Combinatorially Generalizing Video Prediction by learning Object- Centric World Models

Rishi Shah
Department of Computer Science
Indian Institute of Technology
Delhi, India
cs1190394@iitd.ac.in

Himanshu Gaurav Singh
Department of Computer Science
Indian Institute of Technology
Delhi, India
cs1190358@iitd.ac.in

Abstract

We present an architecture for the problem of next frame prediction. Sparse, relational interactions are inherent in the dynamics of large, real world environments and we hypothesize that incorporating these in the latent representation of the world would improve generalisation. In this work, we propose an approach to learn a structured graph-based latent representation of the state dynamics from perceptual input. Graphs provide a cogent framework to encode sparse and local structure. Thus, graph-based representations would generalize better to larger, more complex instances than seen during training. We build upon recent advances in object detection to extract an object-centric representation of the scene and learn an encoder-decoder style architecture for predicting the next frame.

1 Related Work

Learning in imagination of world models [Ha and Schmidhuber, 2018] demonstrated simulation of visual environments by learning a world model using variational autoencoders and recurrent neural networks. [Hafner et al., 2019] learns a controller network on top of a world model, demonstrating effectiveness in a simulated maze and a real robot. [Yarats et al., 2021] achieved human level performance on the Atari benchmark. More recently, DreamerV3(Hafner et al. [2023]) became the first algorithm to mine diamonds in Minecraft, a long standing challenge in RL.

Inferring latent graph structure [Kipf et al., 2018] learns to encode relational structure between entities in a graph from demonstrations of trajectories for each entity. We use it as a starting point to learn a structured latent space. Additionally, [Kazi et al., 2022] provides an end-to-end trainable model that predicts connectivity of the graph between given entities. [Veličković et al., 2020] uses algorithmic biases to learn dynamic graphs.

Object Detection Object detection is a fundamental task in computer vision that aims to locate and classify objects within an image. Over the years, significant progress has been made in this field, with the emergence of several state-of-the-art approaches. One such approach is YOLO (You Only Look Once), which revolutionized real-time object detection by proposing an end-to-end architecture that directly predicts object bounding boxes and class probabilities. YOLO achieves impressive performance with remarkable speed by dividing the input image into a grid and predicting objects at different spatial scales. Another notable method is Faster R-CNN (Region-based Convolutional Neural Network), which introduced the concept of region proposal networks (RPN) to efficiently generate region proposals. By combining a region proposal network with a region-based CNN, Faster R-CNN achieves accurate object detection with improved speed compared to previous methods. However, both of the architectures have to be trained in supervised setting. In our case, we do not use any labelled information about the object locations in the video frames. So, we rely on Segment Anything (Kirillov et al. [2023]) for extracting objects from the scene for further processing. It is

Preprint. Under review.

trained using internet-scale data: often referred to as a *foundation model* for object detection. We use the pre-trained model of Segment Anything(SAM) to segment out the mask for objects from the video frames.

Temporal GNNs Temporal Graph Neural Networks (GNNs) have shown promising results in modeling dynamic phenomena, particularly in the field of traffic forecasting. By incorporating temporal information, GNNs can capture the underlying patterns and relationships between nodes in a graph and predict their future behavior. In the context of future location prediction of objects as nodes, temporal GNNs have emerged as a powerful tool for capturing the spatio-temporal patterns of object movements and predicting their future locations. The incorporation of traffic forecasting techniques in temporal GNNs has also proven to be effective for predicting object movements in traffic-heavy areas. Various temporal GNN models have been proposed for traffic forecasting, including those that incorporate RNNs or CNNs to model temporal dependencies and those that use attention mechanisms to weight the importance of different time steps in the prediction. Some of the state of the art works are A3-TGCN([Bai et al., 2021]) and DCRNN([Li et al., 2017]) where the task is to predict the traffic at nodes for the next time stamp given previous time stamps. We incorporate the architecture of A3-TGCN in our work to predict the future bounding boxes of the objects given the previuos information.

Video Prediction Video prediction is a challenging task in computer vision that involves forecasting future frames or video sequences based on past observations. It has garnered significant attention due to its potential applications in video analysis, robotics, and autonomous systems. Several notable papers have contributed to the advancement of video prediction methods. One of the work is PredNet ([Lotter et al., 2016]) framework, which is inspired by principles from predictive coding in neuroscience. PredNet utilizes a hierarchical recurrent network to predict future video frames based on a series of preceding frames. By exploiting the spatial and temporal hierarchical structure of the data, PredNet has demonstrated promising results in video prediction tasks. Additionally, Generative Adversarial Networks (GANs) have been successfully employed for video prediction. Notable GAN-based approaches, such as VGAN ([Vondrick et al., 2016]) and MoCoGAN ([Tulyakov et al., 2018]), incorporate adversarial training to generate realistic and coherent future video frames. All the works above except MocCoGAN focus on solving the video prediction problem without taking into account the difference between background and the actual moving objects. They rely on techinques like attention to do the work for them. MoCoGAN adopts a motion and content decomposed representation for video generation. Our work is different from the above works as we focus on sparse interactions between objects explicitly by encoding them in the form of graphs.

2 Problem Formulation

Given a sequence of states $\{S_i\}$ in the form of images, our goal is to learn to predict S_t given $S_1, S_2, .., S_{t-1}$ The dataset is in the form of offline demonstrations containing sequence of images. For accurate prediction, the crucial part is to learn to simulate the dynamics of the environment and additionally, produce perceptible output on top of the understanding of the dynamics. In the light of this, we adopt a model-based approach for our problem. We learn the to embed the scene in a latent representation and predict the representation of the next frame from those of the previous k frames (encoder). Thereafter, we learn to produce images of the frame from the previous frames (decoder). An advantage of this two-part approach is that we can train both the encoder and decoder separately but use it end-to-end during inference.

3 Approach

Extracting Features from the Image We use the SAM model to perform the image segmentation task. SAM provides a model for SamAutomaticMaskGenerator which generates masks and predicted intersection over union of those masks. The original SAM model, also takes into input a mask which denotes the area of the original image we are trying to trace the object, and also a prompt which contains information about the object in textual form. As, the task at hand is video prediction, we do not make use of the prompt. The SamAutomaticMaskGenerator works by sampling single-point input masks in a grid over the image, from each of which SAM can predict multiple masks. Then, masks are filtered for quality and deduplicated using non-maximal suppression.

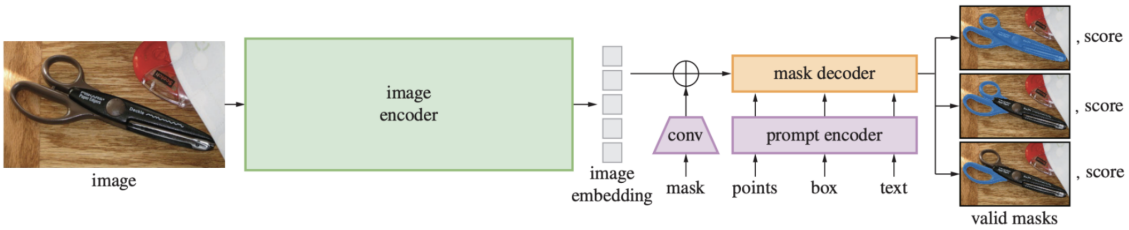


Figure 1: Architecture of Segment Anything(Image taken from [Kirillov et al., 2023])

The image segmentations provided by SAM are not accurate all the time. After SAM provides a list of masks and their predicted IOU, where each masks denotes the mask corresponding to an object or background. However, the masks can have errors like shadow masks, unable to distinguish between two close-by objects, etc. So, to solve this first we put a threshold on the IOU and only keep the filtered masks. The threshold is chosen empirically which performs the best. Secondly, we apply some heuristics such as the mask should always be connected as object is connected, then the size of the mask should be higher than a certain threshold.
Finally, we get corresponding bounding boxes using the mask of the object. We use these bounding boxes as feature vectors in our encoder model.

Modelling as Graph We encode the frames into a graph $G = (V, E)$ where node features are the bounding box location for the corresponding object in the previous frames. However, we still need to track the object in multiple frames to get the complete node feature. Here, we make use of the fact that CNNs are translation invariant. We pass the masked object images after applying standard transformation through a pre-trained resnet model to get the resnet features for every masked image. Then, we need to match objects from the previous frames with the current frame. We solve the linear sum assignment problem which is minimum weight matching in bipartite graphs. Formally, we find a one-one and onto mapping from current to previous frames such that linear norm between the resnet features is minimum. To get E, we use the top-k nearest distance node features. We keep $k = n + (n * (n-1))/2$, the idea is to keep all the self loops and half of the connections. Again, the value of k is a hyperparameter and can be tuned.

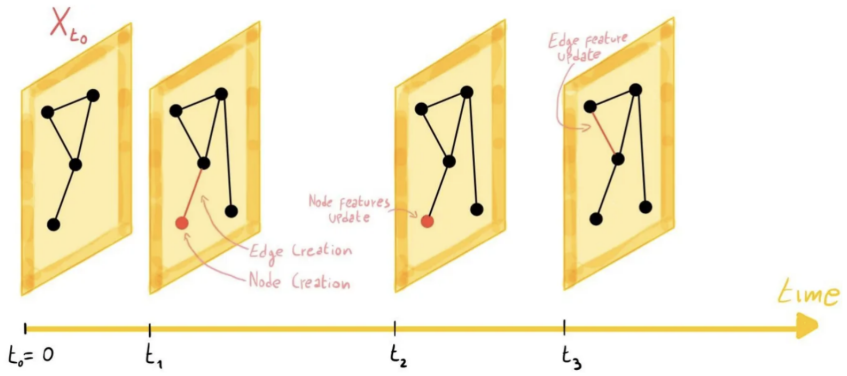
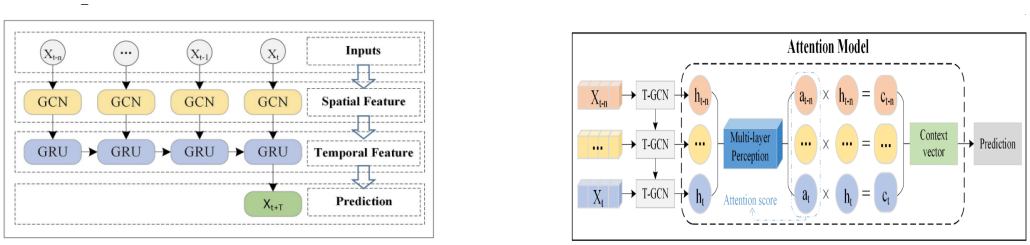


Figure 2: Node Features Changing Over Time

Encoder Model The goal of the encoder model is to predict the node features of all the nodes for the next frame. We use the architecture of A3-TGCN([Bai et al., 2021]) which is build upon the TGCN([Zhao et al., 2019]) framework. TGCN contains a layer of GCN to capture the spatial dependencies and a GRU to capture the temporal dependencies. A3-TGCN adds an attention mechanism to this architecture. Attention is calculated over the temporal features to finally calculate the context vector. The idea of attention is to check which timestamp is more important to predict the next frame. In our case, it can be that a particular timestamp where the collision occured is important

and given more attention. The context vector is passed through an MLP to get our prediction for node features corresponding to the next frame.



(a) TGCN (b) A3-TGCN

Figure 3: Graph Spatio-Temporal Architecture

Decoder Model We use [Johnson et al., 2018] for our decoder model. This architecture takes as input a scene graph in the form of node features and edge index and outputs the corresponding image. First, the scene graph is passed through a GCN. For each node in the processed scene graph (corresponding to an object), the model predicts a layout. The layouts are then combined to produce the final image. We use 3 different losses in the model. The first one is the L1 loss between the pixels of generated and actual image which acts as the regeneration loss. The second one is the mask loss where we minimize the binary cross entropy between each objects predicted and original mask. The third one is the bounding box loss where we minimize the MSE between generated and actual bounding box for every object. Note that the decoder is trained separtely from the encoder.

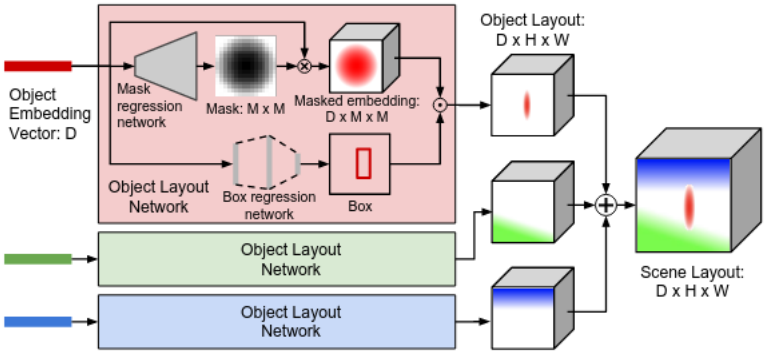


Figure 4: Architecture of the Decoder model (Image taken from [Johnson et al., 2018])

4 Datasets

We evaluate our approach on the videos in the CLEVERER([Yi* et al., 2020]) dataset. CLEVERER is a video-question answering datasets where objects in the CLEVRER videos adopt similar compositional intrinsic attributes as those found in the CLEVR dataset, including three shapes, two materials and eight colors. CLEVRER also includes 300,000 questions and answers related to the action depicted in the videos. We only work with videos of a part of the large dataset, but we make sure to use all the objects from the dataset in our videos.

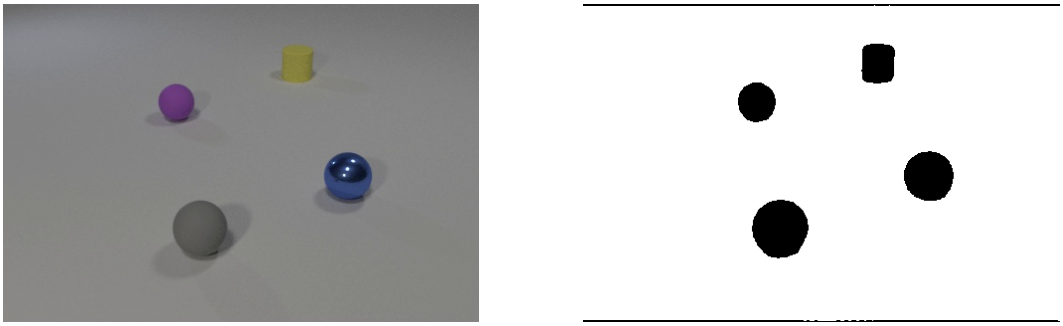
5 Experiments

Pre-processing

We take the given video and break it into frames with the interval gap of 100 milliseconds per frame. In our experiments, this generates sufficiently large number of frames, so the interval gap can be modified based upon the video lengths. In our case, the videos range from 10-20 sec long.

Object Detection

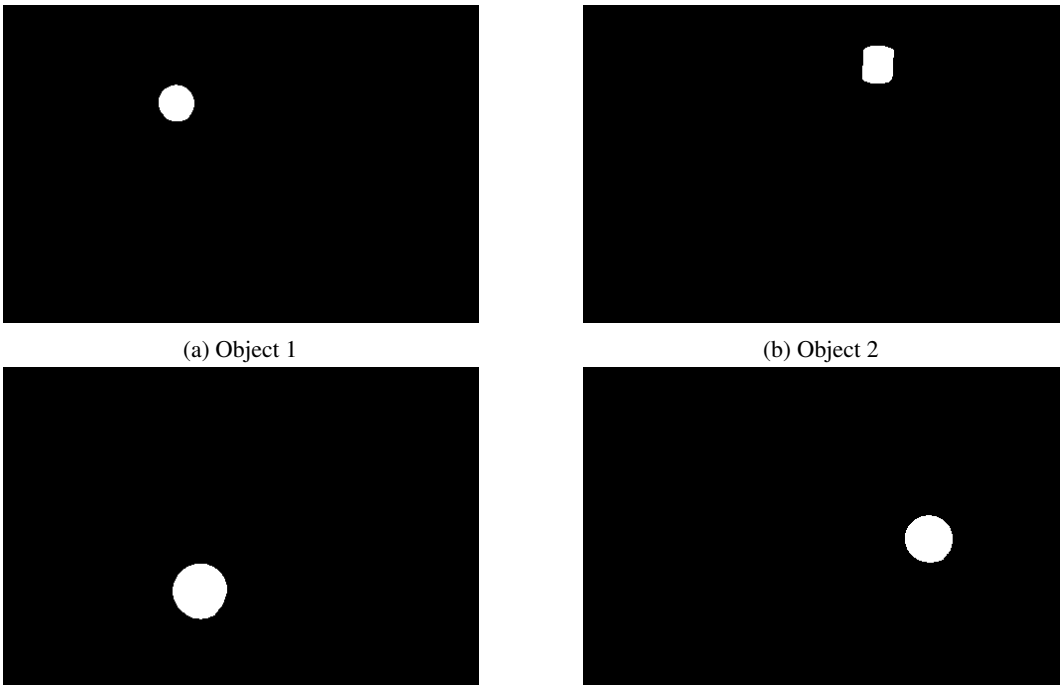
We use the SAM architecture to get the masks for object from the original image. We can see the mask for background has been correctly captured.



(a) Original Image (b) Background Mask

Figure 5: Object Segmentation

These are masks with respect to individual objects after applying the threshold and heuristics mentioned above.



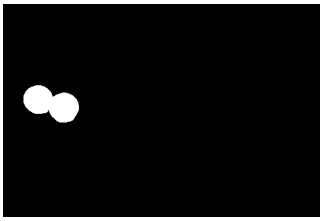
(a) Object 1 (b) Object 2

(c) Object 3 (d) object 4

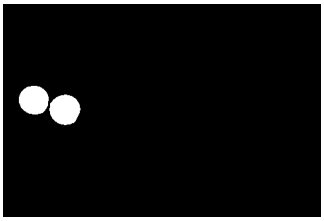
Figure 6: Object Masks

Mislabeled Masks

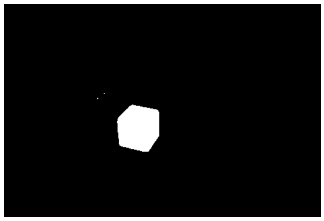
However, we found out that at some instances especially during collision our model can fail. In instance 1 and instance 2, SAM model even after applying heuristics can not distinguish between the two objects. In instance 3, other than the box, there are two more white masks which are very small. So, the bounding box is calculated incorrectly in this case. But here, our heuristic of using the single connected component works and we are able to correctly figure out desired bounding box.



(a) Instance 1



(b) Instance 2

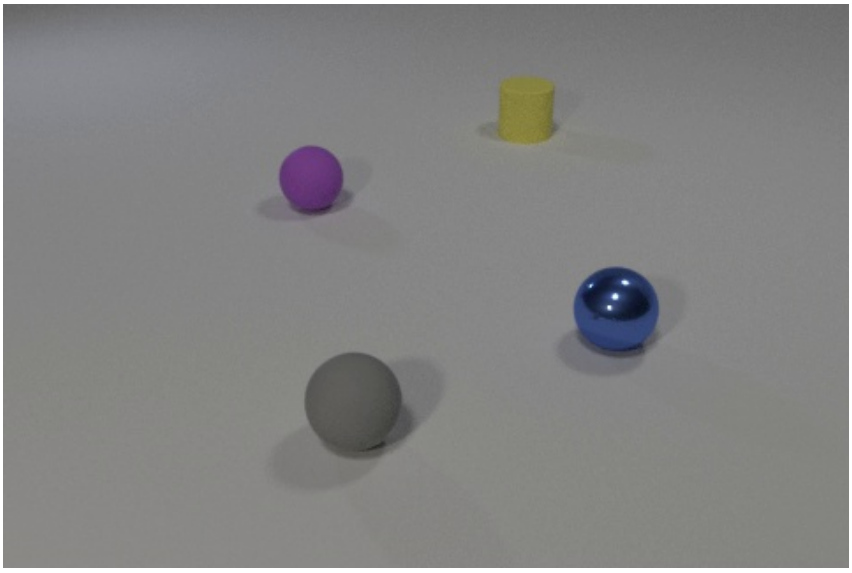


(c) Instance 3

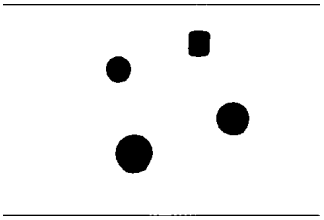
Figure 7: Mislabeled Masks

Object Tracking

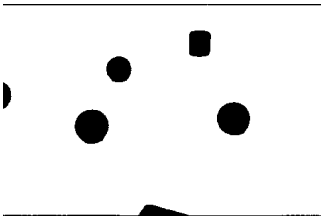
We use the linear sum assignment on resent features to track and match the features between two consecutive frames. We are correctly able to match out the features to the objects due to variation in color and shapes. The figure shows masks for variuos objects in consecutive frames in a video.



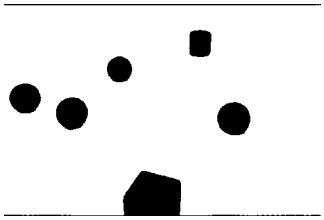
(a) Original - Frame 0



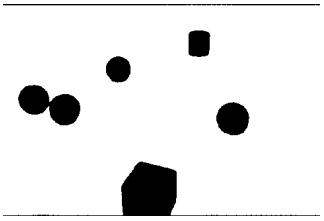
(b) Frame 0



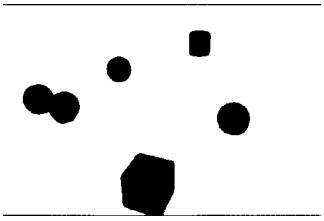
(c) Frame 10



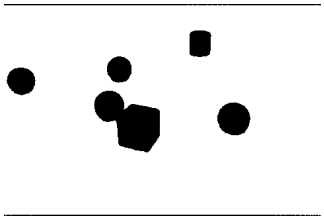
(d) Frame 16



(e) Frame 18



(f) Frame 20

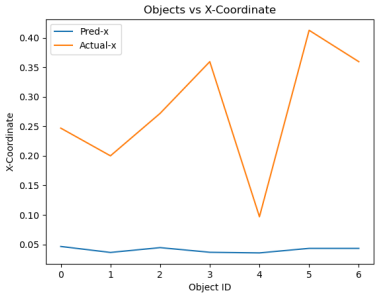


(g) Frame 36

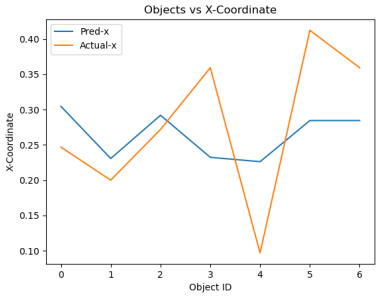
Figure 8: Object Tracking

Bounding Box Prediction

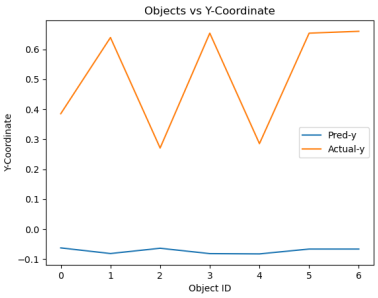
We use the A3-TGCN architecture to predict the next bounding box(node feature) given previous 'k' frames. In our experiments, we varied 'k' from 4 to 10. We report the results on 'k'=6 and we predict the next 2 bounding boxes by running the model in an auto-regressive fashion. These are the results on the input video from test file -



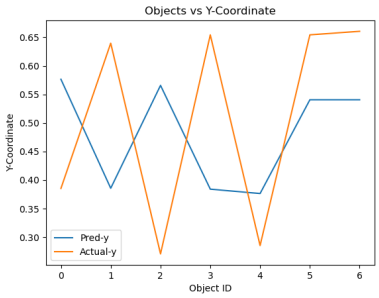
(a) Epoch 1



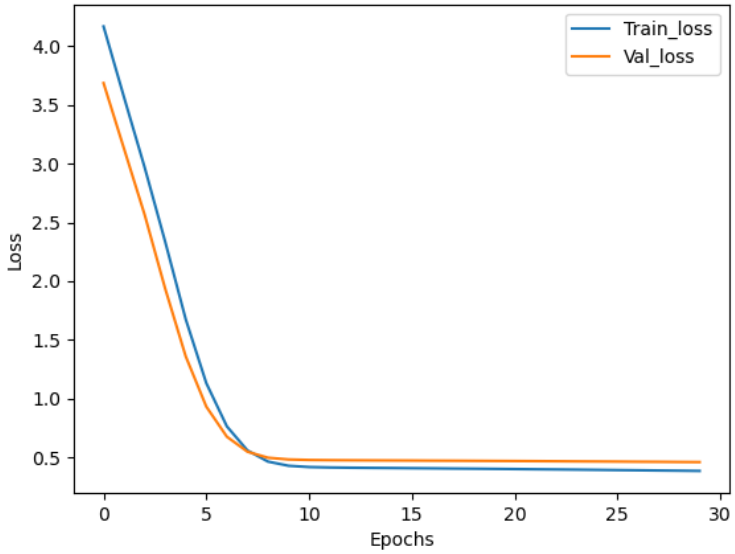
(b) Epoch 30



(c) Epoch 1



(d) Epoch 30



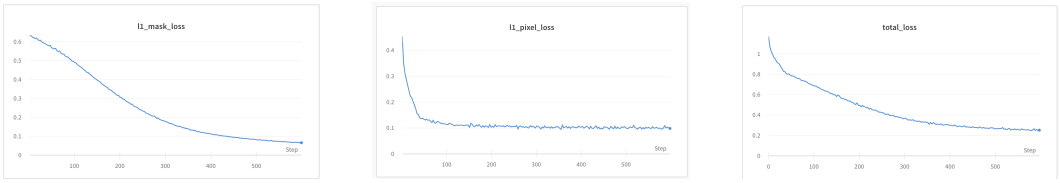
(e) Loss Vs Epoch

Figure 9: Temporal GNN Predictions

We can see that our model can predict the trend for the 'x' coordinate. However, on the 'y' coordinate, for some objects the variation is quite high and our model was not able to capture it.

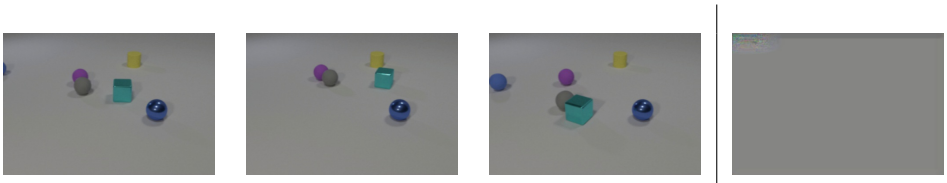
Decoding Next Frames

Finally, the last goal is that using the trained decoder we decode the next frames. The input to the decoder is the image of the previuos frame and object features corresponding to the next frame. However, we found out that our decoder was not able to capture the objects in the prediction frame despite the decreasing loss in the model. We can see that on the top of grey boxes, there are different color patches which correspond to different objects. But the decoder is not able to place the objects on the desired location.



(a) Mark Loss (b) Pixel Loss (c) Total Loss

Figure 10: Loss vs Epochs



Actual Frame 0 Actual Frame 2 Actual Frame 4 Predicted Frame

Figure 11: Generating the Next Frame using the previous frames

6 Future Work

We assume the graph to be static and rely on the attention in message passing framework to incorporate the actual sparse relations between objects. The next work can be done where we generate dynamic graphs depending upon the current frame where the edges represent the relations in the current frame. We need to do more hyper-parameter tuning and better feature extraction to get some better performance. Also, our architecture can scale to different number of objects in videos, so an experiement where we train on fewer objects and test on larger number of objects can be done. Also, we do not include features like speed, direction of the object in the feature set. We are aware of the fact that performance of GNNs are highly feature dependent, so incorporating these features can boost the performance. We use Segment Anything which is pre-trained to perform the object segmentation, which comes with its flaws as it is trained on large internet datasets. So, better pre-trained model which are dataset specific can help in better results. Morever, we directly adopt the architecture of A3-TGCN ignoring the fact that the architecture was build for traffic forcasting where the notion of attention is different as it is essentially made for modelling predictions where future repeats the past. We can develop novel GNN framework which takes into account the collision scenario and various constraints related to prediction of bounding boxes. Finally, we will need to train better decoder model which can efficiently and correctly generate the next frame given the object coordinates and a previuos frame.

References

J. Bai, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, and H. Li. A3t-gcn: Attention temporal graph convolutional network for traffic forecasting. *ISPRS International Journal of Geo-Information*, 10 (7):485, 2021.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.

A. Kazi, L. Cosmo, S.-A. Ahmadi, N. Navab, and M. M. Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2022.

T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

P. Veličković, L. Buesing, M. Overlan, R. Pascanu, O. Vinyals, and C. Blundell. Pointer graph networks. *Advances in Neural Information Processing Systems*, 33:2232–2244, 2020.

C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29, 2016.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

K. Yi*, C. Gan*, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HkxYzANYDB`.

L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.