Transforming Visual Scene Graphs to Image Captions

Xu Yang[1] Jiawei Peng[1] Zihua Wang[1] Haiyang Xu[2*] Qinghao Ye [2]
Chenliang Li[2] Songfang Huang[2] Fei Huang[2] Zhangzikang Li[1] Yu Zhang[1*]

[1] School of Computer Science & Engineering, Key Lab of Computer Network
& Information Integration (Ministry of Education), Southeast Univ., Nanjing, China
[2]Alibaba Group

{101013120, pengjiawei, zihua, zhang_yu}@seu.edu.cn,{shuofeng.xhy, yeqinghao.yqh,
lcl193798, songfang.hsf}@alibaba-inc.com, {fairhuang, lizhangzikang}@gmail.com

Abstract

We propose to TransForm Scene Graphs into more descriptive Captions (**TFSGC**). In TFSGC, we apply multi-head attention (MHA) to design the Graph Neural Network (GNN) for embedding scene graphs. After embedding, different graph embeddings contain diverse specific knowledge for generating the words with different part-of-speech, *e.g.*, object/attribute embedding is good for generating nouns/adjectives. Motivated by this, we design a Mixture-of-Expert (MOE)-based decoder, where each expert is built on MHA, for discriminating the graph embeddings to generate different kinds of words. Since both the encoder and decoder are built based on the MHA, as a result, we construct a **simple and homogeneous** encoder-decoder unlike the previous **heterogeneous** ones which usually apply Fully-Connected-based GNN and LSTM-based decoder. The homogeneous architecture enables us to unify the training configuration of the whole model instead of specifying different training strategies for diverse sub-networks as in the heterogeneous pipeline, which releases the training difficulty. Extensive experiments on the MS-COCO captioning benchmark validate the effectiveness of our TFSGC. The code is in: https://anonymous.4open.science/r/ACL23_TFSGC.

1 Introduction

Image captioning, which aims to generate one sentence for describing multi-aspects of an image, has made huge progress since the proposal of the encoder-decoder framework (Vinyals et al., 2015; Xu et al., 2015). Such a framework contains one visual encoder to extract a series of visual features from the image and one language decoder to generate captions from the extracted visual features. Since the visual encoder is usually well pre-trained by image classification and object detection, the extracted features contain abundant knowledge of the object categories, which enables the captioning model to generate object-abundant captions.

However, object category is not the only visual pattern that matters for high-quality captions (Anderson et al., 2018; Jiang et al., 2020), object attributes and relations also play significant roles in generating descriptive captions, *i.e.*, the caption containing multi-aspects of an image. Motivated by this, researchers propose to incorporate additional semantic knowledge, *e.g.*, object categories, attributes, and relations, into the captioning models by using the scene graph as the mediator (Yao et al., 2018; Yang et al., 2020). Scene graphs assign each object node with certain attribute nodes and some pairwise objects with certain relation nodes. These nodes are represented by the corresponding semantic tags, *e.g.*, as shown in Fig. 1, the object "dog" is assigned with the attribute "black" and the pairwise objects "dog" and "fish" have the relation "bite" in between. To exploit the scene graph, Graph Neural Network (GNN) (Battaglia et al., 2018) is deployed to embed the graphs and the output embeddings are input to the decoder for captioning.

The top part of Fig. 1 shows the pipeline of the previous popular GNN-based captioning model (Yao et al., 2018; Yang et al., 2020), which implements GNN as a few Fully-Connected (FC) and non-linear activation layers. To update the node embedding, this GNN maps the concatenated neighbour embeddings into the new one (Xu et al., 2019). Then the updated graph embeddings are input into the language decoder that contains a few LSTM layers and an attention module. The LSTM layers are used to generate the context vector based on the partially generated captions. This context vector works as the query in the attention module for determining which graph embeddings should be used to generate the next word. Compared with the models without GNN, this GNN-LSTM pipeline usually gets better performance.

*Corresponding authors.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 12427–12440

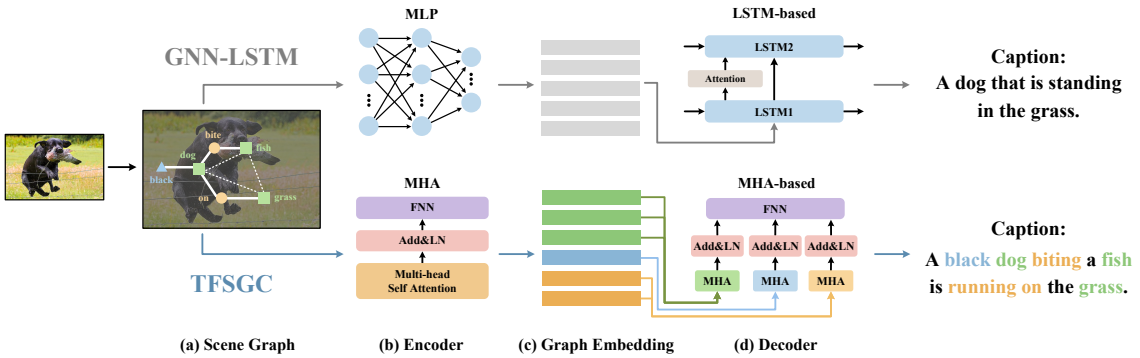

Figure 1: Comparison between traditional heterogeneous GNN-LSTM (top part) and our homogeneous TFSGC model (bottom part). In GNN-LSTM, they use MLP-based GNN and do not discriminate the graph embeddings (grey colour in (c) is used to strengthen such indiscrimination). In TFSGC, we use MHA to design the GNN and the decoder and discriminate diverse graph embeddings (different colours in (c) are used to strengthen such discrimination).

However, this GNN-LSTM framework implies two flaws which hinder the further improvement of applying scene graphs. First, FC-based GNN and LSTM do not share the same building blocks and thus the constructed model is a **heterogeneous** structure, which requires well-chosen training strategies, *e.g.*, choosing different learning rates or optimizers for different sub-networks, to achieve the best performance (Yang et al., 2020). Finding such training configurations is a labour-intensive process. Second, the graph embeddings are indiscriminately selected during captioning (the grey embeddings in Fig. 1 top (c) denote such indiscrimination), which causes less descriptive captions. While intuitively, different kinds of node embeddings should be used for generating the words with diverse part-of-speech (POS), *e.g.*, the object/attribute/relation embeddings should be more responsible for the nouns/adjectives/verbs, respectively (Yang et al., 2019b).

To alleviate the above-mentioned flaws, we propose a novel **homogeneous** captioning model to Transform Scene Graphs (**TFSGC**) into captions. Our TFSGC is built based on the Transformer (Vaswani et al., 2017) since it is more powerful than LSTM in image captioning (Herdade et al., 2019; Li et al., 2019; Cornia et al., 2020). TFSGC is homogeneous since we use multi-head attention (MHA) to design both the graph encoder to embed the scene graphs and the language decoder to generate the caption.

Our design principle is quite simple where we do not need to revise the self-attention operation but only need to reformulate the input data structure. Specifically, to design GNN by MHA, we first linearize the scene graph into a token sequence and introduce a binary mask to index which two nodes are connected in the graph. Then we use the masked MHA operation to deal with this linearized token sequence for graph embedding. In this process, each token embedding is added by a learnable type embedding to index the token type (*e.g.*, object/attribute/relation) and we will show that such type embedding can help distinguish the edge type during the attention calculation.

After graph operation, we get a series of object/attribute/relation embeddings, which will be used in the decoder for captioning. To make the decoder discriminate different embeddings for generating different words, we learn from MOE networks (Jacobs et al., 1991; Xue et al., 2022; Du et al., 2022) to revise the original Transformer decoder with two strategies. First, as Fig. 1 bottom (d) shows, we use three encoder-decoder attention layers, which are built on MHA, as three experts to address object/attribute/relation embeddings, respectively. Second, we incorporate an attention-based soft routing network to discriminate which kinds of embeddings should be more responsible for generating the next word. Both the MOE-decoder and the type embedding in the encoder help distinguish node embeddings for better captions. We carry exhaustive ablation studies and comparisons to validate the effectiveness of TFSGC and it achieves 132.3/138.6/139.5 CIDEr scores when using BUTD/Patch/VinVL features.

2 Related Work

Image Captioning. For a long time, the attention-based CNN-LSTM pipeline (Vinyals et al., 2015;

Xu et al., 2015) is the most popular backbone for captioning and various techniques have been added into it for better performance, including building stronger visual encoders (Lu et al., 2018; Jiang et al., 2020), designing more advanced attention mechanisms (Anderson et al., 2018; Wang et al., 2020), incorporating semantic knowledge (You et al., 2016; Gan et al., 2017; Yao et al., 2018; Yang et al., 2020), and exploiting language structure (Lu et al., 2017; Yang et al., 2019b).

Recently, Transformer (Vaswani et al., 2017) has gradually substituted LSTM as the mainstream language decoder in image captioning (Herdade et al., 2019; Li et al., 2019) since it achieves better performances than the LSTM-based models. Based on this new backbone, researchers develop more advanced strategies for further improving the effectiveness, including designing more sophisticated attention mechanisms (Huang et al., 2019; Pan et al., 2020), introducing additional memory blocks (Cornia et al., 2020; Yang et al., 2021b), distilling knowledge from the large-scale pre-training models (Radford et al., 2021; Li et al., 2021; Xu et al., 2021), and exploiting Transformer-based visual encoders (Wang et al., 2022; Fang et al., 2022), modularized design for large-scale multi-modal pretraining (Li et al., 2022; Xu et al., 2023; Ye et al., 2023). Since the recently proposed SOTA models use Transformer as the backbone, we also built TFSGC based on Transformer for fair comparison.

Graph Neural Network (GNN). Scene Graph abstracts the major visual patterns in a visual scene as a graph. It is usually used as the mediator to narrow the gap between the vision and the language domains. To incorporate scene graphs into deep networks, GNN (Battaglia et al., 2018) is used to embed the discrete node labels into dense embeddings. However, most of the previous GNNs are MLP-based (Yang et al., 2020; Yao et al., 2018; Xu et al., 2019; Milewski et al., 2020; Zhong et al., 2020), which may limit the effectiveness of embedding scene graphs in a Transformer architecture. In our research, we design an MHA-based GNN to remedy this limitation. Moreover, noisy scene graphs may damage the performances (Nguyen et al., 2021), so we use better scene graph parsers to minimize the impact of noise on our model.

Mixture of Experts (MOE). The major idea of MOE is to construct a network with lots of experts where different experts deal with diverse samples (Jacobs et al., 1991; Shazeer et al., 2017). When a sample is input to the MOE network, a routing network will decide which experts should be more responsible for this input. Thus MOE naturally fits our case where we hope diverse experts can discriminate graph embeddings for generating the words with different POS. Different from the existent MOE-based Transformer (Lepikhin et al., 2020; Xue et al., 2022; Du et al., 2022) which applies various feed-forward networks as different experts, we set three encoder-decoder attention layers as different experts where the query is set to the same context vector while the key and value are set to object/attribute/relation embeddings.

3 Revisiting of Transformer

We first revisit the Transformer-based captioning model and then introduce how to revise it to get our TFSGC in the next section. Transformer-based model contains a visual encoder to calculate the contexts of the extracted visual features and the output embeddings will be input into a language decoder for captioning. For both the encoder and the decoder, the most elemental building block is the multi-head attention (MHA). Given the query, key, and value matrices: $\boldsymbol{Q} \in \mathbb{R}^{N_Q \times d}$, $\boldsymbol{K} \in \mathbb{R}^{N_K \times d}$, $\boldsymbol{V} \in \mathbb{R}^{N_V \times d}$, MHA calculates the output $\boldsymbol{Y} = \mathbf{MHA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ as:

$$\begin{aligned}
\textbf{Input:}\quad & \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \\
\textbf{Att:}\quad & \boldsymbol{A}_i = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{W}_i^Q(\boldsymbol{K}\boldsymbol{W}_i^K)^T}{\sqrt{d}}) \\
\textbf{Head}:\quad & \boldsymbol{H}_i = \boldsymbol{A}_i\boldsymbol{V}\boldsymbol{W}_i^V, \\
\textbf{Multi-Head:}\quad & \boldsymbol{H} = [\boldsymbol{H}_1, \boldsymbol{H}_2, ..., \boldsymbol{H}_h]\boldsymbol{W}^H, \\
\textbf{Output:}\quad & \boldsymbol{Y} = \text{LN}(\boldsymbol{H} + \boldsymbol{Q}),
\end{aligned} \tag{1}$$

where $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{d \times d_h}$, and $\boldsymbol{W}_i^H \in \mathbb{R}^{d \times d}$ are all trainable matrices; h is the number of attention heads (set to 8 in the experiments) and $d_h = d/h$; $\boldsymbol{A}_i$ denotes the i-th attention matrix used to calculate the i-th head matrix; $[\cdot]$ means the concatenation operation; and LN is the Layer Normalization operation.

Besides MHA, another important module in Transformer is the Feed-Forward network (FFN):

$$\mathbf{FFN}(\boldsymbol{Y}) = \text{LN}(\text{FC}(\text{ReLU}(\text{FC}(\boldsymbol{Y}))) + \boldsymbol{Y}), \tag{2}$$

where FC denotes the fully-connected layer and ReLU denotes the rectified linear function.

Given MHA and FFN, we can use them to build a Transformer-based captioning model. For the encoder, it stacks 6 identical blocks where each one contains an MHA and an FFN. Given the output of the former block as the input $\boldsymbol{X}$, the next block

calculates its output as:

$$\begin{aligned} \textbf{Input:} \quad & \boldsymbol{X}, \\ \textbf{Self-ATT:} \quad & \boldsymbol{Y} = \textbf{MHA}(\boldsymbol{Q}=\boldsymbol{X}, \boldsymbol{K}=\boldsymbol{X}, \boldsymbol{V}=\boldsymbol{X}), \\ \textbf{Output}: \quad & \boldsymbol{Z} = \textbf{FFN}(Y), \end{aligned} \tag{3}$$

Note that the variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ used here are "local variables" for conveniently introducing the work flow of Transformer architecture, whose values will be set to the specific values when introducing the concrete captioning model. In "Self-ATT", $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are set to the same value and this operation is named as self-attention (Vaswani et al., 2017). After stacking 6 blocks defined in Eq. (3), a visual encoder is built. For the first block, its input is the extracted visual feature set of the given image. The output of the last block will be input into the language decoder.

For the decoder, it also stacks 6 identical blocks where each one contains two MHAs and an FFN. Given the output of the former decoder block $\boldsymbol{X}_D$ and the output of the visual encoder $\boldsymbol{X}_E$, the next decoder block calculates its output:

$$\begin{aligned} \textbf{Input:} \quad & \boldsymbol{X}_D, \boldsymbol{X}_E \\ \textbf{Self-ATT:} \quad & \boldsymbol{Y}_1 = \textbf{MHA}(\boldsymbol{Q}=\boldsymbol{X}_D, \boldsymbol{K}=\boldsymbol{V}=\boldsymbol{X}_D), \\ \textbf{ED-ATT:} \quad & \boldsymbol{Y}_2 = \textbf{MHA}(\boldsymbol{Q}=\boldsymbol{Y}_1, \boldsymbol{K}=\boldsymbol{V}=\boldsymbol{X}_E), \\ \textbf{Output}: \quad & \boldsymbol{Z} = \textbf{FFN}(\boldsymbol{Y}_2), \end{aligned} \tag{4}$$

Note that in "ED-ATT", $\boldsymbol{Q}$ is set to the output of the former decoder block while $\boldsymbol{K}, \boldsymbol{V}$ are set to the output of the visual encoder, and such operation is called encoder-decoder attention (Vaswani et al., 2017). After stacking 6 blocks defined in Eq. (4), a language decoder is built.

For the first block in the decoder, $\boldsymbol{X}_D$ in Eq. (4) is set to the word embedding set of the partially generated captions $\boldsymbol{S} = \{s_1, ..., s_t\}$ at the t-th time step. For all the decoder blocks, the input $\boldsymbol{X}_E$ is set to the same value, which is the output of the visual encoder. The output of the last decoder block $\boldsymbol{Z} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_t\}$ is used to calculate the word distribution of the next word:

$$P(s_{t+1}) = \text{Softmax}(\boldsymbol{z}_t). \tag{5}$$

Given the ground-truth caption $\boldsymbol{S}^*$, we can train this model by minimizing the cross-entropy loss:

$$L_{XE} = -\log P(S^*), \tag{6}$$

or by maximizing a reinforcement learning (RL) based reward (Rennie et al., 2017):

$$R_{RL} = \mathbb{E}_{\boldsymbol{S}^s \sim P(\boldsymbol{S})}(r(\boldsymbol{S}^s; \boldsymbol{S}^*)), \tag{7}$$

where r is a sentence-level metric for the sampled sentence $\boldsymbol{S}^s$ and the ground-truth $\boldsymbol{S}^*$, *e.g.*, the CIDEr-D (Vedantam et al., 2015) metric.

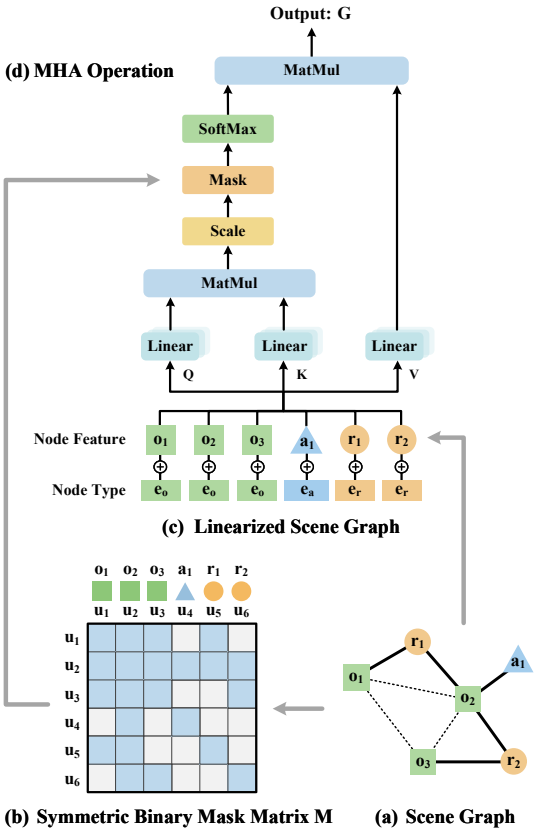

Figure 2: The sketch of the proposed MHA-GNN. In (a), square/triangle/circle demonstrate the object/attribute/relation embeddings, where the dash line means that all the object nodes should be connected for capturing visual contexts. (b) sketches the built binary mask matrix, where the top part shows the original graph embeddings ($\boldsymbol{o}/\boldsymbol{a}/\boldsymbol{r}$). For convenience, we also index the linearized annotation ($\boldsymbol{u}$) in the left and top. (c) shows the linearized scene graph, where the top and bottom parts are respectively node feature and type embeddings. (d) details how MHA achieves the graph operation.

4 Transforming Scene Graphs

In this section, we introduce how to revise the Transformer to get our TFSGC. We will first show how to get an MHA-based GNN and then introduce how to design an MOE-based decoder.

4.1 MHA-GNN

A visual scene graph (Krishna et al., 2017) contains three kinds of node embeddings: object/attribute/relationship embeddings $\boldsymbol{o}/\boldsymbol{a}/\boldsymbol{r}$. These nodes are connected by the following rules: if an object $\boldsymbol{o}_i$ has an attribute $\boldsymbol{a}_k$, $\boldsymbol{o}_i$ and $\boldsymbol{a}_k$ are connected, *e.g.*, $\boldsymbol{o}_1$ connects $\boldsymbol{a}_1$ in Fig. 2 (a). If two objects $\boldsymbol{o}_i$ and $\boldsymbol{o}_j$ have the relation $\boldsymbol{r}_k$, we connect $\boldsymbol{r}_k$ with $\boldsymbol{o}_i$ and $\boldsymbol{o}_j$, *e.g.*, $\boldsymbol{r}_1$ connects $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$. Given an image, we extract a series of visual features from the im-

age as the object embeddings: $\{\boldsymbol{o}_1, ..., \boldsymbol{o}_{N_o}\}$. To get the attribute/relation embeddings, we first use the attribute/relation annotations from VG (Krishna et al., 2017) to train the attribute/relation classifiers to predict the labels. Then we use two learnable embedding layers to respectively transform these labels into the dense attribute/relation embeddings $\{\boldsymbol{a}_1, ..., \boldsymbol{a}_{N_a}\}$/ $\{\boldsymbol{r}_1, ..., \boldsymbol{r}_{N_r}\}$.

Given these original node embeddings, GNN will update each one by aggregating the neighbour embeddings. In the previous GNN-LSTM-based models, GNN is usually deployed by the FC layers, which aggregates the contexts by mapping the concatenated neighbour embeddings to the new one (Yao et al., 2018; Yang et al., 2020). However, in our TFSGC, since Transformer is applied as the backbone, we may more hope to use the basic building block of Transformer to deploy the GNN. Such a design principle has two advantages. First, it alleviates the coding implementation difficulty that we do not need to specify some additional GNN operations. Second, which is more important, when the GNN and the Transformer architecture are homogeneous, the whole model will be more easily trained. For example, we do not need to set different training strategies like the learning rate or optimizer to different sub-networks.

Since MHA (Eq. (1)) can learn the context knowledge between the embeddings (Vaswani et al., 2017), it can naturally be used to define the graph operation for aggregating the knowledge. We apply the following two steps to do this. Firstly, as shown in Fig. 2, we linearize the object, attribute, and relation embeddings into one sequence and add the learnable type embeddings as the linearized token set: $\boldsymbol{U} = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_N\}$:

$$\begin{aligned} &\textbf{Object:} && \boldsymbol{u}_i = \boldsymbol{o}_i + \boldsymbol{e}_o, && 1 \le i \le N_o, \\ &\textbf{Attribute:} && \boldsymbol{u}_{N_o+i} = \boldsymbol{a}_i + \boldsymbol{e}_a, && 1 \le i \le N_a, \\ &\textbf{Relation:} && \boldsymbol{u}_{N_o+N_a+i} = \boldsymbol{o}_i + \boldsymbol{e}_r, && 1 \le i \le N_r, \end{aligned} \quad (8)$$

where $\boldsymbol{e}_o/\boldsymbol{e}_a/\boldsymbol{e}_r$ are respectively learnable type embeddings correspond to object/attribute/relation types and $N = N_o + N_a + N_r$. For example, in Fig. 2, $N_o/N_a/N_r$ is 3/1/2, the objects $\boldsymbol{o_{1:3}}$ become $\boldsymbol{u}_{1:3}$, the attributes $\boldsymbol{a}_1$ becomes $\boldsymbol{u}_4$, and the relations $\boldsymbol{r}_{1:2}$ become $\boldsymbol{u}_{5:6}$.

After linearizing, the topological knowledge of the graph is lost, *i.e.*, this token sequence does not show which two nodes are connected or not. To remedy such knowledge, we use a symmetric binary mask matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N}$ to all Transformer blocks to mask the attention weights of the unconnected nodes to control whether two nodes are connected or not. If two nodes $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are connected in the original scene graph, we set $M_{i,j} = 1$ and $M_{i,j} = 0$ otherwise. Specifically, the values of $\boldsymbol{M}$ are set as follows: 1) If $\boldsymbol{o}_i$ has one attribute $\boldsymbol{a}_j$, we set $M_{i,j+N_o} = 1$, *e.g.*, $\boldsymbol{o}_2$ ($\boldsymbol{u}_2$) and $\boldsymbol{a}_1$ ($\boldsymbol{u}_4$) in Fig. 2 are connected and $M_{2,4} = 1$. 2) If $\boldsymbol{r}_k$ connects with $\boldsymbol{o}_i$ and $\boldsymbol{o}_j$, we set $M_{i,k+N_o+N_a} = M_{j,k+N_o+N_a} = 1$, *e.g.*, $\boldsymbol{r}_1$ ($\boldsymbol{u}_5$) connects $\boldsymbol{o}_1$ ($\boldsymbol{u}_1$) and $\boldsymbol{o}_2$ ($\boldsymbol{u}_2$) and $M_{1,5} = M_{2,5} = 1$. 3) All the object nodes are connected with each other since they are visual features that their contexts play a key role in captioning (Herdade et al., 2019). Thus $\forall i, j \le N_o, M_{i,j} = 1$. 4) Since the scene graph is an undirected graph, $\boldsymbol{M}$ should be symmetric: $M_{i,j} = M_{j,i}$.

After getting $\boldsymbol{U}$ and $\boldsymbol{M}$, we can revise the Transformer encoder to get our MHA-GNN. Specifically, we use $\boldsymbol{U}$ as the input of the encoder defined in Eq.(3) and revise the **Att** operation in Eq. (1) as the following **Masked Att** operation:

$$\boldsymbol{A}_i = \text{Softmax}(M \odot \frac{\boldsymbol{Q}\boldsymbol{W}_i^Q(\boldsymbol{K}\boldsymbol{W}_i^K)^T}{\sqrt{d}}), \quad (9)$$

where $\odot$ denotes the element-wise product. In this way, the graph operation is defined by MHA. Specifically, for each node embedding, it is updated by weighted summing its neighbour embeddings, where the weights are from the attention heads $\boldsymbol{A}_i$ calculated by the **Att** operation in Eq. (1). During weighted summing, the binary matrix control whether two nodes are connected or not. Note that the edge type is implicitly embedded in **Att** operation due to the added node type embedding. For example, after adding node type embeddings $\boldsymbol{e}_o$ and $\boldsymbol{e}_r$ to the object and relation embeddings $\boldsymbol{o}$ and $\boldsymbol{r}$, respectively, the inner-product becomes*:

$$(\boldsymbol{o} + \boldsymbol{e}_o)^T(\boldsymbol{r} + \boldsymbol{e}_r) = \boldsymbol{o}^T\boldsymbol{r} + \boldsymbol{e}_o^T\boldsymbol{r} + \boldsymbol{e}_r^T\boldsymbol{o} + \boldsymbol{e}_o^T\boldsymbol{e}_r, \quad (10)$$

where the right three terms are affected by the node type embedding. Thus, when the edge type changes (*e.g.*, the object-relation edge changes to object-attribute edge), the corresponding node type embeddings also change (*e.g.*, $\boldsymbol{e}_r$ changes to $\boldsymbol{e}_a$), which means Eq. (10) encodes the knowledge of edge types into the embeddings.

By stacking more such layers, the receptive field is increasing and thus each node can be updated by aggregating more neighbour embeddings, which naturally follows the design principle of GNN (Battaglia et al., 2018). The output graph

*For convenience, we omit the trainable matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K$ in Eq. (1) in this inner-product operation.

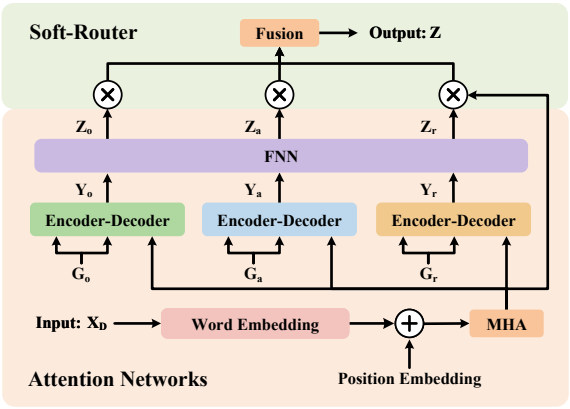

Figure 3: The sketch of the MOE-decoder, we use different colours to denote different experts: green/blue/yellow correspond to object/attribute/relation experts.

embedding set $\boldsymbol{G}$ are input to the decoder for captioning.

4.2 MOE-decoder

As mentioned before, a caption contains different kinds of words for describing diverse visual patterns, *e.g.*, nouns/adjectives/verbs for objects/attributes/relations (Yang et al., 2019b), which suggests that different experts should be used to address diverse visual knowledge for generating the corresponding words. Motivated by this idea, we design an MOE-based (Jacobs et al., 1991; Du et al., 2022) language decoder to discriminate diverse graph embeddings by setting three encoder-decoder attention layers as different experts. As shown in Fig. 1 (c), the graph embeddings $\boldsymbol{G} = \{\boldsymbol{g}_1, ..., \boldsymbol{g}_N\}$ output from the MHA-GNN can be naturally divided according to the original token types in the scene graph: object/attribute/relation sets $\boldsymbol{G}_o = \{\boldsymbol{g}_1, ..., \boldsymbol{g}_{N_o}\}$/ $\boldsymbol{G}_a = \{\boldsymbol{g}_{N_o+1}, ..., \boldsymbol{g}_{N_o+N_a}\}$/$\boldsymbol{G}_r = \{\boldsymbol{g}_{N_o+N_a+1}, ..., \boldsymbol{g}_N\}$. Then we only need to input them into the corresponding experts for discriminating them. Fig. 3 sketches the designed MOE-based decoder, which is got by revising the decoder defined in Eq. (4) as:

$$
\begin{aligned}
\textbf{Input:}\quad & \boldsymbol{X}_D, \boldsymbol{G}_o, \boldsymbol{G}_a, \boldsymbol{G}_r \\
\textbf{SA:}\quad & \boldsymbol{X} = \textbf{MHA}(\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{X}_D), \\
\textbf{EXP}_O\textbf{:}\quad & \boldsymbol{Y}_o = \textbf{MHA}(\boldsymbol{Q} = \boldsymbol{X}, \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{G}_o), \\
\textbf{EXP}_A\textbf{:}\quad & \boldsymbol{Y}_a = \textbf{MHA}(\boldsymbol{Q} = \boldsymbol{X}, \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{G}_a), \\
\textbf{EXP}_R\textbf{:}\quad & \boldsymbol{Y}_r = \textbf{MHA}(\boldsymbol{Q} = \boldsymbol{X}, \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{G}_r), \\
\textbf{FFN:}\quad & \boldsymbol{Z}_o, \boldsymbol{Z}_a, \boldsymbol{Z}_r = \textbf{FFN}(\boldsymbol{Y}_o, \boldsymbol{Y}_a, \boldsymbol{Y}_r), \\
\textbf{SR:}\quad & \boldsymbol{Z} = \textbf{SR}(\boldsymbol{Z}_o, \boldsymbol{Z}_a, \boldsymbol{Z}_r, \boldsymbol{X})
\end{aligned}
\tag{11}
$$

where $\textbf{EXP}_O$, $\textbf{EXP}_A$, and $\textbf{EXP}_R$ denote three different experts (encoder-decoder attentions) used to address object, attribute, and relation embeddings, respectively. They have the same structure while with different parameters.

Note that the input $\boldsymbol{X}_D$ is the word embeddings of the partially generated captions and at t-th step, $\boldsymbol{X}_D = \{\boldsymbol{x}_D^1, ..., \boldsymbol{x}_D^t\}$. Then all the $\boldsymbol{X}/\boldsymbol{Z}_o/\boldsymbol{Z}_a/\boldsymbol{Z}_r$ also contain t elements, *e.g.*, $\boldsymbol{Z}_o = \{\boldsymbol{z}_o^1, ..., \boldsymbol{z}_o^t\}$. Soft Router (**SR**) calculates an ensemble embedding z at each time step to construct the embedding set $\boldsymbol{Z} = \{\boldsymbol{z}^1, ..., \boldsymbol{z}^t\}$. Specifically, for each element $\boldsymbol{x}/\boldsymbol{z}_o/\boldsymbol{z}_a/\boldsymbol{z}_r$ in $\boldsymbol{X}/\boldsymbol{Z}_o/\boldsymbol{Z}_a/\boldsymbol{Z}_r$, a corresponding output $\boldsymbol{z}$ can be got[†]:

$$
\begin{aligned}
\textbf{Input:}\quad & \boldsymbol{x}, \boldsymbol{z}_o, \boldsymbol{z}_a, \boldsymbol{z}_r, \\
\textbf{ATT:}\quad & \boldsymbol{\alpha} = \{\alpha_o, \alpha_a, \alpha_r\} \\
& \quad = \text{Softmax}(\{\boldsymbol{x}^T\boldsymbol{z}_o, \boldsymbol{x}^T\boldsymbol{z}_a, \boldsymbol{x}^T\boldsymbol{z}_r\}) \\
\textbf{Output:}\quad & \boldsymbol{z} = \alpha_o\boldsymbol{z}_o + \alpha_a\boldsymbol{z}_a + \alpha_r\boldsymbol{z}_r,
\end{aligned}
\tag{12}
$$

where **ATT** operation calculates the soft routing weights, since $\boldsymbol{x}$ accumulates the context knowledge of the partially generated caption, it can help judge which kind of word should be generated at the next step. For example, if the last word of this partially generated caption is an adjective "black", the next word is more like to be a noun and thus α_o should be a large value for using more object embeddings instead of the other embeddings.

5 Experiments

5.1 Datasets, Metrics, and Implementation Details

Datasets. MSCOCO. We use MSCOCO (Lin et al., 2014) to validate our TFSGC. This dataset has 123,287 images and each one is labeled with 5 captions. We use two splits in the experiments: the offline Karpathy split (113,287/5,000/5,000 train/val/test images) and the Official online split (82,783/40,504/40,775 train/val/test images).

Visual Genome (Krishna et al., 2017) provides scene graph annotations for training the scene graph parser. We follow (Yang et al., 2020) to filter the noisy dataset (*e.g.*, lots of labels only appear a few times in the dataset) by removing the attribute/relation labels appearing less than 2000 times and use the remained 103/64 attribute/relation labels to train the attribute/relation classifiers.

Implementation Details. In the experiments, we use three kinds of visual features to exhaustively compare to the other SOTA models, which are BUTD (Anderson et al., 2018), Patch (Liu et al., 2021), and VinVL (Zhang et al., 2021). During training and inference, for BUTD/Patch/VinVL, we respectively follow (Yang et al., 2020) and

[†]For convenience, we remove the superscript representing for different time steps of each embedding.

VinVL's official parser[‡] to parse SGs, where the latter is more powerful. For all the visual features, we set the batch size to 20 and use Adam (Kingma and Ba, 2014) as the optimizer. For BUTD/Patch/VinVL features, We sequentially use cross-entropy loss (Eq. (6)) and RL-based reward (Eq. (7)) to train the models 20/20/30 and 30/30/30 epochs. For BUTD/Patch/VinVL features, the learning rate used in the cross-entropy stage is initialized as $5e^{-4}$/$5e^{-4}$/$2e^{-5}$ and both decayed by 0.8 every 5 epochs, the learning rate used in the RL-reward stage is reset to $5e^{-5}$/$2e^{-5}$/$5e^{-6}$ and both decayed by 0.8 every 5 epochs. During inference, we use beam search where the beam size is 5. We evaluate the captions by CIDEr-D (Vedantam et al., 2015), BLEU (Papineni et al., 2002), METEOR(Banerjee and Lavie, 2005), ROUGE (Lin, 2004) and SPICE (Anderson et al., 2016).

5.2 Ablation Studies

To confirm the effectiveness of the proposed MHA-GNN and MOE-decoder, we deploy exhaustive ablations as follows. Note that we use BUTD feature in this section. **BASE**: We directly use the classic Transformer architecture. **SG**: We incorporate the scene graphs into the Transformer by using the node embeddings without any graph operations. **MLP-GNN**: We apply MLP-based Graph Neural Network (Xu et al., 2019) for embedding the scene graphs. **MHA-GNN w/o e**: We apply the proposed MHA-GNN while do not use node type embedding. **MHA-GNN**: We apply the proposed MHA-GNN and keep the decoder unchanged as BASE. **GNN-FC**:We remove the binary mask matrix M from TFSGC. **MOE**: We use the proposed MOE-decoder and do not use GNN but input the original node embeddings into the decoder. **TFSGC**: We apply the integral TFSGC.

Table 1 compares the similarity metrics of the ablation models. Firstly, we can find that the integral TFSGC achieves the highest scores, which confirms its effectiveness. Next, we respectively compare the ablation models to validate the effectiveness of the proposed MHA-GNN and MOE-decoder. By comparing MLP-GNN, SG, and BASE, it is easy to see that using GNN can gain more profits than only using node embeddings. Furthermore, we can find that MHA-GNN has higher CIDEr than MLP-GNN, which suggests that designing GNN by MHA is more powerful than by

[‡]https://github.com/microsoft/scene_graph_benchmark

Models	B@4	M	R	C	S
BASE	38.4	28.5	58.1	128.7	22.0
SG	38.5	28.6	58.1	129.0	22.2
MLP-GNN	38.9	28.8	58.4	129.5	22.4
MHA-GNN w/o e	39.1	28.9	58.5	130.1	22.4
MHA-GNN	39.5	29.2	58.9	130.9	22.8
GNN-FC	39.2	29.0	58.3	130.5	22.3
MOE	39.2	28.8	58.5	130.1	22.5
TFSGC	**39.8**	**29.6**	**59.3**	**132.3**	**23.4**

Table 1: The performances of various ablation models. The metrics: B@N, M, R, C, and S denote BLEU@N, METEOR, ROUGE-L, CIDEr-D, and SPICE.

Models	nouns	adjectives	verbs	prepositions
BASE	43.8	12.7	20.3	40.1
SG	44.2	13.5	20.9	40.8
MLP-GNN	45.4	14.8	21.8	41.6
MHA-GNN w/o e	48.8	16.4	24.3	44.3
MHA-GNN	49.6	17.0	24.7	44.6
MOE	48.4	16.6	24.4	44.0
TFSGC	**52.8**	**18.2**	**25.8**	**45.8**

Table 2: The recalls (%) of five part-of-speech words.

MLP in the Transformer architecture. Next, to see whether discriminating the graph embeddings is beneficial or not, we can compare MHA-GNN with MHA-GNN w/o e and find that using node type embedding performs better. From the comparison between GNN-FC and TFSGC, it can be seen that when removing M, the graph becomes a fully-connected graph, which introduces more noises. Also, it can be seen that MOE and TFSGC respectively achieve better performances than SG and MHA-GNN, which validates the effectiveness of the MOE-decoder.

Besides evaluating these ablation models by similarities, in Table 2, we calculate the recalls of the words with different POS to evaluate the descriptiveness. Table 2 shows that the captions generated from TFSGC have the highest recalls, suggesting that TFSGC generates the most descriptive captions. Also, we can find that both the proposed MHA-GNN (MHA-GNN vs. MLP-GNN) and MOE-based decoder (MOE vs. SG) can boost the recalls, suggesting both of them improve the descriptiveness. Moreover, We use Stanford Parser to get the POS to train the route weights by the cross-entropy loss, then the CIDEr of TFSGC boosts from 132.3 to 132.9, suggesting the advantage of using POS knowledge. Fig. 4 shows 4 examples of the captions generated from diverse models, where we can see that TFSGC generates more descriptive captions. BASE generates less descriptive captions since it does not use scene graphs and thus loses se-

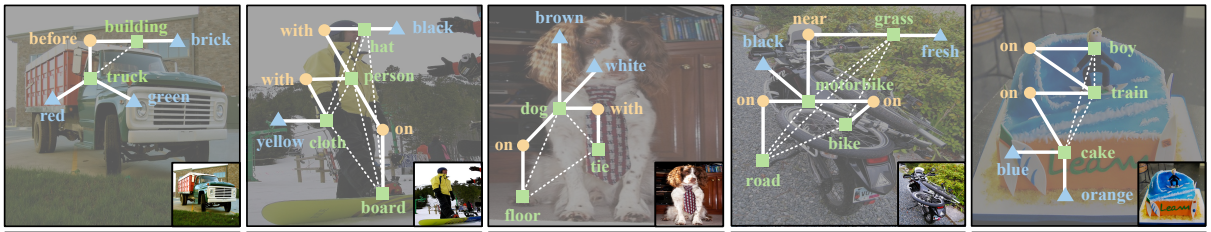

Figure 4: The captions generated by **BASE**, **MHA-GNN**, and **TFSGC**. It can be found that TFSGC generates more descriptive captions, *e.g.*, it describes more attributes like "green truck" and "brick building" in (a), or using more fine-grained nouns like "jacket" in (b). Diverse colors in TFSGC denote these words use more knowledge from different experts: green/blue/orange corresponds to object/attribute/relation expert, which is got by checking which one of $\alpha_o/\alpha_a/\alpha_r$ in Eq (12) is the largest.

mantic knowledge compared with SG. MHA-GNN does not use the soft router and thus can not select the most suitable experts for generating corresponding words, which may lose certain details. Also, we show that which expert is more responsible for the words with different POS, *e.g.*, in Fig. 4 (b), the adjective "yellow" is generated by using more knowledge from the attribute expert.

5.3 Comparisons with SOTA

Recently, various SOTA captioning models with diverse settings have been proposed, including using different language decoders (LSTM, GRU, and Transformer), different visual features, and whether distilling knowledge from large-scale pre-training models (CLIP (Radford et al., 2021) or VInVL (Zhang et al., 2021)). In fairness, we compare our TFSGC with the models that also use Transformer-based decoder by three features: BUTD (Anderson et al., 2018), Patch (Liu et al., 2021), and VinVL (Zhang et al., 2021).

We compare with the following SOTA models: **ETA** (Li et al., 2019), **ORT** (Herdade et al., 2019), **AoANet** (Huang et al., 2019), $\mathcal{M}^2$ **Transformer** (Cornia et al., 2020), **CATT** (Yang et al., 2021b), **APN** (Yang et al., 2021a), **DLCT** (Luo et al., 2021), **X-Transformer** (Pan et al., 2020), **Up-Down** (Anderson et al., 2018), **SGAE** (Yang et al., 2019a), **NG-SAN** (Guo et al., 2020), **PureT** (Wang et al., 2022), and **ViTCAP** (Fang et al., 2022). Specifically, ETA and ORT are preliminary Transformer-based models; AoANet and X-Transformer design stronger attention mechanisms; CATT and $\mathcal{M}^2$ Transformer introduce additional memory networks; and APN exploits the hidden tree structures. We set the training batch size of TFSGC* to 50 as X-Transformer for fairly comparing. These methods use BUTD features, while ViTCAP and PureT use Patch features. RSTNet and ViTCAP distill knowledge from pre-trained vision-language BERTs. Note that in VinVL (Zhang et al., 2021), they use OSCAR (Li et al., 2020) as the captioner, while OSCAR is trained on 6.5 million image-text pairs. To fairly compare, we input VinVL feature into the classic Transformer to generate the captions, which is denoted as **VinVL(Transformer)**. Note that we do not compare with extra large scale models trained by millions of image-text pairs or with excessively large parameter sizes, such as **RSTNet** (Zhou et al., 2020) and **LEMON** (Hu et al., 2022). All the results are shown in Table 3.

From Table 3 we can find that TFSGC almostly achieves the highest scores in different settings, *i.e.*, achieving 132.3, 138.6, 139.5 CIDEr scores when using BUTD, Patch, and VinVL features, respectively. DLCT's visual features extractor (Jiang et al., 2020) is stronger than TFSGC(BUTD) (Anderson et al., 2018) and thus DLCT is a little better than TFSGC(BUTD). Among these compared methods, although other SOTAs do not use scene graphs, they usually have some other training burdens. For example, APN and X-Linear apply more complex attention mechanisms and it requires more computation resource for well-training them, while our TFSGC only apply the simplest attention opera-

Models	Cross-Entroy Loss					CIDEr optimization				
	B@4	M	R	C	S	B@4	M	R	C	S
BUTD feature										
ETA	37.1	28.2	57.1	117.9	21.4	39.3	28.8	58.9	126.6	22.7
ORT	35.5	28.0	56.6	115.4	21.2	38.6	28.7	58.4	128.3	22.6
AoANet	37.2	28.4	57.5	119.8	21.4	38.9	29.2	58.8	129.8	22.4
$\mathcal{M}^2$ Transformer	-	-	-	-	-	39.1	29.2	58.6	131.2	22.6
CATT	37.3	28.5	57.4	119.0	21.5	39.4	29.3	58.9	131.7	22.8
APN	-	-	-	-	-	39.6	29.2	59.1	131.8	23.0
DLCT	-	-	-	-	-	**39.8**	29.5	59.1	**133.8**	23.0
TFSGC	**38.1**	**28.6**	**57.7**	**120.2**	**21.9**	**39.8**	**29.6**	**59.3**	132.3	**23.4**
BUTD feature & Larger Batch Size										
X-Transformer	38.2	**28.8**	**58.0**	122.0	21.9	39.7	29.5	59.2	132.8	23.2
TFSGC*	**38.4**	**28.8**	57.8	**122.3**	**22.1**	**39.9**	**29.8**	**59.4**	**133.0**	**23.4**
Large Visual-language model pretraining										
RSTNet	-	-	-	-	-	40.1	28.9	59.5	135.6	23.3
LEMON$_{base}$	40.3	30.2	-	133.3	23.3	41.6	30.1	-	142.7	25.1
LEMON$_{huge}$	41.5	30.8	-	139.1	24.1	42.6	31.4	-	145.5	25.5
Patch feature										
PureT	-	-	-	-	-	40.9	**30.2**	**60.1**	138.2	24.2
ViTCAP$_{small}$	35.7	28.8	57.6	121.8	22.1	40.1	29.4	59.4	133.1	23.0
TFSGC	**38.8**	**29.4**	**58.2**	**122.2**	**22.3**	**41.4**	30.1	**60.1**	**138.6**	**24.4**
VinVL feature										
VinVL(Transformer)	35.4	28.6	57.5	121.5	21.3	40.6	30.0	59.8	137.3	23.7
TFSGC	**38.5**	**29.2**	**58.8**	**122.7**	**22.4**	**41.7**	**30.5**	**60.4**	**139.5**	**24.6**

Table 3: The performances of SOTA methods on MS-COCO Karpathy split. All models used are single models.

Models	B@4		M		R		C	
	c5	c40	c5	c40	c5	c40	c5	c40
Up-Down	36.9	68.5	27.6	36.7	57.1	72.4	117.9	120.5
SGAE	37.8	68.7	28.1	37.0	58.2	73.1	122.7	125.5
ETA	38.9	70.2	28.6	38.0	58.6	73.9	122.1	124.4
APN	38.9	70.2	28.8	38.0	58.7	73.7	126.3	127.6
NG-SAN	38.8	70.2	29.0	38.4	58.7	74.0	126.3	128.6
TFSGC^S	**39.0**	**70.9**	**29.1**	**38.4**	**58.9**	**74.4**	**127.2**	**129.8**
AoANet	39.4	71.2	29.1	38.5	58.9	74.5	126.9	129.6
X-Transformer	**39.9**	71.8	29.5	39.0	59.3	74.9	129.3	131.4
$\mathcal{M}^2$ Transformer	39.7	**72.8**	29.4	39.0	59.2	74.8	129.3	132.1
TFSGC^E	39.7	72.6	**29.6**	**39.2**	**59.5**	**75.1**	**129.6**	**133.1**

Table 4: The scores on the MS-COCO online test server. ^{S} indicates single model, ^{E} indicates ensemble model.

tion. Moreover, as detailed in Sec 4.1 of ViTCAP, it applies much more training data (9.9M image-text pairs from 4 datasets including VG) to pre-train a concept network to get more powerful discrete tags, while we only use one dataset VG to get scene graphs and achieve better performance, which suggests that connecting discrete tags into a graph is a useful strategy if the discrete tags is not very powerful. Moreover, TFSGC achieves comparable performances with LEMON (Hu et al., 2022) which uses 200 million image-text pairs and uses 12/24-layer, 768/1024-hidden units Transformers(base/huge). To sum up, the advantage of TFSGC is that it can effectively embed and discriminate the semantic knowledge from scene graphs to balance the (usually more) burden of using more training data or of training more complex networks.

We also submit the single model TFSGC^S and 4-ensembled model TFSGC^E trained by regional-based features into the online server for testing, where the results are shown in Table 4. From this table, we can discover that both TFSGC^S and TFSGC^E have the highest CIDEr-D scores, which further confirm the effectiveness of the proposed TFSGC.

6 Conclusion

We proposed to transform the scene graph into captions (TFSGC) by a simple and homogeneous network. Specifically, we use MHA to design the GNN by linearizing the scene graph and remedying the lost topological knowledge with a binary mask matrix. Furthermore, we add learnable type embedding and design an MOE-based decoder to distinguish node embeddings for more descriptive captions. At last, we compared TFSGC with various SOTA models and demonstrated that our model can achieve comparable performances to some strong benchmarks.

Limitations

There are two major limitations of the proposed TFSGC. The first one is that the effectiveness of TFSGC depends on the quality of the scene graph. Since MSCOCO does not have SG annotations, we evaluate the parsers in Visual Genome: for BUTD/Patch and VinVL, the recall@50 of relation/attribute are respectively 65.2/68.4 and 73.4/76.6. We use VinVL's SGs in TFSGC(BUTD) and CIDEr improves from 132.3 to 133.1, suggesting better SGs are beneficial. If the scene graph quality is poor, then TFSGC will not achieve good performance. When an incorrect node in the scene graph, it also affects the output of the caption. *e.g.*, in Fig. 4 (e), the correct object label should be "surfboard" instead of "train". In this paper, we use Visual Genome, which contains abundant and useful scene graph annotations for parsing effective scene graphs, but current performance is not the best, and we will improve the scene graph parser based on the latest scene graph parsing methods in the future.

The second limitation of TFSGC is that if the visual features contain abundant attribute or relation knowledge, then the improvement of TFSGC compared with the classic Transformer will be weakened. For example, compared with the BUTD feature case where the relative improvement of CIDEr-D is 3.6 (TFSGC-BASE in Table 1), the VinVL feature is more powerful since it is trained by much more data samples with more semantic labels, thus the relative improvement is lower, which is 2.2 (TFSGC-VinVL(Transformer) in Table 3).

Acknowledgements

This work is supported by National Key R&D Program of China (2018AAA0100104, 2018AAA0100100), National Science Foundation of China (62206048), Natural Science Foundation of Jiangsu Province (BK20220819, BK20211164), and Young Elite Scientists Sponsorship Program of Jiangsu Association for Science and Technology Tj-2022-027.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 5, page 6.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10578–10587.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Zhiyuan Fang, Jianfeng Wang, Xiaowei Hu, Lin Liang, Zhe Gan, Lijuan Wang, Yezhou Yang, and Zicheng Liu. 2022. Injecting semantic concepts into end-to-end image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18009–18019.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5630–5639.

Longteng Guo, Jing Liu, Xinxin Zhu, Peng Yao, Shichen Lu, and Hanqing Lu. 2020. Normalized and geometry-aware self-attention network for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10327–10336.

Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. 2019. Image captioning: Transforming objects into words. In *Advances in Neural Information Processing Systems*, pages 11137–11147.

Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2022. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17980–17989.

Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on attention for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4634–4643.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. 2020. In defense of grid features for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10267–10276.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Chenliang Li, Haiyang Xu, Junfeng Tian, Wei Wang, Ming Yan, Bin Bi, Jiabo Ye, Hehong Chen, Guohai Xu, Zheng Cao, et al. 2022. mplug: Effective and efficient vision-language learning by cross-modal skip-connections. *arXiv preprint arXiv:2205.12005*.

Guang Li, Linchao Zhu, Ping Liu, and Yi Yang. 2019. Entangled transformer for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705.

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. *ECCV 2020*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, volume 6, page 2.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7219–7228.

Yunpeng Luo, Jiayi Ji, Xiaoshuai Sun, Liujuan Cao, Yongjian Wu, Feiyue Huang, Chia-Wen Lin, and Rongrong Ji. 2021. Dual-level collaborative transformer for image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2286–2293.

Victor Milewski, Marie-Francine Moens, and Iacer Calixto. 2020. Are scene graphs good enough to improve image captioning? *arXiv preprint arXiv:2009.12313*.

Kien Nguyen, Subarna Tripathi, Bang Du, Tanaya Guha, and Truong Q Nguyen. 2021. In defense of scene graphs for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1407–1416.

Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. 2020. X-linear attention networks for image captioning. In *CVPR*, pages 10971–10980.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Li Wang, Zechen Bai, Yonghua Zhang, and Hongtao Lu. 2020. Show, recall, and tell: image captioning with recall mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12176–12183.

Yiyu Wang, Jungang Xu, and Yingfei Sun. 2022. End-to-end transformer based model for image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8053–8072.

Haiyang Xu, Ming Yan, Chenliang Li, Bin Bi, Songfang Huang, Wenming Xiao, and Fei Huang. 2021. E2e-vlp: end-to-end vision-language pre-training enhanced by visual learning. *arXiv preprint arXiv:2106.01804*.

Haiyang Xu, Qinghao Ye, Ming Yan, Yaya Shi, Jiabo Ye, Yuanhong Xu, Chenliang Li, Bin Bi, Qi Qian, Wei Wang, et al. 2023. mplug-2: A modularized multi-modal foundation model across text, image and video. *arXiv preprint arXiv:2302.00402*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. 2022. Go wider instead of deeper. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8779–8787.

Xu Yang, Chongyang Gao, Hanwang Zhang, and Jianfei Cai. 2021a. Auto-parsing network for image captioning and visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2197–2207.

Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. 2019a. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10685–10694.

Xu Yang, Hanwang Zhang, and Jianfei Cai. 2019b. Learning to collocate neural modules for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4250–4260.

Xu Yang, Hanwang Zhang, and Jianfei Cai. 2020. Auto-encoding and distilling scene graphs for image captioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu Yang, Hanwang Zhang, Guojun Qi, and Jianfei Cai. 2021b. Causal attention for vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9847–9857.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2018. Exploring visual relationship for image captioning. In *Computer Vision–ECCV 2018*, pages 711–727. Springer.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5579–5588.

Yiwu Zhong, Liwei Wang, Jianshu Chen, Dong Yu, and Yin Li. 2020. Comprehensive image captioning via scene graph decomposition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 211–229. Springer.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13041–13049.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The Limitation section is given in page 10.

☒ A2. Did you discuss any potential risks of your work?
This work deal with image captioning, which is a task that does not have potential moral risks.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
The Abstract and Introduction sections are given in the page 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Left blank.

☑ B1. Did you cite the creators of artifacts you used?
The creators of artifacts are cited in the "Datasets, Metrics, and Implementation Details" section in the page 6.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
When we use the artifacts, we cite them in the paper, for example, the metrics are cited in the "Datasets, Metrics, and Implementation Details" section in the page 6.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use the evaluation metrics like BLEU or CIDEr-D to measure the similarities between the generated and ground-truth captions, which is consistent with their intended use.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Such information is introduced in the Dataset section in the page 6.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Not applicable. Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☒ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
No response.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
No response.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
No response.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
No response.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.