structure detection problems in other domains like social network analysis [4], recommender systems [34] or drug discovery in the medical domain [8]. And in most of those works authors have used GNNs to achieve the state of art results. The major dissimilarity, when compared to the object structure problem, is that conventional structure recognition problems expect a partial set of connections as input to predict the complete set of connections. But in the object structure detection, only input trajectories of keypoints are given and then the entire set of connections needs to be predicted. To solve this issue, [10, 16] have used a probability distribution over the hidden interactions between keypoints using a variational autoencoder [9] or interaction networks [2] as prior beliefs about the graph structure and then used it as an input to GNNs. Instead of using GNNs, in this project a transformer-based architecture has been proposed to solve the issue of predicting a whole set of interactions just based on keypoints trajectories as the input. GNNs also have the added advantage of being able to handle the varying input sizes (because of the varying number of keypoints) and to produce a symmetric square matrix called adjacency matrix with the size depending on the varying input size. In order to handle this problem, the proposed transformer-based architecture is also supported with padding, a custom masking technique and an inner product layer at the end. This architecture is based on the hypothesis that transformer layers can be trained to generate contextualised embeddings of each of the keypoints such that connected keypoints will have similar embeddings.

Also, most of the past works for object structure detection have not experimented on multiple objects in a single episode setting. Authors in [10, 16] have claimed that they experimented on multiple object interactions but each of those multiple objects was just represented by a single keypoint. The real challenge occurs when those multiple objects are represented with different numbers of keypoints (more than 1). The major challenge that occurs here is that the structure prediction model should not be predicting any connections between keypoints of different objects ideally and also it should be able to handle such a larger number of keypoints because of multiple objects. In this project, the proposed transformer-based architecture has been used to experiment on a similar dataset where a pair of robotic arms are performing random motions in an episode independent of each other. F1-Scores of 81% and 79% have been observed for rigid and dynamic length robotic arms respectively with unseen topologies at the test time. It has also been observed that less than 5% of incorrectly predicted links are connecting keypoints of two different objects.

Chapter 7

Conclusions

7.1 Summary

In this project, the object structure recognition problem was solved in different training settings i.e, seen and unseen topologies at test time and with different kinematics structures i.e, fixed-length or dynamic length robotic arms or multiple objects simulations with varying number of links or keypoints. To solve this problem, two categories of methods have been proposed. In the first category i.e. Link Level models, binary predictions are made for all possible pairs of keypoints and in the second category i.e. Object Level models, the entire structure for an object is predicted at once as an adjacency matrix which is a variable-sized (depending on the number of keypoints) symmetrical square matrix.

Overall three different methods have been proposed in addition to the LSTM Baseline model to solve this problem. In the first method which is a link-level model, time domain signals of a pair of keypoints are used to calculate the corresponding frequency and jerk signals and then all of these three signals are used as input to the LSTM model to predict whether a link exists between this pair of keypoints or not. In the second method which is also a link-level model, again frequency, jerk and time domain signals for a target pair of keypoints are used as input to the LSTM model. But unlike the previous method, this will also use the neighbouring keypoints signals as contextual input to the LSTM model for the link prediction between the target pair of key points. It has been observed that the introduction of frequency signals helped in improving the performance by predicting more correct links and the introduction of neighbouring keypoints information has further increased the performance by removing the falsely predicted links because of frequency signals. LSTM model with frequency and jerk signals and also neighbouring keypoints information turns out to be the best performing model in all of the experiments. Lastly, a transformer-based architecture has been proposed with a specific mask propagation technique between the layers and an inner product layer at the end to ensure that the model can handle a varying number of keypoints and that output satisfies the necessary conditions for an adjacency matrix. Even though the transformer-based architecture is not the best performing model still it would be the preferred solution for a larger number of keypoints of an object or multiple objects (which in turn means a larger number of keypoints), as it can predict the entire object structure at once instead of making a prediction for each possible pair of keypoints which grows drastically with an increase in the number of keypoints.

Almost all of the models are performing pretty well in the case of seen topologies as it's an easy learning task for predicting already seen architecture at test time. The maximum F1 score for the link present class is observed as 95% and 93% for rigid and dynamic length robotic arms respectively using LSTM with frequency, jerk and time domain signals along with neighbouring keypoints information. F1-Score with Masked Multi-headed Self-attention transformer architecture is 87% and 92% for rigid and dynamic length robotic arms respectively. For unseen topologies, on the other hand, the maximum F1 score for the link present class is observed as 89% and 88% for rigid and dynamic length robotic arms respectively using LSTM with frequency, jerk and time domain signals and with neighbouring keypoints information as well. F1-Score with the transformer-based architecture is 83% and 85% for rigid and dynamic length robotic arms respectively. In the case of multiple object settings given the larger number of key points, transformers models are used because of single stage prediction of the entire structure at once. In this case, F1 scores of 81% and 79% have been observed for rigid and dynamic length robotic arms respectively and with unseen topologies at the test time. Additionally, a metric called inter-structure false positives rate is calculated which refers to falsely predicted links between the keypoints of the object which is pretty low in both of the cases. Additionally, the effect of the increasing link length on model performance has also been discussed and the result verifies the fact that all of the models usually find it difficult to predict the larger links when compared to shorter ones. This effect was highest in the case of the baseline model but other proposed architectures have reduced it by introducing the concept of similar frequency signals for linked keypoints and considering the effects of neighbouring keypoints as well either by passing them as another set of inputs in an LSTM network or by passing all of the keypoints at once in a transformer based architecture for better-contextualised embeddings for each keypoints.

7.2 Future Work

In this project, all proposed methods are trained in a supervised setting to predict the object structure. As a next future step, it can be tried to incorporate the same architectures in an unsupervised workflow and observe the performance. One possible way to do so is by following the similar three steps approach as followed in [10, 16]. First train an unsupervised keypoint detection method [13, 6], and use the predicted keypoints as input to the object structure prediction model discussed in this project. Finally, based on the structure detected and previous keypoints positions, predict the future keypoints and then train them in an end-to-end fashion by comparing them with the keypoints predicted by the unsupervised keypoints detector using conventional mean squared error loss.

In this project, an attempt has been made to analyse the model performances in the case of multiple objects in an episode as well but each of those objects was following their own random trajectories independent of each other. As a next step which will be more close to a real-time scenario, performances of object structure detection methods should be tried in a setting where multiple objects in an episode are allowed to interact with each other.

Several experiments with different kinematic structures and training settings have been tried in this project to test the effectiveness of proposed architectures. As a next step performance of these methods could also be analysed in real-time or 3d objects datasets with more complex function controlling the interaction between keypoints. Additionally, instead of just predicting the link existence, several other continuous variables could also be modelled like the rest length of the spring in case of spring connection between keypoints.

Bibliography

[1] Baole Ai, Zhou Qin, Wenting Shen, and Yong Li. Structure enhanced graph neural networks for link prediction. *CoRR*, abs/2201.05293, 2022.

[2] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016.

[3] Susan Carey. The origin of concepts. *Copyright*, 1:37–41, 02 2000.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. *CoRR*, abs/1902.07243, 2019.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[6] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Conditional image generation for learning the structure of visual objects, 06 2018.

[7] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14:201–211, 1973.

[8] Jintae Kim, Sera Park, Dongbo Min, and Wankyu Kim. Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences*, 22(18), September 2021. Funding Information: Funding: WK was funded by National Research Foundation of South Korea (2017M3C9A5028690). Publisher Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland.

[9] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[10] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2688–2697. PMLR, 10–15 Jul 2018.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[12] Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *CoRR*, abs/1906.11883, 2019.

[13] Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *CoRR*, abs/1906.11883, 2019.

[14] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A-statistical Mechanics and Its Applications*, 553:124289, 2020.

[15] Xin Li, Yu Zhu, and Kai-ming Yang. Self-adaptive composite control for flexible joint robot based on rbf neural network. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 2, pages 837–840, 2010.

[16] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. *CoRR*, abs/2007.00631, 2020.

[17] Alberto Dalla Libera, Matteo Terzi, Alessandro Rossi, Gian Antonio Susto, and Ruggero Carli. Robot kinematic structure classification from time series of visual data. *CoRR*, abs/1903.04410, 2019.

[18] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *CoRR*, abs/2106.04554, 2021.

[19] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *CoRR*, abs/1906.07889, 2019.

[20] Abdullah-Al Nahid, Mohamad Ali Mehrabi, and Yinan Kong. Frequency-domain information along with lstm and gru methods for histopathological breast-image classification. In *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 410–415, 2017.

[21] Tuan Pham. Time–frequency time–space lstm for robust classification of physiological signals. *Scientific Reports*, 11:6936, 03 2021.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[23] S. Saminu, G. Xu, S. Zhang, A. E. K. Isselmou, A. H. Jabire, Y. K. Ahmed, H. A. Aliyu, M. J. Adamu, A. Y. Iliyasu, and F. A. Umar. A novel computer aided detection system for detection of focal and non-focal eeg signals using optimized deep neural network. In *2021 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–4, 2021.

[24] Ionut-Cristian Severin and Dan-Marius Dobrea. 6dof inertial imu head gesture detection: Performance analysis using fourier transform and jerk-based feature extraction. In *2020 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW)*, volume 1, pages 118–123, 2020.

[25] Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental science*, 10 1:89–96, 2007.

[26] Antoine J.-P. Tixier. Notes on deep learning for NLP. *CoRR*, abs/1808.09772, 2018.

[27] Ivan Virgala, Michal Kelemen, and Erik Prada. Kinematics of serial manipulators. In Constantin Voloşencu, Serdar Küçük, José Guerrero, and Oscar Valero, editors, *Automation and Control*, chapter 7. IntechOpen, Rijeka, 2020.

[28] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *CoRR*, abs/1808.06601, 2018.

[29] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[30] Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *CoRR*, abs/1904.05046, 2019.

[31] Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter W. Battaglia, and Daniel Zoran. Visual interaction networks. *CoRR*, abs/1706.01433, 2017.

[32] Peter Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.

[33] Patrick Winston. Learning structural descriptions from examples. 08 2005.

[34] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. Graph neural networks in recommender systems: A survey. *CoRR*, abs/2011.02260, 2020.

[35] Guangluan Xu, Xiaoke Wang, Yang Wang, Daoyu Lin, Xian Sun, and Kun Fu. Edge-nodes representation neural machine for link prediction. *Algorithms*, 12:12, 01 2019.

[36] Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Unsupervised discovery of parts, structure, and dynamics. *CoRR*, abs/1903.05136, 2019.

[37] Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. *CoRR*, abs/2011.08843, 2020.

[38] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. pages 575–583, 08 2017.

[39] Tao Zhou and Bertram E. Shi. Simultaneous learning of the structure and kinematic model of an articulated body from point clouds. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 5248–5255, 2016.