2021

Spatio-Temporal Representation for Reasoning with Action Genome

Kesar Tumkur Narasimhamurthy
University of Central Florida

SPATIO-TEMPORAL REPRESENTATION FOR REASONING WITH ACTION GENOME

by

KESAR MURTHY
B.S Visvesvaraya Technological University, 2018

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2021

ABSTRACT

Representing Spatio-temporal information in videos has proven to be a difficult task compared to action recognition in videos involving multiple actions. A single activity consists many smaller actions that can provide better understanding of the activity. This paper tries to represent the varying information in a scene-graph format in order to answer temporal questions to obtain improved insights for the video, resulting in a directed temporal information graph. This project will use the Action Genome dataset, which is a variation of the charades dataset, to capture pairwise relationships in a graph. The model performs significantly better than the benchmark results of the dataset providing state-of-the-art results in predicate classification. The paper presents a novel Spatio-temporal scene graph for videos, represented as a directed acyclic graph that maximises the information in the scene. The results obtained in the counting task suggest some interesting finds that are described in the paper. The graph can be used for reasoning with a much lower computational requirement explored in this work among other downstream tasks such as video captioning, action recognition and more, trying to bridge the gap between videos and textual analysis.

Keywords: Spatio-temporal scene graph, Action Genome, directed acyclic graph, counting, reasoning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Video understanding tasks using action recognition mostly treats the scene and the activity as a single event. Cognition is core to not just recognizing, but reasoning with the visual world. Recent models perform an end-to-end prediction to produce a single label output for a lengthy video sequence. On the other hand, scene understanding requires numerous vision tasks such as object detection, classification of these objects and estimating the pair-wise relationships of objects in the scene.

There has been great interest in image based structured representations. This could potentially help solve problems across domains, such as image retrieval, captioning, relationship modeling, and question answering, even image generation from a scene graph has been tested. However, perception of a visual scene goes beyond recognizing individual objects, since even these models struggle to perceive the interaction between 2 objects and the relationship between the subject and the object. With the recent success in scene-graph understanding, thanks part to datasets such as Visual Genome and Visual Question Answering (VQA), there has been decent growth in relation prediction. Taking this a step further, Action Genome dataset by Jingwei et. al [1] breaks down video events into multiple scene graphs from individual frames to better understand the activity. It provides a scaffold to study the dynamic actions as relationships between people and objects. This information can be used to perform much higher level tasks such as answering spatial and temporal questions, scene description, etc.

Figure 1 describes a toy example of the Spatio-temporal graph as a Directed graph. The nodes represent the subject, object and the time-steps. The edges represent a connection while few edges have a relational attribute. In the image above, the person interacts with 2 objects: a door and her phone. Counting seems to be a relatively easy task where the number of bounding boxes corresponding to the objects can be calculated. However in a video, counting the events that occur is

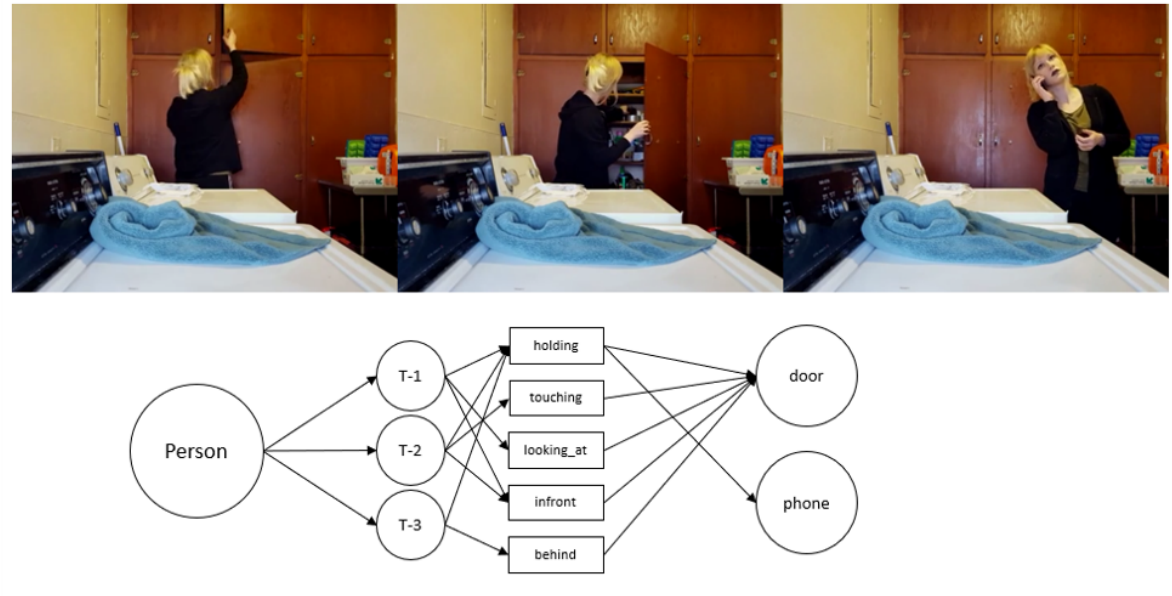


Figure 1.1: Spatio-temporal graph as a Directed graph

still a topic that has been untouched. The complex task of event localization is yet to reach the heights of object localization in an image. Hence we resort to understanding the video through a dynamic graph that can be constructed over time and to retrieve information from the representation. Counting requires the machine to be able to keep every event in memory, to estimate the count. An efficient algorithm is yet to be developed to perform such a task. This paper proposes a novel Spatio-temporal scene-graph for reasoning which makes it a task of counting the isomorphic substructures. Counting the number of such isomorphic substructures in a complex graph is an NP-complete task, luckily the graphs in question are not complex. Even though a simple rule-based technique can be used to solve this task, we will be trying to solve this using learning. However, these substructures that help with reasoning are simple acyclic paths and are easier to solve by using walks. When it comes to training a machine learning model to perform this task, it becomes even more difficult as the context for counting a specific substructure is minimal. We dig deeper to try and understand the reason behind the shortcomings of the counting task by creating another

experiment to solve a simpler sub-task. Some useful insights are gained from the experiments conducted to solve the mystery behind machine counting.

This papers contributions can be summarized as:

1. A Novel Spatio-temporal scene graph representation of an entire video.
2. A new state-of-the-art result for PredCls on Action Genome.
3. Try to solve the problem of counting graph substructures using learning.

CHAPTER 2: LITERATURE REVIEW

Action Recognition in Videos

Action Recognition is a representative task for video understanding. The challenges faced by these networks includes high computation cost, modeling long-range temporal information in videos, designing a classification architecture, etc. Furthermore, definition of a label space for training is also non-trivial because activities are usually composite concepts that comprise of multiple actions, and a hierarchy is not clearly defined. It is important to capture the intra- and the inter-class variations. Secondly, human action recognition requires simultaneous understanding of short-term action-specific information and long-range-temporal information [10].

Over time, computer vision scientists have shifted their focus from 2D to 3D supervised techniques. To facilitate this, there are a number of datasets released such as AVA, Kinetics, UCF101, ActivityNet and Charades. Initial models used dual-stream networks, where the video frames and the optical flow of the video acted as 2 different flows to the network. The assumption was that most actions can be predicted based on the human's pose. Temporal Segment Networks or TSN's used multiple segments of the video to obtain segmental consensus from different streamlines. It was understood that pre-computing optical-flow in the video is computationally expensive. With the rise of 3D CNNs, C3D was born. It was essentially a 3D version of the VGG16 network. It showed strong generalization capabilities, and yet performed poorly. In the year 2017, Cerreira et. al. [4] proposed I3D, which pushed video action recognition due to the fact that the model could be used on larger datasets such as Kinetics400, which was previously not possible. The authors of [10] point out that 3D CNNs are not replacing two-stream networks and are not mutually exclusive.

Long-range temporal modeling has been tried to achieve using multiple stacks of short tempo-

ral convolutions. However, useful temporal information is almost always lost towards the end. Recently, V4D has been proposed to model the evolution of long-range Spatio-temporal representations, which still remains a field of active study.

Object Detection

Object detection in Computer vision has found the most interest, since it can help solve numerous problems. R-CNN proposed by Girshick et. al. [5] proved highly effective in detecting and classifying objects in natural images. It is short for "Region-based Convolutional Neural Networks." The main idea is composed of 2 steps, the model initially uses selective search to identify a number of bounding-boxes which are considered the Region of Interest or ROI. Then it uses Convolutional Neural networks to extract features from the ROI to independently classify these bounding boxes. It proposed the method of Bounding box regression to predict the coordinates. Consider the ground-truth bounding box to be $g = (g_x, g_y, g_w, g_h)$ where x, y signify the coordinates of the center of the box, and w, h signify the width and the height of the box. Upon a forward pass of the image through the network, the model predicts these coordinates given by $p = (p_x, p_y, p_w, p_h)$. The regressor is configured to learn scale-invariant transformations. The transformation function is as follows.

$$\hat{g_x} = p_w d_x(p) + p_x$$

$$\hat{g_y} = p_h d_y(p) + p_y$$

$$\hat{g_w} = p_w exp(d_w(p))$$

$$\hat{g_h} = p_h exp(d_h(p))$$

Hence the targets for the model to learn become,

$$t_x = (g_x - p_x)/p_w$$

$$t_y = (g_y - p_y)/p_h$$

$$t_w = log(g_w/p_w)$$

$$t_h = log(g_h/p_h)$$

A standard regression model using SSE loss with regularization is applied

$$\mathcal{L}_{reg} = \Sigma_{i \in x,y,w,h}(t_i - d_i(p))^2 + \lambda||w||^2$$

To avoid overlapping prediction of multiple repeated detections, the paper also proposes Non-max suppression which discards boxes with low confidence scores based on high IoU (Intersection-over-union). The drawbacks of this proposal however is running selective search for over 2000 region candidates which creates a speed bottleneck.

The next iteration to the R-CNN family aimed to make the prediction closer to real-time, which was again proposed by Ross Girshick [6]. These advancements are driven by sharing the convolutions across proposals in a feature pyramid network which drastically reduced the cost. The significant jump in the performance can be attributed to ROI pooling. This model unified 3 independent models to jointly train by increasing shared computation. The model aggregates the CNN features into one forward pass which is shared by the feature matrix, instead of extracting them individually. These features are then branched out to be used by the classifier and the regressor. This paper also introduced ROI pooling, which is a type of max pooling to convert ROI features into a fixed small window. Although Fast-RCNN is an upgrade to the RCNN, the improvement is not dramatic. To overcome this bottleneck, Ren et. al. [7] proposed the Faster-RCNN network. Since then,

it has gained popularity to effectively extract semantic features from images and is used as the benchmark. The proposed Region proposal network shares the convolutional feature layers and is trained end-to-end. It utilises anchors of various sizes to propose possible region of interests, using a sliding window mechanism. Multiple regions of various scales and ratios are simultaneously predicted. These proposed ROIs pass through the Fast-RCNN framework. The final loss function also combines the losses of bounding box regression and classification as follows.

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$$

where $\mathcal{L}$ is the log loss function over two classes, i.e

$$\mathcal{L}_{cls}(p_i, p_i^*) = -p_i^* log p_i - (1 - p_i^*) log(1 - p_i)$$

$$\mathcal{L}(p_i, t_i) = \frac{1}{N_{cls}} \Sigma_i \mathcal{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \Sigma_i p_i^* \cdot L_1^{smooth}(t_i - t_i^*)$$

where N_{cls} and N_{box} is the normalization of the mini-batches, and λ is the balancing parameter. An accurate figure of the pipeline is provided in image 2.

In this work, ResNext-101 32x8d is used as the backbone for the network. This is similar to a residual block of a ResNet-101 backbone with the addition of the *next* dimension, also called the 'cardinality' dimension. This backbone was 1st runner up in the ILSVRC classification task in 2016. The added dimensions is accurately depicted in the image below. It shows 32 parallel connections whose outputs are aggregated through summation, along with a skip-connection towards the end. These number of complex transformations are controlled by the parameter C.

For each path, a standard bottleneck of a residual block is used which is $1x1conv$-$3x3conv$-$1x1conv$. The internal paths are denoted by d i.e $d = 8$ in our case. The number of paths denoted by C is set to 32. Compared to Inception-ResNet, it requires minimal effort to design each path.

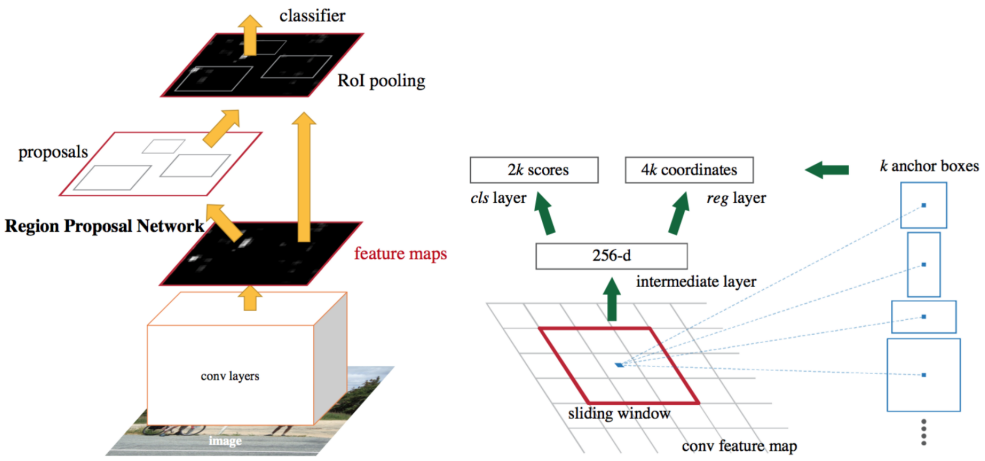


Figure 2.1: Illustration of the Faster-RCNN model

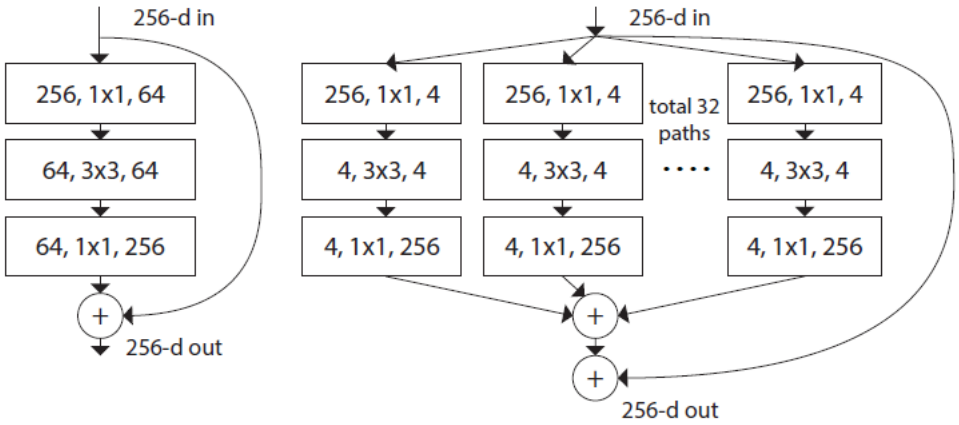


Figure 2.2: Comparing Residual block of ResNet (left) and ResNext (right).

Visual Relationships

Relation Prediction task was widely popularized by the Open Images challenge from Google. Visual Genome enables modeling of relations, to help present it in a structured manner, in terms of a scene-graph. It comprises or modeling interactions between pairs of objects in images. Since re-

lationships require 2 objects, there is a greater skew of rare relationships, as object co-occurrence is very infrequent in any given dataset. Hence, it becomes important to learn these relationships using very few instances. This task on it's own has seen multiple papers being published, year after year.

These relationships can essentially be broken down into multi-faceted categories. The spatial orientation of the object with respect to the subject becomes important to track. Whether the subject is "above", "below", "inside", "outside" can add vital information about the article and how it is being interacted with. These spatial relationships are essentially *prepositions* in the English language. Subsequently, it is key to know whether the person is actively interacting with the object, i.e if the person is either "looking at" or "not looking at" an object. This can be classified as an attentional relationship. Lastly, there are relationships that describe how the person is interacting with the object, described using *verbs*. These comprise of words such as "drinking from", "wearing", "holding", etc.

Relationships have improved object localization, the semantic space of the relationships aid to the cognitive task of mapping images to captions, and have even helped generate images from sentences to improve image search [3].

Recently relation prediction has been in the form of visual phrases, i.e models have shown to improve individual object detection. For example, the detection and localization of a "person" and a "horse" can be improved by detecting that "a person riding a horse." The drawback however is the model is bound by the label map of any given dataset.

There are various visual relationship datasets. It should capture the rich variety of interactions than just objects localized in images. As mentioned above, these interactions might be verbs (ex. hold), spatial prepositions (ex. below), actions (ex. kick) etc. There are a number of datasets that offer these, such as Visual Relationship Detection (VRD), Visual Genome (VG), Hymans Interacting with Common Objects (HICO) etc.

Knowledge Graphs

Knowledge graphs (KG) represent a collection of interlinked entities between real-world events, concepts and objects. These descriptions are in such a manner that it allows humans and machines to efficiently process them. It uses graph-structured data models to integrate topological data. Knowledge graphs have played an important role in Machine Learning since the development of Graph Convolutional Networks (GCN).

It can be a directed or undirected labeled graphs where the labels have well defined meanings. It consists of inter connected nodes and edges. Anything can act as a node. In Social media knowledge graphs, the nodes represent the users of the portal, and the edges represent the friendship. This can vary from social networking website to website. For example, an undirected graph can be used to represent Facebook data, since users are friends with each other. However, in a graph representing Twitter data, the graph would need to be a directed one, where the users choose to follow each other or not.

These graphs also offer weighted edges where connections between a few nodes can be weighted more than other connections. Each node can also contain attributes, i.e the users data like Age, Occupation, Place, etc. These attributes essentially become the node features. Formally, Given a set of nodes N, and a set of labels E, the knowledge graph is a subset of the cross-product $NxExN$. Each member of the set is referred to as a triplet. The choice of how to represent data becomes a design decision, with the pipeline looking like data to information, from information to knowledge, and from knowledge to insight.

Considering a knowledge graph is essentially a set of data points, these features can embedded and used in Machine Learning models. They are usually derivative datasets that are obtained by analyzing and filtering data. These are pre-computed data-points which means they can be analyzed much faster and at scale. This reduces the need for large, labelled datasets, facilitating explainability and transfer learning.

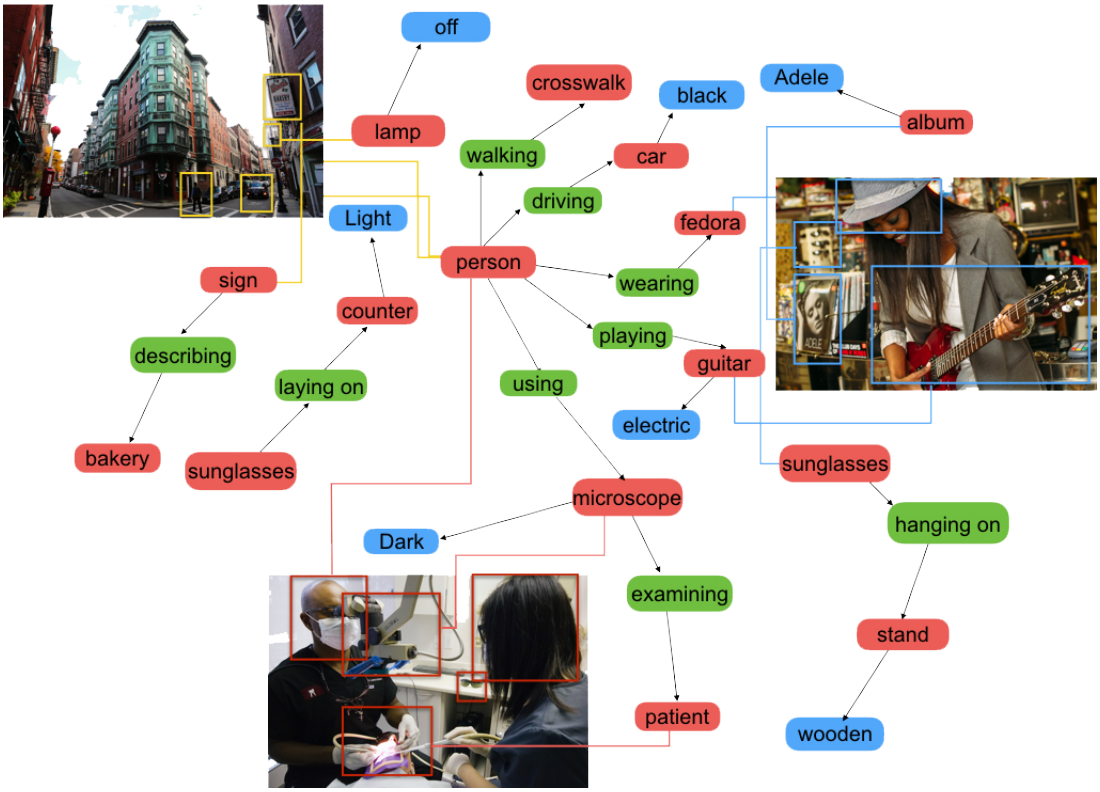


Figure 2.3: A scene-graph Knowledge graph

Scene-Graphs

Understanding a scene requires more than just recognizing individual objects in isolation. It is key to also understand the relationships between objects that constitute rich semantic information about the visual scene [11]. There is an increasing effort to understand images thoroughly, which can be facilitated by using 3 key features according to [2],

1. Grounding of visual concepts to language

2. Complete set of descriptions and QAs

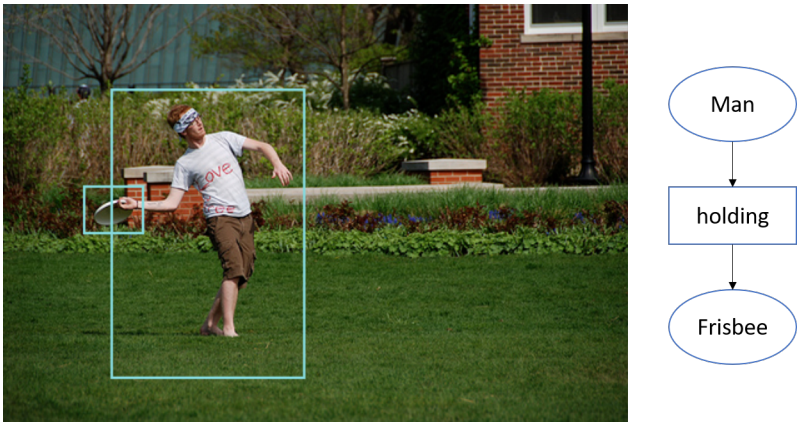


Figure 2.4: An example of a simple scene-graph

3. Formalized representation.

Focusing on the last element, an action can be formally expressed by the *subject-relationship-object* triplet. For example, in the image 2 Man is the subject, Frisbee becomes the object and holding is the relation between the Man and the Frisbee. A structured scene-graph is the union of all triplets extracted from the region descriptions. The nodes in a scene-graph correspond to subject/object bounding boxes, and the edges correspond to their pairwise relationships. Scene graph explicitly model the object and their relationships in a graphical structure.

Visual Genome dataset proposed by Krishna et. al. [2] enables modeling of such relations. The dataset provides a dense annotation of objects, attributes, and relationships within an image. It consists of over 100k images with an average of 21 objects, and 18 pairwise relationships between objects. However, in this paper, we wanted to focus on videos.

There have been numerous end-to-end models that try to predict these structured relationships from an input image taking advantage of contextual cues. Give it's versatile utilities, the papers also explore structured prediction, reinforcement learning, sequential and few-shot prediction, including

graph-based methods.

Visual Question-Answering

Reasoning has emerged as an important domain of research since it tries to bridge the gap between vision and language. Since humans can only understand language, it is vital to convey the information extracted in terms of text from an image/video.

There are numerous datasets that expects a language answer as the output with an image and a free-form natural language question as inputs. These goal-driven tasks differ vastly based on the type of questions. For example, the capabilities required could be object detection for question like ”How many vehicles are there?”, where as for questions like ”is the man eating the pizza?” the task required is activity recognition. There are commonsense reasoning questions which are essentially open ended, while knowledge based reasoning have their answers in the knowledge domain. The answers to these questions can just be of binary output, i.e yes or no, while some would require more grounded information. The questions could even elicit a numerical value or just textual based answers. Hence, VQA poses a rich set of challenges, many of which first require understanding of the question and the type of answer that needs to be extracted. The questions can be broadly classified into (a) open-ended questions or (b) Multiple Choice questions. Open-ended questions require answers to be 1-3 words, while Multiple choice questions are answered by picking the most popular answer among the choices.

VQA for video is still a relatively unexplored territory. Since the task requires understanding temporal information. Once the temporal aspect enters the scene, it becomes essential to have a long term memory to remember all the events that took place before and after the event in question. This task exponentially blows up as the length of the video increases. The semantic information in the video is (a) hard to be expressed and (b) harder to extract. The type of question elicits what

part of the video to actually look at. Whether the question can be answered by just looking at a segment of video, or it requires global context of the entire video sequence.

In this work, we try to answer counting questions. These questions are answered by numerical values. It is very easy for the problem to blow up as the number of events that take place can be any value N. This can even be formulated as a regression problem of some sort, since it requires prediction of the value N. For example, to answer the question "How many times did the person touch the cup?" the model needs to check the number of times the person interacted with the cup in the video. There have been several methods of VQA counting [9] that directly involves counting the values from the feature map of the image. The authors of this paper attribute the problem to be soft-attention in the models introduce a neural network architecture for robust counting from object proposals. This proves to be a slightly easier task of just counting the number of bounding boxes in an image. But to answer a specific question, by counting the items in the image becomes an entirely different task tackled in [9]. It becomes harder when it comes to the video domain. Where it requires counting of events that occur over a period of time.

Action Genome

The action genome dataset is a variation of charades dataset. Charades is composed of 9848 videos of daily indoor activities collected through Amazon Mechanical turk, which includes 267 different users. Charades contains 66,500 temporal annotations for 157 actions. However, Action Genome provides 476,229 object bounding boxes, with 1,715,568 relationships across 234,253 video frames [1]. The objects are annotated from frames, which are uniformly sampled across 5 frames across the activity. The Action Genome dataset was released in the year 2019.

The pairwise-relationships are annotated between the person and those objects. of 3 different categories: Attentional, Spatial and Temporal. The video is broken down into intermittent frames

Dataset	Video hours	# videos	# action categories	Objects				Relationships			
				annotated	localized	# categories	# instances	annotated	localized	# categories	# instances
ActivityNet [8]	648	28K	200			-	-			-	-
HACS Clips [86]	833	0.4K	200			-	-			-	-
Kinetics-700 [9]	1794	650K	700			-	-			-	-
AVA [26]	108	504K	80			-	-	✓		49	-
Charades [65]	82	10K	157	✓		37	-			-	-
EPIC-Kitchen [15]	55	-	125	✓		331	-			-	-
DALY [74]	31	8K	10	✓	✓	41	3.6K			-	-
CAD120++ [90]	0.57	0.5K	10	✓	✓	13	64K	✓	✓	6	32K
Action Genome	82	10K	157	✓	✓	35	**0.4M**	✓	✓	25	**1.7M**

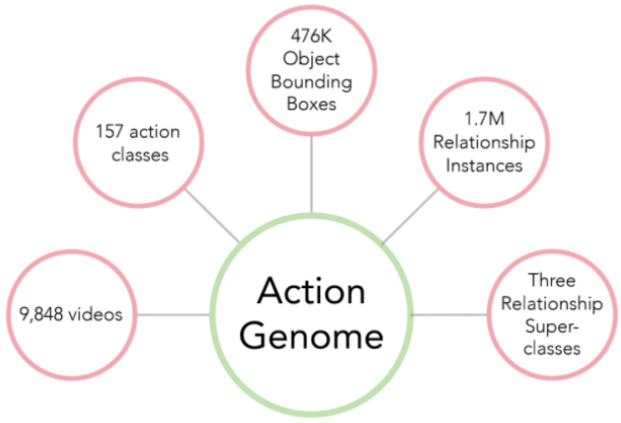


Figure 2.5: Action Genome Statistics

along with their annotations. The authors pose that this decomposition of events into prototypical action-object unit will allow us to improve action recognition. This may even help predict a behaviour based on a pattern. The representation can be viewed as a temporally changing version of Visual Genome scene-graphs, but instead of densely representing the objects in the scene, it aims to decompose actions by annotating only those segments of videos that involves an activity that can be decomposed.

The Table 2 shows the Action Genome dataset compared to some of the other video datasets. It offers annotation and localization for all classes of objects and relationships.

The dataset consists of 35 object classes and a ”person” class which is always the subject in the video. There are a total of 25 relations distributed over the 3 aforementioned categories which are given in Table 2 and the object categories are provided in Table 2.

Table 2.1: Action Genome Objects categories

Objects				
window	bag	bed	blanket	book
box	broom	chair	closet/cabinet	clothes
cup/glass/bottle	dish	door	doorknob	doorway
floor	food	groceries	laptop	light
medicine	mirror	paper/notebook	phone/camera	picture
pillow	refrigerator	sandwich	shelf	shoe
sofa/couch	table	television	towel	vacuum

Table 2.2: Action Genome Relation categories

Attention	Spatial	Contact	
looking at	in front of	carrying	covered by
not looking at	behind	drinking from	eating
unsure	on the side of	have it on the back	holding
	above	leaning on	lying on
	beneath	not contacting	sitting on
	in	standing on	touching
		twisting	wearing
		wiping	writing on

CHAPTER 3: METHODOLOGY

In this work, we tackle the question regarding activity counting. This task requires numerical answers, i.e the number of specific events that occurred in the video, based on the question. This is still an unexplored territory, and all the findings in this work are novel, and focus on tackling this problem.

The process of reasoning using these Spatio-temporal scene-graphs can be broken down into 2 parts. The first part deals with the creation of the aforementioned Spatio-temporal scene graphs. This deals with recognizing the objects in the frames of the video. Thankfully, Action Genome dataset provides us with all the annotation required to do this task.

Part 1: Relation Prediction

An object detection model such as Faster-RCNN is trained for all 36 classes in the dataset. We achieve an average recall of 21.86% for this task using a ResNext-101 32x8d backbone. Pre-trained weights from ImageNet is used, which has a total of 88M parameters with 16B FLOPS. The pre-trained model got Top-5 accuracy of 96.5% on the ImageNet 1K dataset.
The features proposed by the base module are used for future predictions. We use various algorithms for the prediction of relationships, such as

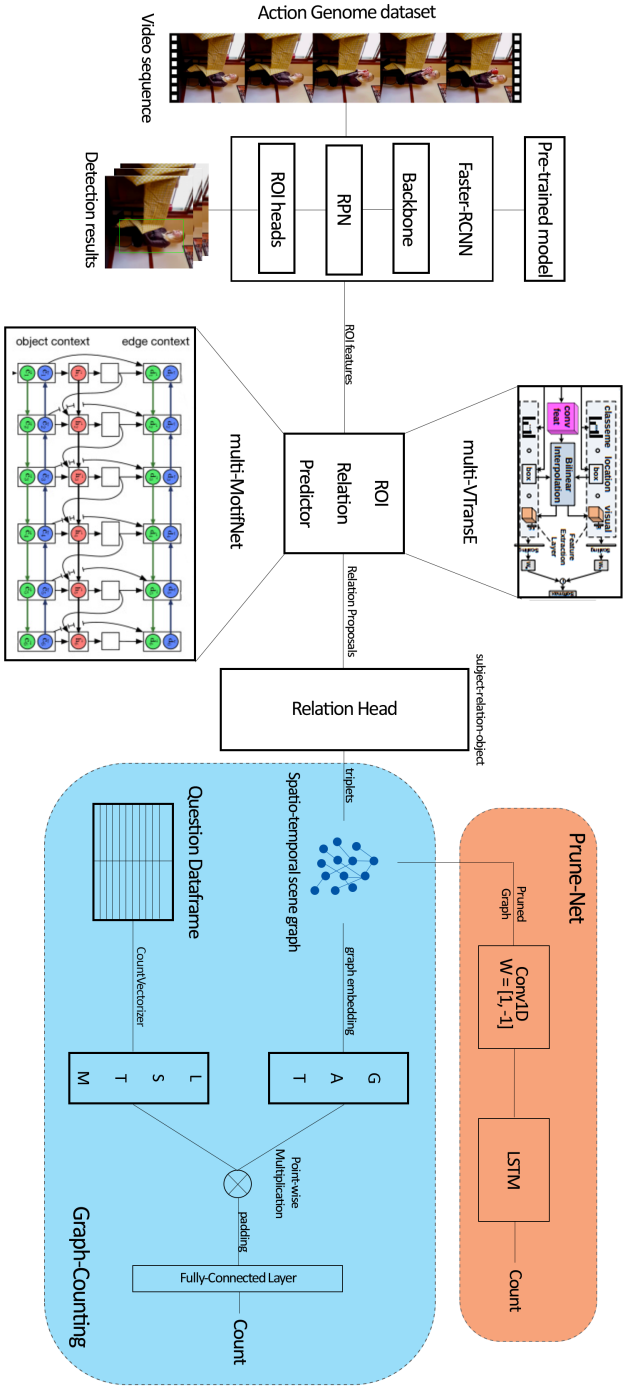


Figure 3.1: Network architecture for Spatio-temporal scene-graph prediction and reasoning

Multi-MotifNet

The algorithm uses the features from the Region Proposal Network to compute the bounding box regions. This along with the global context is used to predict labels for these bounding boxes. Given the boxes and the labels, the model creates a new edge-context representation using Bidirectional LSTMs for edge prediction.

That is, for a given image I, the object detector generates region proposals $B = b_1, b_2, ..., b_n$. For each proposal $b_i \in B$ there exists a feature vector f_i and object label probability vector $l_i \in R^{|C|}$. These elements are organized in a linear sequence $[(b_1, f_1, l_1)...(b_n, f_n, l_n)]$. The Object context C is calculated as

$$C = biLSTM([f_i; W_1 l_i]_{i=1,2,3...n})$$

where C will contains all the hidden states of all elements in linearization of B represented as $C = [c_1, c_2...c_n]$, W_1 is the weight matrix that maps the distribution of predicted classes $l1$ to R^{100}. This also allows the biLSTM to capture information about potential objects from all elements of B.

The contextualized representation C is used to sequentially decode labels for each proposed region, which is conditioned on the previous decoded labels. An LSTM is used to decode the labels given as

$$h_i = LSTM([c_i; \hat{o}_{i-1}])$$

$$\hat{o}_i = argmax(W_o h_i) \in R^{|C|}$$

here $\hat{o}_i$ is a one hot encoded object class commitments from the relation model, obtained by discarding the hidden states h_i. The objects and their bounding regions are used regions are construct contextualized representations D called the edge context using another bidirectional LSTM given

by.

$$D = biLSTM([c_i; W_2\hat{o}_i]_{i=1,2,...n})$$

The weight matrix W_2 maps $\hat{o}_i$ into R. The edge context $D = [d_1, d_2...d_n]$ contains states of each bounding region at the final layer. In a scene-graph, there are quadratic number of possible relations.

Finally, by combining contextualized head, tail, union bounding region information using their outer product the model assigns labels to these edge representations [13]. Note that since it is a multi-class output, there are 3 different heads to predict the final label for Attentional, Spatial and Contacting relationships. Therefore, an input embedding of $3XO$ is passed through the model, where O is the number of objects in the frame, we also expect an output of the same dimensions. That is given the feature vector for the union of boxes $f_{i,j}$ where i, j are 2 possible objects, and the global context D. We compute the probability distribution of the edge having a label $x_{i \to j}$.

$$g_{i,j} = (W_h d_i) \circ (W_t d_j) \circ f_{i,j}$$

$$Pr(x_{i \to j}|B, O) = softmax(W_r g_{i,j} + w_{o_i,o_j})$$

where W_h and W_t project the head and the tail context into R^{4096}. w_{o_i,o_j} is a bias vector specific to the head and tail labels.

Multi-VTransE

VTransE stands for Visual Translation Embedding network, which also takes a similar approach where every pair of objects are fed into the Relation prediction module to extract visual translation embedding and feature extraction using bilinear interpolation of the feature maps [14]. However, it cannot be considered as a relation prediction algorithm stacked on top of a Faster-RCNN module.

The authors of this paper propose a novel feature extraction layer that exploits bi-linear interpolation instead of a non-smooth RoI pooling layer.

TransE offers a simple yet effective linear model fro representing long-tail relations in large knowledge graphs. VTransE does an effective job transferring TransE into the visual domain by mapping the features of the detected objects into the relation space.

Consider $X_s, X_o \in R^M$ as the M dimensional features of subject and object in the image respectively. VTransE aims to learn 2 projection matrices $W_s, W_o \in R^{rxM}$ where $(r << M)$. Thus the visual relation can be represented as

$$W_s x_s + t_p \approx W_o x_o$$

The model samples mini-batches of 256 region proposal boxes generated by the FPN, where we consider only the IOUs of 0.7 of higher. These positive proposals are fed to non-max suppression. The final feature map F of the ResNeXt layer is used by removing the last pooling layer. F encodes the visual appearances of the entire image. The ROI pooling layer is replaced by the bi-linear interpolation later, as it smooths 2 inputs: the object bounding box projected onto F and the feature map F itself. The output V can be represented as

$$V_{i,j,c} = \Sigma_{i'=1}^{W'} \Sigma_{j'=1}^{H'} F_{i',j',c} k(i' - G_{i,j,1}) k(j' - G_{i,j,2})$$

where G records the grid split in the input bounding box and $k(x) = max(0, 1 - |x|)$ is the interpolation kernel. W, H, c represent the width, height and the channels of the image. Gradients of V can be back-propagated to the bounding box coordinates since the grid position G is a linear function of the input box. Similar to MotifNet, we use 3 heads in the final layer, to obtain all categories of relations - Attentional, Spatial and Contact. The loss function is a multi-task loss combining object detection loss $\mathcal{L}_{obj}$ and relation detection loss $\mathcal{L}_{rel}$ with a reasonable loss trade-

off of $\mathcal{L}_{obj} + 0.4\mathcal{L}_{rel}$.

Part 2: Reasoning

Once the relations have been predicted, we create the novel Spatio-Temporal scene graph for videos using these objects and their relations in the form of a Directed-Acyclic Graph. These graphs are carefully constructed in order to maximise the information in the graph. The graph can be broken down into a hierarchical structure with the node 'Person' always being at the very top. It is important to note that the person is always the subject in the Action Genome dataset. The second layer is the time-steps involved in the video. This may vary from 10 time-steps to $N/5$ which is the number of frames in the video. These time-steps are important to capture the frame in which the event actually occurred. The person node is connected to every single node of the time-step that the person appears in. These time-steps are connected to the relation nodes that can be treated as edges. There are always 3 different edges going from the time-step to the object since we always capture attentional, spatial and contacting relations in the graph. These relations are converted to nodes since it becomes an important feature for reasoning. The final layer is the objects themselves that the person is interacting with. The graph accurately depicts all the events occurring in the video, along with their spatial position w.r.t to the person.

Consider an visual example shown in Figure 3 for the video 0D5JP.mp4. Based on this graph, we are able to picture the activity the person is doing over the course of the video. Explainability also becomes easier, since we can see that the person was initially looking at the cup, and then moved on to holding it and then drinking from it. At a single time-step, the person's attention switched to the book, while still holding the cup. Then the person proceeded to touch the book simultaneously holding the cup. Then a window appeared in the scene, where the person moved on the side of the

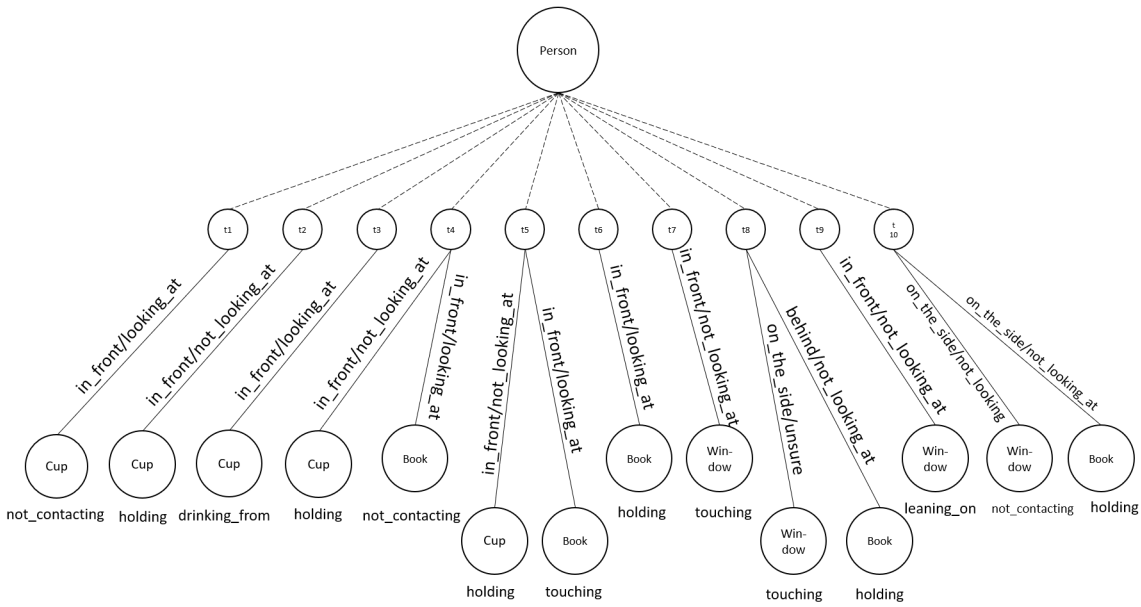


Figure 3.2: An example of a scene-graph before for the video 0D5JP.mp4

window, while holding the book. There exists so much semantic information in the graph, that it can even be substituted for the video for specific tasks.

From this graph, there are some interesting questions that can be asked, such as (1) What was the person doing before/after touching the book? (2) How did the person interact with the window that was in front of him? (3) What other object was the person interacting with while holding the cup? (4) How many times did the person touch the book? etc. Note that the upper bound of the number of questions is the number of objects in the video times the variety of contact relations plus one. All of these questions seem fairly easy to answer by looking at the graph itself. A human could easily could just track an edge to the final object node to figure out what is happening at each time-step/frame. The machine can also do the same function track back the events or the relationships from the object as well. A simple Rule-based algorithm can extract each of these answers from the graph. That is exactly what we use to create the ground-truth answers. The final answer is

the number of discontinuous time-steps. However, to create a Machine Learning model to perform these techniques proves to be a significantly harder task. The rule-based approach involves simple rules that direct the machine to what to look for in a graph. A counter is used for counting the number of events, and for a few of the questions mentioned above which require objects/relationships as the answer, the results are extracted by walking through the graph.

These questions can be broken down into multiple categories. But based on the type of answer it requires, the questions can be divided into numeric questions or text-based questions. It is pretty evident that all the questions have an answer somewhere in the graph i.e these questions are not open-ended. This helps us to a great extent, but to create a Machine learning model that is able to extract these information from the graph proved to be a difficult task. Hence, in this work we only focus on the questions that elicit numerical values as the answer that can be summarized as repetition questions. This on its own is a very interesting task to tackle since extracting the answers from the videos require counting each of these instances and saving them. The questions are created keeping in mind the repetition of these actions throughout the video. The Repetition questions seek a numerical answer. Using the Spatio-temporal scene-graph, it becomes very easy as it just requires going through the graph which is less computationally expensive.

In order to create these questions for each of the videos, we use question templates where the objects and interactions are inserted. We try to answer 4 types of questions.

1. How many times did the Person X Y?
2. How many times did the Person interact with the Y?
3. How many times did the Person look at the Y?
4. How many objects did the Person X?

Where X represents the contacting relation, and Y represents the object. We can clearly see how

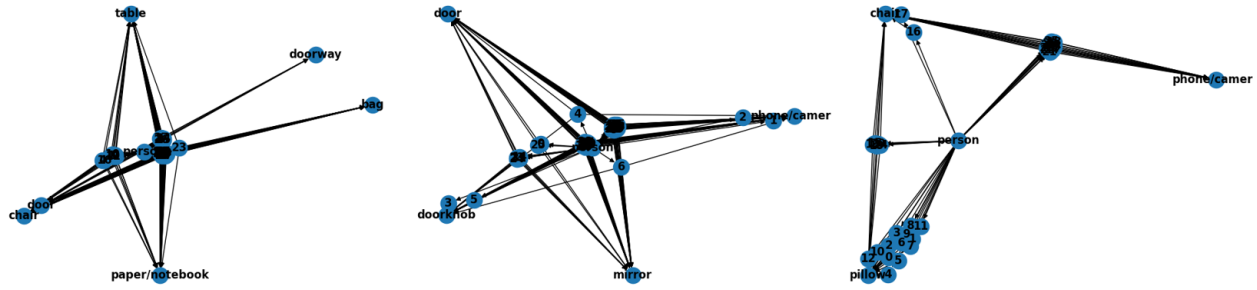


Figure 3.3: Visualizing Scene-graphs using NetworkX

we can involve more types of questions and scale the extent of reasoning from the graph.

To create the ground-truth for these questions for training, the answers are hand-crafted for these graphs by counting the simple-graphs from Person to object. We initially use a Lancaster stemmer module to stem the words in the question. Using these stemmed vectorized words, we use a Part-of-speech tagger to extract the nouns and the verbs in the question. This helps identify what the question is asking about. Based on the objects and the actions asked in the question we count the number of discontinuous paths from Peron to the object through the relationships in question. For instance, For the questions ”How many times did the person hold the cup?”, the answer requires going into the graph to and looking at the number of edges that have hold and are connected to the object Cup. It is important to note that we only consider the discontinuous time-steps as instances, because if the person is holding the cup throughout the video, the answer will only be 1 and not the number of time-steps themselves.

In this work, I use NetworkX to create these Directed acyclic graphs since it provides us with some useful features of extracting adjacency matrix, simple paths between nodes, etc. The visualization of these graphs are shown in Figure 3. Once the dataset is created, we focus on learning to predict these counts.

Graph Counting

The subsequent task is to facilitate this counting task using machine learning. It is important that the question features and the graph features are mappable, i.e they remain consistent throughout the process. The questions can be efficiently vectorized following this principle, but ensuring the same for the graph embedding proved to be a harder task, due to the temporal aspect to the scene-graph which adds more variations to the relations in each frame. The adjacency matrix, nodes along with the node features are passed through a Graph-Attentional Network [8] to obtain the graph features. Since the node features are non-existent in our network, we use simple feature representation from the adjacency matrix to formulate out node features.

Consider A the adjacency matrix of a given graph, and X as the node features. Since the features don't exist, we try out 2 different techniques. (1) The node features are generated based on the node index. But this method bases the importance of each node entirely based on the node's neighbors. (2) We use the category-to-index values of each node, i.e the label index of each of the object present in the frame while the time-steps remain the same integers. The adjacency matrix of the graph is summed with an identity matrix $\hat{A}$. We perform this in order to include it's own node features in the aggregated representation given as $\hat{A} = A * I$ where I is the Identity matrix. We proceed to normalize the feature representation by multiplying it with the inverse of the degree matrix D. The inverse of the degree matrix D^{-1} is a diagonal matrix consisting of normalized row-wise elements i.e the normalized degree of each node. Hence, the embeddings can be formalized as follows

$$G = f(X, A) = D^{-1} \cdot \hat{A} \cdot X$$

These embeddings are passed into the Graph attention network.

Graph Attention Network

Each Graph attentional unit takes in a set of node features, $h = h_1, h_2...h_N, h_i \in R^F$ where N is the number of nodes, and F is the number of features for each of the nodes. Since in our graph, the nodes signify the entity themselves, the number of features is 1 i.e the node label. This outputs another vector h' which has a higher cardinality F'. The architecture leverages a shared linear transformation layer, parameterized by $W \in R^{F'XF}$ applied to each node. Self-attention is performed on these nodes using a shared attention mechanism $a : R^{F'}XR^{F'} \rightarrow R$ computes self-attention coefficients. It is a single layer feed-forward network parameterized by $a \in R^{2F'}$ with LeakyReLU non-linearity and is given as

$$e_{i,j} = a(Wh_i, Wh_j)$$

which encapsulates the importance of node j's features to node i. $e_{i,j}$ is computed for nodes where j is in the first-order neighbourhood of i in the graph. These coefficients are normalized cross j's using the softmax function given as

$$\alpha_{i,j} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\Sigma_k exp(e_{ik})}$$

Formally, the attention mechanism can be expressed as

$$\alpha_{i,j} = \frac{exp(LeakyReLU(a^T[Wh_i||Wh_j]))}{\Sigma_k exp(LeakyReLU(a^T[Wh_i||Wh_k]))}$$

Simultaneously, the question embedding are extracted using a Inverse Document Frequency vectorizer and passed through an LSTM network. The final hidden layer of the of the LSTM is used as the question features. The graph and the text features are combined by point-wise multiplication.

Since this output depends on the number of nodes themselves, the product is padded to 256 to create a combined feature vector. The resulting features are then passed through fully-connected layers and then through a Tanh non-linear function to obtain a numerical output as the output. In this work, we only use the questions that expect an answer of ¡ 20. This can be formalized by the following equation

$$y = Tanh(W(h_i \cdot g_i) + b)$$

where h_i represents the final hidden layer of the LSTM network, and g_i represents the graph features obtained by the Graph Attention Network.

Finally, the predicted value is compared against the ground-truth to obtain the cross-entropy loss value. We jointly update the weight matrices of the network by back propagating from the loss. The idea behind using the following units in the network is simple. The hidden state of the LSTM network does a great job of extracting the important words from the question that are directly related to answering these questions. There is no known graph model that effectively counts the substructure. In a complex graph, counting the graph isomorphisms becomes an NP-complete task. The intuition behind using a Graph-attention network to capture the number of instances was the ability of such a network to focus on one part of the network that is more important than the rest. A simple convolutional operator on the adjacency matrix will not do the trick since the number of nodes in one graph to another vary drastically. The importance of one node is determined by its neighbors. CNN's also require rigid structures, while a graph simply doesn't offer that rigidity.

Prune-Net

More experiments are conducted to dig deeper into the problem of Neural counting. The Spatio-temporal scene graph is pruned to have the only one object and one relation, over multiple time-

steps. We randomly choose an object from a video and track when the person is "looking at" the object, i.e "How many times did the Person look at the $Object$?" The ground-truth answers are again extracted using a Rule-based algorithm, where the integer answer is the number of discontinuous time-steps. When the randomly picked object is not looked at in the video, our answer would be 0. However, since every other relation and object edges are pruned, the question itself becomes irrelevant, since the question is always asking about the only object that is connected to the graph.

The graph embedding is done slightly differently. The adjacency matrix of the graph is defined as A. Since we are only concerned about the number of edges going from time-steps to the relation node, we extract a single column from A corresponding to the relation node and call it G which is of the size $NX1$. This vector G is interesting because it clearly captures the transitions in the connections from time-steps to relation which directly correlates to the ground-truth answer.

The model is fairly simple, since it is only required to count the transitions in G. Since the input is of variable length depending on the number of nodes in a graph, the input G is padded. We perform a 1D convolution on the input with a standard weight matrix $W = [1, -1]$ to effectively capture the transitions from 1 to 0. Essentially the number of these transitions is the our final answer. ReLU non-linearity is applied to remove all the negative values and is then passed through a standard LSTM to count these values. This also makes the process learnable, since the weight matrix is static. We perform Softmax function on the final hidden state, to obtain the probabilities. The discussion about this experiment will further be described in the following chapter.

CHAPTER 4: RESULTS

The object detection model used to predict the objectness in each frame is described in the table below. We keep the aspect ratio of the image similar by resizing the longer edge of the image to 480, and the shorter edge of the image to 270 pixels. Hence the image will always be either (480, 270) or (270, 480). A ResNeXt-101 backbone with FPN is used to obtain the image features. The pixel mean for the dataset is calculated to be [98.047, 102.7969, 98.047] with a standard deviation of [1, 1, 1] which is used to normalize the input images. Anchor sizes of (16, 32, 64, 128, 256) are used with strides (4, 8, 16, 32, 64) with aspect ratios (0.5, 1, 2). The ROI box head uses an an MLP of dimension 1024, with a dilation of 1. The model is trained for 100,000 iterations with a base Learning rate of 0.01 and a learning rate scheduler with a decay of 0.001 at 60000 and 80000 iterations.

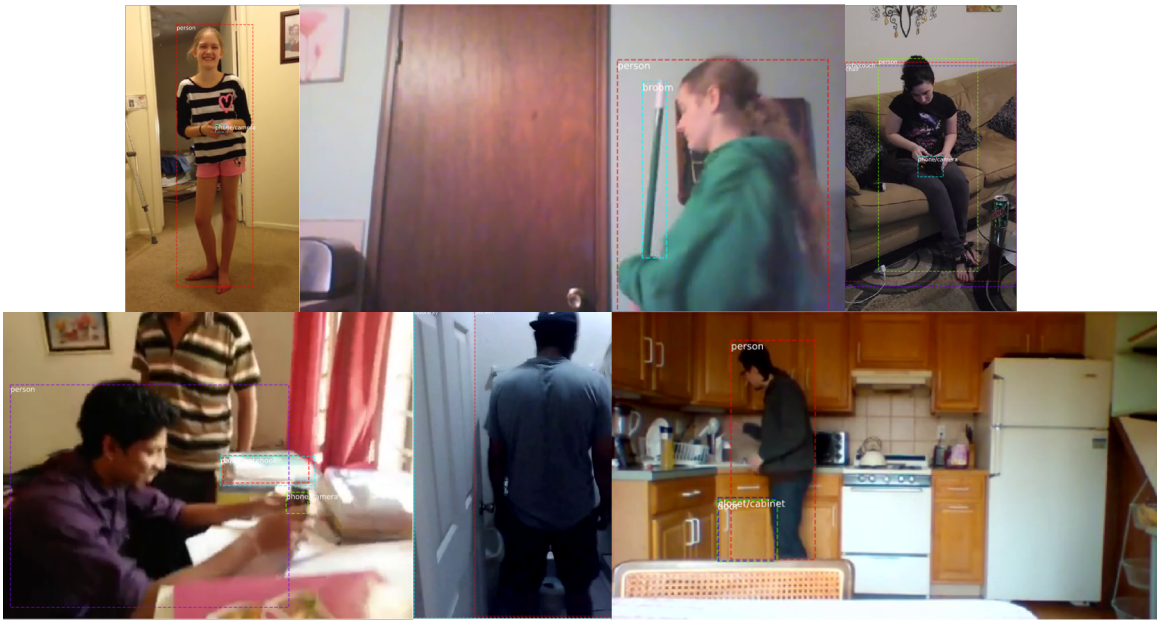


Figure 4.1: Object Detection results for a few random video frames

Table 4.1: Object Detection on Action Genome

Model	AR@100	ARm@100	ARl@100	AR@1000
ResNet-101	0.1860	0.1095	0.2414	0.2172
ResNet-101 32x8d	0.2186	0.1471	0.2629	0.2414

The results of the Object detection model are shown in Table 4.1. Compared to the ResNet-101 model, The ResNet-101 32x8d performs significantly better for a variety of metrics. This is expected as the width of the features in each residual block is higher, and comprises of more connections compared to the vanilla ResNet-101 model.

The proposed features are passed through the Multi-Label Relation algorithm to obtain the subject-relation-object triplet. The relations can be evaluated using 3 different metrics, that require 3 different training processes.

1. Predicate Classifications PredCls: Where the model takes in Ground-truth bounding boxes and labels as input
2. Scene Graph Classification SGCls: Where the model takes in bounding boxes without labels.
3. Scene Graph Detection SGDet: Detecting Scene-graphs from scratch.

For the second part of the spatio-temporal scene-graph reasoning, we use PredCls trained model, as it performs the exceptionally well, beating the state of the art results by a big margin.

The results of the relation prediction algorithm is described in Table 4.2. Since this is a multi-class label prediction, The results are aggregated w.r.t the input embedding. As we can see, MotifNet performs better than VTrans for the validation set.

Reasoning using these predicted relations uses the validation set with a total of 9513 questions for

Table 4.2: Relation Prediction on Action Genome metric: PredCls

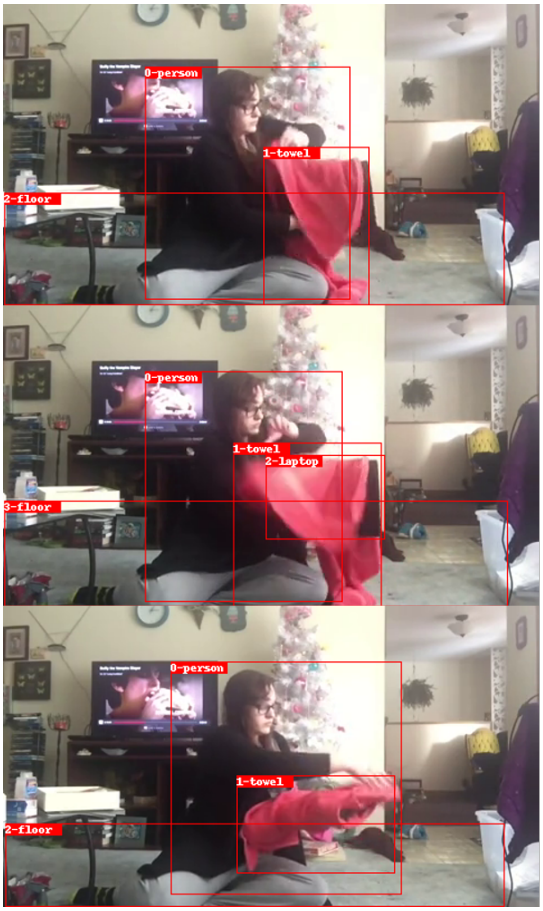
Model	Train		Val	
	AR@20	AR@50	AR@20	AR@50
VRD [3]	24.92	25.20	24.63	24.87
Graph-RCNN [15]	23.71	23.91	23.43	23.60
MSDN [16]	48.05	48.32	47.43	47.67
IMP [11]	48.20	48.48	47.58	47.83
RelDN [17]	49.37	49.58	48.80	48.98
Multi-VTrans (ours)	67.87	67.90	49.23	49.31
Multi-MotifNet (ours)	66.85	66.87	67.01	67.08

690 videos. The model uses Adam optimizer with cross entropy loss. We obtain a an accuracy of 36.20% with an rmse loss of 2.411. From the experiments conducted, it is safe to say that the model does not completely generalize the prediction of accurate number of repetitions. However, the results are interesting in the sense, that it is able to predict values that are uniformly distributed in the graph network. Most of the accurate predictions are in the range of 1-7, which means that the model struggles to count the repetitions beyond that. Hence, we can conclude that learning a graph-substructure counting and generalization requires more work to elicit the exact number of repetitions captured in the graph.

Table 4.3: QA prediction pairs for video described in Figure 4

Question	GT answer	Prediction
How many times did the Person hold the towel?	1	1
How many times did the Person interact with the laptop?	3	5
How many times did the Person look at the floor?	0	1
How many objects did the Person hold?	2	2

Based on the results seen, we identify a few areas where extensive research needs to be done to identify what is going on in the model. Firstly, the graph input to the network is in the form of an embedding vector. In our case, we utilize the node's category value to obtain the initial embeddings




```
('0-person', 'looking_at', '1-towel')
('0-person', 'in_front_of', '1-towel')
('0-person', 'holding', '1-towel')
('0-person', 'not_looking_at', '2-floor')
('0-person', 'beneath', '2-floor')
('0-person', 'sitting_on', '2-floor')
```

```
('0-person', 'not_looking_at', '1-towel')
('0-person', 'in_front_of', '1-towel')
('0-person', 'holding', '1-towel')
('0-person', 'looking_at', '2-laptop')
('0-person', 'in_front_of', '2-laptop')
('0-person', 'holding', '2-laptop')
('0-person', 'not_looking_at', '3-floor')
('0-person', 'beneath', '3-floor')
('0-person', 'sitting_on', '3-floor')
```

```
('0-person', 'looking_at', '1-towel')
('0-person', 'in_front_of', '1-towel')
('0-person', 'holding', '1-towel')
('0-person', 'not_looking_at', '2-floor')
('0-person', 'beneath', '2-floor')
('0-person', 'sitting_on', '2-floor')
```

Figure 4.2: Image triplet prediction for a few frames from the video

to create a $NX1$ embedding, where N signifies the number of nodes in the graph. Essentially, any graph network takes in a set of features to output another set of features where the network identifies a few important features in the input using attention mechanism. Hence the graph features are only as good as the node features. To overcome this issue of flawed graph features, we also tried node2vec to obtain the node features. Node2Vec is a graph traversal algorithm similar to word2vec and is a modification of Deepwalk. This algorithm traverses the graph by moving to random neighbors of each node for a certain number of steps. It samples these random walks and uses skip-gram to maximise the probability of predicting the neighboring nodes. Node2vec has a

walk bias α, and 2 parameters p and q that prioritizes breadth-first and depth-first searches. Using this method only increased the accuracy by 1.4% which is negligible considering the added computation required to perform these random walks. But this method is not feasible since node2vec (1) requires multiple random walks for a single graph which increases the time-complexity and the computation required for each graph before sending these features through the network (2) It also does not preserve the local neighborhood of each node which is vital to count the substructures.

Secondly, most graph neural networks on knowledge graphs tackle the problem of relation prediction or entity prediction. We have seen a variety of papers that solve this problem by either using a ranking based KGQA model where question is provided with options, or models that use semantic parsing. There is little to none work done in counting in a graph, simply because graph traversal does the job.

Lastly, the complexity of the task has proven to be one of the shortcomings for graph substructure counting. In a complex graph, counting the number of simple paths between 2 nodes is a P-complete task. Hence, building a learnable model for such a task without traversing through the graph is still unknown and this work comes close to solving the task.

In order to understand the reason as to why counting is a hard task for Neural models, we conduct the Prune-net experiment. The number of transitions being captured in the graph essentially captures the ground-truth answer for a single relation and a single object. There exists numerous time-step nodes that don't have any edges going to the relation nodes. Since it is only one question per video, it reduces the training corpus significantly. We obtain an accuracy of **78.07%** on the validation set, after running the model for 100 epochs which is when the loss plateaus.

This result shows that counting sub-structures in a graph is not impossible. Identifying a way to construct the graph embedding in a manner that captures these selective transitions among multiple other connections is the first step to solve the problem. This is a much harder task when the number of edges between time-steps and relations increase. Further, a separate task is to focus on the relations and objects in question, which is where attention comes in. Solving these obstacles will

provide an efficient way to reason with these graphs. Using other knowledge-graph techniques, we can further explore Spatio-temporal scene-graphs.

CHAPTER 5: CONCLUSION

In this paper, we find alternate ways that can be used to understand actions in a video. We introduce a novel Spatio-temporal scene graph that maximizes the information captured from a video in the form of a Directed Acyclic graph. As the experiments suggest, we also achieve a new state-of-the-art result for PredCls on Action Genome dataset. Using the novel graph, we tackle a new problem of graph-substructure counting. We perform multiple experiments to evaluate the performance of these models including Node2vec embedding that perform random depth-first walks through the graph. The findings suggest that developing an efficient algorithm for effectively capturing the knowledge-graph embedding while preserving the temporal aspect of these graphs is necessary. This is shown in the second Prune-Net experiment by simplifying the sub-task. It is important to track the transitions of when the action stops occurring in the video in terms of frames. Perhaps providing more context about each of the nodes and increasing the number of features per node can prove to be an effective solution. Researchers can leverage the work done in this paper to build on the neural counting by ensuring the mapping of the questions to sub-graphs by identifying a graph-embedding that is able to hold temporal transitional information along with the correct object and their relations. Since a Spatio-temporal scene-graph is essentially a knowledge graph, we can possibly even create an end-to-end solution for this task, where the edge weights can be altered using a function that also controls the node features. The novel Spatio-temporal scene-graphs can be used for other downstream tasks such as action recognition from graphs, video captioning and can even create a story of the video using the graph along with popular text generation algorithms. This work has opened new doors to view machine understanding of videos and has provided alternate research directions that require less computation for scene understanding. New tasks such as Neural Counting is also addressed, along with their shortcomings.

LIST OF REFERENCES

[1] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, Juan Carlos Niebles, *Action Genome: Actions as Composition of Spatio-temporal Scene Graphs.* Computer Vision and Pattern Recognition (CVPR), 2020.

[2] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, Fei-Fei Li *Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations.* Computer Vision and Pattern Recognition (CVPR), 2016.

[3] Cewu Lu, Ranjay Krishna, Michael Bernstein, Li Fei-Fei. *Visual Relationship Detection with Language Priors* ECCV 2016.

[4] Joao Carreira and Andrew Zisserman. Quo Vadis, *Action Recognition? A New Model and the Kinetics Dataset.* Computer Vision and Pattern Recognition (CVPR), 2017.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation* Computer Vision and Pattern Recognition (CVPR), 2013.

[6] Ross Girshick *Fast-RCNN* Computer Vision and Pattern Recognition (CVPR), 2015.

[7] Shaoqing RenKaiming HeRoss B. GirshickJ. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* EEE Transactions On Pattern Analysis And Machine Intelligence, 2015

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. *Graph Attention Networks.* International Conference on Learning Representations (ICLR), 2018

[9] Yan Zhang, Jonathon Hare, Adam Prügel-Bennett. *Learning to Count Objects in Natural Images for Visual Question Answering.* ICLR, 2018.

[10] Yi Zhu, Xinyu Li, Chunhui Liu, Mohammadreza Zolfaghari, Yuanjun Xiong, *A Comprehensive Study of Deep Video Action Recognition*. https://arxiv.org/abs/2012.06567, 2020.

[11] Xu, Danfei and Zhu, Yuke and Choy, Christopher and Fei-Fei, Li. *Scene Graph Generation by Iterative Message Passing*, Computer Vision and Pattern Recognition (CVPR), 2017.

[12] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, Hanwang Zhang *Unbiased Scene Graph Generation from Biased Training* Computer Vision and Pattern Recognition (CVPR), 2020

[13] Rowan Zellers, Mark Yatskar, Sam Thomson, Yejin Choi, *Neural Motifs: Scene Graph Parsing with Global Context* Computer Vision and Pattern Recognition (CVPR), 2018

[14] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua. *Visual translation embedding network for visual relation detection.* Computer Vision and Pattern Recognition (CVPR), 2017

[15] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. *Graph r-cnn for scene graph generation.* arXiv preprint arXiv:1808.00191, 2018

[16] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. *Scene graph generation from objects, phrases and region captions.* Computer Vision and Pattern Recognition (CVPR), 2017

[17] Ji Zhang, Kevin J Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. *Graphical contrastive losses for scene graph parsing* Computer Vision and Pattern Recognition (CVPR), 2019