SLML2018 Final Project - Image Captioning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Image Captioning is an appealing and challenging task on the meeting point of Computer Vision and Natural Language Processing. To generate descriptions of images automatically, the effective encoder-decoder framework has been proposed with a variety of corresponding approaches. Most recently, Attention mechanism and variant RNN decoders are proposed to improve the performance. In this paper, we innovatively extend the encoder-decoder framework to encoder-connection-decoder framework, as well as apply Transformer architecture for captions generation. We conduct experiments on Flickr30k [1], and the performance of our Transformer-based model is sightly better than DeepVS [2], though it doesn't reach the state-of-the-art performance.

1 Introduction

Generating captions for images has emerged as an appealing task to achieve the final goal of Artificial Intelligence, where machines are able to describe what they see automatically like human being. This task is standing at the meeting point of Computer Vision and Natural Language Processing with far-ranging applications from human-robot interaction, images storytelling to helping visually disabled users. For these reasons above, most recently, this research area has been drawn great attention from both academia and industry with many efficient and innovative approaches proposed.

With the emergence of Deep Neural Network, the most influential architecture for this task was proposed as encoder-decoder framework [3], exploiting Deep Neural Network such as CNN and RNN to encode visual information and generate textual sentences for captioning. Many works concentrated on how to improve the architectures for modeling images or sentences respectively. [4] replaces vanilla RNN with LSTM to generate sentence in decoding process. And [5] applied CNN as decoder to explore fully CNN architecure for captioning task. Besides of this, inspired by attention mechanism proposed in machine-translation task, spatial attention mechanism was applied in models [6], [7] to concentrate on different areas of one image at every time step.

Although the Encoder-Decoder framework improved the performance of this task greatly, there are some limits of the decoder architecture: (1) For RNN decoder: firstly, RNN is hard to be deployed for parallel computing because it has to be executed step-by-step. Secondly, RNN suffers from Long-Term Dependecies Problem. (2) For CNN decoder: though CNN solves the problem of parallel computing, CNN is hard to catch the long-distance relation because of its architecture. And it's hard to get better performance in NLP by increasing CNN layers [8]. Most recently, many Transformer [6]-based model like BERT [9] and ELMo [10] achieved the state-of-the-art performance for Natural Language Processing tasks. It's natural to explore the application of Transformer in Image Captioning task to model the natural language generation, in order to overcome the shortcomings mentioned above about current RNN/CNN decoder.

To the best of our knowledge, it's the first implementation of Transformer-based decoder for Image Captioning task. In this paper, the main contributions are:

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

- Replace the RNN/CNN decoder with Transformer Architecture.
- Explore methods to convert the extracted image features into language embedding space.
- Systemic experiments analysis on Flicker30k Dataset.

2 Related Work

Among the methods about generating captions of images automatically, the most popular framework is Encoder-Decoder framework, which is inspired by the success of sequence-to-sequence model in Natural Language Processing. In captioning task, usually CNN is applied as encoder to extract the feature from image, while RNN is exploited as decoder to generate captions. Under this framework, the performance was improved by different mechanisms like Attention to represent image effciently.

- **Encoder-Decoder Framework** was proposed in m-RNN model [3]. This model applied several CNN models, like AlexNet and VGG, with a vanilla RNN to convert the image information into a single vector at every time step. As for decoding part, the model applied softmax layer to predict words. With the development of RNN, LSTM as applied in decoding process to address the problem of gradient vanishing [4]. For the encoding part, [2] adopted the result from R-CNN and used a Bidirectional RNN to map images into a joint embedding sapce for caption generation.
- **Attention Mechaism** was proposed in [6] to encode the corresponding information selectively. Spatial Attention Mechanism began to be introduced into image captioning task in [6], which proposed two kinds of way to pay attention to different areas of images for generating captions. [11] extend current attention mechanism with a review network, which captures the global properites in a compact vector representation for attention mechanism in decoder. Image Semantics begin to be included in this task [12, 13, 14], like using image annotations produced by image classifier. Semantic Attention model is proposed in [12], to generate the weights from CNN-based senmantics and then apply these semantics into decoding process.
- **Tranformer** [15] was proposed to solve the RNN problems we mentioned for sequence-to-sequence framework. And then ELMo [10] and BERT [9] was proposed based on Transformer to generalize traditional word embedding research along a different dimension and learning word representations bidirectionally. Based on Transformer, BERT and its variants are the state-of-the-art models for most NLP tasks.

3 Methodology

In this section, we introduce the architecture of model and its detailed implementation. In general, our model follows the Encoder-Decoder framework with Transformer which was proposed in [15]. We applied Transformer for decoding processing directly. And for the encoding part, we proposed three methods concerning how to map the visual feature into language embedding space.

3.1 Preliminary

3.1.1 Multi-head Attention

Mutli-head Attention is consist of multiple Scaled Dot-Product Attetion networks in [15]. In details, we compute the attention scores using query matrix Q and key matrix K, to get the weighted results from value matrix V. Scale factor $\frac{1}{\sqrt{d_k}}$ is used in case the dot products grow large in magnitude when matrix dimension d_k is large. The formula is:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

With the experiment results in [15] shown, it's beneficial to linearly project the queries, keys and values h times with different, learned projections to d_k, d_k and d_v dimensions, respectively, and then concatenate the outputs from every projected attention modules.

$$MultiHead(Q, K, V) = Concat\left(head_1, \ldots, head_{\mathrm{h}}\right)W^O \tag{2}$$

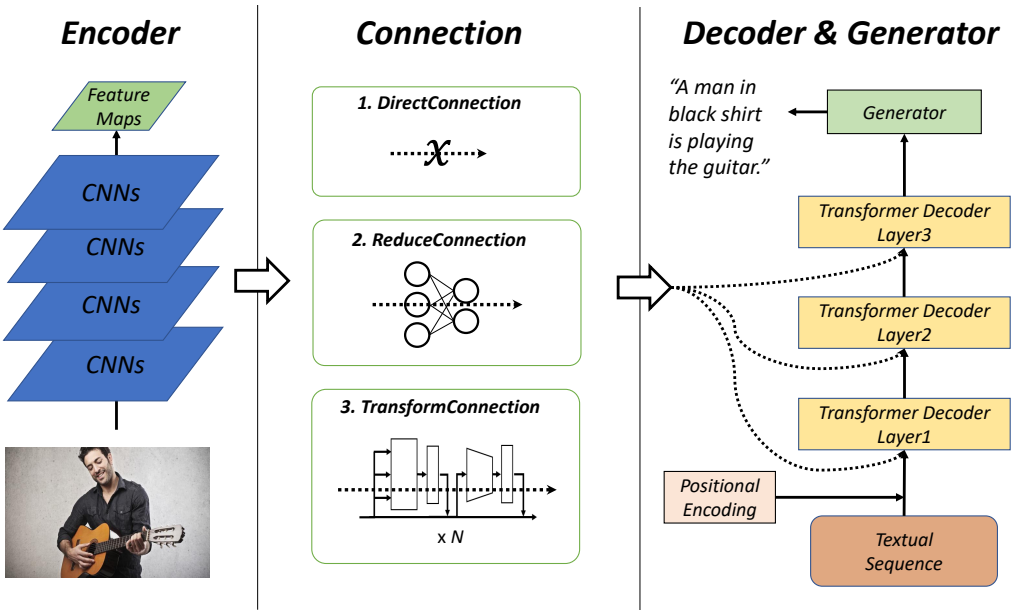


Figure 1: The architecture of our model. Firstly, we extract visual features from CNN-based Neural Network. Secondly, we proposed three methods for connection between visual and textual feature. Thirdly, we generate natural language captions with the converted visual information. The decoder architecture is based on Transformer.

$$head_{\mathrm{i}} = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{3}$$

Where W_i^Q, W_i^K, W_i^V and W^O is the projection matrix.

3.1.2 Transformer

According to [15], the Transformer architecure can be introduced briefly as two parts:

- **Transformer Encoder** is composed of a stack of N identical layers. Each layer is consist of a multi-head self-attention module and a simple fully connected network with Layer Normalization [15] and Residual Connection [16]. We define the encoding function as $Transformer Encoder(\cdot)$ for further use.
- **Transformer Decoder** is similar to transformer encoder, but inserts a third sub-layer to perform multi-head attention over the output of the encoder stack. We also define the encoding function as $Transformer Decoder(\cdot)$ for further use.

3.1.3 Positional Encoding

Position Encoding method is proposed to make use of information of the sequence. Positional Encodings can be sumed on input embeddings to differ the same embedding vector in different positions. We use sine and cosine functions of different frequencies as [15].

3.2 Our Model

3.2.1 Encoder

We simply apply Deep Convolutional Network like VGG [17] and ResNet [16] to encode the visual information. The input features used in our experiment are the feature maps $\mathcal{F}^{n_0 \times d_0}$ from the convolutional layer of ResNet101 [16] where $n_0 = 16 \times 16$ and $d_0 = 2048$.

3.2.2 Connection

The output of encoder will be fed into decoder. To address the problem of high dimension computing time cost and to bridge the gap of visual features with textual representations, we explored three methods as connection module, denoting as function $C(\cdot)$:

- **DirectConnection:** In this part, we regard the visual feature as the input of encoding directly. The connection function is identified mapping:

$$\mathcal{F_C}^{n\times d} = C_{DC}(\mathcal{F}^{n_0\times d_0}) = \mathcal{F}^{n_0\times d_0} \tag{4}$$

  where the dimension of output vectors is as the same as $d = d_0 = 2048$.

- **ReduceConnection:** To address the problem of expensive time cost resulting from high dimension d in identified mapping. We consider inserting a non-linear layer for dimension reduction. Besides of this, the non-linear layer is helpful for mapping the visual feature into the space of natural language. We define the mapping function as C_{RC}:

$$\mathcal{F_C}^{n\times d} = C_{RC}(\mathcal{F}^{n_0\times d_0}) = Activate(\mathcal{F}^{n_0\times d_0}\ W_{RC}) \tag{5}$$

  where $Activate(\cdot)$ is the non-linear activation function and weight $W_{CE} \in \mathcal{R}^{d_0\times d}$ can be used to reduce the dimension of feature map from d_0 to d ($d < d_0$).

- **TransformConnection:** In this method, we process the visual feature from an innovative perspective. From the previous work, it's usual to apply CNN layer on input $\mathcal{F}^{n_0\times d_0}$, where $n_i \in [0, n_0)$ is the index of the specific grid on the feature map and d_0 denotes the number of channels. Compared with that, we can view $\mathcal{F}^{n_0\times d_0}$ as a sequence of vectors from an embedding perspective, where $n_i \in [0, n_0)$ is the index of the sequence, and d_0 is the dimension of every embedding vector.

  However, if we use RNN to model the sequence of embedding vectors $\mathcal{F}^{n_0\times d_0}$, it always suffers from the limited performance for long-range dependencies. However, Transformer solve the long-distance dependence problem by its self-attention architecture [15], where the entire sequence is processed in one time step.

  In TransformConnection, we regard the input visual feature map as sequence of embedding and apply Transformer Encoder (based on multi-head attention) to detect the interested area with given captions, as well as converting the input feature map into memory vectors for the next word prediction. We define C_{TC} as:

$$\mathcal{F_C}^{n\times d} = C_{TC}(\mathcal{F}^{n_0\times d_0}) = TransformerEncoder(\mathcal{F}^{n_0\times d_0}) \tag{6}$$

  where n and d is equal to n_0 and d_0 according to the setting of Transformer. However, to speed up our training in a limited time, we apply a non-linear layer to reduce the dimension. We get the encoder output:

$$\mathcal{F_C}^{n\times d} = C_{TC}(C_{RC}(\mathcal{F}^{n_0\times d_0})) \tag{7}$$

3.2.3 Decoder

Denoting the input textual embedding sequence as $\mathcal{T}^{m\times d}$, where m is the length of sequence and d is the dimension of word embedding. We apply the transformer decoder in [15] for language generation. In details, We employ the Multi-head Self-attention (MS) $MultiHead(\mathcal{T}^{m\times d}, \mathcal{T}^{m\times d}, \mathcal{T}^{m\times d})$ to get the internal result $\mathcal{T}_M^{m\times d}S$. The converted visual information $\mathcal{F_C}^{n\times d}$ and textual information $\mathcal{T}_{MS}^{m\times d}$ is merged in the second sub-layer of decoder, where $MultiHead(\mathcal{T}_{MS}^{m\times d}, \mathcal{F_C}^{n\times d}, \mathcal{F_C}^{n\times d})$. Decoding function can be defined as $D(\cdot)$:

$$\mathcal{T}_D^{m\times d} = D(\mathcal{T}^{m\times d}, \mathcal{F_C}^{n\times d}) = TransformerDecoder(\mathcal{T}^{m\times d}, \mathcal{F_C}^{n\times d}) \tag{8}$$

What's more, we apply mask matrix for $\mathcal{T}^{m\times d}$ during Multi-head Self-attention in case the the next word appearing in the input.

3.2.4 Generator

We simply apply the fully-connected layer with softmax function to obtain the probabilities of words in every position of this sequence. The function can be defined as:

$$Result = softmax(\mathcal{T}_D^{m\times d} W_G + b) \tag{9}$$

where $W_G \in \mathcal{R}^{d\times w}$ and $b \in \mathcal{R}^d$ denote the weight and bias of the fully-connected layer, and w is the size of vocabulary.

3.3 Training

3.3.1 Loss Function

Kullback-Leibler (KL) Divergence is used for our loss function. KL Divergence is a measure of how one probability distrituion is different from another refernce probability distribution. Here we train our network parameters Θ by minimizing the KL Divergence between the output probability $p_\Theta(\mathcal{F}, \mathcal{T})$, where $\mathcal{T}$ will be fed into Masked Attention model, and optimal output probability $p_{\Theta^*}(\mathcal{F}, \mathcal{T})$, where we regard our supervised labels with one-hot format as the optimal output probability. Therefore, our loss function is:

$$\mathcal{L}_{KL}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n} p_{\Theta^*}(\mathcal{F}, \mathcal{T}) \log \frac{p_\Theta(\mathcal{F}, \mathcal{T})}{p_{\Theta^*}(\mathcal{F}, \mathcal{T})} \tag{10}$$

where n is the number of data instances.

3.3.2 Optimizer

We used Adam optimizer [18] and 'warm up' the learning rate in training following this formula:

$$lr = \frac{1}{\sqrt{d}} \ \min(\frac{1}{\sqrt{sn}}, sn \cdot ws \cdot \frac{1}{\sqrt{ws}}) \tag{11}$$

where sn and ws denote the number of steps and the number of 'warming up' steps respectively, where we set $ws = 500$.

4 Experiments

In this section, we present our experiments settings, experiment results and analysis of our proposed model.

4.1 Dataset

Our experiments are conducted on Flickr30k [1] dataset.

Flickr30k consists of 31783 images collected from Flickr. Most images from this dataset contain various human activities. And Each image is described by 5 captions. We split the dataset following the protocol in [2].

Dataset	Training size	Validation size	Testing size	Vocabulary size
Flickr30k	29000	1014	1000	8641

Table 1: Flickr30k dataset summary.

4.2 Settings

To explore the the best type of connection method, we setup three models based on different connection modules mentioned above. The settings are:

It's worth noting that we set the TC-Transformer dimension d as 256 because we consider the model complexity of our connection module and the limited computing resource for us. As for every model,

Model	Connection C	Dimension d
DC-Transformer	*DirectConnection*	2048
RC-Transformer	*ReduceConnection*	512
TC-Transformer	*TransformConnection*	256

Table 2: Model settings for three models based on different connection method and model dimension.

we set $learningrate$ with the 'warm up' scheduler, $batchsize = 32$, number of stack layers $N = 3$ and $dropout = 0.1$.

We set starting symbol <s>, ending symbol </s> and padding symbol <pad> for every sentence. In practice, the length of every sentence is fixed as 90 with padding and truncation.

We mainly implement these three models based on PyTorch 0.4.1 modified from codes of Transformer. We set each of our experiment on single GPU NVIDIA Tesla V100 for one day.

In our experiment, we conduct experiments with the aim of answering the following research questions:

- **RQ1** How do our proposed Transformer-based methods with connection module perform?
- **RQ2** How do the hyper-parameters hidden dimension d affect the model performance?
- **RQ3** How the positional encoding in *TransformConnection* affect the performance?

4.3 Model Performance (RQ1)

At first, we will show the performance of our models based on three connection modules.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	Rouge-L	CIDEr
DC-Transformer	0.569	0.353	0.226	0.149	0.151	0.422	0.156
RC-Transformer	0.549	0.333	0.205	0.128	0.155	0.421	0.173
TC-Transformer	**0.587**	**0.386**	**0.257**	**0.179**	**0.174**	**0.456**	**0.249**

Table 3: Flickr30k results for our three different proposed models.

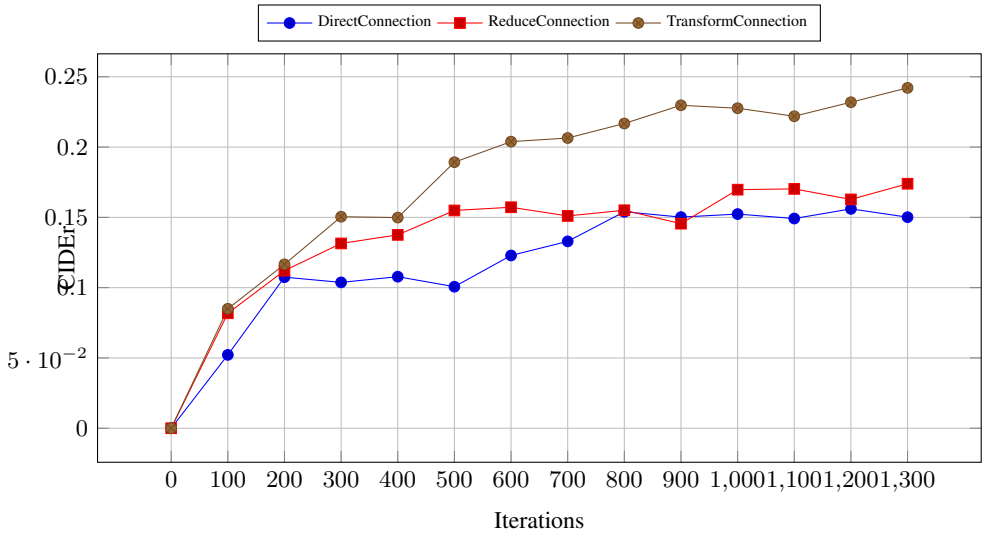


Figure 2: CIDEr under training processing for three different proposed models.

From the results we shown in figure 2 and table 4.3, our analysis run as follows:

- **DC-Transformer** exploits the visual feature directly. Because we should align the dimension as input feature dimension, it's time consuming for training. From the results shown above, we know the performance is poor, which may suffer from the gap between heterogeneous representations like visual and textual information.
- **RC-Transformer** is running efficiently because we reduced the dimension of input visual feature by learned non-linear layer. With the non-linear layer, the performance is better than using visual feature directly in DC-Transformer which improved the CIDEr score by 9.8%.
- **TC-Transformer** regards the visual feature maps from an innovative word embedding perspective, where dimension of embedding is reduced for efficient training. In TC-Transformer, the model exploits self-attention to extract feature from ResNet. Among all metrics above, TC-Transformer outperforms other connection methods significantly.

From the reported results, it's obvious that our Transformer-based model is not as effective as state-of-the-art model like [12]. However, it makes sense to compare our model with the models and baselines used in [5] because it's similar with our paper about proposing novel decoder beyond RNN. We select our strongest model **TC-Transformer** for this comparison:

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	Rouge-L	CIDEr
DeepVS [2]	0.573	0.369	0.240	0.157	0.153	-	0.247
m-RNN [3]	0.60	0.41	0.28	0.19	-	-	-
NIC [4]	0.663	0.423	0.277	0.183	-	-	-
LRCN [19]	0.587	0.39	0.25	0.165	-	-	-
Hard-ATT [6]	**0.669**	**0.439**	**0.296**	**0.199**	0.185	-	-
Soft-ATT [6]	0.667	0.434	0.288	0.191	0.185	-	-
CNN+CNN [5]	0.607	0.425	0.292	0.199	**0.191**	0.422	**0.395**
Ours, TC-Transformer	0.587	0.386	0.257	0.179	0.174	**0.456**	0.249

Table 4: Flickr30k results for our TC-Transformer model and other methods.

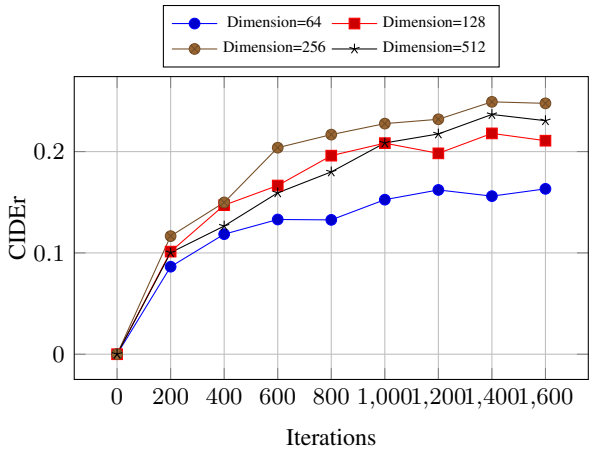
Results of comparison with other models are shown in 4.3. Out analysis is as follows:

- **General comments** According to CIDEr, the performance of our model is sightly better than DeepVS [2] by 0.8%, which shows the effectiveness of our model. But our model is much poorer than CNN-decoder and RNN-decoder according to all metrics except Rouge-L above. We will analyze it below.
- **Analysis of our model** Our model performance isn't as good as we expected, which may be ascribed to these factors
  - **Pre-trained textual model** Since the Transformer architecture has shown its effectiveness in language modeling, it's reasonable to get a better results than RNN-decoder. However, as we known, Transformer is hard to train from scratch like BERT [9], we may need pre-trained Transformer Decoder to improve the performance.
  - **Lack of spacial attention mechanism** We apply self-attention to process the input visual feature, where the query, key and value are identical in Multi-head Attention, therefore, there is not information fed back to visual part for attention when decoding.
  - **Limited computing resource** Much time is required for tuning a deep neural network, especially a totally new architecture for this task. We may need more time to find the optimal model capacity and training strategy.

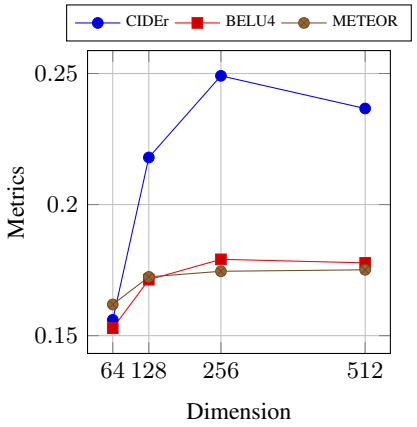
4.4 Hyper-parameter Study (RQ2)

Because our $leaningrate$ is set according to the 'warm up' scheduler and the computing resource is limited to conduct experiments about optimal $batchsize$. We only concentrate on the hyper-parameter dimension d to explore the optimal model complexity for this task:

Results of hyper-parameter dimension d are shown in figure 4.4. From the (a) sub-plot in figure 4.4, we know that when $d = 64$ or $d = 128$, the model suffers from under-fitting problem, where they soon converge . From the (b) sub-plot, we find the turning point occurs when $d = 256$, where $d = 512$ is too complex to train and suffer from the over-fitting problem because of its poorer performance than $d = 256$.



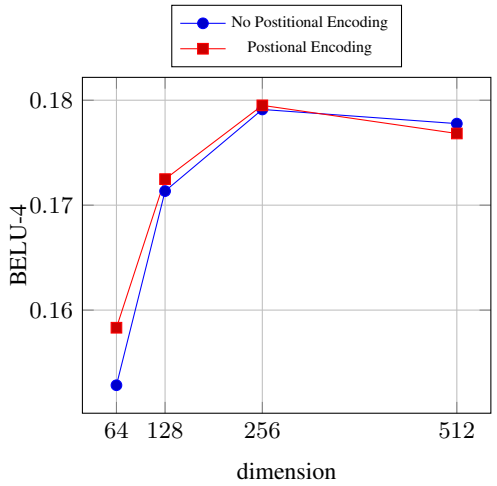
(a) CIDEr under training processing.



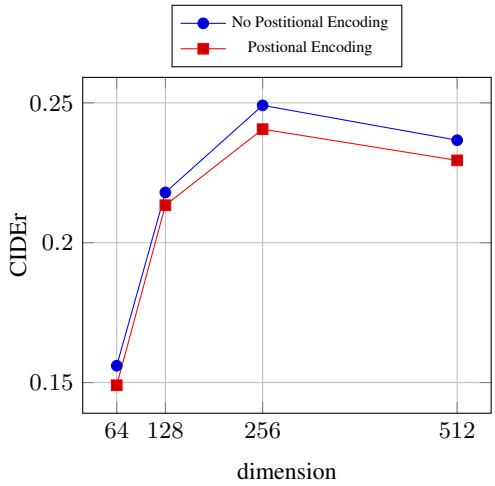
(b) Metrics under different dimensions.

4.5 Positional Encoding in Visual Features (RQ3)

In this section, we would like to explore if positional encoding contributes to our Transformer-based model. The results is shown as follows:



(c) BELU-4 for whether adding Positional Encoding.



(d) CIDEr for whether adding Positional Encoding.

Figure 3: Results of Positional Encoding in Visual Reatures.

From the results in 3, we find that when adding positional encoding to the visual feature, the BELU-4 of our models are not different significantly and CIDEr is slightly worse than not adding positional encoding. We find it make senses because the feature maps from CNNs network owns the attribute about shift invariance from images. According to this analysis and experiments, we remove the positional encoding module in our TransformConnection.

5 Conclusion

Transformer is a great breakthrough for modeling language in Natural Language Processing. It's significant to explore the applications of Transformer for other task like Image Captioning. In this paper, we innovatively set Transformer Decoder to generate captions automatically. What's more, we explored several methods for connection between the visual encoder part and textual decoder part and TC-Transformer achieved the best performance in our experiments. Our model outperformed the

famous model DeepVS [2] totally, which shows the potential of Transformer Architecture. However, there is a long distance with the state-of-the-art method. We still need new architecture to feed the information of textual feature back to visual part, as well as more time and computing resource for parameter fine tuning in the future.

6 Acknowledgments

Thanks to the following open source projects for the great help when we implement our Transformer-based model:

- **Transfomer**[1]: A tutorial and PyTorch implementation for Transformer.
- **COCO-Caption**[2]: A scripts for image captioning metrics.

References

[1] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2(1):67–78, 2014.

[2] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137, 2015.

[3] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *international conference on learning representations*, 2015.

[4] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.

[5] Qingzhong Wang and Antoni B. Chan. Cnn+cnn: Convolutional decoders for image captioning. *arXiv preprint arXiv:1805.09019*, 2018.

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

[7] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3242–3250, 2017.

[8] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. Deep contextualized word representations. In *NAACL HLT 2018: 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237, 2018.

[11] Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W. Cohen. Encode, review, and decode: Reviewer module for caption generation. *arXiv: Learning*, 2016.

[12] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4651–4659, 2016.

[1] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[2] https://github.com/tylin/coco-caption

[13] Qi Wu, Chunhua Shen, Anton van den Hengel, Lingqiao Liu, and Anthony Dick. What value high level concepts in vision to language problems. *arXiv: Computer Vision and Pattern Recognition*, 2015.

[14] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4904–4912, 2017.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *neural information processing systems*, pages 5998–6008, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *international conference on learning representations*, 2015.

[18] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *international conference on learning representations*, 2015.

[19] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.