NODIS: Neural Ordinary Differential Scene Understanding

Yuren Cong[1], Hanno Ackermann[1], Wentong Liao[1], Michael Ying Yang[2(✉)], and Bodo Rosenhahn[1]

[1] Institute of Information Processing, Leibniz University, Hannover, Germany
{cong,ackermann,liao,rosenhahn}@tnt.uni-hanover.de
[2] Scene Understanding Group, University of Twente, Enschede, The Netherlands
michael.yang@utwente.nl

Abstract. Semantic image understanding is a challenging topic in computer vision. It requires to detect all objects in an image, but also to identify all the relations between them. Detected objects, their labels and the discovered relations can be used to construct a scene graph which provides an abstract semantic interpretation of an image. In previous works, relations were identified by solving an assignment problem formulated as (Mixed-)Integer Linear Programs. In this work, we interpret that formulation as Ordinary Differential Equation (ODE). The proposed architecture performs scene graph inference by solving a neural variant of an ODE by end-to-end learning. The connection between (Mixed-)Integer Linear Program and ODEs in combination with the end-to-end training amounts to learning how to solve assignment problems with image-specific objective functions. Intuitive, visual explanations are provided for the role of the single free variable of the ODE modules which are associated with time in many natural processes. The proposed model achieves results equal to or above state-of-the-art on all three benchmark tasks: scene graph generation (SGGEN), classification (SGCLS) and visual relationship detection (PREDCLS) on Visual Genome benchmark. The strong results on scene graph classification support the claim that assignment problems can indeed be solved by neural ODEs.

Keywords: Semantic image understanding · Scene graph · Visual relationship detection

1 Introduction

This paper investigates the problem of semantic image understanding. Given an image, the objective is to detect objects within, label them and infer the relations

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58565-5_38) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12365, pp. 636–653, 2020.
https://doi.org/10.1007/978-3-030-58565-5_38

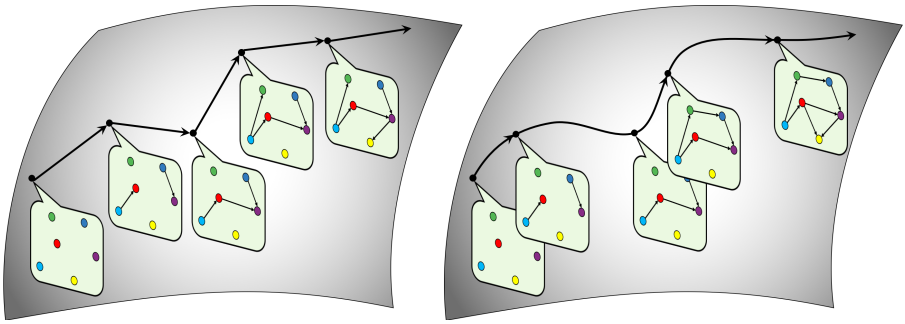

Fig. 1. Visualization of the main contributions made in this work. *Left*: state-of-the-art works rely on multiple embedding layers which amounts to an evenly-spaced, discrete sampling. *Right*: the proposed module relies on ordinary differential equations (ODE), thus it learns a continuous approximation of the trajectory onto the embedding manifolds.

which might exist between objects. These data provide rich semantic information about the image content. So called scene graphs contain all these information and constitute abstract representations of images [15]. Nodes of a scene graph represent objects detected in an image, while edges represent relationships between objects. Applications range from image retrieval [15] to high-level vision tasks such as visual question answering [39], image captioning [44,47] and visual reasoning [35]. The community has been very active in the past years to investigate these problems. Results on public benchmarks such as the Visual Genome database [20] have improved drastically within the past few years [2,41,49].

A naive approach to infer scene graphs is to use a standard object detector, classify the objects, and use a separate network to label the relationships between the objects. This, however, ignores the conditional information shared between the objects. It has therefore become standard practice to also use relationship information for object classification [13,25,28].

Scene graph estimation relies on two classification problems, both for objects and for relationships. Classification, or more generally, assignment problems can be solved by (Mixed) Integer Linear Programs, (M)ILP, for instance in [36]. Powerful solvers are able to even infer globally optimal solutions for many types of problems. A drawback of (M)ILPs is that the optimization objective and all constraints must be defined in advance. Objective functions usually include multiple terms and weights per term. Furthermore, backpropagation through an (M)ILP solver is not possible, thus such pipelines cannot adapt to data.

For many other problems in which adaption to data is less important, (M)ILPs have been successfully employed. In [8], network flows and transport problem are modeled by Partial Differential Equations (PDEs). The arising systems of PDEs are transformed to systems of Ordinary Differential Equations (ODEs) by finite differences. Adding constraints such as minimal or maximal capacities, integral constraints and an objective, for instance energy

minimization, and further applying a piece-wise linear approximation, a solution of the ODE can be obtained by means of an (M)ILP. That implies that for properly constructed (M)ILPs there are systems of ODEs which include the solution of the (M)ILPs, since dropping the optimization objective – thereby switching from an optimization problem to solving an equation system – only increases the number of solutions. While backpropagating through an (M)ILP is generally not possible, backpropagation through an ODE is possible using a recent seminal work *Neural Ordinary Differential Equations* [3].

Using this link, we propose to solve the labeling problem by means of neural ordinary differential equations. This module can be interpreted such that it can learn to perform the same task as an (M)ILP. Since it allows end-to-end training, it adapts to the data, thus it learns a near-optimal objective per image. A further question is which role the time-variable of the ODE plays in our architecture. We will provide visual explanations of that variable, namely that it controls how many objects are classified and how many relationships are classified (cf. Figs. 1 and 4). In our experimental evaluation, we demonstrate results that are equal to state-of-the-art or above in all three benchmark problems: scene graph generation (SGGEN), scene graph classification (SGCLS) and visual relationship prediction (PREDCLS).

The **contributions** made in this work can be summarized as follows: (1) We propose to use ODE modules for scene graph generation. (2) It can be interpreted that it learns the optimal assignment function per image. (3) Intuitive, visual explanations are provided for the role of the single free variable of the ODE modules which are associated with time in many natural processes. (4) The proposed method achieves state-of-the-art results. Our code is published on GitHub[1].

2 Related Work

Context for Visual Reasoning: Context has been used in semantic image understanding [7,13,22,27,28,46]. For scene graph generation, context information has been recently proposed and is still being investigated. Message passing has been used to include object context in images in several works, for instance by graphical models [24,42], by recurrent neural networks (RNN) [40,49], or by an iterative refinement process [19,41]. Context from language priors [30] has been proved to be helpful for visual relationships detection and scene graph generation [25,29,48].

Scene Graph Generation: Scene graphs are proposed in [15] for the task of image retrieval and also potential for many applications [17,18,32]. They consist of not only detected objects but also the object classes and the relationships between the detected objects. The estimation of scene graphs from images is attracting more and more attention in computer vision [6,14,24–26,40,49,51].

[1] https://github.com/yrcong/NODIS.

Several of these methods use message passing to capture the context of the two related objects [23,40,49], or of the objects and their relationships [24,25,41,42]. The general pipeline for message passing is to train some shared matrices to transform the source features into a semantic domain and then to assemble them to update the target features. In some works, an attention mechanism is implemented to weight the propagated information to achieve further improvement [9,12,24,42]. Graph CNNs [16] have been used to propagate information between object and relationship nodes [4,42]. Some works also introduce generative models for scene understanding [5,10].

Contrastive Training: Contrastive losses [1] have been applied for scene understanding [50]. They have also been applied for image captioning, visual question answering (VQA) and vector embeddings [31,34,45]. They are based on the idea that it can be easier to randomly select similar and dissimilar samples. Such losses can then be used even if no label information is available.

Losses: In Visual Genome [20], many semantically meaningful relations are not labeled. Furthermore, relations that can have multiple labels, for instance `on` and `sit`, are usually labeled only once. Networks for visual relationship detection which use cross-entropy as training loss may encounter difficulties during training, since almost identical pairs of objects can have either label, or even none at all. To overcome the problem of such contradicting label information, margin-based [21] and contrastive [50] losses have been proposed.

Support Inference: Inferring physical support, i.e. which structures for instance carry others, e.g. `floor` → `table`, was investigated for object segmentation [36], instance segmentation [52], and also for scene graph inference [43].

Proposed Model: The proposed model uses neither iterations, except for LSTM-cells, nor message passing, nor a Graph CNN. As most recent works do, language priors are included. We further use a standard loss based on cross-entropy. Unlike most state-of-the-art algorithms, we propose to use a new module which has never before been used for semantic image analysis. The ODE module [3] can be interpreted as learning a deep residual model. In contrast to residual networks, the ODE-module *continuously* models its solutions according to a pre-defined, problem-specific precision.

This idea is motivated by works on gas, water, energy and transport networks which need to be modeled by systems of Partial Differential Equations (PDE). In [8], it was proposed to simplify such systems to obtain systems of Ordinary Differential Equations (ODE), add one or multiple optimization objectives and then use a Mixed-Integer Linear Program to compute the solution. That implies that, given an (M)ILP, we can always find a system of ODEs that can generate the same solution among others. Using a neural ODE [3], we can thus learn a function of the system of ODEs. Due to the end-to-end training, that system further encodes near-optimal assignment functions per image.

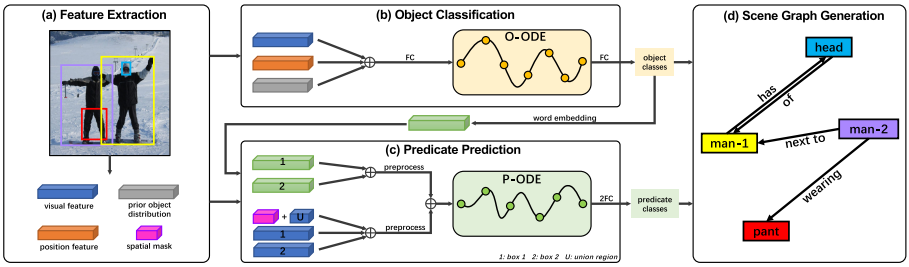

Fig. 2. Overview of the proposed method: (a) given an image, many object regions are proposed, (b) those regions are classified by an Object ODE (O-ODE), (c) after pre-processing the features of object pairs, a Predicate ODE (P-ODE) is applied to predict predicates, (d) a scene graph is generated.

3 Method

In this section, we first define scene graphs, then we define straight-forward ILP models for object and relationship classification in Sect. 3.2. Neural Ordinary Differential Equations [3] are briefly explained in Sect. 3.3. Both models are combined in Sects. 3.4 and 3.5.

3.1 Scene Graphs

A scene graph [15] is an abstract semantic representation of all objects in an image and the relationships between them. The set of objects consists of a set of bounding boxes along with a label for each object. The relationships between two objects, also known as predicates, consists of bounding boxes which usually cover both object bounding boxes, and labels for each relation. As in [49], we include a particular predicate to indicate that there is no relation between two objects.

3.2 Models for Object and Relationship Detection

Assume a graph $G_{obj} = (U_{obj}, E_{obj})$ whose nodes represent the detected yet unlabeled objects in an image. Each node is further assigned to a label $l \in \mathcal{L}_{obj}$. Each label l is associated with a score $\alpha_{u,l}$ with $u \in U$. That score can represent an agreement with some given feature map. The integer variables $x_{u,l} \in \{0, 1\}$ indicate that node u is given label l. We assume that each object can only belong to a single class.

We further include a term $\beta_{u,u',l,l'}$ which models statistical prior knowledge about co-occurrences of object pairs u and u', $u \neq u'$, and their corresponding labels l and l'. We then arrive at the following Integer Linear Problem (ILP) for object classification

$$\max \sum_{u \in U_{obj}} \sum_{l \in \mathcal{L}_{obj}} \alpha_{u,l} x_{u,l} + w_{obj} \sum_{u,\, u' \in U_{obj}} \sum_{l,\, l'} \beta_{u,u',l,l'} x_{u,l} x_{u',l'} \tag{1a}$$

$$s.t. \quad x_{u,l} \in \{0,1\}, \quad \sum_{l \in \mathcal{L}_{obj}} x_{u,l} \leq 1 \tag{1b}$$

where w_{obj} is a scalar weight which balances the two terms. How to exactly determine the scores $\alpha_{u,l}$, $\beta_{u,u',l,l'}$ and the weight w_{obj} constitutes a hyper-parameter search. The large search space makes this problem by itself challenging. Some of the parameters remain invariant for all images.

Assume a further graph $G_{pred} = (V_{pred}, E_{pred})$ whose nodes represent all possible subject-object pairs. Each node is assigned to a label $k \in \mathcal{L}_{pred}$. Each label k is associated with a score $\alpha_{v,k}$ with $v \in V_{pred}$. The integer variables $x_{v,k} \in \{0,1\}$ indicate that node v is given label k, for instance the subject-object pair *dog-street* with a label *sit* or *walk*. The number of labels per node is limited by T_v, and the total number of labels by K. The latter arises from the *recall-at-K* metric commonly used in visual relationship detection and scene graph estimation tasks.

Here, we also include a term $\beta_{v,v',k,k'}$ which models statistical prior knowledge about co-occurrences of subject-object pairs v with pairs v', $v \neq v'$, and their corresponding relationship labels k and k'. We then arrive at the following Integer Linear Problem (ILP) for relationship classification

$$\max \sum_{v \in V_{pred}} \sum_{k \in \mathcal{L}_{pred}} \alpha_{v,k} x_{v,k} + w_{pred} \sum_{v,\, v' \in V_{pred}} \sum_{k,\, k'} \beta_{v,v',k,k'} x_{v,k} x_{v',k'} \tag{2a}$$

$$s.t. \quad x_{v,k} \in \{0,1\}, \quad \sum_{k \in \mathcal{L}} x_{v,k} \leq T_v, \quad \sum_{v \in V_{pred}} \sum_{k \in \mathcal{L}_{pred}} x_{v,k} \leq K, \tag{2b}$$

where w_{pred} is a scalar weight which balances the two terms.

Denote by $\alpha(t)$ the *function* in the single variable t that assigns *continuous* scalar weights which express label strengths. The values that the label weights take on vary with t. Likewise, let $\beta(t)$ be a function in t that assigns scalar weights according to co-occurrence. From [8], we can see that the optimization problems defined by Eqs. 1 and 2 correspond to ordinary differential equation (ODE) models defined by

$$\frac{d}{dt} f = f(x(t),\, t). \tag{3}$$

In other words, we can use Eqs. 1 and 2 to obtain a solution of problem (3) subject to several constraints. A disadvantage of the optimization problem in Eqs. 1 and 2 is that it is not possible to backpropagate through the ILPs. Thus, feature-generating neural networks cannot be trained to adapt to given data when using the ILPs.

3.3 Neural Ordinary Differential Equations [3]

For many problems it is hard to explicitly define functions $\alpha(t)$ and $\beta(t)$. In [3], it was proposed to parameterize $f(t)$ by a function $f_\theta(t)$ based on neural network

with parameters θ. The idea is then to use $f_\theta(t)$ to solve an ODE. Thus, starting with an input $x(0)$ at time $t = 0$, the output at time T can be computed by a standard ODE-solver

$$\frac{d}{dt} f_\theta = f_\theta\,(\,x(t),\,t\,)\,. \tag{4}$$

The dynamics of the system can thereby be approximated up to the required precision. Importantly, model states at times $0 < t < T$ need not be evenly spaced.

For back-propagation, the derivatives for the adjoint $a(t) = -\partial L/\partial x(t)$

$$\frac{d}{dt} a = -a(t)^\top \frac{\partial}{\partial t} f(\,x(t),\,t,\,\theta\,) \tag{5}$$

and

$$\frac{d}{d\theta} L = \int_T^0 a(t)^\top \frac{\partial}{\partial \theta} f(\,x(t),\,t,\,\theta\,)\,dt \tag{6}$$

have to be computed for loss L. This computation, along with the reverse computation of $x(t)$ starting at time T can be done by a single call of the ODE-solver backwards in time.

3.4 Assignments by Neural Ordinary Differential Equations

While Eqs. 1 and 2 can be used to obtain solutions of problem (3), this does not allow to train previous modules if used in a neural network. Furthermore, the hyper-parameters of the ILP must be determined in advance and cannot be easily adapted to the data. On the other hand, we can construct an ODE from Eqs. 1 or 2, respectively. Since a manual construction would be hard, we can use a neural ODE layer to learn the optimal assignment function. In contrast to an ILP, the neural ODE both allows to train a feature generating network in front of it, and can adapt to each image. The latter implies that the hyper-parameters of the corresponding, yet unknown ILP vary with each image.

ODEs involve the variable t which is usually associated with time in many natural processes. Here, we are posed with the question what this variable represents for object or relationship classification. It will be demonstrated in Sect. 4.4 that the two variables of two neural ODE layers control how many objects and relationships are classified correctly (cf. Fig. 4). In other words, specifying particular values of the variable of the relationship module determines the connectivity of estimated scene graph, whereas such values of the object layer determine if too few or too many objects are correctly labeled.

As outlined in Fig. 2, we use two separate ODE layers. The visual features of detected objects, their positional features and the prior object distributions are concatenated into a vector x_v and processed by an ODE layer in the object classifier

$$\frac{d}{dt_1} f_{\theta_u} = f_{\theta_u}\,(\,x_u(t_1),\,t_1)\,. \tag{7}$$

In the following, this ODE layer will be denoted by *O-ODE*.

The word embedding resulting from the object classifier, the spatial mask with the union box and the visual features of two detected and classified object are concatenated and pre-processed to yield vector $x_{v,v'}$ before being processed by an ODE layer for predicate classification (*P-ODE*)

$$\frac{d}{dt_2} f_{\theta_v} = f_{\theta_v}\left(x_{v,v'}(t_2), t_2\right). \tag{8}$$

The variables t_1 and t_2 in Eqs. (7) and (8) control how many objects or relations are labeled. In other words, graphs constructed using different t and t', either for Eq. (7) or (8), result in scene graphs with differently many objects or relations correctly labeled.

3.5 Architecture

Our model is built on Faster-RCNN [33] which provides proposal boxes, feature maps and primary object distributions. There are two fundamental modules in the model: object classifier and predicate classifier. Each of them contains a neural ordinary differential equation layer, Object ODE (O-ODE) and Predicate ODE (P-ODE). For both of them we use bidirectional LSTMs as the approximate function in the ODE solver. The data in the model are organized as sequences with random order.

In the object classifier, the feature maps, bounding box information and primary object distribution from Faster-RCNN are concatenated and fed through a fully connection layer. The object class scores are computed by the O-ODE and a following linear layer. The predicate classifier uses feature maps, spatial and semantic information of object pairs to predict the predicate categories. The spatial masks are forwarded into a group of convolution layers so that the output has the same size as the feature maps of the union box ($512 \cdot 7 \cdot 7$) and can be added per element. Global average pooling is applied on the feature maps of the subject box, object box and union box. The features of the subject, object, and union boxes are concatenated as ($3 \cdot 512$)-dimensional visual vectors. Two 200-dimensional semantic embedding vectors are generated from the subject and object classes predicted by the object classifier and concatenated as 400-dimensional semantic vectors.

The visual vectors and semantic vectors of object pairs can be pre-processed by three methods before the P-ODE: *FC-Layer:* The ($3 \cdot 512$)-dimensional visual vectors and 400-dimensional semantic vectors are forwarded into two independent fully connection layers that both have 512 neurons. Then, the outputs are concatenated together as 1024-dimensional representation vectors for the P-ODE. *GCNN:* The visual vectors and semantic vectors are first concatenated. Then, we use a graph convolutional neural network (GCNN) to infer information about context. Since the number of object pairs in each image is variable, we set each element on the diagonal of the adjacency matrix to 0.8. The weight of 0.2 is uniformly distributed among the remaining entries of each row. The output vectors of the GCNN are passed into the P-ODE. *LSTM:* Similar as for

the first variant, the $(3 \cdot 512)$-dimensional visual vectors and 400-dimensional semantic vectors are fed into two single layer LSTMs. Both of them have the output dimension 512. We concatenate the two outputs for the P-ODE.

The final class scores of the relations are computed by the P-ODE followed by two linear layers.

4 Experiments

In this section, we firstly clarify the experimental settings and implementation details. Then, we show quantitative and qualitative results on the Visual Genome (VG) benchmark dataset [20] in terms of scene graph generation.

4.1 Dataset, Settings, and Evaluation

Dataset. We validated our methods on the VG benchmark dataset [20] for the task of scene graph generation. However, there are varying data pre-processing strategies and dataset splits in different works. For fair comparison, we adopted the data split as described in [41] which is the most widely used. According to the data pre-processing strategy, the most-frequent 150 object categories and 50 predicate types are selected. The training set has 75651 images while the test set has 32422 images.

Settings. For comparison with prior works, we use Faster R-CNN [33] with VGG16 [37] as the backbone network for proposing object candidates and extracting visual features. We adopted the code base and a pre-trained model provided by [49]. As in NeuralMotifs [49], the input image is resized to 592×592, bounding box scales and dimension ratios are scaled, and a ROIAlign layer [11] is used to extract features within the boxes of object proposals and the union boxes of object pairs from the shared feature maps.

Evaluation. There are three standard experiment settings for evaluating the performance of scene graph generation: (1) **Predicate classification** (PREDCLS): predict relationship labels of object pairs given ground truth bounding boxes and labels of objects. (2) **Scene graph classification** (SGCLS): given ground truth bounding boxes of objects, predict object labels and relationship labels. (3) **Scene graph detection** (SGGEN): predict boxes, labels of object proposals and relation labels of object pairs given an image. Only when the labels of the subject, relation, and object are correctly classified, and the boxes of subject and object have more than 50% intersection-over-union (IoU) with the ground truth, it is counted as a correctly detected entity. The most widely adopted recall@K metrics ($K = [20, 50, 100]$) for relations are used to evaluate the system performance.

Table 1. Comparison on VG test set [41] **using graph constraints**. All numbers in %. We use the same object detection backbone provided by [49] for fair comparison. Bold blue numbers indicate results better than competitors by >0.5. Regarding GCL [50], cf. to the text. Methods in the lower part use the same data split as [41]. Results for MSDN⋆ [25] are from [49].

Data Split	Method	SGGEN			SGCLS			PREDCLS		
		R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
[25]	MSDN⋆ [25]	–	11.7	14.0	–	20.9	24.0	–	42.3	48.2
	FacNet [24]	–	13.1	16.5	–	22.8	28.6	–	–	–
[41] split	VRD [29]		0.3	0.5		11.8	14.1		27.9	35.0
	IMP [41]	14.6	20.7	24.6	31.7	34.6	35.4	52.7	59.3	61.3
	Graph R-CNN [42]	–	11.4	13.7	–	29.6	31.6	–	54.2	59.1
	Mem [40]	7.7	11.4	13.9	23.3	27.8	29.5	42.1	53.2	57.9
	MotifNet [49]	21.4	27.2	30.3	32.9	35.8	36.5	58.5	65.2	67.1
	MotifNet-Freq	20.1	26.2	30.1	29.3	32.3	32.6	53.6	60.6	62.2
	GCL [50]	21.1	28.3	32.7	36.1	36.8	36.8	66.9	68.4	68.4
	VCTREE [38]	22.0	27.9	31.3	35.2	38.1	38.8	60.1	66.4	68.1
	CMAT [2]	22.1	27.9	31.2	35.9	39.0	39.8	60.2	66.4	68.1
	Ours-FC	21.5	27.5	30.9	**37.7**	**41.7**	**42.8**	58.6	66.1	68.1
	Ours-GCNN	21.4	27.1	30.6	33.2	38.2	39.7	52.0	60.9	63.8
	Ours-LSTM	21.6	27.7	31.0	**37.9**	**41.9**	**42.9**	58.9	66.0	67.9

Training. We train our model with the sum of the cross entropy losses for objects and predicates. We collect all annotated relationships in the image and add negative relationships so that the relation sequences in the batch have identical length if there are sufficiently many ground truth boxes. We randomly select one if the object pairs are annotated with multiple relationships. For SGCLS and SGGEN, ground truth object labels are provided to the semantic part at the training stage. For fair comparison we use the same pre-trained Faster-RCNN as [49] and freeze its parameters. An ADAM optimizer was used with batch size 6, initial learning rate 10^{-4}, and cross-entropy as loss both for object and relationship classification. We choose *dopri5* as ODE solver and set the absolute tolerance $atol = 0.01$ and the relative tolerance $rtol = 0.01$. The integral time t_{end} is set to 1.5 (cf. Sect. 4.4). Please confer to the supplementary for a more detailed explanation of our model architecture.

4.2 Quantitative Results and Comparison

Our results **using graph constraints** are shown in Table 1. The middle block indicates methods that all use the same data split that is used in [41]. Two methods that use different splits are listed in the top section of the table. For MSD-net [25] in this part, we show the results reported in [49].

We used bold blue numbers to indicate the best result in any column that was at least 0.5% larger than the next best competing method. Since GCL [50] used a different VGG backbone than other works, and also larger input images

than all other methods (1024×1024 compared to 592×592), results cannot be fairly compared with other methods, so we did not highlight best results in this row (SGGEN-R@100 and PREDCLS-R@20).

The bottom part of Table 1 shows results of the proposed ODE layers using three different front-ends: (1) One in which the function used inside the ODE layer is taken to be a linear layer (**ours-FC**), (2) the second in which a Graph-CNN is used (**ours-GCNN**), (3) and the third in which a bidirectional LSTM is used (**ours-LSTM**).

As can be seen from Table 1, most recent methods (MotifNet [49], VCTREE [38] and CMAT [2]) including ours are very similar in performance with respect to the scores in SGGEN and PREDCLS. The only exception among state-of-the-art works is GCL [50] in two out of nine scores, yet by using a more powerful front-end.

For SGCLS, however, the proposed ODE layer turns out to be very effective. Apparently both the version using a linear layer inside the ODE layer (**ours-FC**) and the one using the bidirectional LSTM (**ours-LSTM**) perform very similar. Their scores improve state-of-the-art (CMAT [2]) between $\approx 2\%$ (SGCLS-R@20), $\approx 3\%$ (SGCLS-R@50) and $\approx 3\%$ (SGCLS-R@100). Compared with GCL [50] our results improve by up to 5% (SGCLS-R@100) although our VGG was trained on coarser images. This demonstrates the effectiveness of the proposed ODE layer. It is sufficient to use a simple linear layer, since the ODE layer is so powerful that it produces similar outputs to those of a function based upon a more complicated and slower bidirectional LSTM.

4.3 Ablation Studies

For ablation studies, we consider several variants of the proposed network architecture. Results are shown in Table 2. For model-1 (first row), we removed the ODE networks and directly classified the output of the feature generating networks. This measures the impact of both ODE modules. In model-2 (second row), we removed only the ODE layer for object classification (O-ODE) in the proposed network. The third model shows the results when the relationship ODE (P-ODE) is removed. Finally, the fourth rows shows the results if both ODE modules are present.

Table 2. Ablation study demonstrating the effect if both ODE layers are removed (first row), only the layer for object classification (second row), only the layer for predicate classification (third row), or if both are present (last row).

Model	O-ODE	P-ODE	SGGEN		SGCLS		PREDCLS	
			R@50	R@100	R@50	R@100	R@50	R@100
1	–	–	27.1	30.4	35.3	36.1	65.4	67.4
2	–	✓	27.3	30.6	35.9	36.6	65.9	67.9
3	✓	–	27.3	30.7	41.4	42.5	65.4	67.5
4	✓	✓	27.7	31.0	41.9	42.9	66.0	67.9

As can be seen, removing the ODE layer for predicate classification has a negligible effect since the PREDCLS scores hardly change. Removing the ODE layer for object classification has a strong, negative effect, however. This study shows that the ODE module can have a very positive impact. It further confirms our main claim that classification/assignment problems can be solved by means of neural ODEs.

Regarding the scores on PREDCLS, we conjecture that the noise in VG (missing relationship labels, incorrect labels) is so strong that scores cannot improve anymore. This hypothesis is supported by the fact that results in the past two years have not improved since [49].

4.4 Neural ODE Analysis

We propose two neural ordinary differential equations: Object ODE (O-ODE) and Predicate ODE (P-ODE) in the object classifier and predicate classifier respectively. Here, we tune the hyper-parameter, integral time t_{end} in the ODE solver, to understand how that variables influences neural ordinary differential equations and final performance. Moreover, we extract the hidden states at different points in time and generate the scene graph to visualize how features are refined in the ODE space.

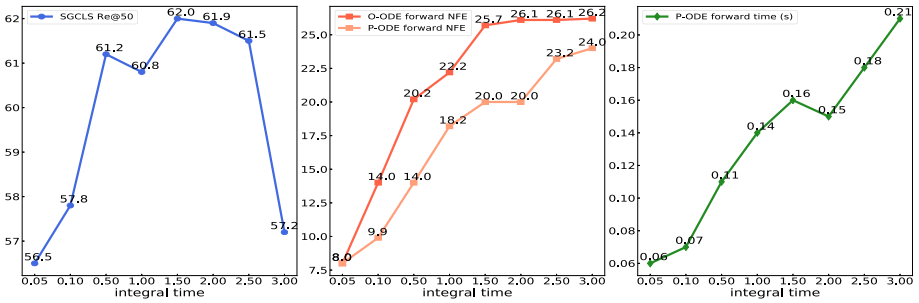

Fig. 3. The effect of integral time in the ODE solver on SGCLS Re@50, average forward NFE (number of function evaluation) and average forward running time per image of O-ODE and P-ODE. Because of different number of objects in different images the experimental results are based on 5000 samples from the training set.

Since the O-ODE and P-ODE both work in the setting SGCLS, SGCLS Re@50 is used as performance indicator. Because different images contain different number of objects, we randomly sample 5000 images from the training set to compute the average number of function evaluation (NFE) for a forward pass and the computation time per image. This experiment is implemented on a single GTX 1080Ti GPU. We vary the integral time t_{end} from 0.05 to 3.00 (x-axes).

According to the left plot in Fig. 3, SGCLS Re@50 increases gradually from 56.5 to 62.0 until $T = 1.50$ and then decreases; (middle plot) the average forward

NFE of O-ODE increases from 8.0 to 26.2 and from 8.0 to 24.0 for P-ODE; (right plot) the average forward time of the P-ODE layer increases from 0.06 s to 0.21 s whereas the average forward time of the O-ODE layer is negligibly small ($<10^{-4}$ s), thus we omit these measurements. The P-ODE requires much more time than the O-ODE due to the large amount of object pairs. Considering the above points, we set $t_{end} = 1.5$ for all other experiments.

To provide an intuitive interpretation of the parameters t_1 and t_2 in the O-ODE and the P-ODE, respectively, we use a model trained on SGGEN with the hyper-parameter $t_{end} = 1.5$ and evaluate it using different latent vectors corresponding to different points between 0 and 2.5. The hidden states at $t = 2.0$ and $t = 2.5$ imply extrapolation. The detection results and scene graphs generated by the different hidden states are shown in Fig. 4. The features are more and more refined by the neural ODE layers if t_1 and t_2 increase, hence more objects and relationships are correctly classified. After $t_{end} = 1.5$, increasingly many relationships are misclassified due to the extrapolation.

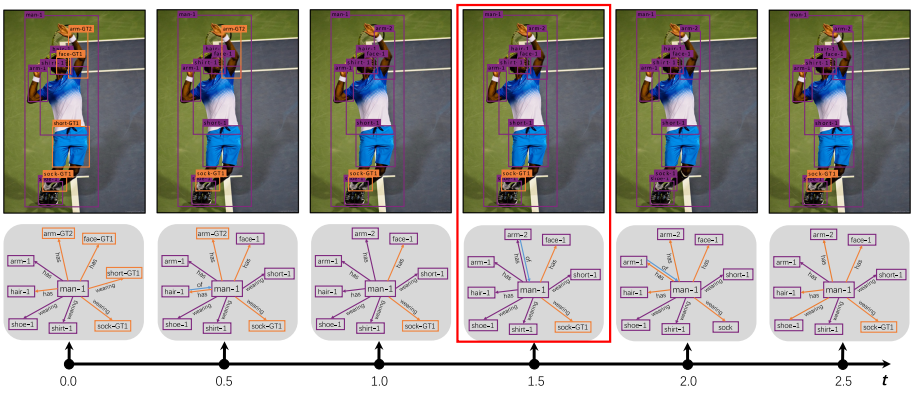

Fig. 4. We evaluate the model trained with the hyper-parameter $t_{end} = 1.5$, i.e. during training the hidden vector at $t = 1.5$ is used for classifications. The upper row visualizes detection results while the lower row shows the corresponding scene graphs. The red box indicates the state corresponding to $t = 1.5$. The orange boxes and edges indicate the objects and predicates in the ground truth that are not detected, purple that the objects and predicates are predicted correctly. Blue edges show false positives. (Color figure online)

4.5 Qualitative Results

Qualitative results for scene graph generation (SGGEN) are shown in Fig. 5. The images include object detections. The purple color indicates correctly detected and classified objects and relations, whereas orange means failure. The blue color indicates false negatives, i.e. relationships that are not in the ground truth.

Most of the errors stem from the object detection stage. Whenever an object is not detected, relationships connecting this object are also not present in the

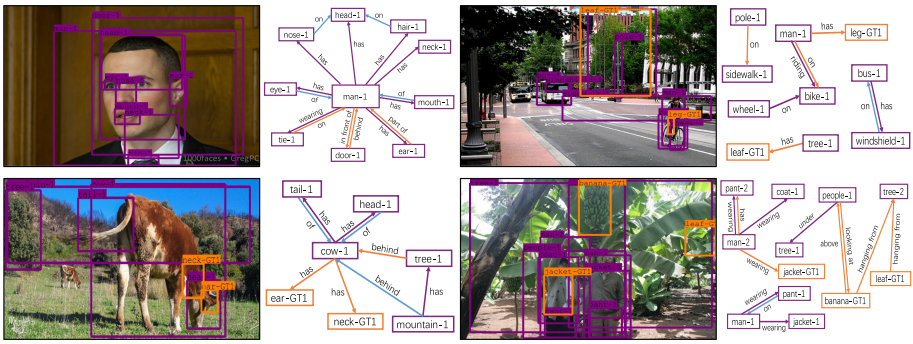

Fig. 5. Qualitative results from our model in the scene graph generation setting. Purple boxes denote correctly detected objects while orange boxes denote ground truth objects that are not detected. Purple edges correspond to correctly classified relationships at the R@20 setting while orange edges denote ground truth relationships that are not detected. Blue edges denote detected relationships that do not exist in ground truth annotations (false positives). (Color figure online)

scene graph. The cause of these errors is the Faster R-CNN detector which is used in all previous works.

There are a few false positives (blue links) which are semantically meaningful, for instance `eye-of-man` in the upper left example. In other words, the ground truth lacks this particular relationship in such cases. Several false positives result from semantically indistinguishable classes, for instance `has` and `of` in the lower left example.

5 Conclusions

We presented Neural Ordinary Differential Equations for Scene Understanding (NODIS). The idea of this work is based on the fact that Mixed-Integer Linear Programs can be used to solve problems defined by ordinary differential equations; therefore, given a particular (M)ILP, we can find a system of ODEs that can produce the solution of the (M)ILP within a time series. Since it is not possible to manually define the system of ODE to solve, we draw on recent advances in machine learning and use a trainable function approximator instead of an explicitly defined system of ODEs. In other words, the proposed network *learns* the optimal function to solve the assignment problem, whereas previous works manually define modules to do so.

We use this newly defined module for object classification and relationship classification. The proposed model using ODE layers shows large improvements on SGCLS, between 2% and 3% compared with the best SOTA in that category. We believe that ODE layers can be valuable improvements for neural architectures in semantic image understanding.

Acknowledgement. This work was partially supported by the DFG grant COVMAP (RO 2497/12-2) and EXC 2122.

References

1. Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., Saunshi, N.: A theoretical analysis of contrastive unsupervised representation learning. In: Proceedings of Machine Learning Research (PMLR) (2019)
2. Chen, L., Zhang, H., Xiao, J., He, X., Pu, S., Chang, S.F.: Counterfactual critic multi-agent training for scene graph generation. In: IEEE International Conference on Computer Vision (ICCV), pp. 4613–4623 (2019)
3. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Neural Information Processing Systems (NeurIPS), pp. 6571–6583 (2018)
4. Chen, T., Yu, W., Chen, R., Lin, L.: Knowledge-embedded routing network for scene graph generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6163–6171 (2019)
5. Chen, V.S., Varma, P., Krishna, R., Bernstein, M., Re, C., Fei-Fei, L.: Scene graph prediction with limited labels. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2580–2590 (2019)
6. Dai, B., Zhang, Y., Lin, D.: Detecting visual relationships with deep relational networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3076–3086 (2017)
7. Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M.: An empirical study of context in object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1271–1278 (2009)
8. Fügenschuh, A., Herty, M., Klar, A., Martin, A.: Combinatorial and continuous models for the optimization of traffic flows on networks. SIAM J. Optim. **16**(4), 1155–1176 (2006)
9. Gkanatsios, N., Pitsikalis, V., Koutras, P., Maragos, P.: Attention-translation-relation network for scalable scene graph generation. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (2019)
10. Gu, J., Zhao, H., Lin, Z., Li, S., Cai, J., Ling, M.: Scene graph generation with external knowledge and image reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1969–1978 (2019)
11. He, K., Georgia, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE International Conference on Computer Vision (ICCV), pp. 2961–2969 (2017)
12. Herzig, R., Raboh, M., Chechik, G., Berant, J., Globerson, A.: Mapping images to scene graphs with permutation-invariant structured prediction. In: Advances in Neural Information Processing Systems, pp. 7211–7221 (2018)
13. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3588–3597 (2018)
14. Hu, T., Liao, W., Yang, M.Y., Rosenhahn, B.: Exploiting attention for visual relationship detection. In: Fink, G.A., Frintrop, S., Jiang, X. (eds.) DAGM GCPR 2019. LNCS, vol. 11824, pp. 331–344. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33676-9_23
15. Johnson, J., et al.: Image retrieval using scene graphs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3668–3678 (2015)

16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2016)
17. Kluger, F., Ackermann, H., Yang, M.Y., Rosenhahn, B.: Temporally consistent horizon lines. In: ICRA (2020)
18. Kluger, F., Brachmann, E., Ackermann, H., Rother, C., Yang, M.Y., Rosenhahn, B.: CONSAC: robust multi-model fitting by conditional sample consensus. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4634–4643 (2020)
19. Krishna, R., Chami, I., Bernstein, M., Fei-Fei, L.: Referring relationships. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
20. Krishna, R., et al.: Visual genome: connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vis. (IJCV) **123**(1), 32–73 (2017). https://doi.org/10.1007/s11263-016-0981-7
21. Krishnaswamy, N., Friedman, S., Pustejovsky, J.: Combining deep learning and qualitative spatial reasoning to learn complex structures from sparse examples with noise. Association for the Advancement of Artificial Intelligence (AAAI), vol. 33, pp. 2911–2918 (2019)
22. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.S.: Graph cut based inference with co-occurrence statistics. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6315, pp. 239–253. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15555-0_18
23. Li, Y., Ouyang, W., Wang, X.: Vip-CNN: visual phrase guided convolutional neural network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1347–1356 (2017)
24. Li, Y., Ouyang, W., Zhou, B., Shi, J., Zhang, C., Wang, X.: Factorizable net: an efficient subgraph-based framework for scene graph generation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 346–363. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_21
25. Li, Y., Ouyang, W., Zhou, B., Wang, K., Wang, X.: Scene graph generation from objects, phrases and region captions. In: IEEE International Conference on Computer Vision (ICCV), pp. 1261–1270 (2017)
26. Liang, X., Lee, L., Xing, E.P.: Deep variation-structured reinforcement learning for visual relationship and attribute detection. In: IEEE International Conference on Computer Vision (ICCV), pp. 848–857 (2017)
27. Liao, W., Rosenhahn, B., Shuai, L., Yang, M.Y.: Natural language guided visual relationship detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019)
28. Liu, Y., Wang, R., Shan, S., Chen, X.: Structure inference net: object detection using scene-level context and instance-level relationships. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6985–6994 (2018)
29. Lu, C., Krishna, R., Bernstein, M., Fei-Fei, L.: Visual relationship detection with language priors. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 852–869. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_51
30. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
31. Nagaraja, V.K., Morariu, V.I., Davis, L.S.: Modeling context between objects for referring expression understanding. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 792–807. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_48

32. Reinders, C., Ackermann, H., Yang, M.Y., Rosenhahn, B.: Learning convolutional neural networks for object detection with very little training data. In: Multimodal Scene Understanding, pp. 65–100. Elsevier (2019)
33. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Neural Information Processing Systems (NeurIPS), pp. 91–99 (2015)
34. Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., Schiele, B.: Grounding of textual phrases in images by reconstruction. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 817–834. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_49
35. Shi, J., Zhang, H., Li, J.: Explainable and explicit visual reasoning over scene graphs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8376–8384 (2019)
36. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_54
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
38. Tang, K., Zhang, H., Wu, B., Luo, W., Liu, W.: Learning to compose dynamic tree structures for visual contexts. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
39. Teney, D., Liu, L., van den Hengel, A.: Graph-structured representations for visual question answering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3233–3241 (2017)
40. Wang, W., Wang, R., Shan, S., Chen, X.: Exploring context and visual pattern of relationship for scene graph generation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8188–8197 (2019)
41. Xu, D., Zhu, Y., Choy, C.B., Fei-Fei, L.: Scene graph generation by iterative message passing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5410–5419 (2017)
42. Yang, J., Lu, J., Lee, S., Batra, D., Parikh, D.: Graph R-CNN for scene graph generation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 690–706. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_41
43. Yang, M.Y., Liao, W., Ackermann, H., Rosenhahn, B.: On support relations and semantic scene graphs. ISPRS J. Photogram. Remote Sens. (ISPRS) **131**, 15–25 (2017)
44. Yang, X., Tang, K., Zhang, H., Cai, J.: Auto-encoding scene graphs for image captioning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10685–10694 (2019)
45. Yang, X., Zhang, H., Cai, J.: Shuffle-then-assemble: learning object-agnostic visual relationship features. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11216, pp. 38–54. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01258-8_3
46. Yao, B., Fei-Fei, L.: Modeling mutual context of object and human pose in human-object interaction activities. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 17–24 (2010)

47. Yao, T., Pan, Y., Li, Y., Mei, T.: Exploring visual relationship for image captioning. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 711–727. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_42
48. Yu, R., Li, A., Morariu, V.I., Davis, L.S.: Visual relationship detection with internal and external linguistic knowledge distillation. In: IEEE International Conference on Computer Vision (ICCV), pp. 1974–1982 (2017)
49. Zellers, R., Yatskar, M., Thomson, S., Choi, Y.: Neural motifs: Scene graph parsing with global context. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5831–5840 (2018)
50. Zhang, J., Shih, K.J., Elgammal, A., Tao, A., Catanzaro, B.: Graphical contrastive losses for scene graph parsing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
51. Zhuang, B., Liu, L., Shen, C., Reid, I.: Towards context-aware interaction recognition for visual relationship detection. In: IEEE International Conference on Computer Vision (ICCV), pp. 589–598 (2017)
52. Zhuo, W., Salzmann, M., He, X., Liu, M.: Indoor scene parsing with instance segmentation, semantic labeling and support relationship inference. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5429–5437 (2017)