Rethinking Scene Graphs for Action Recognition

Mathieu Riand, Patrick Le Callet, Laurent Dollé

Rethinking Scene Graphs for Action Recognition

Mathieu Riand[1,3], Patrick Le Callet[1,2]
[1]*Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004*
F-44000 Nantes, France
[2]*Institut Universitaire de France*
mathieu.riand@univ-nantes.fr, patrick.le-callet@univ-nantes.fr

Laurent Dollé[3]
[3]*CEA*
CEA Tech Pays de la Loire
F-44340 Bouguenais, France
laurent.dolle@cea.fr

***Abstract*—Over the last years, Graph Neural Networks (GNNs) have been widely used in a variety of applications, including action recognition. Scene graphs are extracted from videos and fed to a GNN in order to predict the action represented. However, in previous works, choices regarding the design of such scene graphs are often arbitrary; for instance, directed temporal edges are added without giving the GNN the capacity to use this information. In this work, we rethink the way scene graphs are built, taking inspiration from line graphs in order to propose a new design that can be applied to any type of human activity. We perform our experiments on 2 datasets and show that adapting our GNN so that it can make use of temporal edges improves its precision up to 7.5% for action recognition. We also show that adopting our alternate design for scene graphs further improves performance by an additional 14%, bringing new perspectives to this field.**

***Index Terms*—Action recognition, scene graphs, graph neural network**

I. Introduction

Recently, Graph Neural Networks (GNNs) [1] and graph representations in general have seen a growing interest from the scientific community to solve a wide range of problems: from chemical property prediction for molecules or proteins [2] to link prediction in social networks [3], graphs have the capacity to represent very complex information in a symbolic way, allowing networks to capture features otherwise hard to extract from raw data.

Action recognition [4] can also be solved using graph representations, by considering objects and hands in a video as the nodes of a **scene graph**, in which edges represent symbolic spatial relations between objects. Those graphs have the advantage of being easily interpreted, since they are a direct transcription of the scene. However, they require a certain level of knowledge about the scene in order to use the right set of spatial relations. Time is often represented through **temporal edges**; they are simple directed edges added in order to link the same instance of an object across the duration of a video, as in [5].

In this paper, we propose more general scene graphs where we use the raw 3D position of objects instead of symbolic relations. We also argue that the addition of temporal edges in scene graphs can be useful if and only if the GNN used to classify them is able to differentiate between different types of edges, which is not the case in [5] for instance. Furthermore, we think that this field of research would benefit from using scene graphs that are better suited to the capacities of GNNs; in this paper, we propose a design inspired from **line graphs** where each node contains two objects and their relative positions. This simplifies our GNN and its training since it is no longer needed to use spatial relations on edges.

To sum up, in this work:

- We show that the inclusion of edge types in order to differentiate spatial and temporal edges, paired with a GNN able to grasp this information, brings improvement to the action classification precision by up to 7.5%.
- We propose a new design for scene graph, inspired from line graphs, and show that it greatly improves the generalization power of our GNN when classifiying actions, bringing us to state-of-the-art performances without using expert information such as symbolic spatial relations.

We perform these experiments on the Bimanual Actions dataset [5] and also on a personal dataset containing mainly manipulation tasks on a conveyor.

II. Related works

Many approaches have been proposed over decades to tackle action recognition in computer vision [4], evolving with the parallel advances in deep learning. We present only methods that used graphs for action recognition; for a broad review of the domain, we refer the reader to [6].

A family of works considers that a large part of the needed information to predict the performed action is carried by the person. Thus, those methods extract the human skeleton and train networks on this information to perform action recognition. In [7] and [8] the authors represent sequences of skeleton data as a 3D information that is then fed to a CNN, along with the motion information of the skeleton (computed the same way as the optical flow for images), to predict the performed action. Other works like [9] convert skeleton sequences into RGB images to use CNNs originally built to process images, such as YOLO [10], in order to process those newly formed skeleton images. Lastly, [11] and [12] use the fact that skeletons can be naturally represented as graphs, where body joints are vertices and bones are edges, and train GNNs to perform action recognition.

However, for some actions, only having the skeleton information can be limiting; this is why works such as [5] included objects in their graph representation of the scenes

This work was supported by Conseil Régional des Pays de la Loire.

along with important body joints, mainly the hands. Such graphs containing objects, humans, and their relations are called scene graphs. However, the relations they contain are often specific to the use-case, and cannot be generalized to other works; for instance, in [13] objects can only be in contact, in front of each other or not touching. In [14], authors are using very specific contact relations, such as "sat on" or "eating", which are already actions that need to be recognized.

In [5] and [11], directed edges, or temporal edges, allow to link similar joints or objects across time; this is a way of including temporal information initially contained in the videos directly into the graphs. However, we believe that adding temporal edges as simple directed edges makes it harder for the GNN to use this information; in this work, we explore ways of adding temporal edges in a more meaningful way in our graphs, notably by adapting the convolutional layer of our GNN.

On top of this, classical scene graphs are modeled directly on the video; in [5] for instance, an object is represented by one node. It is possible to "paste" the graph over the frame it corresponds to, which makes it very interpretable; but this limits the amount of different graphs that could be generated from one scene, and perhaps this formalism is not the best adapted to GNNs. This is why in this work we propose to start exploring new designs for scene graphs in order to find the ones bringing good performance on action recognition.

III. Method

The scene graphs we propose share similarities with the ones used in [5], as seen in Figure 1. Nodes represent objects and human hands involved in the task; compared to previous works, we separate actors (namely hands) from targets (manipulated objects). This allows to also differentiate between edges linking an actor to a target, which we call **actor-target inter-edges**, and edges connecting two actors or two targets together, called **intra-edges**. In our past experiments, we noticed that removing intra-edges boosts the performance of our GNN, so we do not include them in this work. Each node contains the position and the class of the object it represent; as in [5], we also create our scene graphs by concatenating ten successive frames so that we can capture the movement of each object.

For each scene graph we generate 6 variants, each having a different way to include the temporal dimension of our videos. We first propose to add a timestamp on the nodes, representing the frame number at which the object is detected; this is a direct way to represent time in the graph since objects can now be followed using this value in ascending order.

As in [5], we also introduce directed temporal edges (see Figure 2) connecting nodes corresponding to the same object or hand instance across time. However, as mentionned in the introduction, we also add a flag on these edges in order to differentiate them from the others; in practice, it is simply a value that is read by our GNN and that allows it to learn separately on spatial and temporal edges. In short, each graph can have a timestamp on each node, temporal edges, and edge types only when temporal edges are used, which totals to 6 variants.

Finally, we propose an alternative to classical scene graphs, which we call **line scene graphs**; we take inspiration from line graphs in the sense that what was previously an edge representing the relation between two objects is now a node containing both objects, encoded as a 3D vector representing the distance between them $(\Delta x, \Delta y, \Delta z)$, and their classes (hand, bowl, etc). They also feature a temporal marker: its value is 0 if the two objects are at the same instant (formerly a spatial edge) and 1 if they are consecutive in time (formerly a temporal edge). Their design is summarized by Figure 3.

We generate our graphs directly from ground truth when available; we do this in order to have access to the position of all objects with the least amount of noise possible, so that our results depend only on the graph design we chose.

For the training model, we inspire from the one proposed in [15] (see Figure 4), consisting of 4 graph convolutional layers whose outputs are aggregated using a sum pooling operation, followed by 2 fully-connected layers and a final linear layer to output classification scores. We changed the convolutional layers to match the information available on our edges depending on the graph design.

IV. Experiments

A. Datasets

As mentionned in the introduction, we perform our experiments on two datasets to make more general observations on scene graph design choices. We first use the Bimanual Actions dataset [5] (see Figure 1, containing 540 videos of 6 different actors performing 9 tasks, ranging from cooking to hammering. These demonstrations contain few objects, which makes for very sparse scene graphs.

This is why we also perform our experiments on a dataset that we recorded (see Figure 2), which contains 760 videos of 19 actors performing 8 different manipulation scenarios. The tasks presented in this dataset are simpler, but they contain a lot of objects (sometimes more than 20) which allows to create

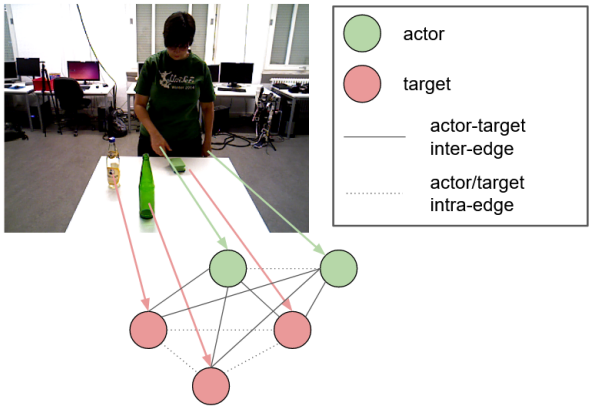

Fig. 1. Extracting a scene graph from a frame: we make the difference between actors and targets for the nodes involved in the action.

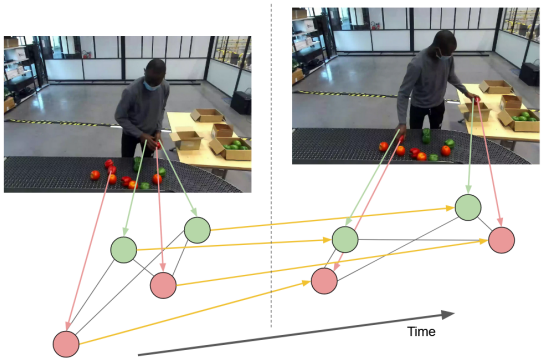

Fig. 2. Including temporal edges in scene graphs (represented as yellow directed edges).

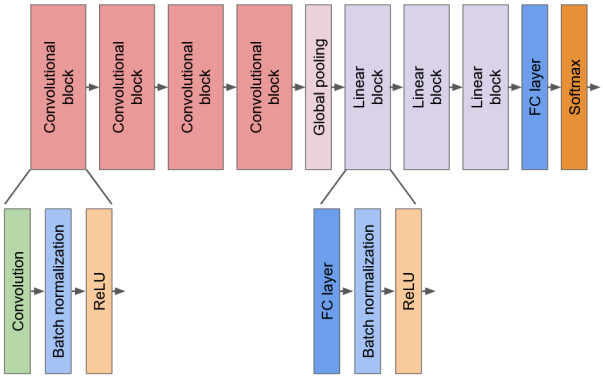

Fig. 4. The GNN used for action classification.

wider scene graphs; additionally, more variants of each action are available since the number of actors is much higher.

B. Training parameters

For each experiment, the maximum number of epochs is set to 200, and the learning rate to 10e-3. We also perform early stopping to avoid overfitting; we stop the training after 20 epochs without any improvement on the validation loss. For the scene graphs extracted from Bimanual Actions [5], we performed leave-one-subject-out cross-validation, each time using 4 actors for the training set, one for the validation set and the last one as test set; the results are then averaged over the 6 folds. For our dataset, we also performed a cross-validation, this time simply using a 60/20/20 split for training/validation/test. In each experiment we tried to predict the action performed by the right hand.

C. Representing the time dimension

In this experiment, we changed the convolutional layers of our network for RGCNConv [16], which allows the inclusion of edge types; when training, RGCNConv learns separate transform functions for each type of edge. For the cases where we did not want to separate between spatial and temporal edges, all edges were set to the same unique type. We tried every combination of the different ways to represent the time dimension that we proposed in Section III, namely adding a timestamp on the nodes, adding temporal edges and including edge types. Those are denoted in the following tables as "Node feat.", "Temp. edges" and "Type" respectively.

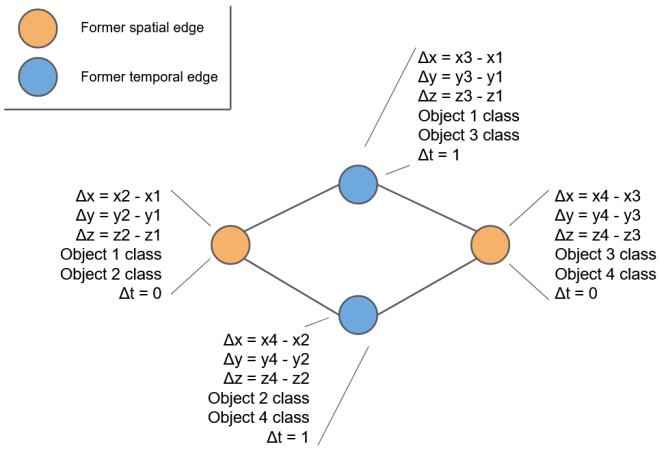

Fig. 3. The design of our line scene graphs.

First, Table I displays all results from this experiment applied to the Bimanual Actions dataset; we give precision, recall and F1-score macro-averaged over all action classes; best performances are highlighted in bold and worst ones in italic. We note that the addition of timestamps increases the precision of the network by approximately 4% when edge types are not used. Adding edge types further increases the precision by up to 7.5%.

From those results, we show first that timestamps on objects' features are used by the network to grasp the temporal information contained in the graph. Another interesting conclusion is that adding temporal edges without specifying any edge type doesn't seem to yield significant improvements on our target task; it can even damage the overall performance of the network. However, when using edge types (spatial and temporal) the quality of the action prediction seems to improve greatly. This supports our assumption that in order for the network to grasp the usefulness of temporal edges, they need to be treated separately. Finally, the time stamp information seems to be of no importance when temporal edges with a specific edge type are used, meaning that those edges alone carry enough information to grasp the temporal nature of the graphs.

Next, we perform the same experiment on our dataset; our results are shown in Table II. This time, we can see that the addition of temporal edges actually deteriorates the performance of our GNN on action recognition; including edge types only helps regaining what has been lost by the inclusion of the temporal edges. Furthermore, using timestamps on node features doesn't help the network either.

Those results make us conclude that the inclusion of temporal edges in scene graphs should not be systematic, as it often is in the literature. In some cases, such as in our dataset, they do not bring useful information; this brings us to think

TABLE I
ACTION RECOGNITION PERFORMANCE ON [5] FOR DIFFERENT TIME REPRESENTATIONS.

Graph design			Top prediction		
Node feat.	Temp. edges	Type	Prec.	Rec.	F1
X	X	X	*0.3949*	*0.3657*	0.3368
✓	X	X	0.4347	0.4249	0.3628
X	✓	X	0.3965	0.3700	*0.3293*
✓	✓	X	0.4437	0.4139	0.3665
X	✓	✓	0.4731	0.4355	0.3901
✓	✓	✓	**0.4732**	**0.4547**	**0.4067**
[5] - object positions			/	/	0.31
[5] - symbolic relations			0.63	0.63	0.63

TABLE II
ACTION RECOGNITION PERFORMANCE ON OUR DATASET FOR DIFFERENT TIME REPRESENTATIONS.

Graph design			Top prediction		
Node feat.	Temp. edges	Type	Prec.	Rec.	F1
X	X	X	**0.3510**	0.3301	**0.3031**
✓	X	X	0.3311	0.3136	0.2766
X	✓	X	0.3128	*0.3044*	*0.2739*
✓	✓	X	*0.3108*	0.3155	0.2772
X	✓	✓	0.3418	0.3124	0.2891
✓	✓	✓	0.3494	**0.3383**	0.2924

that maybe classical scene graphs are not always an optimal solution to represent videos, and that we must propose new ways to represent our scenes.

D. Line scene graphs

In [5], edges carry symbolic values representing spatial relations between nodes; this results in simpler graphs and in better performance on action recognition. However, this is a restrictive representation of the environment; in this last experiment, we generated graphs where spatial edges carry the euclidean distance between connected objects. To be able to use edge features, we again changed the convolutional layer in our GNN; NNConv [17], for instance, can take edge attributes as an additional input.

Macro-averaged results are presented in Table III; note that here the training has been done only on one fold. The precision of the network making use of the euclidean distance is 11.2% lower than the base one using no edge features.

We can see that the model having access to the euclidean distance performed worse. The overall precision of the network on action recognition is lower, and training time is much higher. Adding low-level information on edges seems to not be beneficial to the model; however it is difficult to affirm this since different layers were used for the two setups, so the difference in performance might as well come from NNConv itself.

The use of NNConv to process edge features is computationally expensive; we argue that this comes from the fact that scene graphs are currently representing the scene in a non-optimal way. For the distance to be easily used by a GNN, it should be contained on the nodes, which is why we proposed our line scene graphs in Section III.

TABLE III
EDGE FEATURES INFLUENCE ON ACTION RECOGNITION (BIMANUAL ACTIONS DATASET).

Information level on edges	Prec.	Rec.	F1	Training duration (s)
No edge features	0.7306	0.7333	0.7147	343
Euclidean distance	0.6181	0.5279	0.4949	1587

TABLE IV
COMPARING THE ACCURACY OF THE SAME GNN ON CLASSICAL SCENE GRAPHS AND OUR PROPOSED LINE SCENE GRAPHS (BIMANUAL ACTIONS DATASET).

Graph type	Accuracy		
	Training set	Validation set	Test set
Classical scene graphs	0.817	0.497	0.470
Line scene graphs	0.817	0.630	0.612

We use the same model as before with GCNConv [18] as our convolutional layer since we no longer have edge types or edge features, and train it on the dataset from [5] using our new "line" scene graphs. For a fair comparison, we also retrain a network on classical graphs, using the same data split. Table IV shows our results averaged over 4 trainings.

We can see that the accuracy is improved by more than 13% when using our line scene graphs, which is very close the performance reported in [5] (63% accuracy). However, we use no symbolic relations on edges, meaning that our graphs can be applied to any other use case, such as the demonstrations from our own dataset.

Additionally, the accuracy obtained on the training set is the same for both designs: this shows that line scene graphs have a much higher generalization power when exposed to new data. When analyzing the confusion matrix of our GNN, we noticed that our line scene graphs also allowed to detect rare classes that were previously never recognized using classical scene graphs.

V. CONCLUSION

In this work we have shown that it is possible to achieve good performance for action recognition using generic scene graphs while maintaining their interpretability. By rethinking the way scene graphs are usually constructed, we achieve performances comparable to the ones obtained using symbolic graphs, without the extra step of detecting those symbolic relations.

On top of this, we proved that the addition of temporal edges in classical scene graphs must be done using edge types, so that the GNN can treat temporal and spatial edges separately; including them this way improved action recognition accuracy by up to 7.5%.

In a future work, we would like to share our results on applying self-supervised learning with scene graphs in order to further improve the generalization power of our GNN, as well as our dataset of video demonstrations along with their annotations. Additionally, we want to apply our scene graphs to other datasets to confirm their general purpose.

REFERENCES

[1] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," Proceedings. 2005 IEEE International Joint Conference on Neural Networks, vol. 2, 2005, pp 729–734.

[2] X. Wang, Z. Li, and M. Jiang, "Molecule property prediction based on spatial graph embedding," Journal of chemical information and modeling, vol. 59, 2019, pp. 3817–3828.

[3] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," Proceedings of the 32nd International Conference on Neural Information Processing Systems, vol. 31, 2018, pp. 5171-–5181.

[4] J. Yamato, J. Ohya and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992, pp. 379–385.

[5] C. R. G. Dreher, M. Wächter and T. Asfour, "Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks," in IEEE Robotics and Automation Letters, vol. 5, no. 1, Jan. 2020, pp. 187–194.

[6] Y. Zhu, X. Li, C. Liu, M. Zolfaghari, Y. Xiong, C. Wu, Z. Zhang, J. Tighe, R. Manmatha, and M. Li, "A comprehensive study of deep video action recognition," arXiv preprint arXiv:2012.06567, 2020.

[7] C. Li, Q. Zhong, D. Xie and S. Pu, "Skeleton-based action recognition with convolutional neural networks," 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2018, pp. 597–600.

[8] C. Li, Q. Zhong, D. Xie and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 786–792.

[9] J. Wu, Y. Li, L. Wang, K. Wang, R. Li, and T. Zhou, "Skeleton Based Temporal Action Detection with YOLO," Journal of Physics: Conference Series, vol. 1237, 2019, p. 022087.

[10] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[11] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI'18/IAAI'18/EAAI'18), 2018, Article 912, pp. 7444-–7452.

[12] L. Shi, Y. Zhang, J. Cheng and H. Lu, "Skeleton-Based Action Recognition With Directed Graph Neural Networks," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 7904–7913.

[13] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 398—405.

[14] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles, "Action genome: Actions as compositions of spatio-temporal scene graphs," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10236—10247.

[15] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," Proceedings of the 34th International Conference on Neural Information Processing Systems, vol. 33, 2020, pp. 5812—5823.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," European Semantic Web Conference, 2018, pp. 593–607.

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for Quantum chemistry," Proceedings of the 34th International Conference on Machine Learning, vol. 70, 2017, pp. 1263—1272.

[18] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," International Conference on Learning Representations (ICLR 2017), 2017.