# Dynamic Regions Graph Neural Networks for Spatio-Temporal Reasoning

**Iulia Duta,**[1*] **Andrei Nicolicioiu,** [1*]
[1] Bitdefender, Romania
iduta@bitdefender.com, anicolicioiu@bitdefender.com

**Abstract**

Graph Neural Networks are perfectly suited to capture latent interactions occurring in the spatio-temporal domain. But when an explicit structure is not available, as in the visual domain, it is not obvious what atomic elements should be represented as nodes. They should depend on the context and the kinds of relations that we are interested in. We are focusing on modeling relations between instances by proposing a method that takes advantage of the locality assumption to create nodes that are clearly localised in space. Current works are using external object detectors or fixed regions to extract features corresponding to graph nodes, while we propose a module for generating the regions associated with each node dynamically, without explicit object-level supervision. Conditioned on the input, for each node we predict the location and size of a region and use them to pool node features using a differentiable mechanism. Constructing these localised, adaptive nodes makes our model biased towards object-centric representations and we show that it improves the modeling of visual interactions. By relying on a few localized nodes, our method learns to focus on salient regions leading to a more explainable model. Our model achieves superior results on video classification tasks involving instance interactions.

## 1 Introduction

Spatio-temporal data, and videos in particular, are characterised by an abundance of interactions between concepts (Chen et al. 2019) and instances (Wang and Gupta 2018). The general meaning of a class is given by an abstract, *semantic concept*, which can be particularised into specific *instances*, each with its own identity. The relations between the concepts define a context and clarify the semantics, while different spatio-temporal environments, containing the same set of concepts, are distinguished by their specific composition of instances.

For proper modeling, both types of interactions are needed. In classic convolutional networks their ratio is implicitly defined, while graph neural networks offer more flexibility in choosing inductive biases, favouring one or the other. Some current works applying graph networks in visual domain lean towards relating concepts in a semantic space (Chen et al. 2019; Liang et al. 2018) while others focus on the interactions between entities given by object detectors (Wang and Gupta 2018; Sun et al. 2018). In a similar vein as the second approach, we propose a Graph Neural Network method for visual understanding, focusing on the creation of *instance-oriented nodes* relating entities in the scene. By entity, we denote a localised visual unit that represents an entire or a part of an instance.

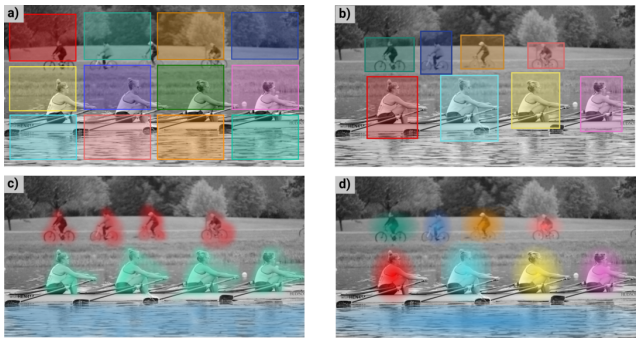


Figure 1: Multiple kinds of latent visual interactions could be captured by graph models with nodes created in different ways: a) fixed spatio-temporal regions b) bounding-boxes given by object detectors c) semantic maps capturing multiple concepts d) local regions capturing distinct instances. This work aims for the last approach by proposing localised graph nodes, oriented towards capturing instances.

By the locality assumption, local correlations are stronger than distant ones, thus we create graph nodes features by pooling from clearly defined regions in the input. The pooling is done in a differentiable way w.r.t. regions' location and size such that we could predict them dynamically, conditioned on the input, in order to better adapt to the current scene. The process does not involve objects identified by external detectors and does not use any kind of object-level supervision, only relying on video classification signal.

Object-centric representations improve the learning capabilities of visual models (Locatello et al. 2020) and we argue that our instance-oriented nodes produce such representations for the following reasons. First, our model extracts local information that better correlates with instances. Second, graph methods work best on top of well-defined structures where nodes have a clear meaning on their own (Battaglia

[*]Equal contribution.

et al. 2018), thus the learning process should lead to node representations that best fulfil these requirements. We also validate experimentally that the predicted kernels correlate with object locations.

Our methods learns to adapt the level of granularity of the nodes' regions according to the current task, to cover well the salient entities in the scene. This is in contrast to the approaches based on object detection that are restricted by a set of pre-defined object annotations. Focusing on the nodes' location, our method *Dynamic Regions Graph Neural Networks* (DyReG) is well suited for tasks that rely heavily on the position of different entities, such as action recognition or human-object interaction.

**Our main contributions** are summarised as follow:

1. We design a novel method to create **localised** graph nodes, **dynamically** predicted from the input, that are biased towards **object-centric** representations.
2. Our model can discover salient regions, suitable for relational processing, **without** object-level **supervision**.
3. Localising the nodes gives a form of hard attention that makes the model more **explainable** and we quantitatively show that they correlate with objects locations.
4. We obtain state-of-the-art results on a video classification task where instances play a key role.

## 2 Related work

**Graph Neural Networks** Graph neural networks have been recently used in many domains where the data has a non-uniform structure (Bruna et al. 2013; Battaglia et al. 2016; Gilmer et al. 2017; Li et al. 2018). In vision tasks, it is important to model the relations between different entities appearing in the scene (Baradel et al. 2018; Qi et al. 2018) and Graph Neural Networks have strong inductive biases towards relations (Battaglia et al. 2018), 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. We classify recent approaches taking into account *how* they create the nodes and *what* type of information each node represents.

Regarding how the nodes are created, recent literature generally follows two directions. In the first one, the graph nodes represent convolutional points or fixed regions (Santoro et al. 2017; Wang et al. 2018; Chen et al. 2018b; Gao, Zhang, and Xu 2019), while in the second one nodes are associated with objects given by pre-trained external detectors (Wang and Gupta 2018; Herzig et al. 2019; Zhang et al. 2019; Materzynska et al. 2020). Our work draws from both directions, covering dynamically predicted regions without object-level supervision.

The idea of forming relations from visual elements given by convolutional features appears in (Santoro et al. 2017) where they process pairs of features from every location, to capture distant interactions in the scene. The Non-Local (Wang et al. 2018) method creates graph's nodes from every point in the convolutional features and uses self-attention (Vaswani et al. 2017) mechanism to achieve long-range connections. Following, (Nicolicioiu, Duta, and Leordeanu 2019) extract nodes from larger fixed regions at different scales and processes them recurrently.

Instances from the visual scene, each having their own spatial identity such as objects, are involved in complex interactions that are modeled by several works using Graph Neural Networks. In (Wang and Gupta 2018), information between object features is propagated over two different graph structures, one given by location and one given by similarity between nodes. Other works combine both worlds by sending messages between nodes corresponding to points and object features (Sun et al. 2018; Girdhar et al. 2019) or by introducing propagation over class or concept embeddings (Chen et al. 2018a; Mavroudi, Béjar, and Vidal 2020).

Regarding the kind of information represented by each node, we could distinguish two types of interactions in the visual world, one relating high-level concepts, living in a purely semantical space, and one between instances associated with specific spatio-temporal locations. For example, as illustrated in Figure 1, all the boats in an image are purely semantically associated with the river, while a specific boat relates to its rowers by an instance interaction. This is similar to the usage of the terms semantic and instance segmentation. Depending on the task, methods are designed to model them in different proportions. The approaches of (Chen et al. 2019; Li and Gupta 2018; Kipf, van der Pol, and Welling 2020; Locatello et al. 2020) capture the purely semantic interactions by reasoning over global graph nodes, each one receiving information from all the points in convolutional input, regardless of spatio-temporal position. In (Chen et al. 2019) the nodes assignments are predicted from the input, while in (Li and Gupta 2018) the associations between input and nodes are made by a soft clusterization. The work of (Locatello et al. 2020) is able to discover different representation groups by using an iterative clusterization based on self-attention similarity. In (Liang et al. 2018) the semantic features of global nodes are also augmented with concept information given by class embeddings and define the connectivity by a knowledge graph structure.

The downside of these approaches is that individual instances, especially those belonging to the same semantic class, are not distinguished in the graph processing. This information is essential in tasks such as capturing human-object interactions, instance segmentation or tracking. Alternatively, we associate nodes with features from specific regions, predicted from the input, giving our model bias for modeling instance interactions.

Concurrently, the method (Rahaman et al. 2020) uses multiple position-aware nodes that take into account the spatial structure. This makes their method 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 generalise to new environments.

Similar relations are captured by methods involving features of detected objects as node features. However, these models not only depend on the performance of an external

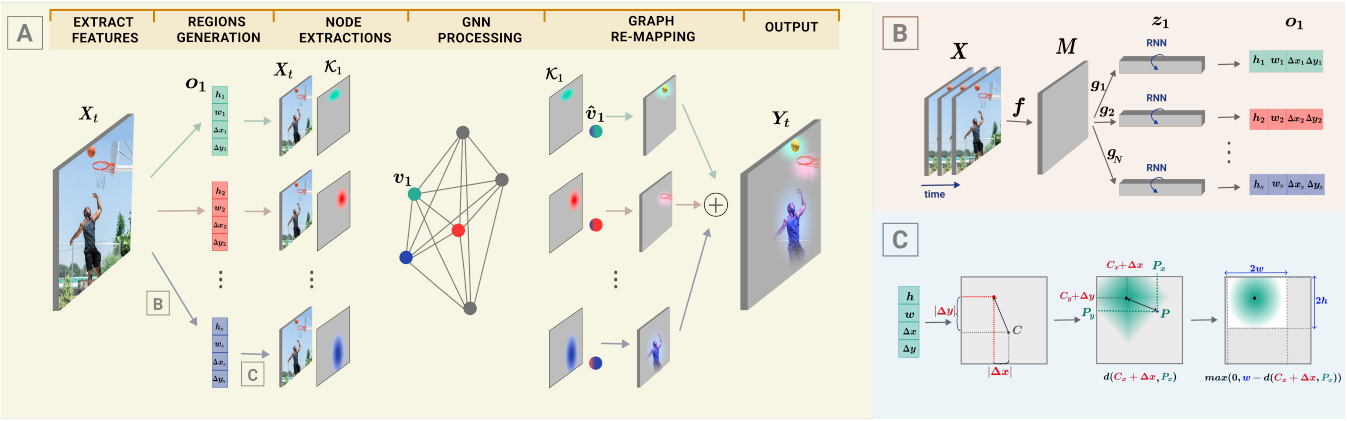


Figure 2: (**Left**) Architecture of our DyReG model that extracts localised node representations, useful for relational processing. For each node $i$, from the feature volume $X_t$, we extract parameters $\mathbf{o}_i$ denoting the location and size of a region. They define a kernel $K_i$, used to extract the localised features $\mathbf{v}_i$ from the corresponding region of $X_t$. We process the nodes with a spatio-temporal GNN and project each node features $\hat{\mathbf{v}}_i$ into its initial location, according to the same kernel $K_i$. **(Right)** B) Node Region Generation: To extract regions params $\mathbf{o}_i$ we use positional aware functions $f$ and $\{g_i\}$. $f$ extracts latent representation shared between nodes, while each $g_i$ has different parameters for each node $i$. C) Node Features Extraction: A kernel, created from each set of params $\mathbf{o}_i$, is used in a differentiable pooling w.r.t. $\mathbf{o}_i$ that allows the optimisation of these parameters.

object detector but are also unable to adapt to the requirements of the current task, being limited to the set of predefined object annotations. Different from them, our proposed module predicts salient regions conditioned on the input and optimise these predictions for the current task, using only the video classification signal. This is related to the trend in supervised object detection where external object proposals have been replaced by self-predicted locations (Ren et al. 2015) while also pooling using bilinear interpolation (Dai, He, and Sun 2016; He et al. 2017).

These two types of interactions are captured in the dual attention model (Fu et al. 2019) by using spatial attention for instance relations and channel attention for semantic ones.

**Dynamic Networks** Several works use second-order computations by dynamically predicting different parts of their model from the input, instead of directly optimising parameters. The method (Jia et al. 2016), replaces the learnable convolutional parameters with weights predicted from the input using a distinct convolution, resulting in a dynamically generated filter. While in standard convolutions the kernel is multiplied with features from fixed points, deformable convolutions (Dai et al. 2017; Zhu et al. 2019) predict an offset for each position based on the input. Similar, (Zhang et al. 2020b) use the same idea of predicting offsets but in a graph network formulation. As in Non-Local, their nodes correspond to all the convolutional feature points but they use the offset to control the connectivity of each node. They take advantage of sparse, non-local connections between points, while we focus on creating instance-oriented nodes that have a more clear identity. This kind of processing, involving a small set of powerful modules, is also highlighted in works like (Goyal et al. 2019) and (Rahaman et al. 2020).

**Activity Recognition** Video classification has been influenced by methods designed for 2D images (Yue-Hei Ng et al. 2015; Donahue et al. 2015; Ma et al. 2018; Zhou et al. 2018). More powerful 3D convolutional networks, inflated from their 2D counterpart, have been later proposed (Carreira and Zisserman 2017), while other methods factorise the 3D convolutions (Xie et al. 2018; Tran et al. 2018, 2019) bringing both computational speed and accuracy. Methods like TSM (Lin, Gan, and Han 2019) and (Fan et al. 2020) showed that a simple shift in the convolutional features results in improved accuracy at low computational budget.

## 3 Dynamic Regions GNNs

We investigate how we can create node representations that are useful for modeling visual interaction using Graph Neural Networks. While there are many formulations of graph processing in the visual domain, little attention is given in the literature to the way the nodes are extracted from convolutional features.

Our method can distinguish and relate different instances existing in the visual scene, as each node is a high-level module that represents an entity, having a definite spatio-temporal location. To adapt to the approached task and the current visual scene, we *dynamically* assign each node to a location, based on the input. Since each node strongly relies on the predicted position this could lead to an object-centric representation of an entity.

The main architecture of our DyReG model is illustrated in Figure 2. From a spatio-temporal feature volume $X \in \mathbb{R}^{T\times H\times W\times C}$, we use a differentiable pooling operation to create graph nodes, each one extracted from a specific region of the video. At each time step $t$, we estimate $N$ regions, defined by a kernel function $\mathcal{K}$ with 4 parameters that corresponds to its location and size. The nodes containing features from these predicted locations are processed by a graph neural network to capture interactions across space and time. The resulting node features, enriched by the rela-

tional processing, are then projected to their initial position according to the kernel function. The module can be inserted at any intermediate level in a standard convolutional model. In the following sections, we explain in more detail each part of our model.

### 3.1 Node Region Generation

Our method processes a few object-centric nodes so it is crucial to assign them to the salient regions. To accomplish this, we propose a global processing (as illustrated in Figure 2 B) that aggregates the entire input features to produce regions defined by four parameters for each node: two for determining the location of the center $(\Delta x, \Delta y)$ and two to control the width and height $(w, h)$.

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. 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.

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

We process each of the $N$ latent representations independently, with a GRU (Cho et al. 2014) recurrent network, to achieve consistency across time. At each time step, the final parameters are obtained by a linear projection, modulated by a set of parameters $\alpha$ used to control the initialisation.

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

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

### 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 spatial locations $p \in \mathbb{R}^2$ are interpolated according to the kernel function $\mathcal{K}^{(i)}(p)$ as presented in Figure 2 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 $i$. 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 $\Delta x_i$ are the predicted parameters from Eq. 4.

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

For each time step $t$, the features for node $i$ are obtained by interpolating using the kernel function all the points in the input $X_t$. 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.

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

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 requires 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.

We note that setting $w_i = 1$ leads to the standard bilinear interpolation kernel, but optimising it allows the network to learn the size of the regions dynamically adapted for each node. Interpolating from larger regions also results in a more stable optimisation of the predicted region parameters. In the Appendix, we present some experiments regarding the regions' size and analyse how different non-linear functions over the distance determine their shape.

### 3.3 Graph Processing

For processing the node features, different Graph Neural Networks could be used. Generally, they follow a framework (Gilmer et al. 2017; Battaglia et al. 2018) of sending messages between connected nodes, aggregating them using simple permutation invariant functions (Xu et al. 2019) or attention mechanisms (Velikovi et al. 2018) and updating them to form the final nodes representations.

The specific message-passing mechanism is not the focus of the current work, thus we follow a general formulation similar to (Nicolicioiu, Duta, and Leordeanu 2019) for spatio-temporal graph processing, using two different stages: one happening between all the nodes at a single time step and the other updating each node across time. For each time step $t$, we send messages between each pair of two nodes, computed as an MLP and aggregates them using a dot product attention coefficient $a(v_i, v_j) \in \mathbb{R}$.

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

We incorporate temporal information through a 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 several times.

### 3.4 Graph Re-Mapping

The graph propagation produces higher-level information, by modeling the global interactions between position-aware nodes. We map the node features back into the same space as

the input $X_t$, in order to further benefit from complementary local processing, such as convolutional layers. 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, distributing their features into a local region defined by the initial kernel $\mathcal{K}^{(i)}$:

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

The output $\mathbf{y}$ could be used for the final prediction or could be further processed with any spatio-temporal model.

## 4 Experiments

We test our model on two video classification datasets. For the first one, we use a variant of the SyncMNIST (Nicolicioiu, Duta, and Leordeanu 2019) dataset that is challenging and requires spatio-temporal reasoning, while allowing fast experimentation. We then evaluate on Something-Something-V2 (Goyal et al. 2017), a real-world dataset that involves complex human-object interactions.

### 4.1 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 variant of the dataset, that we call MultiSyncMNIST, consisting in scenes with 5 moving digits where a subset of them moves synchronously. A video can contain multiple instances of the same class and the goal is to find the smallest and the largest digit class that move in the same way, resulting in a video classification task with 56 classes. The dataset contains 600k training videos and 10k validation videos with 10 frames each.

The task is challenging because it requires to distinguish between multiple instances and to identify the subset that is moving synchronously. This entails modeling complex instance-based relationships.

We use this dataset to identify key challenges in video-processing and show how our model is able to address them.

**Baselines** We start with a 2D ResNet-12 (He et al. 2016) baseline, also used as a backbone for our method, applied independently at each time step, that is not able to solve the task and only takes advantage of the dataset biases. In Table 1 we show an oracle model *2D Oracle* that predicts the smallest and largest digits in the video, ignoring the movement, and observe that the 2D model achieves similar performance, hinting that it only learns to recognise the digits. It has been shown (Wang et al. 2018) that there is a need to model spatio-temporal relations in video and graph models are well suited for this task (Wang and Gupta 2018). Thus we introduce a series of graph models, that uses the features from the second stage of the ResNet-12 as input for a graph module and processed them as explained in Section 3.3.

We argue for two key ideas, first that graph methods should require nodes that are *localised* in space, biased toward instances and second that nodes should *dynamically adapt* according to the input. In the following two subsections, we experimentally validate each of them.

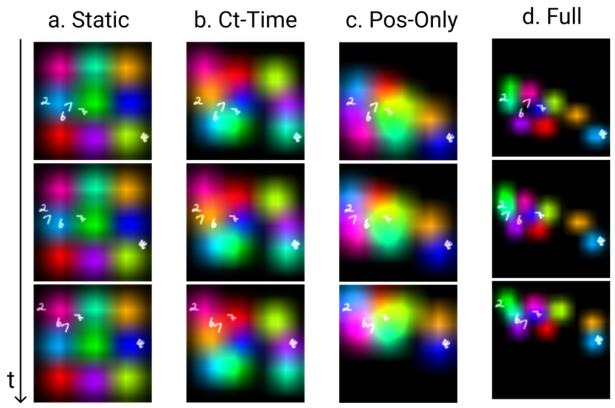


Figure 3: Visualisation of the kernel functions associated to each node, as learned by different model variants a) *Static* model, ignoring the input, learns a regular grid such that it covers the expected digits' positions over the whole dataset; b) *Constant-Time* model predicts a single set of regions covering the zones of movement in the current video; c-d) *m*odel generally follow the digits locations at each time steps while our *Full* model also adapts the regions' size.

**Dynamic Nodes Importance** We investigate different types of localised nodes, each adapting to the input to a varying degree, and show the benefits of our design choices. We keep the same graph processing and constrain the node regions in different ways.

We start by having node features extracted from regions arranged on a grid, with a fix location and size, similar to the approach used in (Nicolicioiu, Duta, and Leordeanu 2019). We refer to this experiment as *Grid* in Table 1 and Table 2 and observe that this model is capable of spatio-temporal reasoning, but achieves poorer results than when allowing the regions to move.

In order to investigate the importance of dynamic regions, we construct a model (referred to as *Static*) with node zones that are optimised from the dataset but do not take into consideration the current input. This is achieved by replacing each $\mathbf{z}_i$ features from Eq. 4 with a set of learnable parameters for all time steps, thus ignoring the input.

We also validate that the regions should be adapted to each time step, by constructing a model, denoted as *Constant-Time*, where the nodes are dynamic, but receive the same regions in all time steps. This is done by applying network $f$ in Eq. 1 on the input features temporally aggregated by mean pooling and omitting the GRU for the prediction of region parameters (since it receives a single time step).

The regions used in our DyReG model are defined by location and size, and we can either pre-determine a fixed size for all the regions, based on existing biases in data, (*Position-Only* model) or directly predict it from the input as in our complete model (*Full* model).

We observe from the previous experiments, summarised in Table 1, that the fixed region approach achieves the worst

| Model | Optimise Position | Time Variant | Dynamic Position | Dynamic Size | Accuracy |
|---|---|---|---|---|---|
| Grid | | | | | 78.85 |
| Static | ✓ | | | | 81.48 |
| Ct-time | ✓ | | ✓ | | 86.77 |
| Pos-Only | ✓ | ✓ | ✓ | | 93.41 |
| Full | ✓ | ✓ | ✓ | ✓ | 95.09 |

Table 1: Ablation study on the MultiSyncMNIST dataset showing the importance of dynamically adapting node regions to the visual content. All the models use the same graph processing and constrain the node generation in different ways. We observe that it is crucial to have regions that depend on the input, varying at each time step, while adapting their size.

| Model | Parameters (M) | Accuracy |
|---|---|---|
| 2D Oracle | - | 54.40 |
| ResNet-12 | 2.790 | 52.29 |
| Grid | 2.824 | 78.85 |
| Semantic | 2.853 | 82.41 |
| DyReG-Lite | 2.833 | 91.43 |
| DyReG | 3.081 | 95.09 |

Table 2: Results of our main model and different baselines on MultiSyncMNIST. We show that the instance-oriented node regions of our DyReG model are better suited than semantic nodes' maps obtained by the Semantic model for the current task.

results, slightly improving when the regions are allowed to change according to the learned statistics of the dataset in the Static model. Adapting to the input is shown to be beneficial, the performance improving even when the regions are invariant in time and achieving the maximum performance when both the location and the size of the regions are dynamically predicted from the input.

In Figure 3 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* model starts with bigger regions but learns to reduce their size and we observe that they closely follow the movement of the digits.

**Localised Node Importance** 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 instance nodes that are obtained from local regions biased towards individual entities.

In *Semantic* model we create nodes similar to (Chen et al. 2019) and (Liang et al. 2018). For each node, we directly predict from the features at each spatial position $p$ a weighting scalar. Node features are computed by global average pooling according to these coefficients. This is equivalent to predicting each position $\mathcal{K}_t^{(i)}(p)$ directly from the corresponding position in the input $X_{t,p}$. Thus each node extracts features from all the spatial locations and could represent a semantic concept. A disadvantage of this approach is that it does not distinguish between positions with the same features, making it harder to reason about different instances.

In order to model the instances presented in the video, we leverage our full *DyReG* model explained in Section 3. We experiment with two variants of the $g_i$ functions to obtain a lighter model (*DyReG-Lite*) comparable in terms of the number of parameters (as seen in Table 2 ) with the previous one, and a bigger, more accurate one (*DyReG*). Both models obtain nodes that are localised in space and we observe that

| Model | BB | #F | Top 1 | Top 5 |
|---|---|---|---|---|
| TRG | R50 | 16 | 59.8 | 87.4 |
| GST | R50 | 16 | 62.6 | 87.9 |
| v-DropPath | DNet121 | 16 | 62.9 | 88.0 |
| TSM | R50 | 16 | 63.4 | 88.5 |
| SmallBig | R50 | 16 | 63.8 | 88.9 |
| STM | R50 | 16 | 64.2 | **89.8** |
| DyReG - r4 | R50 | 16 | 64.3 | 88.9 |
| DyReG - r3-4-5 | R50 | 16 | **64.8** | 89.4 |

Table 3: State-of-the-art models on the validation set of Something-Something-v2. We place one instance of DyReG model in the res4 stage of the backbone or three instances placed at res3, res4 and res5 stages. We achieve improvements over the TSM backbone and obtain superior results.

both of them improve the previous approach that uses purely semantical nodes. More details about the implementation of all the presented models are found in the Appendix.

Table 2 presents the performance of the previously explained models and shows that models like Semantic or DyReG, with either type of adaptive nodes, improves upon a fixed node approach. For the current task, that involves reasoning about the position of entities in a scene, we observe that our instance-based model is more appropriate than purely semantical processing.

**Implementation details** All models share the ResNet-12 backbone with 3 stages, where the graph receives the features from the second stage and sends its output to the third stage. We use a number of $N = 9$ graph nodes and repeat the graph propagation for three iterations. In the main dynamic model $f$ from Eq. 1 is implemented as 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 Appendix. The graph offsets are initialized such that all the nodes' regions start in the center of the frame by properly setting $\alpha$ in Eq. 4. In all experiments, we use SGD optimizer with learning rate $0.001$ and momentum $0.9$, trained on a single GPU.

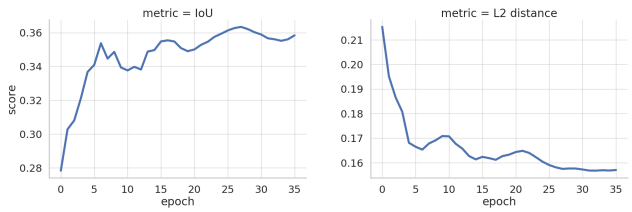


Figure 4: IoU and mean $L_2$ distance between DyReG regions and the GT boxes on validation set of Something-Something-v2. During training, it improves the scores, hinting that the model learns object-centric representations.

### 4.2 Human-Object Interactions Experiments

We experiment on a real-world dataset, Something-Something-V2 (Goyal et al. 2017) that involves classifying complex scene involving human-object interactions. It is challenging because solving it requires both temporal reasoning and modeling spatial relations between objects in the environment. It consists of 169K training videos and 25K validation videos, having 174 classes.

We use TSM-ResNet-50 (Lin, Gan, and Han 2019) as our backbone and add instances of our module at multiple stages. 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 for a small number of epochs to kick-start the optimisation process. This involves supervising the nodes' regions from the single graph models for the first $10\%$ of the training iterations then continue the learning process with only the video classification signal. Distilling the model for the first $10\%$ or $40\%$ training iterations arrives at similar results, hinting that it is useful only for fixing the optimisation problems in the beginning of training.

In Table 3 we compare to recent methods from the literature such as TRG (Zhang et al. 2020a), v-DropPath (Zhou et al. 2020), GTS (Luo and Yuille 2019), TSM, SmallBig (Li et al. 2020), STM (Jiang et al. 2019). Our method improves over the TSM backbone by $1.4\%$ in Top-1 accuracy and achieves superior results compared to all the other methods.

**Object-centric representations** The localised nodes make our model capable of discovering salient regions, leading to object-centric node representations. The nodes represent the core processing units and their localisation enforces a clear decision on what specific regions to focus on while completely ignoring the rest, as a form of hard attention. By inspecting their predicted kernels we have a better understanding of the elements influencing the model predictions, thus making the method more explainable. Visualisations of our nodes' regions (Figure 5) reveal that generally, they cover the objects in the scene.

We quantify this by measuring the mean Intersection over Union (IoU) and the mean $L_2$ distance between the predicted regions and ground truth objects given by (Materzynska et al. 2020). The metrics are completely defined in the Appendix. In Figure 4 we present the two scores and observe that they improve during the learning process hinting that the model actually learns object-centric representations. We note that we do not supervise the model in any way with information from objects but during optimisations, it obtains representations that correlate with objects.

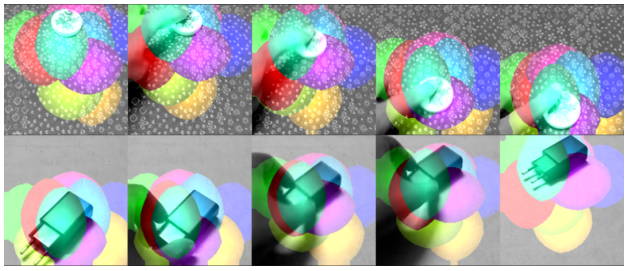

Figure 5: Visualisation of kernel functions learned by our DyReG model, on videos from Something-Something-V2 validation dataset, where each node is represented with a different color. We observe that the regions are temporally coherent and generally cover the main object instances.

**Implementation Details** In all of our experiments we follow the training setting of (Lin, Gan, and Han 2019), by using 16 frames resized such that the shorter side has size 256, and randomly sample a crop of size $224 \times 224$. We add our dynamic graph module to a TSM (Lin, Gan, and Han 2019) backbone based on ImageNet (Russakovsky et al. 2015) pre-trained ResNet-50 model. To benefit from this ImageNet pre-training, we add our graph module as a residual connection and initialize the final normalisation of the module such that it is ignored at the start of the optimisation.

For the evaluations, we follow the setting in (Lin, Gan, and Han 2019) of taking 3 spatial crops of size $256 \times 256$ 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.

The code for the entire model will be soon released.

## 5 Conclusion

In this paper we propose Dynamic Regions Graph Neural Networks (DyReG), a method to create localised nodes that complements the relational modeling of spatio-temporal data using Graph Neural Networks. We describe and analyze several contributions: 1) graph nodes mapped to dynamically predicted regions consisting of object-centric representations that favour instance interactions; 2) a differentiable pooling operation that facilitates the dynamic prediction of salient input regions, without object-level supervision. The resulting method 3) improves the explainability of the model and leads to 4) state-of-the art results in a human-object interactions classification task.

## References


Baradel, F.; Neverova, N.; Wolf, C.; Mille, J.; and Mori, G. 2018. Object Level Visual Reasoning in Videos. In *ECCV*.

Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J.; et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, 4502–4510.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* .

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral Networks and Locally Connected Networks on Graphs. *CoRR* abs/1312.6203. URL http://arxiv.org/abs/1312.6203.

Carreira, J.; and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 4724–4733. IEEE.

Chen, X.; Li, L.-J.; Fei-Fei, L.; and Gupta, A. 2018a. Iterative Visual Reasoning Beyond Convolutions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 7239–7248.

Chen, Y.; Kalantidis, Y.; Li, J.; Yan, S.; and Feng, J. 2018b. Aˆ 2-Nets: Double Attention Networks. In *Advances in Neural Information Processing Systems*, 350–359.

Chen, Y.; Rohrbach, M.; Yan, Z.; Shuicheng, Y.; Feng, J.; and Kalantidis, Y. 2019. Graph-Based Global Reasoning Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 433–442.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. 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)*, 1724–1734.

Dai, J.; He, K.; and Sun, J. 2016. Instance-Aware Semantic Segmentation via Multi-task Network Cascades. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 3150–3158.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 764–773.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2625–2634.

Fan, L.; Buch, S.; Wang, G.; Cao, R.; Zhu, Y.; Niebles, J. C.; and Fei-Fei, L. 2020. RubiksNet: Learnable 3D-Shift for Efficient Video Action Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Fu, J.; Liu, J.; Tian, H.; Li, Y.; Bao, Y.; Fang, Z.; and Lu, H. 2019. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3146–3154.

Gao, J.; Zhang, T.; and Xu, C. 2019. Graph Convolutional Tracking. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4644–4654.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1263–1272.

Girdhar, R.; Carreira, J.; Doersch, C.; and Zisserman, A. 2019. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 244–253.

Goyal, A.; Lamb, A.; Hoffmann, J.; Sodhani, S.; Levine, S.; Bengio, Y.; and Schölkopf, B. 2019. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893* .

Goyal, R.; Kahou, S. E.; Michalski, V.; Materzynska, J.; Westphal, S.; Kim, H.; Haenel, V.; Fruend, I.; Yianilos, P.; Mueller-Freitag, M.; et al. 2017. The" Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *ICCV*, volume 1, 3.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778.

Herzig, R.; Levi, E.; Xu, H.; Gao, H.; Brosh, E.; Wang, X.; Globerson, A.; and Darrell, T. 2019. Spatio-temporal action graph networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 0–0.

Jia, X.; De Brabandere, B.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, 667–675.

Jiang, B.; Wang, M.; Gan, W.; Wu, W.; and Yan, J. 2019. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2000–2009.

Kipf, T.; van der Pol, E.; and Welling, M. 2020. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations*.

Li, X.; Wang, Y.; Zhou, Z.; and Qiao, Y. 2020. SmallBigNet: Integrating Core and Contextual Views for Video Classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 1089–1098.

Li, Y.; and Gupta, A. 2018. Beyond grids: Learning graph representations for visual recognition. In *Advances in Neural Information Processing Systems*, 9225–9235.

Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; and Battaglia, P. 2018. Learning Deep Generative Models of Graphs.

Liang, X.; Hu, Z.; Zhang, H.; Lin, L.; and Xing, E. P. 2018. Symbolic Graph Reasoning Meets Convolutions. In *Advances in Neural Information Processing Systems 31*, 1853–1863.

Lin, J.; Gan, C.; and Han, S. 2019. TSM: Temporal Shift Module for Efficient Video Understanding. In *Proceedings of the IEEE International Conference on Computer Vision*.

Locatello, F.; Weissenborn, D.; Unterthiner, T.; Mahendran, A.; Heigold, G.; Uszkoreit, J.; Dosovitskiy, A.; and Kipf, T. 2020. Object-Centric Learning with Slot Attention. *arXiv preprint arXiv:2006.15055* .

Luo, C.; and Yuille, A. 2019. Grouped Spatial-Temporal Aggretation for Efficient Action Recognition. In *Proceedings of the IEEE International Conference on Computer Vision*.

Ma, C.-Y.; Chen, M.-H.; Kira, Z.; and AlRegib, G. 2018. TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *Signal Processing: Image Communication* .

Materzynska, J.; Xiao, T.; Herzig, R.; Xu, H.; Wang, X.; and Darrell, T. 2020. Something-Else: Compositional Action Recognition with Spatial-Temporal Interaction Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mavroudi, E.; Béjar, B.; and Vidal, R. 2020. Representation Learning on Visual-Symbolic Graphs for Video Understanding. In *The European Conference on Computer Vision (ECCV)*.

Nicolicioiu, A.; Duta, I.; and Leordeanu, M. 2019. Recurrent Space-time Graph Neural Networks. In *Advances in Neural Information Processing Systems 32*, 12838–12850. Curran Associates, Inc. URL http://papers.nips.cc/paper/9444-recurrent-space-time-graph-neural-networks.pdf.

Qi, S.; Wang, W.; Jia, B.; Shen, J.; and Zhu, S.-C. 2018. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 401–417.

Rahaman, N.; Goyal, A.; Gondal, M. W.; Wuthrich, M.; Bauer, S.; Sharma, Y.; Bengio, Y.; and Schölkopf, B. 2020. S2RMs: Spatially Structured Recurrent Modules. *arXiv preprint arXiv:2007.06533* .

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 211–252.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 30*, 4967–4976.

Sun, C.; Shrivastava, A.; Vondrick, C.; Murphy, K.; Sukthankar, R.; and Schmid, C. 2018. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 318–334.

Tran, D.; Wang, H.; Torresani, L.; and Feiszli, M. 2019. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 5552–5561.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 6450–6459.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Velikovi, P.; Cucurull, G.; Casanova, A.; Romero, A.; Li, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 4.

Wang, X.; and Gupta, A. 2018. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 399–417.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 305–321.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Yue-Hei Ng, J.; Hausknecht, M.; Vijayanarasimhan, S.; Vinyals, O.; Monga, R.; and Toderici, G. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4694–4702.

Zhang, J.; Shen, F.; Xu, X.; and Shen, H. 2020a. Temporal Reasoning Graph for Activity Recognition. *IEEE Transactions on Image Processing* 29: 5491–5506.

Zhang, L.; Xu, D.; Arnab, A.; and Torr, P. H. 2020b. Dynamic Graph Message Passing Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, Y.; Tokmakov, P.; Hebert, M.; and Schmid, C. 2019. A structured model for action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9975–9984.

Zhou, B.; Andonian, A.; Oliva, A.; and Torralba, A. 2018. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 803–818.

Zhou, Y.; Sun, X.; Luo, C.; Zha, Z.-J.; and Zeng, W. 2020. Spatiotemporal Fusion in 3D CNNs: A Probabilistic View. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9829–9838.

Zhu, X.; Hu, H.; Lin, S.; and Dai, J. 2019. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9308–9316.

# Appendix: Dynamic Regions Graph Neural Networks for Spatio-Temporal Reasoning

In this Appendix, we provide details about our model and some additional ablation studies. In Section A we present more technical details about how the regions are generated. In Section B we describe the dataset used in the synthetic setting (B.1), present in more detail the models used in the ablation studies from the main paper (B.2) and also show two additional ablation studies involving the size (B.3) and shape (B.4) of the nodes' kernels. In Section C we present the distillation used at the start of the training (C.1), the metrics used to measure the correlation between our regions and the existing objects in the scene (C.2) and also the runtime analysis of our proposed models (C.3).

## A 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 $X_t \in H \times W \times C$ 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 $\mathbb{R}^{16\times16\times32}$ to $\mathbb{R}^{7\times7\times16}$.

**Functions $\{g_i\}$** For each $i$ 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 $1\times1$ 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 (Vaswani et al. 2017). 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 2 of the main paper we could use the second approach for the $\{g_i\}$ functions and also skip the $f$ processing.

**Constraints** As explained in the main paper, from Equation 4 we obtain for each node $i$ a set of region parameters $\mathbf{o}_i = (\Delta x_i, \Delta y_i, w_i, h_i) = \alpha \odot W\mathbf{z}_i$. 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. We design the non-linearities such that $w_i, h_i > 0$ and $\Delta x_i + C_x \in [0, W]$ and , $\Delta y_i + C_y \in [0, H]$.

$$\tilde{h} = e^h h_{init} \qquad \tilde{w} = e^w w_{init} \tag{11}$$

$$\Delta\tilde{x} = \tanh\left(\Delta x + \frac{W}{2}\,\mathrm{arctanh}\,\frac{2C_x}{W} + \frac{W}{2}\right) - C_x \tag{12}$$

$$\Delta\tilde{y} = \tanh\left(\Delta y + \frac{H}{2}\,\mathrm{arctanh}\,\frac{2C_y}{H} + \frac{H}{2}\right) - C_y \tag{13}$$

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

## B Synthetic Setting

### B.1 Dataset details

Based on (Nicolicioiu, Duta, and Leordeanu 2019) we create MultiSyncMNIST. It consists of videos, each having 10 frames of size $128 \times 128$, 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 a subset of any size 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.

### B.2 Ablation details

We give more details about the models used in the ablation studies performed on the synthetic setting.

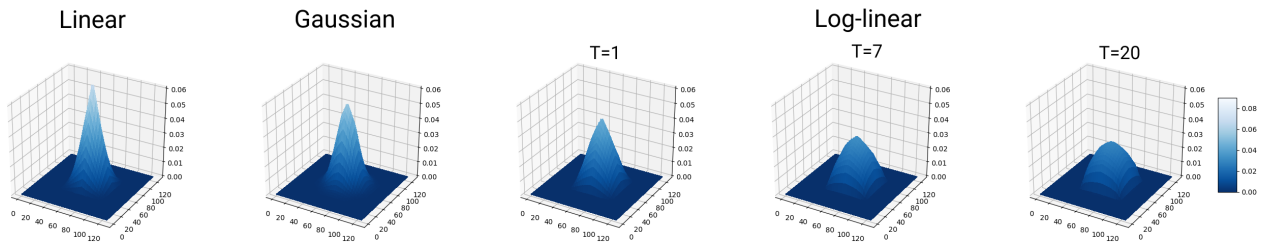


Figure 6: Visualisation of variants of the kernel functions used by the differentiable pooling mechanism to extract node features.

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

Table 4: 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.

| Linear | Gaussian | Log-linear $T = 1.0$ | Log-linear $T = 7.0$ | Log-linear $T = 20.0$ |
|---|---|---|---|---|
| 93.41 | 91.74 | 91.52 | 90.46 | 89.85 |

Table 5: Experiments on MultiSyncMNIST exploring the shape of the nodes' regions. We use different kernel function and observe that the results improve with the sharpness of the kernel as seen in Figure 6. In the main paper, we use as default the Linear kernel function.

**2D Oracle** We take the ground-truth set of digits and predict the pair formed from the smallest and the largest digits among them. This approach completely ignores the movement, acting as a single-frame upper-bound.

**2D ResNet-12 Model** This is a convolutional network as defined in torchvision library, applied independently at each frame. The resulting features are temporally aggregated by an average pooling and projected to obtain the final prediction.

All the graph models presented in this section use the ResNet backbone and add a single graph module with $N = 9$ nodes, placed at the second stage. In the following, we vary the way we create the graph nodes, keeping the same graph processing.

**Grid Model** This model keeps the nodes' regions at the same location and size as the initialization. In order to cover the whole image, the regions are arranged on a $3 \times 3$ grid.

**Static Model** In this model, the regions' locations are optimized from the whole dataset, not predicted from the current input as in our full model. This is achieved by replacing each $\mathbf{z}_i$ features from Eq. 4 in the main paper with learnable parameters. We keep the size of the regions fixed and we use a single set of parameters for all time steps to control the location of each node.

**Constant-Time Model** This model predicts the same nodes' regions at each time step, base on the whole video, without adapting to each individual frame. The size of the regions is kept fixed and the $f$ function is applied only once over the mean pooling of the features from all time steps and the GRU is omitted since it receives a single time step.

**Position-Only Model** This is similar to our full model but used to optimise only the nodes' location, keeping the size of the regions fixed to the initial value.

**DyReG (Full) Model** This is our proposed DyReG model, that predicts both location and size of the nodes' regions according to the current input, allowing different regions for each frame. We use the default setting for Node Region Generation with the convolutional network as the $f$ function and the $\{g_i\}$ functions implemented as fully connected layers, as detailed in the previous section.

**DyReG-Lite Model** This is the lighter version (in terms of parameters) of our proposed DyReG model, that skips the $f$ processing and uses a weighted average pooling for the $\{g_i\}$ functions, as explained in the previous section.

**Semantic Model** In this model, we follow an approach similar to (Chen et al. 2019) and (Liang et al. 2018) to create global nodes that are biased towards capturing semantic concepts. This is complementary to our method that is focused on instance-oriented nodes, clearly localised in the scene. For each node, the model predicts a map that is used to globally pool the input features. The map is predicted from the input features using a $1 \times 1$ convolution.

### B.3 Ablation: Regions' Size

In the previous section, we validated that adapting the size of each region according to each video leads to better results.

In this subsection, we conduct more experiments to find a fixed size that is as close as possible to the performance of the learned one. 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 4. Setting $\lambda = 8$ corresponds to regions having approximately the expected values of the regions predicted by the Full DyReG 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 observe 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 any other model and obtains poorer results. This is probably caused by two reasons. First, the regions cover a small area thus they must be more precise to cover small entities and also they could not cover large entities in their entirety. Second, due to the small receptive field of each node, the resulting gradients used to update the region parameters are noisier.

### B.4 Ablation: Kernel Shape

In our model, we pool the graph features using a kernel function $\mathcal{K}$ defined by two sets of parameters corresponding to the center and size of the kernel, normalised such that the sum of its elements is 1. For node $i$, we define a separable kernel at any position $p$ as:

$$\mathcal{K}^{(i)}(p_x, p_y) = k_x^{(i)}(p_x) k_y^{(i)}(p_y) \in \mathbb{R} \tag{14}$$

We experiment with different types of kernel functions, allowing us to change the shape of the regions and the distribution of the pooling weights. We present how the kernel is defined for one axis.

In all our main experiments we use as default the following kernel. On each axis, it decreases linearly with the distance to the center of the kernel.

$$k_x^{(i)}(p_x) = \max(0, w_i - |c_{i,x} + \Delta x_i - p_x|) \tag{15}$$

We also experiment with a Gaussian kernel with standard deviation controlled by the region size parameters $h_i, w_i$.

$$k_x^{(i)}(p_x) = \exp\Big(-\frac{(c_{i,x} + \Delta x_i - p_x)^2}{2w_i^2}\Big) \tag{16}$$

Then we tested a log-linear kernel, where the temperature $T$ controls the sharpness of the distribution.

$$k_x^{(i)}(p_x) = \log\Big(1 + T \cdot \max(0, w_i - |c_{i,x} + \Delta x_i - p_x|)\Big) \tag{17}$$

The kernels are presented in Figure 6 and we use them for training Position-Only models with results shown in Table 5. The shape of the kernel should depend on the processed entities and in this task we observe that the performance increases with the sharpness of the kernel.

## C Human-Object Interactions

### C.1 Distillation for kick-starting the optimisation

We observed that when we add three graph modules at different layers in the TSM backbone, the optimisation of the learned regions for two of them behave poorly, resulting in a model with the same accuracy as one with a single graph. By visualizing the interpretable kernels, we notice that only the one corresponding to the last module behaves well showing that indeed a single graph module is effectively used. We solve this problem by guiding the learning of the kernels at the start of the optimisation. We first learn three separate models each having a single graph module placed at a different stage, as these models do not exhibit optimisation issues. Then we distill the predicted regions of each model into a larger model containing three graph modules.

This involves supervising the nodes' region parameters of the larger model from the parameters predicted by the models having a single graph module at the corresponding layer. This distillation happens in the first few epochs of the training iterations to kick-start the learning process, then the training continues with only the video classification signal. We note that distilling for $40\%$ or $10\%$ of the training iterations leads to similar results. The performance of DyReG models, one trained with the distillation procedure and one without it is shown in Table 6. Both of them improve over the TSM backbone and by visualizing the kernels we observe that, using the distillation kick-start, the graphs from all three stages learn to adapt their regions.

| Model | Layer | Top 1 | Top 5 |
|---|---|---|---|
| TSM | - | 61.1 | 86.5 |
| DyReG | r3-r4-r5 | 62.1 | 87.4 |
| DyReG Distill | r3-r4-r5 | 62.8 | 87.7 |

Table 6: Results on Something-Something-v2 validation dataset, using a single $224 \times 224$ central crop. We observe that DyReG models improve over the TSM backbone and that it is crucial to have the kick-start given by the distillation to learn multiple dynamic graph modules.

### C.2 Object-centric metrics

Our method focuses on extracting a set of few nodes, representing salient regions in the scene. By predicting regions clearly localised in space, the nodes' features are biased towards capturing object-centric representation.

We propose two metrics to quantify to what degree the nodes cover existing ground truth (GT.) objects in the scene annotated by (Materzynska et al. 2020). First, we measure the distance between the center of the predicted regions and the center of the GT. objects. To take into account also the size of the regions we also measure the intersection over union (IoU) between them.

In the following, we describe the first one. 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.

$$Dist_p = \frac{1}{NF} \sum_{f=1}^{F} \sum_{i=1}^{N} min_j |C_i + \Delta_i - B_j|_2 \tag{18}$$

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

| Model | Frames | FLOPS | Params |
|---|---|---|---|
| I3D | 32 | 153.0G | 28.0M |
| I3D+NL | 32 | 168.0G | 35.3M |
| I3D+NL+GCN | 32 | 303.0G | 62.2M |
| TSM | 16 | 65.8G | 23.9M |
| STM | 16 | 66.5G | 24.0M |
| DyReG r4 | 16 | 66.4G | 25.7M |
| DyReG r3-4-5 | 16 | 67.4G | 28.7M |

Table 7: Comparison in terms of the number of operations and parameters for a single video of size $224 \times 224$.

$$Dist_r = \frac{1}{N_B F} \sum_{f=1}^{F} \sum_{j=1}^{N_B} min_i |C_i + \Delta_i - B_j|_2 \qquad (19)$$

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.

Similarly, we compute for each node the maximum Intersection over Union to all GT. boxes and vice versa.

### C.3 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 $224 \times 224$ in batches of 16 on a single NVIDIA GTX 11080 Ti GPU. Our TSM backbone runs at a rate of $35.7$ videos per second while DyReG-r4 runs at $34.8$ videos per second and DyReG-r3-4-5 runs at $32.7$ videos per second. In Table 7, we compare in terms of number of parameters and operations against other current standard models used in video processing: I3D-ResNet50 (Carreira and Zisserman 2017), I3D+NL (Wang et al. 2018), I3D+NL+GCN (Wang and Gupta 2018), TSM (Lin, Gan, and Han 2019), STM (Jiang et al. 2019). 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 RPN 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 precise object proposals given by external models.