Dynamic Regions Graph Neural Networks for Spatio-Temporal Reasoning

Iulia Duta*
Bitdefender, Romania
iduta@bitdefender.com

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 occurring in the spatio-temporal domain (e.g. videos) but when an explicit structure is not available, it is not obvious what atomic elements should be represented as nodes.

For video processing, we design nodes that are clearly localised in space, with an inductive bias for modeling the relations between instances. Current works are using external object detectors or fixed regions to extract graph nodes, while we propose a module for generating the regions associated with each node dynamically, without explicit object-level supervision.

Constructing these localised, adaptive nodes gives our model a bias 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

Current works applying Graph Neural Networks (GNNs) in visual domain lean towards relating concepts in a semantic space [1–5] while others focus on the interactions between instances. One approach suitable for instance modeling is to create graph nodes from convolutional points or fixed regions [6–10] while in another one nodes are associated with objects given by pre-trained external detectors [11–19]. We propose a GNN method, focusing on the creation of *instance-oriented nodes* to relate entities in the scene using only the classification signal, without relying on object-level supervision. We refer as entity any localised visual unit representing an entire or a part of an instance.

Object-centric representations improve the learning capabilities of visual models [5] 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 [20], thus the learning process should lead to node representations that best fulfil these requirements. We validate experimentally that the predicted kernels correlate with object locations.

Our method predicts, based on the input, for each node the location and size for an associated region and extract the node features using a pooling differentiable w.r.t. these coordinates. The model learns

*Equal contribution.

NeurIPS 2020 Workshop on Object Representations for Learning and Reasoning, Vancouver, Canada.

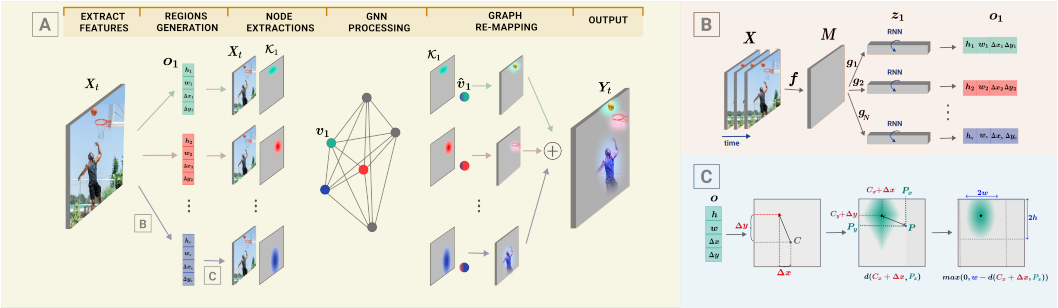

Figure 1: Our DyReG model that extracts localised nodes, useful for relational processing (Fig. A). From X_t, for each node i we extract parameters $\mathbf{o}_i$ denoting the location and size of a region (Fig. B). They define a kernel K_i, used to extract the localised features $\mathbf{v}_i$ from the corresponding region of X_t (Fig. C). We process the nodes with a spatio-temporal GNN and project each node $\hat{\mathbf{v}}_i$ into its initial location, according to the kernel K_i.

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, *unsupervised* at the object-level, that can *discover salient regions* and we quantitatively show that they correlate with objects locations. 2) Our model creates *localised* graph nodes that *improves the relational processing* and obtains superior results on video classification tasks where instances play a key role. 3) Localising the nodes gives a form of hard attention leading to a more *explainable* model.

2 Dynamic Regions GNNs

We investigate how to create node representations 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 DyReG model (shown in Fig. 1) receives feature volume $X \in \mathbb{R}^{T\times H\times W\times C}$ and at each time step t a kernel function is used to estimate N 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. Our method relates a few object-centric nodes so it is crucial to assign them to the salient regions. A global processing (see Fig. 1 B) 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 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, $M_t = f(X_t) \in \mathbb{R}^{H'\times W'\times C'}$. Each g_i has different parameters for each region i and could be instantiated as an MLP or as global pooling enriched with spatial positional information. $\hat{\mathbf{m}}_{i,t} = g_i(M_t) \in \mathbb{R}^{C'}, \forall i \in \overline{1,N}$. A GRU [21] is applied independently for each region and the final parameters are modulated by a set of parameters α to control the initialisation.

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

$$\mathbf{o}_{i,t} = (\Delta x_{i,t}, \Delta y_{i,t}, w_{i,t}, h_{i,t}) = \alpha \odot W\mathbf{z}_{i,t} \in \mathbb{R}^4 \tag{2}$$

3.2 Node Features Extraction. In the following, we ignore the time index for clarity. The feature of the region i are extracted by a differentiable pooling w.r.t. the predicted region parameters $\mathbf{o}_i$. All locations $p \in \mathbb{R}^2$ are interpolated by the kernel $\mathcal{K}^{(i)}(p)$ as shown in Fig. 1 C. The kernel is separable $\mathcal{K}^{(i)}(p_x, p_y) = k_x^{(i)}(p_x)k_y^{(i)}(p_y) \in \mathbb{R}$, thus we present a single axis.

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 Δx_i are the predicted parameters from Eq. 2.

$$k_x^{(i)}(p_x) = \max(0, w_i - d(c_{i,x} + \Delta x_i, p_x)) \tag{3}$$

Model	Opt Reg	Time Var	Dynamic Pos	Dynamic Size	Acc
Fixed					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.

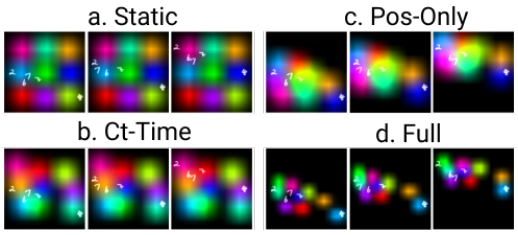

Figure 2: Visualisation of the kernels learned by model variants on MultiSyncMNIST.

The features for node i are obtained by interpolating using the kernel function all the points in the input X_t, for each time step t. By modifying $(\Delta x_i, \Delta y_i)$ and (h_i, w_i) the network controls the location and the size of the regions respectively. $\mathbf{v}_{i,t} = \sum_{p_x=1}^{W} \sum_{p_y=1}^{H} \mathcal{K}^{(i)}(p_x, p_y)\mathbf{x}_{t,p_x,p_y} \in \mathbb{R}^C$.

The position of the nodes' regions provide an identity for each node and could help the relational processing thus we compute a positional embedding by linear projecting the kernel $\mathcal{K}_i$. Setting $w_i = 1$ leads to standard bilinear interpolation, but optimising it allows the model to adapt the size of each region and larger regions results in a more stable optimisation.

3.3 Graph Processing. For processing the node features, different GNNs could be used. Generally, they follow a framework [20, 22] of sending messages between connected nodes, aggregating and updating them. The specific message-passing mechanism is not the focus of the current work, thus we follow a general formulation similar to [10] 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 $a(v_i, v_j) \in \mathbb{R}$.

$$\mathbf{v}_{i,t} = \sum_{j=1}^{N} a(\mathbf{v}_{j,t}, \mathbf{v}_{i,t}) \text{MLP}(\mathbf{v}_{j,t}, \mathbf{v}_{i,t}) \in \mathbb{R}^C \tag{4}$$

We incorporate temporal information through a recurrent function across time, applied independently for each node: $\hat{\mathbf{v}}_{i,t+1} = \text{GRU}(\hat{\mathbf{v}}_{i,t}, \mathbf{v}_{i,t}) \in \mathbb{R}^C$. 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, to further benefit from complementary local processing, such as convolutional layers. The features of each node are sent to all locations in the input according to the weights used in the initial pooling from Section 2, distributing their features into a local region defined by the initial kernel : $\mathbf{y}_{p_x,p_y,t} = \sum_{i=1}^{N} \mathcal{K}_t^{(i)}(p_x, p_y)\hat{\mathbf{v}}_{i,t} \in \mathbb{R}^C$ obtaining the final output of our module.

3 Experiments

4.1 Synthetic Experiments. We use a variant (MultiSyncMNIST) of the video dataset from [10] with 5 moving digits where a classification task is defined as to identify the subset of digits that move in the same way. The task is challenging as it requires to distinguish and relate multiple instances.

A 2D ResNet-18 [23] is used as a backbone for multiple variants of our graph module. We argue that nodes should *dynamically adapt* according to the input and validate this experimentally (results in Table 1). 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 (presented in Section 2) and constrain the node regions in different ways. **Fixed Model** extracts node features from regions arranged on a grid, with a fix location and size, similar to the approach used in [10]. **Static Model** investigates the importance of dynamic regions by optimising regions based on the whole dataset but do not take into consideration the current input. Effectively, the features $\mathbf{z}_i$ from Eq. 2 become learnable parameters, thus ignoring the input. **Constant-Time Model** has regions adapted to the current video but they do not change in time and the results validate that the regions should be adjusted to each time step. **DyReG Model** predicts regions that 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**).

	Model	Top 1	Top 5
non-Graph	GST [24]	62.6	87.9
	TSM [25]	63.4	88.5
	SmallBig [26]	63.8	88.9
	STM [27]	64.2	89.8
Graph	TRG [28]	59.8	87.4
	DyReG - r4	64.3	88.9
	DyReG - r3-4-5	64.8	89.4

Table 2: Results on Something-Something-V2.

Model	mean IoU
Detector	47.1
DyReG	39.0
Center	27.8
Grid	26.8

Table 3: IoU comparison between our best DyReG model, a supervised detector and simple baselines.

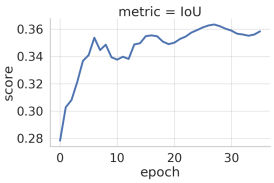

Figure 3: IoU between our DyReG regions and the GT boxes on Smt-Smt-V2.

	Model	Top 1	Top 5
non-Graph	TSM [25]	48.4	78.1
	GST [24]	48.6	77.9
	SmallBig [26]	50.0	79.8
	STM [27]	50.7	80.4
Graph	ORN [12]	36.0	-
	NL I3D [11]	44.4	76.0
	NL GCN [11]	46.1	76.8
	TRG [28]	48.1	80.4
	RSTG [10]	49.2	78.8
	DyReG - r3-4-5	49.9	79.0

Table 4: Results on Something-Something-V1.

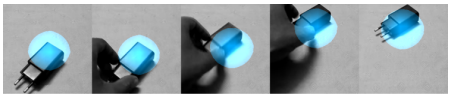

Figure 4: Visualisation of a single predicted kernel on Something-Something-V2.

These experiments, summarised in Table 1, show that the fixed region approach achieves the worst 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 Fig. 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* model starts with bigger regions but learns to reduce their size and we observe that they closely follow the movement of the digits.

We present additional ablation experiments in the Appendix, showing the importance of localised as opposed to semantic nodes and investigate the influence of the regions size and shape.

4.2 Human-Object Interactions Experiments. We experiment on real-world datasets, Something-Something V1 and V2 [29] that classify complex scene involving human-object interactions and requires spatial and temporal reasoning between instances.

We use TSM-ResNet-50 [25] as our backbone and add instances of our module at multiple stages (r3-4-5). We noticed that models using multiple graphs have problems learning to adapt the regions from certain stages. 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.

For testing we follow the setting in [25] of taking 3 spatial crops of size 256×256 with 2 temporal samplings. Our method improves over the TSM backbone by 1.4% (Table 2) and 1.5% (Table 4) respectively in Top-1 accuracy and achieves superior results compared to all the graph based methods.

4.3 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. As seen in Fig. 4, the regions generally cover objects in the scene.

We quantify this by measuring the mean Intersection over Union (IoU) between the predicted regions and ground truth objects given by [18]. We compare the regions given by our best DyReG model to

the $N = 9$ most confident boxes of an external Faster-RCNN and 2 simple baselines: regions that are kept fixed on a grid or in the center of the each frame.

The evaluation proves that the proposed regions are relevant and correlate with objects. In Fig. 3 we present the score and observe that it improves during the learning process hinting that the model actually learns object-centric representations. Although DyReG regions are not as accurate as a trained detector 3, this quantitatively shown that DyReG regions correlate with objects. Contrary to the detector that is supervised to detect objects, our model obtains regions without any object-level supervision while training for video classification.

4 Conclusion

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) a method that facilitates the dynamic prediction of salient input regions, without object-level supervision; 2) graph nodes mapped to dynamically predicted regions consisting of object-centric representations that favour instance interactions. The resulting method 3) augments the relational processing leading to results at state-of-the art level in a human-object interactions classification task and 4) improves the explainability of the model.

Acknowledgment We would like to thank Florin Brad, Elena Burceanu and Florin Gogianu for their valuable feedback on earlier drafts of this work. This work has been supported in part by Bitdefender and UEFISCDI, through projects EEA-RO-2018-0496 and TE-2016-2182

References

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

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

This Appendix provides 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 study from the main paper (B.2) and also show four additional ablation studies illustrating the importance of localised as opposed to semantic nodes (B.3, a comparison between our generated regions and GT. boxes (B.4), the influence of the regions size (B.5) and shape (B.6). We also provide some implementation details for the synthetic setting. (B.7).

In Section C we present more details about the human-object interactions experiments. We give additional details about the distillation used at the start of the training (C.1), the metric used to measure the correlation between our regions and the existing objects in the scene (C.2), implementation details C.3 and also the runtime analysis of our proposed models (C.4).

In Section D we present an extended related work.

We will publicly release the code for our entire method upon acceptance.

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 1×1 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 [30]. 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 7,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 2 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 $w_i, h_i > 0$, $\Delta x_i + C_x \in [0, W]$ and $\Delta y_i + C_y \in [0, H]$.

$$\tilde{h} = e^h h_{init} \qquad \tilde{w} = e^w w_{init} \tag{5}$$

$$\Delta\tilde{x} = \frac{W}{2}\tanh\Big(\Delta x + \operatorname{arctanh}\big(\frac{2C_x}{W} - 1\big)\Big) + \frac{W}{2} - C_x \tag{6}$$

$$\Delta\tilde{y} = \frac{H}{2}\tanh\Big(\Delta y + \operatorname{arctanh}\big(\frac{2C_y}{H} - 1\big)\Big) + \frac{H}{2} - C_y \tag{7}$$

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 [10] we create MultiSyncMNIST. It consists of videos, each having 10 frames 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 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.

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-18 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.

Fixed 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 3×3 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 Equation2 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.

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 5: 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 6: 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.

Model	Params (M)	Acc
2D Oracle	-	54.40
DyReG-GT. Boxes	2.824	97.30
ResNet-18	2.790	52.29
DyReG-Fixed	2.824	78.85
DyReG-Semantic	2.853	82.41
DyReG-Lite	2.833	91.43
DyReG (Full)	3.081	95.09

Table 7: 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.

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 [1] and [3] 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 1×1 convolution.

B.3 Ablation: 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 with a bias for individual entities.

In *Semantic* model we create nodes similar to [1] and [3]. 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 2. 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 7) with the previous one, and a bigger, more accurate one (*DyReG*). Both models obtain nodes that are localised in space and we observe that both of them improve the previous approach that uses purely semantical nodes.

Table 7 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.

B.4 Ablation: Ground-Truth Boxes

To evaluate the quality of our proposed regions, we train our model using ground-truth boxes instead of generated regions. This acts as an oracle in terms of capturing the instances and gives an upper

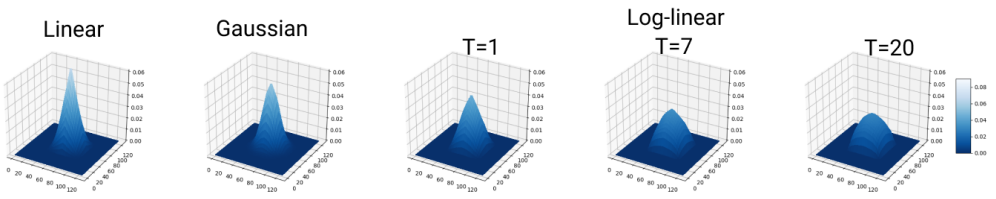

Figure 5: Visualisation of variants of the kernel functions used by the differentiable pooling mechanism to extract node features.

bound for our method. We compare this model to our DyReG ablations in Table 7. Our full DyReG model obtains close results to the oracle model, proving the utility of the node generation.

B.5 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 5. 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.6 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:

$$\mathcal{K}^{(i)}(p_x, p_y) = k_x^{(i)}(p_x) k_y^{(i)}(p_y) \in \mathbb{R} \tag{8}$$

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.

$$k_x^{(i)}(p_x) = \max(0, w_i - |c_{i,x} + \Delta x_i - p_x|) \tag{9}$$

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

$$k_x^{(i)}(p_x) = \exp\Big(-\frac{(c_{i,x} + \Delta x_i - p_x)^2}{2w_i^2}\Big) \tag{10}$$

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

$$k_x^{(i)}(p_x) = \log\Big(1 + T \cdot \max(0, w_i - |c_{i,x} + \Delta x_i - p_x|)\Big) \tag{11}$$

The kernels are presented in Figure 6 and we use them for training Position-Only models with results shown in Table 6. 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.

B.7 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 a number of $N = 9$ graph nodes and repeat the graph propagation for three iterations. The graph offsets are initialized such that all the nodes' regions start in the center of the frame by properly setting α in Eq. 2. In all experiments, we use SGD optimizer with learning rate 0.001 and momentum 0.9, trained on a single GPU.

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 8: Results on Something-Something-V2 validation dataset, using a single 224×224 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 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 8. 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.

C.2 Object-centric metric

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 a metric to quantify to what degree the nodes cover existing ground truth (GT.) objects in the scene, as annotated by [18]. We measure the intersection over union (IoU) between the predicted regions and the GT. objects boxes. For each node region in each frame, we compute the maximum IoU to all GT. object bounding boxes and average all of them.

$$\text{IoU}_p = \frac{1}{NF} \sum_{f=1}^{F} \sum_{i=1}^{N} max_j \text{IoU}(R_i, B_j) \tag{12}$$

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

$$\text{IoU}_r = \frac{1}{N_B F} \sum_{f=1}^{F} \sum_{j=1}^{N_B} max_i \text{IoU}(R_i, B_j) \tag{13}$$

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, R_i represents the bounding box associated with the i-th node's region and B_j the bounding box 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.

C.3 Implementation Details

In all of our experiments we follow the training setting of [25], by using 16 frames resized such that the shorter side has size 256, and randomly sample a crop of size 224×224. We add our dynamic graph module to a TSM [25] backbone based on ImageNet [31] pre-trained ResNet-50 model.

Model	Frames	FLOPS	Params
I3D [32]	32	153.0G	28.0M
I3D+NL [7]	32	168.0G	35.3M
I3D+NL+GCN [11]	32	303.0G	62.2M
TSM [25]	16	65.8G	23.9M
STM [27]	16	66.5G	24.0M
DyReG r4	16	66.4G	25.7M
DyReG r3-4-5	16	67.4G	28.7M

Table 9: Comparison in terms of the number of operations and parameters for a single video of size 224×224.

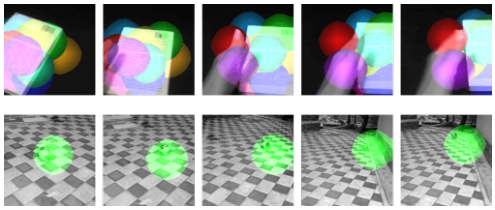

Figure 6: Additional visualisations of kernel functions learned by our DyReG model, on videos from Smt-Smt-V2 validation dataset, where each node is represented with a different color.

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 [25] 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.

C.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. 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 9, 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 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.

D Related work

In this section we present more thoroughly some related work and how our methods relates to them.

D.1 Graph Neural Networks

Graph neural networks have been recently used in many domains where the data has a non-uniform structure [22, 33–35]. In vision tasks, it is important to model the relations between different entities appearing in the scene [12, 14] and Graph Neural Networks have strong inductive biases towards relations [20], 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 [6–9], while in the second one nodes are associated with objects given by pre-trained external detectors [11, 16–18]. 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 [6] where they process pairs of features from every location, to capture distant interactions in the scene. The Non-Local [7] method creates graph's nodes from every point in the convolutional features and uses self-attention [30] mechanism to achieve long-range connections. Following, [10] 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 [11], 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 [13, 36] or by introducing propagation over class or concept embeddings [15, 19].

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 all the boats in a scene 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 [1, 2, 4, 5] 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 [1] the nodes assignments are predicted from the input, while in [2] the associations between input and nodes are made by a soft clusterization. The work of [5] is able to discover different representation groups by using an iterative clusterization based on self-attention similarity. In [3] 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 an inductive bias for modeling instance interactions.

Concurrently, the method [37] 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 object detector but are also unable to adapt to the requirements of the current task, being limited to the set of pre-defined 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 [38] while also pooling using bilinear interpolation [39–41].

These two types of interactions are captured in the dual attention model [42] by using spatial attention for instance relations and channel attention for semantic ones.

D.2 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 [43], 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 [44, 45] predict an offset for each position based on the input. Similar, [46] 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 [47] and [37].

D.3 Activity Recognition

Video classification has been influenced by methods designed for 2D images [48–51]. More powerful 3D convolutional networks, inflated from their 2D counterpart, have been later proposed [32], while other methods factorise the 3D convolutions [52–54] bringing both computational speed and accuracy. Methods like TSM [25] and [55] showed that a simple shift in the convolutional features results in improved accuracy at low computational budget.