Situation Recognition with Graph Neural Networks

Ruiyu Li1, Makarand Tapaswi2, Renjie Liao2, Jiaya Jia1,3, Raquel Urtasun2,4,5, Sanja Fidler2,5

1The Chinese University of Hong Kong, 2University of Toronto, 3Youtu Lab, Tencent
4Uber Advanced Technologies Group, 5Vector Institute

ryli@cse.cuhk.edu.hk, {makarand,rjliao,urtasun,fidler}@cs.toronto.edu, leojia9@gmail.com

Abstract

We address the problem of recognizing situations in images. Given an image, the task is to predict the most salient verb (action), and fill its semantic roles such as who is performing the action, what is the source and target of the action, etc. Different verbs have different roles (e.g., attacking has weapon), and each role can take on many possible values (nouns). We propose a model based on Graph Neural Networks that allows us to efficiently capture joint dependencies between roles using neural networks defined on a graph. Experiments with different graph connectivities show that our approach that propagates information between roles significantly outperforms existing work, as well as multiple baselines. We obtain roughly 3-5% improvement over previous work in predicting the full situation. We also provide a thorough qualitative analysis of our model and influence of different roles in the verbs.

1. Introduction

Object [14, 33, 36], action [35, 40], and scene classification [50, 51] have come a long way, with performance in some of these tasks almost reaching human agreement. However, in many real-world applications such as robotics we need a much more detailed understanding of the scene. For example, knowing that an image depicts a repairing action is not sufficient to understand what is really happening in the scene. We thus need additional information such as the person repairing the house, and the tool that is used.

Several datasets have recently been collected for such detailed understanding [22, 27, 47]. In [22], the Visual Genome dataset was built containing detailed relationships between objects. A subset of the scenes were further annotated with scene graphs [17] to capture both unary (e.g., attributes) and pairwise (e.g., relative spatial info) object relationships. Recently, Yatskar et al. [47] extended this idea to actions by labeling action frames where a frame consists of a fixed set of roles that define the action. Fig. 1 shows a frame for action repairing. The challenge then consists of assigning values (nouns) to these roles based on the image content. The number of different role types, their possible values, as well as the number of actions are very large, making it a very challenging prediction task. As shown in Fig. 2, the same verb can appear in very different image contexts, and nouns that fill the roles are vastly different.

In [47], the authors proposed a Conditional Random Field (CRF) to model dependencies between verb-role-noun pairs. In particular, a neural network was trained in an end-to-end fashion to both, predict the unary potentials for verbs and nouns, and to perform inference in the CRF. While their model captured the dependency between the verb and role-noun pairs, dependencies between the roles were not modeled explicitly.

In this paper, we aim to jointly reason about verbs and their roles using a Graph Neural Network (GNN), a generalization of graphical models to neural networks. A GNN defines observation and output at each node in the graph,
2. Related Work

Situation recognition generalizes action recognition to include actors, objects, and location in the activity. There has been work to combine activity recognition with scene or object labels [7, 12, 44, 45]. In [13, 31], visual semantic role labeling tasks were proposed where datasets are built to study action along with localization of people and objects. In another line of work, Yatskar et al. [47] created the imSitu dataset that uses linguistic resources from FrameNet [10] and WordNet [29] to associate images not only with verbs, but also with specific role-noun pairs that describe the verb with more details. As a baseline approach, in [47], a Conditional Random Field (CRF) jointly models prediction of the verb and verb-role-noun triplets. Further, considering that the large output space and sparse training data could be problematic, a tensor composition function was used [46] to share nouns across different roles. The authors also proposed to augment the training data by searching images using query phrases built from the structured situation.

Different from these methods, our work focuses on explicitly modeling dependencies between roles for each verb through the use of different neural architectures.

Understanding Images. There is a surge of interest in joint vision and language tasks in recent years. Visual Question Answering in images and videos [1, 38] aims to answer questions related to image or video content. In image captioning [19, 39, 42, 26], a natural language sentence is generated to describe the image. Approaches for these tasks often use the CNN-RNN pipelines to provide a caption, or a correct answer to a specific question. Dependencies between verbs and nouns are typically being implicitly learned with the RNN. An alternative is to list all important objects with their attributes and relationships. Johnson et al. [17] created scene graphs, which are being used for visual relationship detection [27, 30, 48] tasks. In [25], the authors exploit scene graphs to generate image captions.

In Natural Language Processing (NLP), semantic role labeling [11, 18, 20, 32, 43, 52] involves annotating a sentence with thematic or semantic roles. Building upon resources from NLP, and leveraging collections such as FrameNet [10] and WordNet [29], visual semantic role labeling, or situation recognition, aims to interpret details for one particular action with verb-role-noun pairs.

Graph Neural Networks. There are a few different ways for applying neural networks to graph-structured data. We divide them into two categories. The first group defines convolutions on graphs. Approaches like [2, 6, 21] utilized the graph Laplacian and applied CNNs to spectral domain. Differently, Duvenaud et al. [9] designed a special hash function such that a CNN can be used on the original graphs.

The second group applies feed-forward neural networks to every node of the graph recurrently. Information is propagated through the network by dynamically updating the hidden state of each node based on their history and incoming messages from their neighborhood. The Graph Neural Network (GNN) proposed by [34] utilized multi-layer perceptrons (MLP) to update the hidden state. However, their learning algorithm is restrictive due to the contraction map assumption. In the following work, the Gated Graph Neural Network (GGNN) [23] used a recurrent gating function [4] to perform the update, and effectively learned model parameters using back-propagation through time (BPTT).

Other work [24, 37] designed special update functions based on the LSTM [16] cell and applied the model to tree-structured or general graph data. In [28], knowledge graphs and GGNNs are used for image classification. Here we use GGNNs for situation recognition.

3. Graph-based Neural Models for Situation Recognition

Task Definition. Situation recognition as per the imSitu dataset [47] assumes a discrete set of verbs $V$, nouns $N$, roles $R$, and frames $F$. The verb and its corresponding frame that contains roles are obtained from FrameNet [10], while nouns come from WordNet [29]. Each verb $v \in V$ is associated with a frame $f \in F$ that contains a set of semantic roles $E_f$. Each role $e \in E_f$ is paired with a noun value
shows framework to recognize situations in images. Each image in one direction and updates one “node” per time step. Contrast, a standard unrolled RNN only moves information in the graph at each propagation step. The hidden states are based on its history and incoming messages from its neighbors. For example, in the verb ‘riding’ is associated with three role-noun pairs, i.e., \{agent:dog, vehicle:surfboard, place:sidewalk\}. The hidden state vector that is updated in a recurrent fashion on graphs have prior work. Gated Graph Neural Networks (GGNNs) are one approach that learns the representations of a frame \(f\) with a set of roles \(E_f\). We instantiate a graph \(G_f\) for each image that consists of one verb node, and \(|E_f|\) (number of nodes associated with the frame) role nodes. To capture the dependency between roles to the full extent, we propose creating undirected edges between all pairs of roles. Fig. 3 shows two example graph structures of this type. We explore other edge configurations in the evaluation.

To initialize the hidden states for each node, we use features derived from the image. In particular, for every image \(i\), we compute representations \(\phi_v(i)\) and \(\phi_n(i)\) using the penultimate fully-connected layer of two convolutional neural network (CNN) pre-trained to predict verbs and nouns, respectively. We initialize the hidden states \(h \in \mathbb{R}^D\) of the verb node \(a_v\) and role node \(a_e\) as

\begin{align}
    h^0_{a_v} &= g(W_{tv}\phi_v(i)) \\
    h^0_{a_e} &= g(W_{tn}\phi_n(i) \odot W_e e \odot W_v \hat{v}),
\end{align}

where \(\hat{v} \in \{0, 1\}^{|\mathbb{V}|}\) corresponds to a one-hot encoding of the predicted verb and \(e \in \{0, 1\}^{|\mathbb{R}|}\) is a one-hot encoding of the role that the node \(a_e\) corresponds to. \(W_v \in \mathbb{R}^{D \times |\mathbb{V}|}\) is the verb embedding matrix, and \(W_e \in \mathbb{R}^{D \times |\mathbb{R}|}\) is the role embedding matrix. \(W_v\) and \(W_e\) are parameters that transform image features to the space of hidden representations. \(\odot\) corresponds to element-wise multiplication, and \(g(\cdot)\) is a non-linear function such as \(\tanh(\cdot)\) or ReLU \((g(x) = \max(0, x))\). We normalize the initialized hidden states to unit-norm prior to propagation.

For any node \(a\), at each time step, the aggregation of incoming messages at time \(t\) is determined by the hidden states of its neighbors \(a'\):

\[ x_a^t = \sum_{(a', a) \in \mathcal{B}} W_p h_{a'}^{t-1} + b_p. \]

Note that we use a shared linear layer of weights \(W_p\) and biases \(b_p\) to compute incoming messages across all nodes.

After aggregating the messages, the hidden state of the node is updated through a gating mechanism similar to the Gated Recurrent Unit [4, 23] as follows:

\begin{align}
    z_a^t &= \sigma(W_z x_a^t + U_z h_a^{t-1} + b_z), \\
    r_a^t &= \sigma(W_r x_a^t + U_r h_a^{t-1} + b_r), \\
    \tilde{h}_a^t &= \tanh(W_h x_a^t + U_h (r_a^t \odot h_a^{t-1}) + b_h), \\
    h_a^t &= (1 - z_a^t) \odot \tilde{h}_a^t + z_a^t \odot h_a^{t-1}.
\end{align}

This allows each node to softly combine the influence of the aggregated incoming message and its own memory. \(W_z, U_z, b_z, W_r, U_r, b_r, W_h, U_h, b_h\) are the weights and biases of the update function.

Output and Learning. We run \(T\) propagation steps. After propagation, we extract node-level outputs from GGNN.
to predict the verb and nouns. Specifically, for each image, we predict the verb and a set of nouns for each role associated with the verb frame using a softmax layer:

$$p_v = \sigma(W_{hv} h_a + b_v)$$  \hspace{1cm} (5)$$
$$p_{c:n} = \sigma(W_{hn} h_a + b_h) .$$  \hspace{1cm} (6)$$

Note that the softmax function $\sigma$ is applied across the class space for verbs $V$ and nouns $N$. $p_{c:n}$ can be treated as the probability of assigning noun $n$ to role $e$.

Each image $i$ in the $imSitu$ dataset comes with three sets of annotations (from three annotators) for the nouns. During training, we accumulate the cross-entropy loss at verb and noun nodes for every annotation as

$$L = \sum_i \sum_{j=1}^3 \left( y_v \log(p_v) + \frac{1}{|E_f|} \sum_e y_{c:n} \log(p_{c:n}) \right) ,$$  \hspace{1cm} (7)$$

where $y_v$ and $y_{c:n}$ correspond to the ground-truth verb for image $i$ and the ground-truth noun for role $e$ of the image, respectively. Different to the Soft-OR loss in [47], we encourage the model to predict all three annotations for each image. We use back-propagation through time (BPTT) [41] to train the model.

**Inference.** At test time, our approach first predicts the verb $\hat{v} = \arg \max_v p_v$ to choose a corresponding frame $f$ and obtain the set of associated roles $E_f$. We then propagate information among role nodes and choose the highest scoring noun $\hat{n}_e = \arg \max_n p_{c:n}$ for each role. Thus our predicted situation is

$$\hat{S} = (\hat{v}, \{(e, \hat{n}_e) : e \in E_f\}) .$$  \hspace{1cm} (8)$$

To reduce reliance on the quality of verb prediction, we explore beam search over verbs as discussed in Experiments.

**3.2. Simpler Graph Architectures**

An alternative to model dependencies between nodes is to use recurrent neural networks (RNN). Here, situation recognition can be considered as a sequential prediction problem of choosing the verb and corresponding noun-role pairs. The hidden state of the RNN carries information across the verb and noun-role pairs, and the input at each time-step dictates what the RNN should predict.

**Chain RNN.** An unrolled RNN can be seen as a special case of a GGNN, where nodes form a chain with directed edges between them. However, there are a few notable differences, wherein the nodes receive information only once from their (left) neighbor. In addition, the nodes do not perform $T$ steps of propagation among each other and predict output immediately after the information arrives.

In the standard chain structure of a RNN, we need to manually specify the order of the verb and roles. As the choice of the verb dictates the set of roles in the frame, we predict the verb at the first time step. We observe that the $imSitu$ dataset and any verb-frame in general, commonly consist of $place$ and $agent$-like roles (e.g. semantic role teacher can be considered as the agent for the verb teaching). We thus predict $place$ and $agent$ roles as the second and third roles in the chain $^1$. We make all other roles for the frame to follow subsequently in descending order of the number of times they occur across all verb-frames. Fig. 4 shows an example of such a model.

For a fair comparison to the fully connected roles GGNN, we employ the GRU update in our RNN. The input to the hidden states matches node initialization (Eqs. 1 and 2). We follow the same scheme for predicting the output (linear layer with softmax), and train the model with the same cross-entropy loss.

**Tree-structured RNN.** As mentioned above, the $place$ and $agent$ semantic roles occur more frequently. We propose a structure where they have a larger chance to influence prediction of other roles. In particular, we create a tree-structured RNN [37] where the hidden states first predict the verb, followed by agent and place, and all other roles. Fig. 5 shows examples of resulting structures.

The tree-structured RNN can be deemed as a special case of GGNN, where nodes have the following directed edges:

$$B = \{(a, a') : a' \in Z\} \cup \{(a', a) : a' \in Z, a \in E_f \backslash Z\} ,$$  \hspace{1cm} (9)$$

where $Z = \{agent, place\}$, and $E_f \backslash Z$ represents all roles in that frame other than agent and place. Similar to the chain RNN, we use GRU update and follow the same learning and inference procedures.

$^1$Predicting place requires a more global view of the image compared to agent. Changing the order to verb $\rightarrow$ agent $\rightarrow$ place $\rightarrow$ ... results in 1.9% drop of performance.
Table 1. Situation prediction results on the development set. We compare several variants of our fully-connected roles model to show the improvements achieved at every step. $T$ refers to the number of time-steps of propagation in the fully connected roles GGNN (FC Graph). $\text{BS}=10$ indicates the use of beam-search with beam-width of 10. $\text{vOH}$ (verb, one-hot) is included when the embedding of the predicted verb is used to initialize the hidden state of the role nodes. $g=\text{ReLU}$ refers to the non-linear function used after initialization. All other rows use $g=\tanh()$. Finally, Soft-OR refers to the loss function used in [47]. Best performance is in bold and second-best is italicized.

<table>
<thead>
<tr>
<th>Method</th>
<th>top-1 predicted verb</th>
<th>top-5 predicted verbs</th>
<th>ground truth verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verb</td>
<td>value</td>
<td>value-all</td>
</tr>
<tr>
<td>1 Unaries</td>
<td>36.32</td>
<td>23.74</td>
<td>13.86</td>
</tr>
<tr>
<td>2 Unaries, $\text{BS}=10$</td>
<td>36.39</td>
<td>23.74</td>
<td>14.01</td>
</tr>
<tr>
<td>3 FC Graph, $T=1$</td>
<td>36.25</td>
<td>25.99</td>
<td>17.02</td>
</tr>
<tr>
<td>4 FC Graph, $T=2$</td>
<td>36.43</td>
<td>26.08</td>
<td>17.22</td>
</tr>
<tr>
<td>5 FC Graph, $T=4$</td>
<td>36.46</td>
<td>26.26</td>
<td>17.48</td>
</tr>
<tr>
<td>6 FC Graph, $T=4$, $\text{BS}=10$</td>
<td>36.70</td>
<td>26.52</td>
<td>17.70</td>
</tr>
<tr>
<td>7 FC Graph, $T=4$, $\text{BS}=10$, $\text{vOH}$</td>
<td>36.93</td>
<td>27.52</td>
<td>19.15</td>
</tr>
<tr>
<td>8 FC Graph, $T=4$, $\text{BS}=10$, $\text{vOH}$, $g=\text{ReLU}$</td>
<td>36.26</td>
<td>27.22</td>
<td>19.10</td>
</tr>
<tr>
<td>9 FC Graph, $T=4$, $\text{BS}=10$, $\text{vOH}$, Soft-OR</td>
<td>36.75</td>
<td>27.33</td>
<td>18.94</td>
</tr>
</tbody>
</table>

4. Evaluation

We evaluate our methods on the imSitu dataset [47] and use the standard splits with 75k, 25k, and 25k images for the train, development, and test subsets, respectively. Each image in imSitu is associated with one verb and three annotations for the role-noun pairs.

We follow [46] and report three metrics: (i) verb: the verb prediction performance; (ii) value: the semantic verb-role-value tuple prediction performance that is considered to be correct if it matches any of the three ground truth annotations; and (iii) value-all: the performance when the entire situation is correct and all the semantic verb-role-value pairs match at least one ground truth annotation.

4.1. Implementation Details

Image Representations. We adopt two pre-trained VGG-16 CNNs [36] for extracting image features by removing the last fully-connected and softmax layers, and fine-tuning all weights. The first CNN ($\phi_v(i)$) is trained to predict verbs, and second CNN ($\phi_n(i)$) predicts the top $K$ most frequent nouns ($K = 2000$ cover about 95% of nouns) in the dataset.

Unaries. Creating a graph with no edges, or equivalently with $T = 0$ steps of propagation corresponds to using the initialized features to perform prediction. We refer to this approach as Unaries, which will be used as the simplest baseline to showcase the benefit of modeling dependencies between the roles.

Learning. We implement the proposed models in Torch [5]. The network is trained using RMSProp [15] with mini-batches of 256 samples. We choose the hidden state dimension $D = 1024$, and train image ($W_{iv}$, $W_{in}$), verb ($W_v$) and role ($W_r$) embeddings. The image features are extracted before training the GGNN or RNN models.

The initial learning rate is $10^{-3}$ and starts to decay after 10 epochs by a factor of 0.85. We use dropout with a probability of 0.5 on the output prediction layer (c.f. Eqs. 5 and 6) and clip the gradients to range $(-1, 1)$.

Mapping agent Roles. The imSitu dataset [47] has situations for 504 verbs. Among them, we notice that 19 verbs do not have the semantic role $\text{agent}$ but instead with roles of similar meaning (e.g. verb $\text{educating}$ has role $\text{teacher}$). We map these alternative roles to $\text{agent}$ when determining their position in the RNN architecture. Such a mapping is not used for the fully connected GGNN model.

Variable Number of Roles. A verb has a maximum of 6 roles associated with it. We implement our proposed model with fixed-size graphs involving 7 nodes. To deal with verbs with less than 6 roles, we zero the hidden states at each time-step of propagation, making them not receive or send any information.

4.2. Results

We first present a quantitative analysis comparing different variants of our proposed model. We then evaluate the performance of different architectures, and compare results with state-of-the-art approaches.

Ablative Analysis A detailed study of the GGNN model with fully connected roles (referred to as FC Graph) is shown in Table 1. An important hyper-parameter for the GGNN model is the number of propagation steps $T$. We found that the performance increases by a small amount when increasing $T$, and saturates soon (in rows 3, 4, and 5). We believe that this is due to the use of a fully-connected graph, and all nodes sharing most of the information at the first-step propagation. Nevertheless, the propagation is important, as revealed in the comparison between Unaries ($T = 0$) from row 1 and $T = 1$ in row 3. We obtain a mean improvement of 3.8% in all metrics.

During test we have the option of using beam search, where we hold $B$ best verb predictions and compute the
role-noun predictions for each of the corresponding graphs (frames). Finally, we select the top prediction using the highest log-probability across all $B$ options. We use a beam width of $B = 10$ in our experiments, which yields small improvement. Rows 1 and 2 of Table 1 show the improvement using beam search on a graph without propagation. Rows 5 and 6 show the benefit after multiple steps of propagation.

Rows 6 and 7 of Table 1 demonstrate the impact of using embeddings of the predicted verb ($v_{OH}$) to initialize the role nodes’ hidden states in Eq. (2). Notable improvement is obtained when using the ground-truth verb (3-4%). The value-all for the top-1 predicted verb increases from 17.70% to 19.15%. We also tested different non-linear functions for initialization, i.e., tanh (row 7) or ReLU (row 8), however, the impact is almost negligible. We thus use tanh for all experiments.

Finally, comparing rows 7 and 9 of Table 1 reveals that our loss function to predict all annotations in Eq. (7) performs slightly better than the Soft-OR loss that aims to fit at least one of the annotations [47].

**Baseline RNNs.** Table 2 summarizes the results with different structures on the dev set. As expected, Unaries perform consistently worse than models with information propagation between nodes on the value and value-all metrics. The Tree-structured RNN provides a 2% boost in value-all for top-1 predicted verb, while the Chain RNN provides a 3.9% improvement. Owing to the better connectivity between the roles in a Chain RNN (especially place and agent), we observe better performance compared to the

<table>
<thead>
<tr>
<th>Model</th>
<th>top-1 predicted verb</th>
<th>top-5 predicted verbs</th>
<th>ground truth verbs</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verb</td>
<td>value</td>
<td>value-all</td>
<td>verb</td>
</tr>
<tr>
<td>Unaries</td>
<td>36.39</td>
<td>23.74</td>
<td>14.01</td>
<td>61.65</td>
</tr>
<tr>
<td>Chain RNN</td>
<td>34.62</td>
<td>24.67</td>
<td>17.94</td>
<td>60.09</td>
</tr>
<tr>
<td>Tree-structured RNN</td>
<td>34.62</td>
<td>24.24</td>
<td>16.04</td>
<td>58.86</td>
</tr>
<tr>
<td>Chain GGNN, $T=8$</td>
<td>36.63</td>
<td>27.27</td>
<td>19.03</td>
<td>68.40</td>
</tr>
<tr>
<td>Tree-structured GGNN, $T=6$</td>
<td>36.78</td>
<td>27.48</td>
<td>19.54</td>
<td>68.45</td>
</tr>
<tr>
<td>Fully-connected GGNN, $T=4$</td>
<td>36.93</td>
<td>27.52</td>
<td>19.15</td>
<td>68.40</td>
</tr>
</tbody>
</table>

Table 2. Situation prediction results on the development set for models with different graph structures. All models use beam search, predicted verb embedding, and $g = \tanh(\cdot)$. Best performance is highlighted in **bold**, and second-best in each table section is *italicized.*

<table>
<thead>
<tr>
<th>Model</th>
<th>top-1 predicted verb</th>
<th>top-5 predicted verbs</th>
<th>ground truth verbs</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verb</td>
<td>value</td>
<td>value-all</td>
<td>verb</td>
</tr>
<tr>
<td>CNN+CRF [47]</td>
<td>32.25</td>
<td>24.56</td>
<td>14.28</td>
<td>58.64</td>
</tr>
<tr>
<td>Tensor Composition [46]</td>
<td>32.91</td>
<td>25.39</td>
<td>14.87</td>
<td>59.92</td>
</tr>
<tr>
<td>Tensor Composition + DataAug [46]</td>
<td>34.20</td>
<td>26.56</td>
<td>15.61</td>
<td><strong>62.21</strong></td>
</tr>
<tr>
<td>Chain RNN</td>
<td>34.62</td>
<td>24.67</td>
<td>17.94</td>
<td>61.09</td>
</tr>
<tr>
<td>Fully-connected Graph</td>
<td>36.93</td>
<td>27.52</td>
<td>19.15</td>
<td>61.80</td>
</tr>
</tbody>
</table>

Table 3. We compare situation prediction results on the development and test sets against state-of-the-art models. Each model was run on the test set only once. Our model shows significant improvement in the top-1 prediction on all metrics, and performs better than a baseline that uses data augmentation. The performance improvement on the value-all metric is important for applications, such as captioning and QA. Best performance is highlighted in **bold**, and second-best is *italicized.*
Tree-structured RNN. Note that as the RNNs are trained jointly to predict both verbs and nouns, and as the noun gradients dominate, the verb prediction takes a hit.

Different Graph Structures. We can also use chain or tree-structured graphs in GGNN. Along with the FC graph in row 6 of Table 2, rows 4 and 5 present the results for different GGNN structures. They show that connecting roles with each other is critical and sharing information helps. Interestingly, the Chain GGNN needs more propagation steps \( (T=8) \), as it takes time for the left-most and right-most nodes to share information. Smaller values of \( T \) are possible when nodes are well-connected as in Tree-structured \( (T=6) \) or FC Graph \( (T=4) \). Fig. 6 presents prediction from all models for two images. The FC Graph is able to reason about associating cheese and pizza rather than sprinkling meat or food on it.

Comparison with State-of-the-art. We compare the performance of our models against state-of-the-art on both the dev and test sets in Table 3. Our CNN predicts the verb well. Beam search leads to even better performance (2-4% higher) in verb prediction. We note that Tensor Composition + DataAug actually uses more data to train models. Nevertheless, we achieve the best performance on all metrics when using the top-1 predicted verb.

Another advantage of our model is in improvement for the value-all metric. It yields +8% when using the ground-truth verb, +6% with top-5 predicted verbs, and +4.5% with top-1 predicted verb, compared with the baseline without data augmentation. Interestingly, even with data augmentation, we outperform [46] by 3-4% in value-all for top-1 predicted verb. This property attributes to information sharing between role nodes, which helps in correcting errors and better predicts frames. Note that value-all is an important metric to measure a full understanding of the image. Models with higher value-all will likely lead to better captioning or question-answering results.

4.3. Further Discussion

We delve deeper into our model and discuss why the FC Graph outperforms baselines.

Learned Structure. A key emphasis of this model is on information propagation between roles. In Fig. 7, we present the norms of the propagation matrices. Each element in the matrix \( P(a',a) \) is the norm of the incoming message from role \( a' \) to \( a \) averaged across all images (in dev set) at the first time-step, i.e., \( \| x_{T \times t = 1} \| \) regarding Eq. (3). In this example, tool is important for the verb fastening and influences all other roles, while agent and obstacle influence roles in jumping.

Wrong Verb Predictions. We present a few examples of top scoring results where the verb prediction is wrong in Fig. 8. Note that in fact these predicted verbs are plausible options for the given images. The metric value treats them as wrong, and yet we can correctly predict the role-noun pairs. One example is the middle one of slouching vs. sitting. Fig. 8 (bottom) shows that choosing a different verb might lead to the selection of different roles (goalitem vs. item, destination). Nevertheless, predicting book for browsing is a good choice.
Figure 9. Images with top-1 predictions from the development set. For all samples, the predicted verb is correct, shown below the image in bold. Roles are marked with a blue background, and predicted nouns are in green when correct, and red when wrong.

### Predictions with Correct Verb.

Fig. 9 shows several examples of prediction obtained by FC Graph, where the predicted verb matches the ground-truth one. The top row corresponds to samples where the metric value-all scores correctly as all role-noun pairs are correct. Note that the roles are closely related (e.g. (agent, clungto) and (material, dye)) and help each other choose the correct nouns. In the bottom row, we show some failure cases in predicting role-noun pairs. First, the model favors predicting place as outdoor (a majority of place is outdoor in the training set). Second, for the sample with verb picking, we predict the crop as apple, which appears 79 times in the dataset compared with cotton that appears 14 times. Providing more training samples (e.g. [46]) could help remedy such issues.

In the latter three samples of the bottom row, although the model makes reasonable predictions, they do not match the ground-truth. For example, the ground-truth annotation for the verb taxiing is agent:jet and for the verb camping is agent:persons. Therefore, even though each image comes with three annotations, synonymous nouns and verbs make the task still challenging.

### 5. Conclusion

We presented an approach for recognizing situations in images that involves predicting the correct verb along with its corresponding frame consisting of role-noun pairs. Our Graph Neural Network (GNN) approach explicitly models dependencies between verb and roles, allowing nouns to inform each other. On a benchmark dataset imSitu, we achieved ~4.5% accuracy improvement on a metric that evaluates correctness of the entire frame (value-all). We presented analysis of our model, demonstrating the need to capture the dependencies between roles, and compared it with RNN models and other related solutions.

### 6. Acknowledgements

This work is in part supported by a grant from the Research Grants Council of the Hong Kong SAR (project No. 413113). We also acknowledge support from NSERC, and GPU donations from NVIDIA.
References


