Discovering Dynamic Salient Regions for Spatio-Temporal Graph Neural Networks

Iulia Duta*
Bitdefender, Romania
id366@cam.ac.uk

Andrei Nicolicioiu*
Bitdefender, Romania
anicolicioiu@bitdefender.com

Marius Leordeanu
Bitdefender, Romania
Institute of Mathematics of the Romanian Academy
University "Politehnica" of Bucharest
marius.leordeanu@imar.ro

Abstract

Graph Neural Networks are perfectly suited to capture latent interactions between various entities in the spatio-temporal domain (e.g. videos). However, when an explicit structure is not available, it is not obvious what atomic elements should be represented as nodes. Current works generally use pre-trained object detectors or fixed, predefined regions to extract graph nodes. Improving upon this, our proposed model learns nodes that dynamically attach to well-delimited salient regions, which are relevant for a higher-level task, without using any object-level supervision. Constructing these localized, adaptive nodes gives our model inductive bias towards object-centric representations and we show that it discovers regions that are well correlated with objects in the video. In extensive ablation studies and experiments on two challenging datasets, we show superior performance to previous graph neural networks models for video classification.

1 Introduction

Spatio-temporal data, and videos, in particular, are characterised by an abundance of events that require complex reasoning to be understood. In such data, entities or classes exist at multiple scales and in different contexts in space and time, starting from lower-level physical objects, which are well localized in space and moving towards higher-level concepts which define complex interactions. We need a representation that captures such spatio-temporal interactions at different level of granularity, depending on the current scene and the requirements of the task. Classical convolutional nets address spatio-temporal processing in a simple and rigid manner, determined only by fixed local receptive fields [1]. Alternatively, space-time graph neural nets [2, 3] offer a more powerful and flexible approach modeling complex short and long-range interactions between visual entities.

In this paper, we propose a novel method to enhance vision Graph Neural Networks (GNNs) by an additional capability, missing from any other previous works. That is, to have nodes that are constructed for spatial reasoning and can adapt to the current input. Prior works are limited to having either nodes attached to semantic attention maps [4] or attached to fixed locations such as grids [5, 3, 6]. Moreover, unlike works that require external object detectors [7] our method relies on a learnable mechanism to adapt to the current input.

*Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

We propose a method that learns to discover salient regions, well-delimited in space and time, that are useful for modeling interactions between various entities. Such entities could be single objects, parts or groups of objects that perform together a simple action. Each node learns to associate by itself to such salient regions, thus the message passing between nodes is able to model object interactions more effectively. For humans, representing objects is a core knowledge system [8] and to emphasize them in our model, we predict salient regions [9] that give a strong inductive bias towards modeling them.

Our method, Dynamic Salient Regions Graph Neural Network (**DyReg-GNN**) improves the relational processing of videos by learning to discover salient regions that are relevant for the current scene and task. Note that the model learns to predict regions only from the weak supervision given by the high-level video classification loss, without supervision at the region level. Our experiments convincingly show that the regions discovered are well correlated with the objects present in the video, confirming the intuition that action recognition should be strongly related to salient region discovery. The capacity to discover such regions makes DyReg-GNN an excellent candidate model for tackling tasks requiring spatio-temporal reasoning.

Our main contributions are summarised as follow:

1. We propose a novel method to **augment spatio-temporal GNNs** by an additional capability: that of learning to create localized nodes suited for spatial reasoning, that adapt to the input.
2. The salient regions discovery **enhance the relational processing** for high-level video classification tasks: creating GNN nodes from predicted regions obtains superior performance compared to both using pre-trained object detectors or fixed regions
3. Our model leads to **unsupervised salient regions discovery**, a novelty in the realm of GNNs: it predicts such regions in videos, with only weak supervision at the video class level. We show that regions discovered are well correlated with actual physical object instances.

2 Related work

Graph Neural Networks in Vision. GNNs have been recently used in many domains where the data has a non-uniform structure [10, 11, 12, 13]. In vision tasks, it is important to model the relations between different entities appearing in the scene [14, 15] and GNNs have strong inductive biases towards relations [16, 17], thus they are perfectly suited for modeling interactions between visual instances. Since an explicit structure is not available in the video, it is of critical importance to establish what atomic elements should be represented as graph nodes. As our main contribution revolves around the creation of nodes, we analyse other recent GNN methods regarding the type of information that each node represents, and group them into two categories, *semantic* and *spatial*.

The approaches of [4, 18, 19, 20, 21, 22] capture the purely *semantic* interactions by reasoning over global graph nodes, each one receiving information from all the points in the input, regardless of spatio-temporal position. In [4] the nodes assignments are predicted from the input, while in [18] the associations between input and nodes are made by a soft clusterization. The work of [22] discovers different representation groups by using an iterative clusterization based on self-attention similarity.

The downside of these semantic approaches is that individual instances, especially those belonging to the same category, are not distinguished in the graph processing. This information is essential in tasks such as capturing human-object interactions, instance segmentation or tracking.

Alternatively, multiple methods, including ours, favour modeling instance interactions by defining *spatial* nodes associated with certain locations. We distinguish between them by how they extract the nodes from spatial location: as fixed regions or points [23, 24], or detected object boxes [25, 26, 27, 28, 29]. The method [5] creates nodes from every point in 2D convolutional features maps, while Non-Local [30] uses self-attention [31] between all spatio-temporal positions to capture distant interactions. Further, [3] extract nodes from larger fixed regions at different scales and processes them recurrently. Recent methods based on Transformer [32, 6, 33] also model the interactions between fixed locations on a grid using self-attention. In [7], nodes are created from object boxes extracted by an external detector and are processed using two different graph structures, one given by location and one given by nodes similarity. A related approach is used in [27] in a streaming setting while [34] learns to hop over unnecessary frames. Hybrid approaches use nodes corresponding to points and object features [35, 36] or propagate over both semantic and spatial nodes [37, 38, 39].

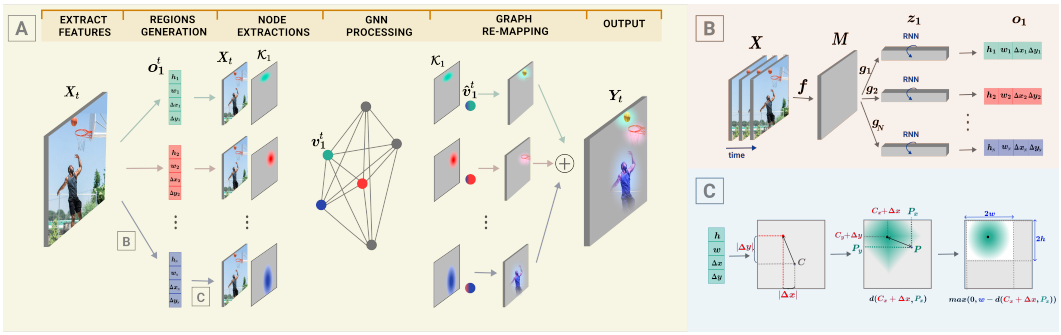

Figure 1: (**Left**) DyReg-GNN extracts localized node useful for relational processing of videos. For each node i, from the features X_t, we predict params $\mathbf{o}_i$ denoting the location and size of a region. They define a kernel K_i, used to extract the localized features $\mathbf{v}_i$ from the corresponding region of X_t. We process the nodes with a spatio-temporal GNN and project each node $\hat{\mathbf{v}}_i$ into its initial location. **(Right)** B) Node Region Generation: Functions f and $\{g_i\}$ generate the regions params $\mathbf{o}_i$; f extracts a latent representation shared between nodes, while each g_i has different params for each node i. C) Node Features Extraction: Each $\mathbf{o}_i$ creates a kernel that is used in a differentiable pooling w.r.t. $\mathbf{o}_i$. This allows us to optimize the generation of these regions' params from the final classification loss, resulting in an unsupervised discovery of salient regions.

However, methods that rely on external modules trained on additional data, such as object detectors, are too dependent on the module's performance. They are unable to adapt to the current problem, being limited to the set of pre-defined annotations designed for another task. Differently, our module is optimized to discover regions useful for the current task, using only the video classification signal.

Recently, the method [40] uses multiple position-aware nodes that take into account the spatial structure. This makes it more suitable for capturing instances, but the nodes have associated a static learned location, where each one is biased towards a specific position regardless of the input. On the other hand, we dynamically assign a location for each node, based on the input, making the method more flexible to adapt to new scenes.

Dynamic Networks. Several works use second-order computations by dynamically predicting different parts of their model from the input, instead of directly optimising parameters. Our work is related to STN [41] that aggregates features by interpolating from an area given by a predicted global transformation and to the differentiable pooling used in some object detectors [42, 43, 44]. The method [45], replaces the parameters in a standard convolution with weights predicted from the input, resulting in a dynamically generated filter. Deformable convolutions [46, 47] predict, based on the input, an offset for each position in the convolutional kernel. Similar, [48] use the same idea of predicting offsets but in a graph formulation. The common topic of these methods is to predict dynamically a support for all points in a convolutional operation while we dynamically generate the input for a set of nodes designed to process high-level interactions. Related ideas, involving high-level processing of a small set of powerful modules, is also highlighted in [49] and [40].

Unsupervised Object Representations. There is an entire area of work devoted to extracting representations centered on objects [50] in a fully unsupervised setting [51, 52, 53, 22]. They are successful in leveraging a reconstruction task to decompose the scene into objects, for synthetic images. In [54] it is shown that representations learned from unsupervised decomposition are also helpful in relational reasoning tasks. Methods for generating unsupervised keypoints or entities [55, 56, 57, 58] have been generally used in synthetic setting. The method [55] generates keypoints from real images of people and faces but they use an image reconstruction objective that could not be aligned with the downsteam task. Our goal is to relate spatio-temporal entities, but without enforcing a clear decomposition of the scene into objects. This allows us to use a simpler but effective method that learns from classification supervision of real-world videos and obtain representations that are correlated to objects.

Activity Recognition. Video classification has been influenced by methods designed for 2D images [59, 60, 61, 62]. More powerful 3D convolutional networks have been later proposed [63], while

other methods factorise the 3D convolutions [64, 65, 66] bringing both computational speed and accuracy. Methods like TSM [67] and [68] showed that a simple shift in the convolutional features results in improved accuracy at a low computational budget.

3 Dynamic Salient Regions GNNs

We investigate how to create node representations that are useful for modeling visual interactions between various entities in space and time using GNNs. Our proposed Dynamic Salient Regions GNN model (DyReg-GNN) learns to dynamically assign each node to a certain interesting region. By dynamic, we mean that we have a fixed number N of regions that change their position and size according to the input at each time step. The regions assigned to each of the N nodes can change from one moment of time to the next depending on their saliency.

The main architecture of our DyReg-GNN model is illustrated in Figure 1. Our model receives feature volume $X \in \mathbb{R}^{T\times H\times W\times C}$ and at each time step t we predict the location and size of N regions. From these regions, a differentiable pooling operation creates graph nodes that are processed by a GNN and then are projected to their initial position. This module can be inserted at any intermediate level in a standard convolutional model.

3.1 Node Region Generation

We want to attend only to a few most relevant entities in the scene, thus a small number of nodes are used in DyReg-GNN (in our experiments $N = 9$) and it is crucial to assign them to the most salient regions. The number of nodes is a hyperparameter that we choose such that it exceeds the expected number of relevant entities in the scene, to increase the robustness of the model. Thus, we propose a global processing (shown in Figure 1 B) that aggregates the entire input features to produce regions defined by parameters indicating their location $(\Delta x, \Delta y)$ and size (w, h).

To generate N salient regions, we process the input X_t using position-aware functions f and $\{g_i\}_{i\in\overline{1,N}}$ that retain spatial information. Nodes should be consistent across time, thus we generate their regions in the same way at all time steps, by sharing in time the parameters of f and $\{g_i\}$. The function f is a convolutional network that highlights the important regions from the input.

$$M_t = f(X_t) \in \mathbb{R}^{H'\times W'\times C'} \tag{1}$$

For each node i, we generate a latent representation of its associated region using the $\{g_i\}$ functions. Each g_i has the same architecture, but different parameters for each node and could be instantiated as a fully connected network or as global pooling enriched with spatial positional information. We generate the node regions from a global view to make the decision as informed as possible.

$$\hat{\mathbf{m}}_{i,t} = g_i(M_t) \in \mathbb{R}^{C'}, \forall i \in \overline{1, N} \tag{2}$$

Each of the N latent representations is processed independently, with a GRU [69] recurrent network (shared between nodes), to take into account the past regions' representations.

$$\mathbf{z}_{i,t} = \text{GRU}(\mathbf{z}_{i,t-1}, \hat{\mathbf{m}}_{i,t}) \in \mathbb{R}^{C'}, \forall i \in \overline{1, N} \tag{3}$$

At each time step, the final parameters are obtained by a linear projection $W_o \in \mathbb{R}^{C'\times 4}$, transformed by a function α to control the initialisation of the position and size (e.g. regions would start at reference points either in the center of the frame or arranged on a grid). For more details about how to set the transformation α we refer to the Supplemental Materials.

$$\mathbf{o}_{i,t} = (\Delta x_{i,t}, \Delta y_{i,t}, w_{i,t}, h_{i,t}) = \alpha(W_o \mathbf{z}_{i,t}) \in \mathbb{R}^4 \tag{4}$$

3.2 Node Features Extraction

The following operations are applied independently at each time step thus, in the current subsection, we ignore the time index for clarity. We extract the features corresponding to each region i using a differentiable pooling w.r.t. the predicted region parameters $\mathbf{o}_i$. All the input spatial locations $p \in \mathbb{R}^2$ are interpolated according to the kernel function $\mathcal{K}^{(i)}(p)$ as presented in Figure 1 C.

We present the operation for a single axis since the kernel is separable, acting in the same way on both axes:

$$\mathcal{K}^{(i)}(p_x, p_y) = k_x^{(i)}(p_x) k_y^{(i)}(p_y) \in \mathbb{R} \tag{5}$$

We define the center of the estimated region $c_{i,x} + \Delta x_i$, where $c_{i,x}$ is a fixed reference point for node i (located in the frame's center or arranged on a grid). The values of the kernel decrease with the distance to the center and is non-zero up to a maximal distance of w_i, where w_i and Δx_i are the predicted parameters from Eq. 4.

$$k_x^{(i)}(p_x) = \max(0, w_i - |c_{i,x} + \Delta x_i - p_x|) \tag{6}$$

For each time step t, node i is created by interpolating all points in the input X_t using the kernel function. By modifying $(\Delta x_i, \Delta y_i)$ the network controls the location of the regions, while (h_i, w_i) parameters indicate their size.

$$\mathbf{v}_i = \sum_{p_x=1}^{W} \sum_{p_y=1}^{H} \mathcal{K}^{(i)}(p_x, p_y) \mathbf{x}_{p_x, p_y} \in \mathbb{R}^C \tag{7}$$

Setting $w_i = 1$ leads to standard bilinear interpolation, but optimising it allows the model to adapt region's size and we observe that larger ones result in a more stable optimisation (see node size ablations from Supp. Material).

The position of the region associated with each node should be taken into account. It helps the relational processing by providing an identity for the node and is also useful in tasks that require positional information. We achieve this by computing a positional embedding for each node i using a linear projection of the kernel $\mathcal{K}_i$ into the same space as the feature vector v_i and summing them.

Key Properties. By construction, the nodes in our method are *localized*, meaning that they are clearly associated with a location: they pool information from clearly delimited area in space and they maintain position information from the positional embedding. These two aspects could be helpful in tasks involving spatio-temporal reasoning.

The *dynamic* aspect refers to the key capability of adapting the region's position and size according to the saliency of the input at each time step. This is done by predicting the regions from the input with the operations from equations (1–4).

An essential aspect of this method is that the final classification loss is *differentiable* with respect to regions' parameters as the gradients are passing from the nodes outputs v_i through the kernels k_i to the parameters w_i and Δx_i. This allows us to learn regions from the final loss, *without direct supervision for the region generation*. Thus the method has more flexibility in learning relevant regions as appropriate for the task.

3.3 Graph Processing

For processing the nodes' features, different spatio-temporal GNNs could be used. Generally, they follow a framework [12] of sending messages between connected nodes, aggregating [70, 71] and updating them.

The specific message-passing mechanism is not the focus of the current work, thus we follow a general formulation similar to [3] for recurrent spatio-temporal graph processing. It uses two different stages: one happening between all the nodes at a single time step and the other one updating each node across time. For each time step t, we send messages between each pair of two nodes, computed as an MLP (with shared parameters) and aggregates them using a dot product attention $a(v_i, v_j) \in \mathbb{R}$.

$$\mathbf{v}_{i,t} = \sum_{j=1}^{N} a(\mathbf{v}_{j,t}, \mathbf{v}_{i,t}) \text{MLP}([\mathbf{v}_{j,t}; \mathbf{v}_{i,t}]) \in \mathbb{R}^C \tag{8}$$

We incorporate temporal information through a shared recurrent function across time, applied independently for each node.

$$\hat{\mathbf{v}}_{i,t+1} = \text{GRU}(\hat{\mathbf{v}}_{i,t}, \mathbf{v}_{i,t}) \in \mathbb{R}^C \tag{9}$$

The GRU output represents the updated nodes' features and the two steps are repeated $K = 3$ times.

Table 1: Results on val. set of Smt-Smt-V2 showing the **importance of salient regions discovery.** We compare our predicted (unsupervised) regions to fixed grid regions or boxes given by an object detector using the same GNN model. The mean L_2 distance between the regions and gt. objects proves that DyReG-GNN has regions correlated with objects, while also having superior accuracy and efficiency.

Model	Regions discovery	FLOPS ↓	Dist ↓	Acc (%)↑
TSM-R50	-	65.8G	-	63.4
+ GNN+Fixed	Grid	+1.4G	0.170	64.1
+ GNN+Detector	Obj detector	+41.1G	0.125	64.0
+ DyReg-GNN	Unsupervised	+1.6G	0.129	**64.8**

Table 2: **Consistent improvements over different backbones** on the validation set of Smt-Smt-V1 using central crop evaluation.

Model	Acc (%)
TSM-R18	33.7
TSM-R18 + DyReg-GNN	35.6 (↑ 1.9)
I3D-R50	44.0
I3D-R50 + DyReg-GNN	45.4 (↑ 1.4)
TSM-R50	47.2
TSM-R50 + DyReg-GNN	48.8 (↑ 1.6)

3.4 Graph Re-Mapping

To use our method as a module inside any backbone, we produce an output with the same shape as the convolutional input $X_t \in \mathbb{R}^{H \times W \times C}$. The resulting features of each node are sent to all locations in the input according to the weights used in the initial pooling from Section 3.2.

$$\mathbf{y}_{p_x,p_y,t} = \sum_{i=1}^{N} \mathcal{K}_t^{(i)}(p_x, p_y)\hat{\mathbf{v}}_{i,t} \in \mathbb{R}^C \tag{10}$$

4 Experimental Analysis

While much effort is put into the creation of different video datasets used in the literature, such as Kinetics [63] or Charades [72], it has been argued [73] that they contain biases that make them solvable without complex spatio-temporal reasoning. CATER [73] is proposed to alleviate this, but it is too small (5500 videos) and still has biases that make the last few frames sufficient for good performance [34]. We test our model on two video classification datasets that seem to offer the best advantages, being large enough and requiring abilities to model complex interactions. We evaluate on real-world datasets, Something-Something-V1&V2 [74], while we also test on a variant of the SyncMNIST [3] dataset that is challenging and requires spatio-temporal reasoning, while allowing fast experimentation. The code for our method can be found in our repository [2].

4.1 Human-Object Interactions Experiments

Something-Something-V1&V2 [74] datasets classify scenes involving human-object complex interactions. They consist of 86K / 169K training videos and 11K / 25K validation videos, having 174 classes. Unless otherwise specified, all experiments on Something-Something datasets use TSM-ResNet-50 [67] as a backbone and we add instances of our module at multiple stages.

Studying the Importance of Salient Regions Discovery. We test the importance of the dynamic regions for GNNs vision methods by training models where we replace the predicted regions with the same number of fixed regions on a grid (GNN + Fixed Regions) or boxes (GNN + Detector) as given by a Faster R-CNN [75] trained on MSCOCO [76].

The detector based model has comparable results to the one with fixed regions, seemingly being unable to fully benefit from the correctly identified objects. The relative weaker performance of this model could be due to the fact that the pre-trained detector is not well aligned to the actual salient regions that are relevant for the classification problem.

On the other hand, this weakness is not applicable for DyReg-GNN that learns suitable regions for the current task and it obtains the best performance as seen in Table 1. Not only that it does not require object annotations, but it is also more computationally efficient. Running the detector on a video of

[2] `https://github.com/bit-ml/DyReg-GNN`

Table 3: **Results on val. set of Smt-Smt-V1.** Our model achieves competitive results compared to recent works (best results in red), while it outperforms all other graph-based methods (best results in blue). All the methods use ResNet50 as backbone.

	Model	Regions discovery	#F	Top 1	Top 5
non-Graph	TSM [67]	-	16	48.4	78.1
	S3D [64]	-	64	48.2	78.7
	GST [78]	-	16	48.6	77.9
	SmallBig [79]	-	16	50.0	79.8
	STM [80]	-	16	50.7	80.4
	MSNet [81]	-	16	**52.1**	**82.3**
Graph	ORN [14]	Objects	8	36.0	-
	NL I3D [7]	Grid	32	44.4	76.0
	NL GCN [7]	Objects	32	46.1	76.8
	TRG [82]	Frames	16	48.1	**80.4**
	RSTG [3]	Grid	32	49.2	78.8
	TSM+DyReg	Dynamic	16	**49.9**	79.0

Table 4: **Results on val. set of Smt-Smt-V2.**, in comparisons to recent works. DyReg-GNN improves the TSM-ResNet50 backbone when using either one (r4) or three (r3-4-5) modules of graph processing and it obtains top results.

Model	BB	Top 1	Top 5
TRG [82]	R50	59.8	87.4
GST [78]	R50	62.6	87.9
v-DP [83]	D121	62.9	88.0
SmallBig [79]	R50	63.8	88.9
STM [80]	R50	64.2	**89.8**
MSNet [81]	R50	64.7	89.4
TSM [67]	R50	63.4	88.5
TSM+DyReg-r4	R50	64.3	88.9
TSM+DyReg-r3-4-5	R50	**64.8**	89.4

size 224×224 would add 39.7 GFLOPS on its own, comparing to the 1.6G of three DyReg-GNN modules, from which 0.2G represents the regions prediction.

Overall, our method, with unsupervised regions obtains superior performance in terms of accuracy and computational efficiency representing a suitable choice for relational processing of a video.

Object-centric representations. The nodes represent the core processing units and their localization enforces a clear decision on what specific regions to focus on while completely ignoring the rest, as a form of hard attention. Different from other works [77], our hard attention formulation is differentiable. To better understand what elements influence the model predictions, we could inspect the predicted kernels, thus introducing another layer of interpretability to the model, on top of the capabilities offered by the convolutional backbone. Visualisations of our nodes' regions reveal that generally, they cover the objects in the scene. For example, in the first row of Figure 3 the nodes are placed around the phone in the first frames and then separate into two groups, one for the phone one for the hand.

The localized nodes make our model capable of discovering salient regions, leading to object-centric node representations. We quantify this capacity by measuring the mean L_2 distance (normalised to the size of the input) between the predicted regions and ground-truth (gt.) objects given by [28]. The metric is completely defined in the Supp. Materials. We observed that the score improves during the learning process (it reaches 0.129 starting from 0.201), although the model is not optimized for this task. This suggests that the model actually learns object-centric representations.

In Table 1 we also compare the final L_2 distance of our best DyReg-GNN model to an object detector and to fixed grid regions. Although our method is not designed and supervised to find object regions, we observe that it is able to predict locations that are fairly close to gt. objects. The L_2 distance is similar to the one obtained by an external model (0.129 vs 0.125), trained especially for detecting objects.

We observe that learning the regions' size is important for the stability of the optimisation and thus for the final performance (see Tab.5 and Supp. Material - Regions' Size section). However, the predicted size is not as well aligned with the size of the true objects. This gives us a hint that for the action classification task it is important to have good region locations, but their size is less relevant. We leave a more thoroughly investigation for futures work.

These experiments prove that the high-level classification task is well inter-related with the discovery of salient regions and that, in turn, these regions improve the relational processing in the recognition task. First, we show that DyReg-GNN's region obtain superior accuracy and efficiency than other methods of extracting nodes and second, these regions are well correlated to gt. object locations.

Comparison to recent methods. DyReg-GNN can be used with any convolutional model and we show that it consistently boosts the performance of multiple backbones(Table 2). We compare to recent methods from the literature in Table 3 and Table 4. Our method improves the accuracy over the

TSM-ResNet50 backbone on both Smt-Smt-V1 and Smt-Smt-V2 by 1.5% and 1.4% respectively and achieves competitive results. Compared to all the other graph based methods we obtain superior results, showing that our discovery of dynamic regions is effective for space-time relational processing.

Implementation Details Unless otherwise specified, we use TSM-ResNet50 (pre-trained on ImageNet [84]) as our backbone and add instances of our module in the last three stages. To benefit from ImageNet pre-training, we add our graph module as a residual connection. We noticed that models using multiple graphs have problems learning to adapt the regions from certain layers. We fix this by training models containing a single graph at each single considered stage, as the optimisation process is smoother for a single module, and distill their learned offsets into the bigger model. The distillation is done for the first 10% of the training iterations to kick-start the optimization process and then continue the learning process using only the video classification signal.

In all experiments we follow the training setting of [67], using 16 frames resized to have the shorter side of size 256, and randomly sample a crop of size 224×224. For the evaluations, we follow the setting in [67] of taking 3 spatial crops of size 256×256 with 2 temporal samplings and averaging their results. For training, we use SGD optimizer with learning rate 0.001 and momentum 0.9, using a total batch-size of 10, trained on two GPUs. We decrease the learning rate by a factor of 10 three times when the optimisation reaches a plateau.

4.2 Synthetic Experiments

SyncMNIST is a synthetic dataset involving digits that move on a black background, some in a random manner, while some move synchronously. The task is to identify the digits that move in the same way. We use a harder variant of the dataset (MultiSyncMNIST), where the videos could include multiple digits of the same class. The challenge consists in finding useful entities and model their relationships while being able to distinguish between instances of the same class. Each video contain 5 digits and the goal is to find the smallest and the largest digit class among the subset that moves in the same way. This results in a video classification task with 56 classes. The dataset contains 600k training videos and 10k validation videos with 10 frames each.

Studying the Importance of Dynamic Nodes. We validate our assumption that the nodes should be dynamic, meaning that their regions position and size should be adapted according to the input at each time step. We investigate (Table. 5) different types of localized nodes, each adapting to the input to a varying degree, and show the benefits of our design choices. We experiment with variants of our model, all having the same backbone (2D ResNet-18 [85]), the same graph processing and same pre-determined number of regions, but we constrain the node regions in different ways.

Fixed Model extracts node features from regions arranged on a grid, with a fix location and size.

Static Model investigates the importance of dynamic regions by optimising regions based on the whole dataset but do not take into account the current input. Effectively, the features $\mathbf{z}_i$ from Eq. 4 become learnable parameters.

Constant-Time Model has regions adapted to the current video but they do not change in time.

DyReg-GNN Model predicts regions defined by location and size, and we can either pre-determine a fixed size for all the regions (*Position-Only Model*) or directly predict it from the input as in our complete model (*DyReg-GNN Model*).

These experiments (Table 5), show that the fixed region approach (Fixed Model) achieves the worst results, slightly improving when the regions are allowed to change according to the learned statistics of the dataset (Static model). Adapting to the input is shown to be beneficial, the performance improving even when the regions are invariant in time (Constant-Time Model), and further more when predicting different regions at every time steps (Position-Only). The best performance is achieved when both the location and the size of the regions are dynamically predicted from the input (DyReg-GNN).

In Figure 2 we show examples of the kernels obtained for each of these models. We observe that the Static Model's kernels are learned to be arranged uniformly on a grid, to cover all possible movements in the scene, while the Constant-Time Model's kernels are adapted for each video such that they cover the main area where the digits move in the current video. The full DyReg-GNN Model learns to reduce the size of its regions and we observe that they closely follow the movement of the digits.

The previous experiments show that performance increases when the model becomes more dynamic, proving that our model benefits from nodes that are adapted to a higher degree to the current input.

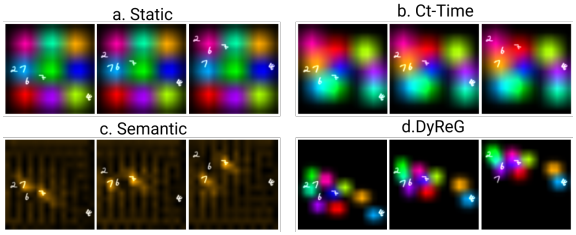

Figure 2: **Nodes' regions** on MultiSyncMNIST for 3 frames. a) *Static* Model, ignoring the input, learns a regular grid; b) *Constant-Time* predicts the same regions for all time steps, covering the movement in the video; c) The attention map of a single node of *Semantic* that can't distinguish between different instances of the same digit; d) *DyReg-GNN* generally follows the digits locations at each time steps while also adapting the regions' size.

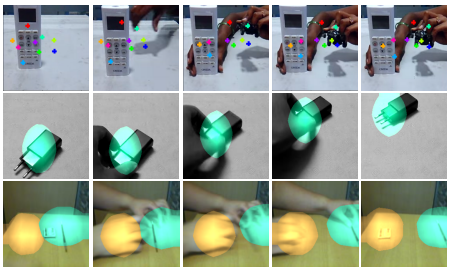

Figure 3: **Nodes' regions** on Smt-Smt-V2. We show (1st row) the center of all the N regions as predicted by DyReg-GNN (each color for a node). Each node region (last 2 rows) corresponds to a zone from the *latent* conv features pooled by a node.

Table 5: **Ablation of dynamic nodes on MultiSyncMNIST.** It is crucial to have regions that adapt based on the input (Dynamic), both their position (Pos.) and size at each time step.

Model	Optimise Pos.	Time Varying	Dynamic Pos.	Dynamic Size	Acc
Fixed					78.85
Static	✓				81.48
Ct-Time	✓		✓		86.77
Pos-Only	✓	✓	✓		93.41
DyReg-GNN	✓	✓	✓	✓	**95.09**

Table 6: **Semantic vs spatial nodes** on MultiSyncMNIST. The localized (spatial) node regions of DyReg-GNN are better suited than semantic nodes' maps obtained by the Semantic Model.

Model	Params (M)	Acc
ResNet-18	2.79	52.29
Fixed	2.82	78.85
Semantic	2.85	82.41
DyReg-GNN-Lite	2.83	91.43
DyReg-GNN	3.08	**95.09**

Studying the Importance of Localized Nodes. We argue that nodes should pool information from different locations according to the input, such that the extracted features correspond to meaningful entities. Depending on the goal, we could balance between semantic nodes globally extracted from all spatial positions or localized (spatial) nodes that are obtained from well-delimited regions.

Semantic Model creates nodes similar to [4, 19] where each node extracts features from all the spatial locations and could represent a semantic concept. Each node is extracted by a global average pooling where the weights at every position p are directly predicted from the input features at that location. Practically, we replace the spatially delimited kernel used in our model with this global attention map.

A major downside of this approach is that it does not distinguish between positions with the same features, making it harder to reason about different instances. Figure 2.C shows the attention map of a single node and we observe that it has equally high activations for both instances of the same digit, thus making it hard to distinguish between them.

This limitation does not exist in our DyReg-GNN model, as it predicts localized nodes that favour the modeling of instances. For comparison, we use two variants with a different number of parameters and show that they clearly outperform the semantic model (Table 6). These experiments prove that in cases that involve spatial reasoning of entities, such as the current task, DyReg-GNN is a perfect choice, showing its benefits for spatio-temporal modeling.

Implementation details. All models share the ResNet-18 backbone with 3 stages, where the graph receives the features from the second stage and sends its output to the third stage. We use $N = 9$ graph nodes and repeat the graph propagation for three iterations. In our main model, f from Eq. 1 is a small convolutional network while g is a fully connected layer. For the lighter model that implements g as a global pooling enriched with spatial positional information, we refer to the Supp. Materials. The graph offsets are initialized such that all the nodes' regions start in the center of the frame. In all experiments, we use SGD optimizer with learning rate 0.001 and momentum 0.9, trained on a single GPU.

Key Results. In the previous section, we experimentally validated that: **1.** DyReg-GNN consistently improves multiple backbones (Table 2) obtaining competitive results (Table 3, 4); **2.** learned dynamic

regions are crucial for good performance (Table 5) and **3.** these regions are preferable to fixed regions or external object detectors for space-time GNNs (Table 1); **4.** predicted nodes correspond to salient regions (Fig. 2-3) and are well correlated with objects (Table 1).

5 Conclusions

We propose Dynamic Salient Regions Graph Neural Networks (DyReg-GNN), a relational model for processing spatio-temporal data (videos), that augments visual GNNs by learning to predict localized nodes, adapted for the current scene. This novel method enhances the relational processing of spatio-temporal GNNs and we experimentally prove that it is superior to having nodes anchored in fixed predefined regions or linked to external pre-trained object detectors. Although we do not use region level supervision, the learning dynamics of high-level classification produces salient regions that are well correlated with object instances. We believe that our method of learning dynamic, localized nodes is a valuable direction that could lead to further advances to the growing number of powerful relational models in spatio-temporal domains.

Acknowledgment We would like to thank Florin Brad, Elena Burceanu and Florin Gogianu for their valuable feedback and discussions of this work. This work has been supported in part by Bitdefender and UEFISCDI, through projects EEA-RO-2018-0496 and PN-III-P4-ID-PCE-2020-2819.

Appendix: Discovering Dynamic Salient Regions for Spatio-Temporal Graph Neural Networks

In this Appendix we present an impact statement, discuss some limitations of the method and then we provide more technical details about DyReg-GNN model and include some additional visualisations and ablation studies.

Section A presents some views on the broader impact of this work.

Section B identifies some limitations of the methods.

Section C presents more details about how the regions are generated.

Section D shows a qualitative analysis of the regions predicted by our model.

Section E shows additional ablation studies in the synthetic setting in relation to the number of nodes, the regions size, the importance of recurrence when generating the nodes and comparisons to using ground-truth boxes or other baselines.

Section F presents some training details, describe the metric used to measure the correlation between our regions and the existing objects in the scene and have a runtime analysis of our proposed module.

We provide our full code as supplementary material and we will release it online upon the paper publication. Beside this Appendix, we also provide some videos, visualising the regions discovered by our DyReg-GNN model.

A Broader Impact

We research novel methods that would improve current general models for spatio-temporal processing. Our goal is to investigate models that emphasize a small number of relevant nodes having the potential to be more explainable and that could lead to more interpretable reasoning. Although this is not fully realised in this paper, we believe that this work is a good step in this direction. Our model enhances any convolutional backbone for video processing and thus inherits the benefits and also the possible harms brought by such models.

When developing our model, we used a synthetic dataset of moving digits and a public dataset for human-object interactions. Our model is kept generic, with no parts specially designed for these tasks. The models trained on these datasets have no obvious direct real application, as the first one is a toy dataset and the second one has restrictive classes meant only to evaluate the capabilities of the models. But developing better models for video understanding leads to more effective applications. On one hand, it could lead to better applications helping visually impaired people navigate the world and on the other hand it could lead to stricter automatic surveillance of workers. In order for ML technology to have a positive broader impact, more discussions between different actors in society should be conducted leading to the development of guidelines and practices.

The proposed work does not rely on using object detectors and only uses video level supervision. Object detectors have a predefined list of objects, that would not be sufficient for many practical cases leading to biases in the system. Moreover, this way we eliminate a possible source of biases coming from the object-level annotations.

B Limitations

By design, DyReg-GNN uses a fixed number of nodes, that we treat as a hyperparameter. This way the model is forced to produce the same number of regions regardless of the complexity of the scene. From simpler scenes, the model learns to group the nodes in overlapping regions, creating redundancy. On the other hand, more complex scenes have an increased number of relevant regions, tending to require distinct regions. This could lead to a discrepancy that would increase the difficulty of the optimisation process. Changes in scene's complexity could be also observed in a single video when the scene suffer major changes in time. For example when elements appear, disappear or are occluded from view, the number of regions predicted by the model remains the same and it is

harder to properly model all the elements. Ideally, we want a system that adapts to the complexity of the scene by dynamically predicting the number of nodes. This is a challenging task that requires additional investigations and we leave it for future work.

Preliminary experiments reveals that our method requires a relative large amounts of data to be properly trained. This seems not to be an issue for Something-Something dataset that has 80k-160k training videos, but could be an issue for smaller datasets. On MultiSyncMNIST we could train models with high accuracy on 10% of the whole dataset of 600k videos. But when using only 1% (6k videos) of the data, the predicted regions would not change during training. Given the size of the recent video dataset, this is not a big limitation.

C Node Region Generation

The goal of this sub-module is to generate the regions that correspond to salient zones in the input. We achieve this by processing the input globally with position-aware functions f and $\{g_i\}$.

Function f. We use f function to aggregate local information from larger regions in the input while preserving sufficient positional information. The input $X_t \in \mathbb{R}^{H \times W \times C}$ is first projected into a lower dimension C' since this representation should only encode saliency without the need to precisely model visual elements. Then we increase the receptive field by applying two conv layers, followed by a transposed conv and then a final conv layer. This results in a feature map $M_t = f(X_t) \in \mathbb{R}^{H' \times W' \times C'}$. Depending on the backbone and the stage where the graph is added H, W have different values and we adapt the hyperparameters of the convolutional layers such that H' and W' are not smaller than 6. For example, in the synthetic experiments f reduces the input from $\mathbb{R}^{16 \times 16 \times 32}$ to $\mathbb{R}^{7 \times 7 \times 16}$.

Functions $\{g_i\}$. For each node i we use g_i to extract a global latent representation from which we predict the corresponding region parameters. We present two variant of g_i function, a larger and more precise one and a smaller, more computational efficient one.

For the bigger one, we use a simple fully connected layer of size $C \times (H' * W' * C')$ that takes the whole M_t and produces a vector of size C. This way g_i could distinguish and model the spatial locations of the $H' \times W'$ grid.

The second approach consists in a weighted global average pooling for each node i. The weight associated to each location p is predicted directly from the input $M_{t,p}$ by a 1×1 convolution. But this results in a translation-invariant function g_i that losses the location information. We alleviate this by adding to each of the $H' \times W'$ location a positional embedding similar to the one used in [31]. This approach predicts regions of slightly poorer quality as the location information is not perfectly encoded in the positional embeddings. For a lighter model, such as the one presented in Table 6 of the main paper we could use the second approach for the $\{g_i\}$ functions and also skip the f processing.

Constraints Equation 4 in the main paper could be expended as:

$$\begin{aligned} \tilde{\mathbf{o}}_i &= (\Delta\tilde{x}_i, \Delta\tilde{y}_i, \tilde{w}_i, \tilde{h}_i) = \gamma \odot W_o \mathbf{z}_i \in \mathbb{R}^4 \\ \mathbf{o}_i &= \alpha(\tilde{\mathbf{o}}_i) \end{aligned} \tag{11}$$

To constrain the model to predict valid image regions and also to start from regions with favourable position and size, we apply non-linear functions for each component $\mathbf{o}_i = \alpha(\tilde{\mathbf{o}}_i)$. We design the non-linearities such that $w_i, h_i > 0$ and $\Delta x_i + C_x \in [0, W]$ and , $\Delta y_i + C_y \in [0, H]$, where C is a fixed reference point. In experiments, all nodes share the same constant C , representing the center of the image.

$$h = e^{\tilde{h}} h_{init} \qquad w = e^{\tilde{w}} w_{init} \tag{12}$$

$$\Delta x = \frac{W}{2} \tanh\left(\Delta\tilde{x} + (\frac{2C_x}{W} - 1)\right) + \frac{W}{2} - C_x \tag{13}$$

$$\Delta y = \frac{H}{2} \tanh\left(\Delta\tilde{y} + (\frac{2C_y}{H} - 1)\right) + \frac{H}{2} - C_y \tag{14}$$

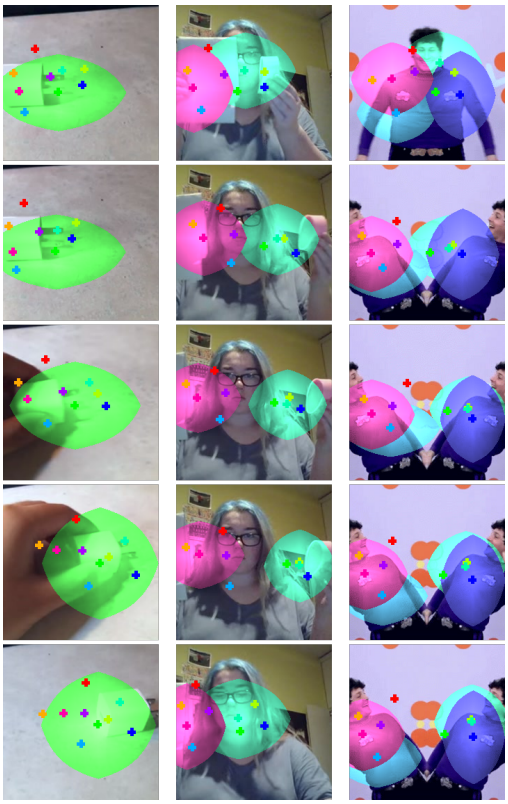

Figure 4: Visualisations of salient regions associated with each node, as predicted by our DyReg-GNN model on videos from Smt-Smt-v2 dataset (Left and Center) and on out-of-distribution real-world videos (Right). Each node learns to move to different relevant regions in the input. For each video, we show the centers corresponding to all the nodes and, for a better visualisation, a subset of the predicted regions. Due to the receptive field of the backbone, the nodes are actually influenced by larger regions in the initial input.

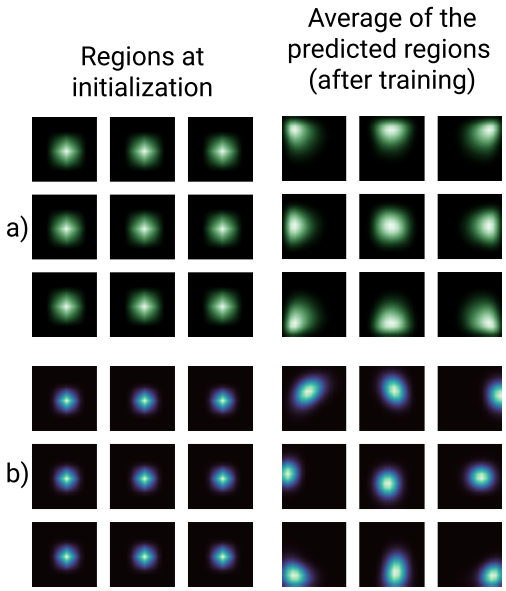

Figure 5: Visualisation of the average locations of the salient regions associated with DyReg-GNN's nodes, computed over the validation set of (a) MultiSyncMNIST and (b) Smt-Smt-v2. In these visualisations the order of the regions is manually selected. In the left column we show the regions at initialisation, and in the right column we present the mean regions as predicted by our learned DyReg-GNN model. Here, we keep the size of the regions fixed, each node has a preferred location in space and assigns salient regions around it. This behaviour is learned by the model to break the symmetry of the nodes.

By initialising $\gamma = 0$ we obtain $h = h_{init}$, $w = w_{init}$ and $\Delta y = \Delta x = 0$. This means that all regions are initialized centered in the reference point C and start with the predefined size. By default we set $h_{init} = \frac{H}{6}$, $w_{init} = \frac{W}{6}$.

D Visualising the nodes' regions

The region associated with each node is clearly delimited in space and we can easily visualize them. We train a model on Something-Something-V2 dataset of human-object interactions and in Figure.4 we show its predicted nodes' regions for two videos from the dataset and one out-of-distribution video. Generally the nodes follow relevant regions in the input. We note that the visualisation of the regions is only an approximation of the actual regions that send information to the graph nodes. Each node pools info from a low-resolution region in the latent convolutional features, that corresponds to the high-resolution visualized region. But, the actual area that contributes to each node is actually larger, due to the receptive field of the convolutional network. Moreover, the backbone also contains temporal processing (e.g. in the form of temporal feature shifting in the case of TSM or 3D conv for I3D) such that each node receives information from adjacent time steps. Thus, we expect some misalignment in the visualizations both in space and time.

To better understand how each node attends to the input, we compute the average of its associated regions over the entire evaluation dataset (see Figure. 5). We observe that the regions are initialized in the center of the image and, after training, each node learns to attend to regions around a specific location. For each video, a node predicts a different region, according to the input, but it is situated mostly around a certain part of the image. This behaviour is learned by the model to break the symmetry of the nodes and be able to create an implicit matching between relevant parts of the input and the nodes.

E Synthetic Setting

E.1 Dataset details

Based on [3] we create MultiSyncMNIST. It consists of 10 frames videos of size 128×128, where MNIST digits move on a black background. Each video has 5 moving digits and a subset of them moves synchronously. Different from the original version, each video could contain multiple instances of the same digit class and any subset can move in the same way. This is done to make it more difficult to distinguish between multiple visual instances. The goal is to detect the smallest and largest digit class among the subset of synchronous digits with each pair of two digits forming a label. In total, we have 55 possible pairs of two digits, and adding a class for videos without synchronous digits results in a 56-way classification task. For example, if a video contains the digits: $\{2, 4, 6, 7, 7\}$ and the subset $\{4, 6, 7\}$ is moving in the same way, it has the label associated with the pair: $\{4, 7\}$. The dataset contains 600k training videos and 10k validation videos.

E.2 Ablation: Number of Nodes

We investigate the effect on the performance of the number of nodes for different environments, of varying difficulty. We conduct experiments varying the complexity of MultiSyncMNIST dataset, by changing the number of moving digit ($D \in \{3, 5, 9\}$). As expected, we observe that for good performance, it is necessary to set a number of nodes that exceeds the number of relevant entities in the scene.

E.3 Ablation: Regions' Size

In this subsection, we conduct experiments to investigate the effect of the size of the node regions on the final performance. Each node pools information from latent convolutional features of size $H \times W = 16 \times 16$. We fix the size of each region to $\frac{H}{\lambda}$ where $H = 16$ and $\lambda \in \{6, 7, 8, 11, 16\}$ and show the results of the corresponding models in Table 11. Setting $\lambda = 8$ corresponds to regions having approximately the expected values of the regions predicted by the full DyReg-GNN model. We note that the model is relatively robust to reasonable choices of size but the best performance is achieved when the size of each region is dynamically predicted from the input. We also note that by setting $\lambda = H = 16$ we arrive at the standard bilinear interpolation kernel. This setting leads us to a model that is more unstable in training than the others and obtains poorer results. There are two probable reasons for this. First, the regions cover a small area thus they must be more precise to cover small entities while also being unable to cover large entities in their entirely. Second, the gradients used to update the region parameters are noisier for small regions. This is because, the gradients of the offsets depend on the features of the predicted regions, and for gradients of the offsets to be informative it means that the features in the regions should also be relevant for the final prediction. Smaller regions have a smaller chance of achieving this.

E.4 Ablation: Comparison to Keypoints Extractor

We conduct an experiment to compare our dynamic way of generating nodes with a previous method [55] that detects keypoints from images. For a fair comparison, we replace the part in our model that predicts the locations of the node regions with a method similar their encoder. The rest of our model would remain the same and will be learned in the same way by the video classification loss. We chose the architecture such that the number of parameters remains the same. As in the original paper, this method predicts the positions but keeps the shape fixed, thus we compare it with our module that has fixed shape, although it obtains poorer results than the full model. We report the results in Table 10.

Table 7: Results on MultiSyncMNIST when varying the number of nodes on datasets of different complexity (with increasing number of moving digits). It is crucial that the number of nodes exceeds the number of important entities in the scene. In the bottom of the table, we also show two additional baselines with the same number of parameters.

Model	Dataset		
# digits(D)	D=3	D=5	D=9
DyReg 5 Nodes	98.2	89.6	64.4
DyReg 9 Nodes	98.1	95.1	79.3
DyReg 16 Nodes	**98.4**	**95.6**	**83.0**
R18 + Conv-LSTM	-	89.1	50.5
R18 + NL	-	93.2	67.5

Table 8: Results on Smt-Smt-V2 val. set, using a single 224×224 central crop. We observe that DyReg-GNN models improve over the TSM backbone and that it is important to have the kick-start given by the distillation to learn multiple dynamic graph modules.

Model	Top 1	Top 5
TSM	61.1	86.5
DyReg-GNN r3-4-5	62.1	87.4
DyReg-GNN r3-4-5 Distill	62.8	87.7

Table 9: Ablation on MultiSyncMNIST for showing the importance of recurrence for predicting the regions.

Model	Accuracy
DyReg-GNN without GRU	91.91
DyReg-GNN	95.09

Table 10: Comparison to Keypoints-based method on MultiSyncMNIST.

Model	Acc
ResNet-18	52.29
Fixed	78.85
Keypoints	90.60
DyReg-GNN - Pos-Only	93.41
DyReg-GNN	**95.09**

Note that the dynamic regions obtained using the keypoints method (denoted as Keypoints) improve over the fixed-regions approach, reinforcing the idea that dynamic regions are helpful for relational processing. However, our DyReg-GNN models obtain better results both when the size of the regions is fixed and especially when the size is also predicted.

E.5 Ablation: Importance of Recurrence for Region Generation

We conduct an experiment (Table 9) on the MultiSyncMNIST dataset, where we omit the GRU from Eq. 3, thus predicting the regions at each time step only from the features of frame. The performance drops from (DyReg-GNN) to (DyReg-GNN without GRU). This experiment suggests that the temporal modeling in region generation is important for good performance.

E.6 Ablation: Ground-Truth Boxes

To evaluate the quality of our proposed regions on MultiSyncMNIST, we train our model using ground-truth (gt.) boxes instead of generated regions. As this task is defined by the exact movements of digits, the gt. boxes represents the ideal regions for the relational model, giving an upper bound for our method. This oracle model obtains 97.30% accuracy, while the DyReg-GNN model obtains 95.09%. Comparing to the other baselines in the main paper, our DyReg-GNN model obtains closer results to the oracle model, proving the utility of the node generation.

E.7 Comparison to other baselines

We compare our method to additional baselines by replacing our entire DyReg-GNN module with two other models, as seen in Table 7. The first baseline (R18+Conv-LSTM) consists in a convolutional encoder that reduces the spatial dimensions, a shared LSTM applied independently on each spatial position followed by a convolutional decoder. The second baseline (R18-NL) consists in a Non-Local[30] network. Both modules are applied over the same ResNet 18 backbone and have the same number of parameters as DyReg-GNN. DyReg-GNN surpasses the other baselines and the difference in performance is more significant in the hardest setting.

Table 11: Experiments on MultiSyncMNIST investigating the size of the learned regions. The best performance is obtained when the size is dynamically predicted while the worst is given by a model with the regions kept at the minimum value, corresponding to the standard bilinear interpolation kernel.

Learnable (Full)	Fix $\lambda = 6$	Fix $\lambda = 7$	Fix $\lambda = 8$	Fix $\lambda = 11$	Fix bilinear
95.09	93.41	94.11	94.04	94.03	90.99

Table 12: Comparison in terms of the number of operations and parameters for a single video of size 224×224. Comparing to assigning nodes to boxes from external detectors (as in I3D+NL+GCN), our module has a smaller computational overhead.

Model	Frames	FLOPS	Params
I3D [63]	32	153.0G	28.0M
I3D+NL [30]	32	168.0G	35.3M
I3D+NL+GCN [7]	32	303.0G	62.2M
STM [80]	16	66.5G	24.0M
TSM [67]	16	65.8G	23.9M
TSM + Fixed GNN r4	16	66.3G	24.9M
TSM + DyReg-GNN r4	16	66.4G	25.7M
TSM + Fixed r3-4-5	16	67.2G	26.1M
TSM + DyReg-GNN r3-4-5	16	67.4G	28.7M

F Human-Object Interactions

F.1 Distillation for kick-starting the optimisation

When training models with multiple DyReg-GNN modules, we observe that only the regions of the last module behaves well, thus a single graph module is effectively used. To alleviate this problem we train models with a single graph at different stages, and use their region predictions to distill the larger model, for the first 10% of the training iterations. This kick-starts the learning of all graph modules, improving the overall results, as seen in Table 8.

F.2 Implementation Details for Detector Experiment

In the main paper, for the comparison with the regions extracted using object detectors (Section 4.1), we use a Faster RCNN ResNet-50 FPN detector[3] pre-trained on MSCOCO dataset. We extract the top-9 detected boxes based on the confidence score and temporally match them using the hungarian algorithm to maximize the IoU between boxes at consecutive time steps.

F.3 Object-centric metric

To quantify to what degree the nodes cover existing ground-truth objects in the scene, we propose the following metric. We measure the distance between the center of the predicted regions and the center of the gt. objects. For each node region in each frame, we compute the minimum L_2 distance to all gt. object bounding boxes and average all of them.

$$Dist_p = \frac{1}{NF} \sum_{f=1}^{F} \sum_{i=1}^{N} min_j |C_i + \Delta_i - B_j|_2 \tag{15}$$

Vice versa we compute for each gt. box the minimum L_2 distance to all predicted regions and average all of them.

$$Dist_r = \frac{1}{N_B F} \sum_{f=1}^{F} \sum_{j=1}^{N_B} min_i |C_i + \Delta_i - B_j|_2 \tag{16}$$

[3] `https://github.com/facebookresearch/detectron2`

In the previous equations, F is the number of frames in the whole dataset, N the number of nodes, N_B the number of objects in the current frame, $C_i + \Delta_i$ is the center of i-th node's region and B_j the center of the j-th object in the current frame and we average over the whole dataset.

The first score (representing precision) ensures that all the predicted regions are close to real objects, while the second (recall) ensures that all the objects are close to at least one predicted region. To balance them, we present as our final score their harmonic mean.

F.4 Runtime Analysis

We compute the number of operations, measured in FLOPS, the parameters and the inference time for our model. We evaluate videos of size 224×224 in batches of 16 on a single NVIDIA GTX 1080 Ti GPU. TSM backbone, TSM + DyReg-GNN-r4, and DyReg-GNN-r3-4-5 run at 35.7, 34.8, and 32.7 videos per second respectively, showing that our DyReg-GNN module does not add a large overhead over the backbone. In Table 12, we compare in terms of number of parameters and operations against other current standard models used in video processing. Note that the I3D-based models uses 32 frames but for our method, the number of operations increases linearly with the number of frames so it is easy to make a fair comparison. The I3D+NL+GCN model counts also the parameters and the operations of the detector module used to extract object boxes. This is characteristic to all the relational models where the nodes are extracted using object detectors. Contrary to this approach, our method has a smaller total complexity by directly predicting salient regions instead of using precise object proposals given by external models.

References

[1] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4898–4906. Curran Associates, Inc., 2016.

[2] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] Andrei Nicolicioiu, Iulia Duta, and Marius Leordeanu. Recurrent space-time graph neural networks. In *Advances in Neural Information Processing Systems 32*, pages 12838–12850. Curran Associates, Inc., 2019.

[4] Y. Chen, M. Rohrbach, Z. Yan, Y. Shuicheng, J. Feng, and Y. Kalantidis. Graph-based global reasoning networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 433–442, 2019.

[5] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 30*, pages 4967–4976, 2017.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[7] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 399–417, 2018.

[8] Elizabeth S Spelke. Core knowledge. *American psychologist*, 55(11):1233, 2000.

[9] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2010.

[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[11] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[12] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, 2017.

[13] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs, 2018.

[14] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, June 2018.

[15] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–417, 2018.

[16] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[17] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020.

[18] Yin Li and Abhinav Gupta. Beyond grids: Learning graph representations for visual recognition. In *Advances in Neural Information Processing Systems*, pages 9225–9235, 2018.

[19] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems 31*, pages 1853–1863. 2018.

[20] Songyang Zhang, Xuming He, and Shipeng Yan. Latentgnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pages 7374–7383. PMLR, 2019.

[21] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020.

[22] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[23] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *Advances in Neural Information Processing Systems*, pages 350–359, 2018.

[24] J. Gao, T. Zhang, and C. Xu. Graph convolutional tracking. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4644–4654, 2019.

[25] Roei Herzig, Elad Levi, Huijuan Xu, Hang Gao, Eli Brosh, Xiaolong Wang, Amir Globerson, and Trevor Darrell. Spatio-temporal action graph networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[26] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9975–9984, 2019.

[27] Hao Huang, Luowei Zhou, Wei Zhang, Jason Corso, and Chenliang Xu. Dynamic graph modules for modeling object-object interactions in activity recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 101.1–101.12. BMVA Press, September 2019.

[28] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[29] Moshiko Raboh, Roei Herzig, Jonathan Berant, Gal Chechik, and Amir Globerson. Differentiable scene graphs. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[30] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[32] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[33] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229, 2020.

[34] Honglu Zhou, Asim Kadav, Farley Lai, Alexandru Niculescu-Mizil, Martin Renqiang Min, Mubbasir Kapadia, and Hans Peter Graf. Hopper: Multi-hop transformer for spatiotemporal reasoning. In *International Conference on Learning Representations*, 2021.

[35] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018.

[36] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[37] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018.

[38] Effrosyni Mavroudi, Benjamın Béjar, and René Vidal. Representation learning on visual-symbolic graphs for video understanding. In *The European Conference on Computer Vision (ECCV)*, 2020.

[39] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[40] Nasim Rahaman, Anirudh Goyal, Muhammad Waleed Gondal, Manuel Wuthrich, Stefan Bauer, Yash Sharma, Yoshua Bengio, and Bernhard Schölkopf. S2rms: Spatially structured recurrent modules. *arXiv preprint arXiv:2007.06533*, 2020.

[41] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2017–2025, 2015.

[42] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3150–3158, 2016.

[43] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[44] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[45] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[46] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[47] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019.

[48] Li Zhang, Dan Xu, Anurag Arnab, and Philip HS Torr. Dynamic graph message passing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[49] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[50] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

[51] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2424–2433. PMLR, 09–15 Jun 2019.

[52] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[53] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.

[54] Qian Huang, Horace He, Abhay Singh, Yan Zhang, Ser-Nam Lim, and Austin R. Benson. Better set representations for relational reasoning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[55] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[56] 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. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[57] Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. In *Advances in Neural Information Processing Systems*, volume 33, pages 9180–9192. Curran Associates, Inc., 2020.

[58] Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1439–1456. PMLR, 30 Oct–01 Nov 2020.

[59] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[60] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[61] Chih-Yao Ma, Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib. Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *Signal Processing: Image Communication*, 2018.

[62] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

[63] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017.

[64] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[65] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[66] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

[67] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[68] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[69] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[70] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[71] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[72] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016.

[73] Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning. In *International Conference on Learning Representations*, 2020.

[74] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 3, 2017.

[75] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[76] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. Microsoft coco: Common objects in context. In *ECCV*. European Conference on Computer Vision, September 2014.

[77] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[78] Chenxu Luo and Alan Yuille. Grouped spatial-temporal aggretation for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[79] X. Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1089–1098, 2020.

[80] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2000–2009, 2019.

[81] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *European Conference on Computer Vision*, pages 345–362. Springer, 2020.

[82] J. Zhang, F. Shen, Xing Xu, and H. Shen. Temporal reasoning graph for activity recognition. *IEEE Transactions on Image Processing*, 29:5491–5506, 2020.

[83] Yizhou Zhou, Xiaoyan Sun, Chong Luo, Zheng-Jun Zha, and Wenjun Zeng. Spatiotemporal fusion in 3d cnns: A probabilistic view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9829–9838, 2020.

[84] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 211–252, 2015.

[85] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.