

Dynamic Graph Generation Network: Generating Relational Knowledge from Diagrams

Daesik Kim^{1,2} YoungJoon Yoo³ Jeessoo Kim¹ Sangkuk Lee^{1,2} Nojun Kwak¹

¹Seoul National University ²V.DO Inc. ³Clova AI Research, NAVER Corp.
{daesik.kim|kimjiss0305|sanguklee|nojunk}@snu.ac.kr youngjoon.yoo@navercorp.com

Abstract

In this work, we introduce a new algorithm for analyzing a diagram, which contains visual and textual information in an abstract and integrated way. Whereas diagrams contain richer information compared with individual image-based or language-based data, proper solutions for automatically understanding them have not been proposed due to their innate characteristics of multi-modality and arbitrariness of layouts. To tackle this problem, we propose a unified diagram-parsing network for generating knowledge from diagrams based on an object detector and a recurrent neural network designed for a graphical structure. Specifically, we propose a dynamic graph-generation network that is based on dynamic memory and graph theory. We explore the dynamics of information in a diagram with activation of gates in gated recurrent unit (GRU) cells. On publicly available diagram datasets, our model demonstrates a state-of-the-art result that outperforms other baselines. Moreover, further experiments on question answering shows potentials of the proposed method for various applications.

1. Introduction

Within a decade, performances on classical vision problems such as image classification [7], object detection [5, 16], and segmentation [17] have been largely improved by the use of deep learning frameworks. Based on the great successes of deep learning for such low-level vision problems, a next step could be deriving semantics from images such as relations between objects. For example, to understand a given soccer scene more deeply, it would be very important not only to detect the objects in the image, such as players and a ball but also to figure out the relationships between the objects.

This work was supported by NRF of Korea (2017M3C4A7077582).

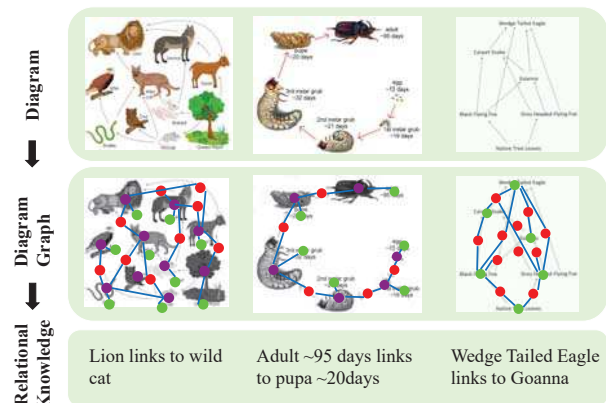


Figure 1. Examples of how relational knowledge can be generated from a diagram. In the first row, inputs are only diagrams which have various types of topics, illustrations, texts and layouts. Our model can **infer a graphical structure** in a diagram as in the second row. In the end, we can extract relational knowledge in the form of sentence from the generated graphs.

In this work, among various vision problems, we aim to understand diagram images, which have played a major role in classical knowledge representation and education. Previously, most machine learning algorithms have focused on extracting knowledge from the information described by natural languages or structured databases (e.g. Freebase [3], Wordnet [20]). In contrast to language-based knowledge, a diagram contains rich illustrations including text, visual information and their relationships, which can depict human’s perception of objects more succinctly. As shown in Figure 1, some complicated concepts such as “food web in a jungle” and “life cycle of a moth” can be easily described as a diagram. On the other hand, a single natural image or a single sentence may not be sufficient to deliver the same amount of information to the readers.

Whereas the diagram has good characteristics of knowledge abstraction, it requires composite solutions to properly

analyze and extract the contained knowledge. Since diagrams in a science textbook employ a wide variety of methods for explaining concepts in their layout and composition, understanding a diagram can be a challenging problem of inferring human’s general perception of structured knowledge. Unlike conventional vision problems, this task must involve inference models for vision, language and particularly relations among objects which can be a novel point. Despite the noted arbitrariness, we believe that a simple method generally exists to analyze and interpret the knowledge conveyed in a diagram.

There have not been many studies on diagram analysis yet, but Kembhavi *et al.* [9] recently proposed a pioneering work analyzing the diagram’s structure (DSDPnet). The main flow of the algorithm is twofold: 1) Object detection: Objects in the diagram are detected and segmented individually by conventional methods such as those in [2, 11]. 2) Relation inference: The relations among detected objects are inferred by a recurrent neural network (RNN) to transmit contexts sequentially. However, this approach has several limitations. First, concatenating separated methods results in a long pipeline from input to output, which can cause accumulated errors and lose contexts on a diagram. Second, and more importantly, the vanilla RNN is not fully capable of dealing with the information formed as a graph structure.

In this paper, we propose a novel method to solve the aforementioned issues. Our contributions are twofold. First, using a robust object detection model instead of conventional methods and a novel graph-based method, a *unified diagram parsing network* (UDPnet) is proposed to understand a diagram by jointly solving the two tasks of object detection and relation matching, which tackles the first limitation of the existing work. Second, we propose a RNN-based *dynamic graph generation network* (DGGN) to fully exploit the diagram information by describing with a graph structure. To solve the problem, we propose a dynamic adjacency tensor memory (DATM) for the DGGN to store information about the relationships among the elements in a diagram. The DATM has features of both an adjacency matrix in graph theory and a dynamic memory in recent deep learning. With this new type of memory, the DGGN suggests a novel way to propagate information through the structure of a graph. In order to demonstrate the effectiveness of the proposed DGGN, we evaluated our model on a couple of diagram datasets. We also analyzed the inside of GRU [4] cells to observe the dynamics of information in the DGGN.

2. Related Works

Visual relationships: Studies on visual relationships have been emerging in the field of computer vision. This line of research includes detection of visual relationships [18, 14]

and generation of a scene graph [8]. Most of these approaches are based on algorithms for grouping elements by relationships, and aiming to find relationships among the elements. Recently, this research field has focused on the scene graph analysis algorithm, which tackles the problem of understanding general scenes in natural images. Lu *et al.* [18] incorporated language prior to reasoning over a pair of objects and Xu *et al.* [24] solved scene graph inference using GRUs via iterative message passing. Whereas most of the previous studies dealt with natural images, we aim to infer visual relationships and generate a graph based on these relationships. Moreover, our method extracts knowledge from an abstracted diagram by inferring human’s general perception.

Neural networks on a graph: Generalization of neural networks for arbitrarily structured graphs has drawn attention in the last few years. *Graph neural networks* (GNNs) [21] were introduced as an RNN-based model that iteratively propagates nodes in the graph until the nodes reach a stable fixed point. Later, Li *et al.* [15] proposed *gated graph neural networks* (GG-NNs), which apply GRU as an RNN model for the task. In contrast to the RNN-based models, Marino *et al.* [19] proposed *graph search neural network* (GSNN) to build knowledge graphs for multi-label classification problems. GSNN iteratively predicts nodes based on current knowledge by a way of pre-trained ‘importance network’. The main difference between previous methods and ours is that our model can generate a graph structure based on relationships between nodes. Since generating a graph should involve in dynamic establishment or removal of edges between nodes, we also adopt RNN for DGGN as most neural-network-based methods for a graph. The proposed DGGN not only works by message-passing between nodes, but also builds the edges of a graph online, which provides great potential for graph generation and solution of inference problems.

Memory augmented neural network: Since Weston *et al.* [22] proposed a memory network for the question answering problem, a memory augmented network became popular in natural language processing. This memory component has shown great potential to tackle many problems of neural networks such as catastrophic forgetting. In particular, Graves *et al.* applied the memory component in Neural Turing machine [6], and showed that the neural network can update and retrieve memory dynamically. Using this concept, a number of dynamic memory models have been proposed to solve multi-modal problems such as visual question answering [13, 23]. In this paper, we incorporate the scheme of the dynamic memory in DGGN to easily capture and store the graph structure.

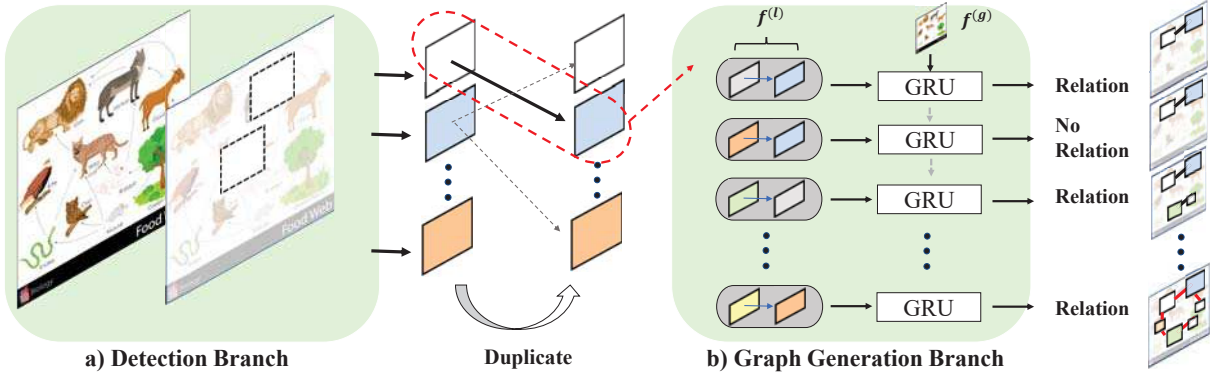


Figure 2. Overview of the *unified diagram parsing network* (UDPnet). In (a) the detection branch, an object detector can extract n objects of 4 different types. Then in order to exploit pairs of objects, we produce n^2 relationship candidates with duplicated objects. (b) In the graph generation branch, we pass local features $f^{(l)}$ from n^2 candidates to the *dynamic graph generation network* (DGGN) with a global feature $f^{(g)}$. In the final step, each relationship candidate can be determined whether it is valid or not. At last, we can establish a relationship graph with nodes and edges.

3. Proposed Method

Figure 2 shows a overall framework of the proposed UDPnet. The proposed network consists of two branches: 1) an object detection network, and 2) a graph generation network handling the relations among the detected objects. In the first branch, a set of objects $O = \{o_i\}_{i=1}^n$ in a diagram image is detected. In the second branch, the relations $R = \{r_j\}_{j=1}^m$ among the objects are generated. We define an object o_i as $\langle \text{location}, \text{class} \rangle$, and a relation r_j in the form of $\langle o_1, o_2 \rangle$. Both branches can be optimized simultaneously by a multi-task learning method in an end-to-end manner. After the optimization process, we can use the generated relational information to solve language-based problems such as question-answering.

3.1. Detecting Constituents in a Diagram

As seen in the Figure 1, various kinds of objects can be included in a diagram depending on the information being conveyed. Those objects are usually described in a simplified manner and the number of object classes is huge, which makes detecting and classifying objects more difficult. In our work, instead of detecting classical object types such as cats and dogs, we define objects in four categorical classes which are adequate for diagrams: blob (individual object), text, arrow head and arrow tail. As a detector, we used SSD [16] which has been reported to have a robust performance.

3.2. Generating a Graph of relationships

3.2.1 Overall Procedure of Graph Generation

In our method, the relation matching for objects in a diagram is conducted by predicting the presence of an edge between a pair of vertices using graph inference. The nodes and edges of a graph match to the objects and the relations of paired objects, respectively. Therefore, the graph is de-

scribed as a bipartite graph,

$$G = (V, E), \quad (1)$$

where $V = X \cup Y$ represents the set of paired disjoint vertices $X \subset V$ and $Y \subset V$, and E denotes edges of the graph each of which connects a pair of nodes $x \in X$ and $y \in Y$. To construct a bipartite graph, we duplicate the detected objects O as O_x and O_y and assume that those two sets are disjoint. Then we predict whether an edge between the nodes $o_x \in O_x$ and $o_y \in O_y$ exists.

The connection between nodes is determined by their spatial relationship and the confidence score for each object class which is provided by the object detector. Note that we do not use convolution features from ROI pooling because there can be various kinds of objects in a diagram, whose shape and texture are hard to be generalized. Instead, we define a feature $f_x \in \mathbb{R}^{13}$ for the object o_x including location (xmin, ymin, xmax, ymax), center point (xcenter, ycenter), width, height and confidence scores. Thus, the relationship between two objects o_x and o_y is described as local feature $f^{(l)} = [f_x, f_y] \in \mathbb{R}^{26}$, and the feature vector $f^{(l)}$ acts as an input to a RNN layer. To prevent the order of local features in a sequence from affecting the performance, we randomly shuffle the order of the features before training in every iteration.

Furthermore, to extract the layout of a diagram and spatial information of all objects, a global feature $f^{(g)}$ is utilized as an input to the RNN. The global feature $f^{(g)} \in \mathbb{R}^{128}$ is constructed by the sum of the convolution feature of conv-7 layer ($256 \times 1 \times 1$) of backbone network in the first branch and the binary mask feature of a diagram (128×1). To match the dimension of conv-7 feature as that of hidden units, we use a fully connected layer in the last step. For the mask feature, we pass the $\mathbb{R}^{n_h \times n_w \times n_c}$ dimensional binary mask map to the 4 layered convolution and max pooling to

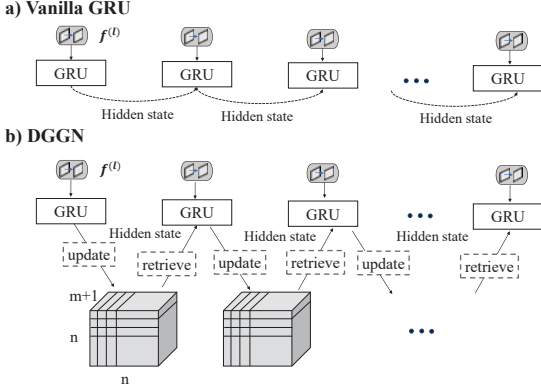


Figure 3. Comparison of the vanilla GRU and the proposed DGGN. (a) In vanilla GRU, information is sequentially transmitted only to a randomly selected next cell. (b) In DGGN, past hidden states are calculated with the dynamic adjacency memory, and the information on the entire graph is propagated in both the update and the retrieval processes simultaneously.

match the dimension to the hidden unit, where n_h and n_w are the width and height of an image, and n_c is the number of object classes.

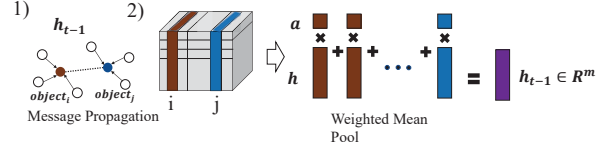
3.2.2 Dynamic Graph Generation Network

In our problem, the local feature vector $f_{i,j}^{(l)}$, ($i, j = 1, \dots, n$) contains the connection information between the nodes $o_i \in X$ and $o_j \in Y$. For simplicity, instead of two indices i and j , we will use one index t to denote the local feature, i.e. $f_t^{(l)}$, ($t = 1, \dots, n^2$). In the previous work [9], vanilla RNN was used and the connection vector $f_t^{(l)}$ was inputted sequentially to train the RNN. The problem is that there is no guarantee that the input $f_t^{(l)}$ will be associated with the $f_{t+1}^{(l)}$ because the vector $f_t^{(l)}$ is randomly shuffled in stochastic gradient training. Besides, while we define this problem as the bipartite graph inference, vanilla RNN could not capture the graph structure and propagate it into the next unit.

To solve the aforementioned problem, we propose the DGGN method which incorporates GRU as a base model. As shown in Figure 3, the proposed method of propagating previous states to the next step is completely different from that of the vanilla GRU. In order to exploit the graph structure, instead of just sequentially transferring features as in vanilla GRU (Figure 3(a)), we aggregate messages from adjacent edges (Figure 3(b)). To pass the messages from adjacent edges, the proposed DGGN requires a dynamic programming scheme which can build the graph structure in an online manner.

In this paper, we incorporate the adjacency matrix in the graph theory which has been mainly used to propagate message through known structure of the graph [21].

a) Retrieve step of DGGN



b) Update step of DGGN

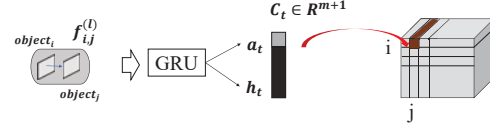


Figure 4. Specific explanations of *update* and *retrieve* steps with the DATM in DGGN. (a) In the retrieve step, past messages are transmitted from adjacent edges (a-1). Specifically, to obtain previous hidden state, we conduct weighted mean pool with extracted matrix at indexes of objects. (b) In the update step, model can store the inferred information into the DATM with a concatenated vector at indexes of input objects.

However, in our problem, the adjacency matrix is unknown, which has to be estimated. Therefore, we propose a dynamic memory component into this problem which holds the connection information among nodes. In this work, we expand 2-dimensional adjacency matrix to 3-dimensional memory. The *dynamic adjacency tensor memory* (DATM) $D \in \mathbb{R}^{n \times n \times (m+1)}$ is defined as a concatenation of the adjacency matrix $A \in \mathbb{R}^{n \times n}$ and the corresponding hidden unit H whose (i, j) element $h_{i,j}$ is an m dimensional hidden vector of the GRU which is related to the connection between the nodes o_i and o_j . The adjacency matrix A represents the connection status between each of n nodes in the directed graph. Each cell in the adjacency matrix only indicates whether the corresponding pair of nodes has a directed arc or not. Then both *retrieve* and *update* steps with tensor D are implemented to aggregate messages from adjacent edges and to build up graph simultaneously.

Retrieve Step: Figure 4(a) shows the *retrieve step* of DGGN. We can get the previous hidden state \hat{h}_{t-1} which collects messages propagated through adjacent edges (Figure 4 (a-1)). In doing so, as shown in Figure 4(a-2) and equation (2), we take average of the adjacent vectors of o_i and o_j weighted by the probability of the existence of an edge. Formally, we extract an adequate hidden unit \hat{h}_t for the input vector $f_{t+1}^{(l)}$ representing the connection with node i and j , as in

$$\hat{h}_t = \sum_{k=1}^n a_{k,i} h_{k,i} + \sum_{k=1}^n a_{k,j} h_{k,j} + f^{(g)}. \quad (2)$$

Here, $a_{i,j}$ represents the (i, j) element of the matrix A , and $h_{i,j} \in \mathbb{R}^m$ is the hidden unit stored in the (i, j) location

of the tensor H . In this step, the probability $a_{i,j}$ works as weights for aggregating messages which represents the philosophy that more credible adjacent edges should give more credible messages. Before transmitted to GRU layer, the global feature $f^{(g)}$ is added to reflect the global shape of the diagram.

Update Step: In the *update step* shown in Figure 4(b), we update the cell D_{ij} with an $m + 1$ length vector that concatenates the output a_t and the hidden state h_t from a GRU cell (8).

$$r_t = \sigma(W_{xr}f_t + W_{hr}\hat{h}_{t-1} + b_r), \quad (3)$$

$$z_t = \sigma(W_{xz}f_t + W_{hz}\hat{h}_{t-1} + b_z), \quad (4)$$

$$\bar{h}_t = \tanh(W_{xh}f_t + W_{hh}(r_t \odot \hat{h}_{t-1}) + b_h), \quad (5)$$

$$h_t = z_t \odot \hat{h}_{t-1} + (1 - z_t) \odot \bar{h}_t, \quad (6)$$

$$a_t = \sigma(W_l h_t + b_l), \quad (7)$$

$$D_{ij} = [a_t, h_t]. \quad (8)$$

Here, $\sigma(\cdot)$ is a sigmoid function. To obtain the hidden state \hat{h}_t , the vectors \hat{h}_{t-1} and $f_t^{(l)}$ are used as previous hidden state and input vectors of the standard GRU, respectively. Update gate z_t has a role to adjust influx of previous information \hat{h}_{t-1} in the GRU cell (6). The binary output a_t is obtained after fully connected layer (7).

3.3. Multi-task Training and Cascaded Inference

In this work, the proposed UDPnet shown in Figure 2 is trained in an end-to-end manner. Because the UDPnet consists of two branches (object detection by SSD and graph generation by DGGN), by nature, the problem is a multi-task learning problem. Thus, different losses for each branches are combined into the overall loss L as follows:

$$L = \alpha L_c + \beta L_l + \gamma L_r. \quad (9)$$

The overall loss is a weighted sum of the classification loss L_c and the location regression loss L_l for the object detection branch, and the relation classification loss L_r for the graph generation network.

As defined in original SSD, the classification loss L_c is the cross-entropy loss over confidences of multiple classes and the location regression loss L_l is a smooth L1 loss [5] between the predicted box and the ground truth box. The relation classification loss L_r is the cross-entropy loss over two classes, adjacent or not. For a faster convergence, we first pre-trained object detection branch alone, then fine-tuned both branches jointly with the overall loss.

During training, matching strategy between the candidates and the ground truths is important for both box detection and relationship inference. To solve the issue, we

set our own strategy for matching candidate pairs and the ground truth. First, given n objects detected at the first branch of object detection, we generate n^2 pairs of relation candidates. For each relation candidate, the two intersection over unions (IOUs), each of which is computed between one of the detected objects and the closest ground truth object, are averaged. Then, each ground truth relationship is matched with the best overlapped relation candidate. To consider the imbalance in the number of detected objects among different diagrams, we should sample the same number of relation candidates from each training diagram.

At inference, we first detect objects in a diagram. Then we apply non maximum suppression (NMS) with an IoU threshold of 0.45 on boxes with scores higher than 0.01. Unlike in training, we should use all boxes that were detected to generate candidate pairs for next branch. Next, we apply graph generation branch to all relation candidates to infer relationship to each other. Finally, we can obtain a diagram graph composed of adjacent edges between nodes with confidence scores higher than 0.1.

After graph inference, we can post-process the generated relational information to further generate knowledge sentences which can be inputs of question answering models. Thus, our methods can make a bridge between visual inference and linguistic reasoning. Actually, we applied proposed pipeline in this paper to Textbook Question Answering competition [10] and the details on the post-processing can be found in the supplementary material.

4. Evaluation

In this section, we validate the performance of the proposed algorithm for the two sub-problems: graph generating and question answering.

Datasets. We performed experiments on two different datasets: AI2D [9] and FOODWEBS [12]. AI2D contains approximately 5,000 diagrams representing scientific topics at an elementary school level. Overall, the AI2D dataset contains class-annotation for more than 118K constituents and 53K relationships among them, including segmentation mask for each of the elements. AI2D also contains more than 15,000 multiple choice questions about diagrams. The polygons for segmentation provided with the AI2D dataset were reshaped into rectangles for simplicity and efficiency. FOODWEBS consists of 490 food web diagrams and 5,208 questions encountered on eighth grade science exams. FOODWEBS focuses on question answering using questions about environmental problems. Unlike AI2D, the diagrams in FOODWEBS do not have annotations for relations among objects, and we used this dataset only as a benchmark of question answering.

Baseline. we used the following ablation models to com-

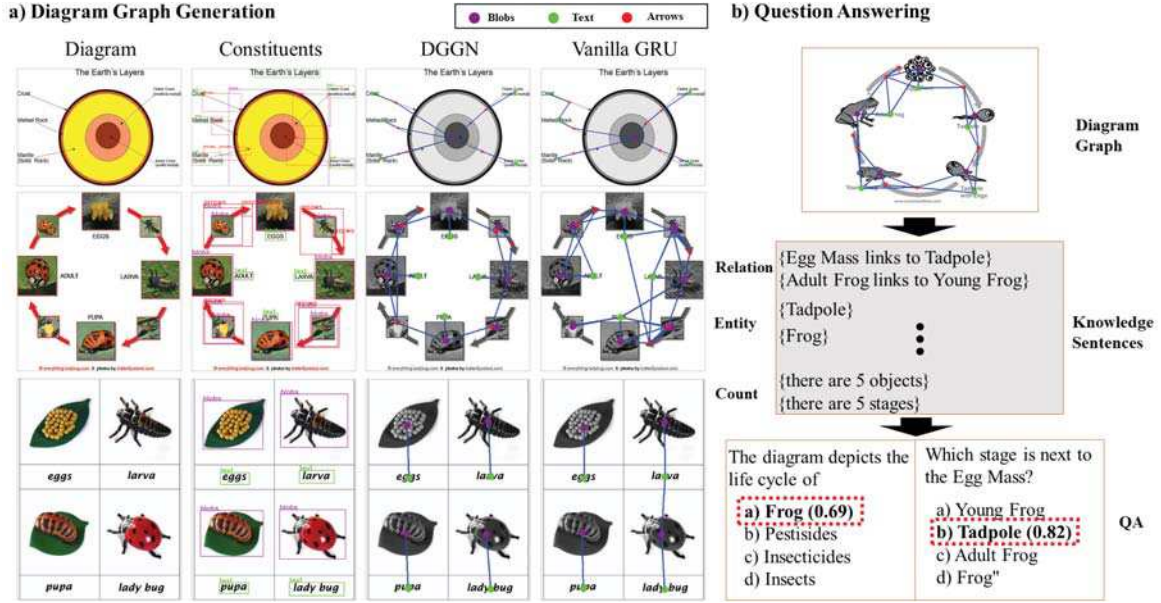


Figure 5. Qualitative results of diagram graph generation and a pipeline to solve question answering problem. (a) Each row shows an example of various kinds of diagram. From the left, original diagrams and diagrams with detected constituents are presented. In last two columns, comparison between the DGGN and the Vanilla GRU with final results is shown. (b) From a diagram graph, we extract knowledge sentences, then solve multi-choice problems.

pare with our method:

- Fully connected layer - only incorporating the object detection branch in our model and replacing the graph generation branch with fully connected layers.
- Vanilla GRU - similar to the previous baseline but using a vanilla GRU instead of the graph generation branch.
- DGGN w/o global feature - exploiting the same structure as our model but excluding the global feature from inputs in the second branch.
- DGGN w/o weighted mean pool - averaging hidden vectors of adjacent edges without multiplying weights which represent the strength of each adjacency.
- DGGN w/ ROI-pooled feature - concatenating a 2×2 ROI-pooled feature in the local feature f , expanding it into a 34 dimensional vector.

Metrics. We propose to measure mean Average Precision (AP) for edge evaluation and IoU for graph completion. First, AP can measure both the recall and precision of a model in predicting the existence of edges. Since our relation candidate should have two boxes, we use average IoUs of those boxes with ground truth boxes as IoU for a relation. We used IoU thresholds $\tau \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ for experiments and report the mean AP by averaging the results of all the thresholds.

Additionally, we adopt an IoU metric to measure completion of entire graph, since JIG metric was not clearly defined in the work of DSDPnet. For both of nodes and

Table 1. Comparison results of AP on the AI2D test set.

Method	mAP	AP_{30}	AP_{50}	AP_{70}
Fully connected layer	8.87	9.22	8.92	8.24
Vanilla GRU	39.28	39.89	43.11	31.54
DGGN				
w/o global feature	39.34	40.51	43.03	31.11
w/o weighted mean pool	42.15	44.22	44.99	34.37
w/ ROI-pooled feature	39.73	43.09	42.19	31.38
DGGN	44.08	44.23	47.13	38.97

edges, we define IoU of node and edge as the number of the intersection divided by the number of the union. Note that we only use the number of overlapped nodes or edges instead of using overlapped area in the original IoU metric.

Implementation Details. We implemented the first branch based on SSDv2 modified from the original SSD. For the second branch, we use 1 layer GRU with 128 hidden states. During training, we sample 160 positive and negative relationship candidates at a ratio of 1 to 7. The training and testing codes are built on Pytorch. Additional experiments about QA on diagrams utilized the implementation¹ under the same conditions of previous work (Dqa-Net) [9].

4.1. Quantitative Results

Table 1 shows comparisons DGGN with baselines on the AI2D dataset. Our results demonstrate that the DGGN out-

¹<https://github.com/allenai/dqa-net>

Table 2. Comparison results of IoU on the AI2D test set.

Method	IoU_{node}	IoU_{edge}
Vanilla GRU	70.06	15.58
DGGN		
w/o global feature	70.95	14.44
w/o weighted mean pool	69.48	24.84
w/ ROI-pooled feature	69.24	23.00
DGGN	69.77	25.86

Table 3. Accuracy of Question Answering on AI2D and FOODWEBS. The results of VQA and DQA-Net(Dsdp) on AI2D and FOODWEBS are refer to [9] and [12], respectively.

Method	AI2D [9]	FOODWEBS [12]
Dqa-Net(GT)	41.55	-
VQA	32.90	56.50
Dqa-Net(Dsdp)	38.47	59.30
Dqa-Net(Ours)	39.73	58.22

performs baselines. In the second row of the table 1, the *Fully connected layer* model shows 8.87 mAP, which is extremely low. This is because the relational information among the nodes (elements) is not reflected to fully connected layer. The *vanilla GRU* shows 39.28 mAP, which is lower than those of any variants of *DGGN*. This implies that the *vanilla GRU* model was not successful for embedding the relational information among the nodes, because the GRU model can only learn the sequential order of the input. In this problem, however, the shuffled order of the relation candidates does not have meaningful sequential knowledge of the relationship.

Next, we performed ablation studies with variants of *DGGN* as presented in the middle of Table 1. In the table, we can see that *DGGN w/o global feature* achieved the largest margin to the best model, and this indicates that the global feature can significantly enhance the performance. On the other hand, the result of *DGGN w/o weighted mean pool* is slightly lower than the best model which shows that weights might not be meaningful to the performance. Interestingly, *DGGN w/ ROI-pooled feature* scored a lower mAP in spite of the additional information. One possible reason is that ROI-pooled feature can cause overfit without a sufficient amount of training data, since objects in diagrams are hard to be generalized.

Table 2 shows comparisons of the modified IoU metric for measuring completion of a graph. In the case of the edge inference, we set 0.5 as the threshold of mean IoU of each predicted box intersecting with a ground truth box and set 0.01 as the threshold of confidence for the adjacency of edges. Since all models use the same SSD model at the object detection branch, results of the IoU_{node} are similar to each other. They have slight different performance because of the end-to-end fine-tuning process. For IoU_{edge} , the *DGGN* shows a better performance than other baselines. Like the results of mAP in Table 1, the usage of global fea-

ture has a significant impact on the performance.

Table 3 shows the results of the question answering experiments conducted on AI2D and FOODWEBS. We only compared to previous works in QA accuracy rather than JIG metric due to the difference of detecting methods. For AI2D, we first evaluated Dqa-net with ground truth annotations of diagrams as our upper bound. Our model shows an accuracy of 39.73% which outperforms previous work and approaches upper bound by 2 % margin. On FOODWEBS, we only deploy on trained model with AI2D and extract diagram graphs from entire data. The results show our model demonstrates comparative results. Overall, our model performs better when compared with the VQA [1] method, which estimates the answer directly from a diagram image. These question answering tests reveal a potential for expansion to the linguistic field. Also, this result is meaningful in that our model is not directly designed to solve the QA problem.

4.2. Qualitative Results

In this section, we analyze qualitative results as shown in Figure 5. Three diagrams which have different layouts and topics are presented to compare qualitatively in Figure 5(a). For example, diagrams for the same topic “life cycle of a ladybug” in the second and third row have different layouts. Nevertheless, our model can understand different layouts and generate graphs according to the intentions of the diagrams. In the second column, the detection results of the object detection branch, finding four kinds of objects (blob, text, arrow and arrow head) in the diagram, are presented. In the third column, we present the results of graph generation on various diagrams. Then we compare our results to those of the baseline (*vanilla GRU*) as shown in the last column. As seen in the results, we confirmed that our model correctly connected the links between the objects according to their intended relation, in most case.

Figure 5(b) shows a sample describing a pipeline of solving question-answering from a diagram graph. After the diagram of “life cycle of a frog” is converted to a relation graph, we can generate knowledge sentences such as “Adult Frog links to Young Frog” with three categories : “relation”, “entity” and “count”. Using those sentences, we solved the multi-choice QA problems. For instance, the second question asks for the relationship among the objects in the diagram. We have already generated a knowledge sentence “Egg Mass links to Tadpole”, so the QA model can easily respond “b) tadpole” with a confidence of 0.82. This process can contribute to the solution of various problems related to knowledge of relationships.

5. Discussion

In this section, we discuss the effectiveness of *DGGN* by investigating the GRU cells, and we analyzed the

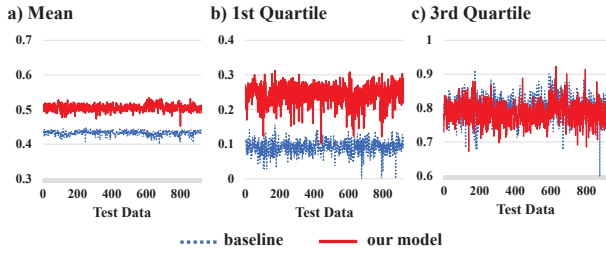


Figure 6. Three statistics of activation value of update gate on AI2D test sets. (a) Mean of activation values. (b) The first quartile statistics of activation values. (c) The third quartile statistics of activation values.

dependency of candidate order of DGGN to compare the results between our model and baseline (vanilla GRU).

Activation of gates. To understand the DGGN better, we analyze information dynamics in DGGN. For this, we extracted the activation values of the update gate. In equation (6), update gate z_t obtained from equation (4) determines the amount of the received information of the cell from the previous \hat{h}_{t-1} . By investigating the graph of the update gate’s activation, we can observe that this model meaningfully exploits messages from the past. Obviously, the more update gates activate, the richer the transmitted information becomes.

We plot three statistics of activation values of update gates using 920 test samples. In Figure 6(a), we presented the mean of activation values which shows the significant margin between our model (red solid line) and the baseline (blue dotted line), and this shows that our model can generally activate update gates more effectively than the baseline does. While the first quartile statistics in Figure 6(b) show a larger margin than the an aforementioned result, the third quartile statistics do not show meaningful differences between our model and the baseline in Figure 6(c). Those two results show that our model encouraged activation in relatively inactive update gates. Overall, we can conclude that DGGN delivers more informative messages based on the graphical structure to GRU cells and induces more influx of information from the past which can lead to better results.

For a study in terms of time steps, we extract activation values of update gates in GRU cells from the second diagram in Figure 5(a). Then we average this quantity over hidden cells with respect to time steps. As shown in Figure 7, our model performs higher than the baseline over almost all the steps. Specifically, almost all the yellow dots in the graph, depicting the candidates of the connected edge, show that the activation values of our model are higher than baseline. Therefore, as we discussed in the previous chapter, the cells of our model successfully infer the relationships by accepting more adjacent information

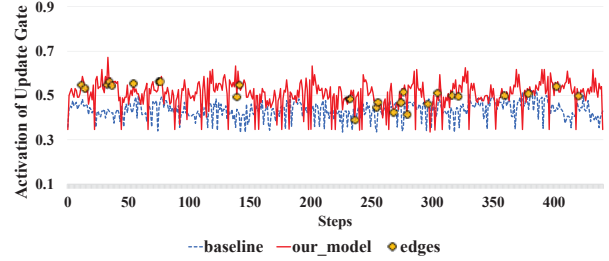


Figure 7. Mean of activation values of update gate on second diagram of Figure 5.

with respect to time steps.

Order of relation candidates. To explore a mechanism of aggregating messages in DGGN, we verified the effect of the order of relation candidates. We evaluated 50 results (AP_{50}) repeatedly with randomly ordered candidates for the baseline and our model on the AI2D. Then we extracted variation statistics from the results. For the baseline, variance and standard deviation of results are $2.27e^{-5}$ and $4.76e^{-3}$, respectively. Our model shows a variance and a standard deviation of results of $1.03e^{-7}$ and $3.22e^{-4}$, respectively. The result shows that the variance and the standard deviation of our model are much lower (around 13 times smaller standard deviation) than those of the baseline.

During the training process, we shuffled the order of candidates before transmitted into GRU cells for both models, to avoid order dependency. However, the statistics show that our model is more robust against the order of relation candidates compared to the baseline. We can confirm that the proposed model successfully extracts the graph structure regardless of the order of the input sequence due to its ability to aggregate messages from the past.

6. Conclusion

In this work, we proposed *UDPnet* and *DGGN* to tackle the problem of understanding a diagram and generating a graph by the neural network. For diagram understanding, we combine an object detector and a network that generates relations among detected objects. A multi-task learning scheme is used to train the *UDPnet* in an end-to-end manner. Moreover, we propose a novel RNN module to propagate message based on graph structure and generate a graph simultaneously. We demonstrated that the proposed *UDPnet* provides state-of-the-art quantitative and qualitative results on problems of generating relation for a given diagram. We also analyzed how our model works better than strong baselines. Our work can be a meaningful step in diagram understanding and reasoning problem beyond natural image understanding. Moreover, we believe that the *DGGN* could benefit other tasks related to graph structure.

References

- [1] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh. Vqa: Visual question answering. *arXiv preprint arXiv:1505.00468*, 2015.
- [2] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.
- [3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.
- [4] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [6] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [8] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3668–3678, 2015.
- [9] A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision*, pages 235–251. Springer, 2016.
- [10] A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [11] I. Kokkinos. Highly accurate boundary detection and grouping. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2520–2527. IEEE, 2010.
- [12] J. Krishnamurthy, O. Tafjord, and A. Kembhavi. Semantic parsing to probabilistic programs for situated question answering. *arXiv preprint arXiv:1606.07046*, 2016.
- [13] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387, 2016.
- [14] Y. Li, W. Ouyang, and X. Wang. Vip-cnn: A visual phrase reasoning convolutional neural network for visual relationship detection. *arXiv preprint arXiv:1702.07191*, 2017.
- [15] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [18] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, pages 852–869. Springer, 2016.
- [19] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.
- [20] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
- [22] J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.
- [23] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*, pages 2397–2406, 2016.
- [24] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. *arXiv preprint arXiv:1701.02426*, 2017.