Attention based image to caption using a Transformer based network

Willem Diepeveen
TU Delft
w.diepeveen@student.tudelft.nl

Daphne van Tetering
TU Delft
d.a.l.vanTetering@student.tudelft.nl

Niek van der Laan
TU Delft
n.vanderLaan-1@student.tudelft.nl

Paul Verkooijen
TU Delft
p.j.m.verkooijen@student.tudelft.nl

Kasper Wendel
TU Delft
k.wendel@student.tudelft.nl

The implementation of this network is available online at https://github.com/kwendel/deeplearning.

1. Introduction

The canonical method for sequential modelling in deep learning is the Recurrent Neural Network (RNN). In recent years, other architectures have been proposed and were shown to perform better in several cases. Especially using convolution or attention to model recurrence got more popular after the papers by Bai et al. [2] that proposes the Temporal Convolutional Network, and by Vaswani et al. [5] that proposes the Transformer network. The latter of the two networks is something special as it does not use any recurrences nor convolutions, making it very attractive due to its high degree of parallelism. The network is purely built up from attention mechanisms and fully connected layers. Although these networks have been tested against and outperform the accepted RNN variations, these transformer networks are not the standard yet for many applications. In the task of image captioning, which entails generating a caption describing the picture given as input, state-of-the-art work still makes use of recurrent networks to generate the captions. We hypothesize that by replacing these for the Transformer network, we can improve performance as a result of high parallelism.

2. Problem Statement

In this work we want to test the Transformer network on yet another task: image caption generation. This problem entails generating a caption describing the picture given as input. Xu et al. [6] used a state of the art network consisting of a VGGNet and a LSTM to make an algorithm that turns an image into a caption in their paper *Show, Attend and Tell*. Previous research already showed that the Transformer can work very well on its own. We want to know whether this still holds when it gets its input from the VGGNet to incorporate attention. The central question will be whether the VGGNet-Transformer pair can outperform the network with the LSTM in terms of captioning performance.

Inspired by this image-to-caption task, our main objectives are:

- Implement a trainable VGGNet-Transformer combination for the image-to-caption task
- Compare the performance with the VGGNet-LSTM network

2.1. Dataset

During this research, the Flickr8k dataset will be used. This dataset consists of 8000 images with five captions each, where both the image and text data needs to be preprocessed differently. Each image-caption pair is used separately to prevent data from being unused. Presumably, this also has a regularization effect as it prevents the network from learning only one caption per image and instead learns the structure that is present in the image. For the captions, the provided lemmatized captions were used to simplify the vocabulary in the training data.

2.2. Expected Results

As stated before, the goal of the project is twofold: implement a trainable network and investigate performance. For the trainability, we expect to build a network that is able to overfit on a smaller subset of the Flickr8k dataset. If the proposed network is indeed trainable, we expect the network to give a reasonable result when trained on the whole Flickr8k dataset.

2.3. Evaluation

If we find a converging network, the evaluation will also be done twofold: using the BLEU score and SPICE. If not, the remainder of the research will be devoted to trying to solve possible problems.

2.3.1 BLEU

BLEU (bilingual evaluation understudy) [4] is an often used evaluation metric in machine translation tasks. BLEU scores will be computed to enable performance comparison with the work of Vaswani et al. [5]. The BLEU score determines the similarity between reference and predicted caption by matching the n-grams in a sentence.

2.3.2 SPICE

SPICE is a novel evaluation metric designed specifically for image captioning, presented in the work by Anderson et. al [1]. Unlike other metrics like BLEU or CIDEr, SPICE determines the quality of a caption by analyzing its semantic content. To do so, all predicted and reference captions are formed into a scene graph that encodes objects, attributes and relationships found in each caption. The similarity between a generated caption c and its reference captions $S = s_1, ..., s_m$ is then defined as:

$$SPICE(c,S) = F_1(c,S) = \frac{2 \cdot P(c,S) \cdot R(c,S)}{P(c,S) + R(c,S)} \quad (1)$$

where P and R are Precision and Recall respectively. For further explanation on the SPICE-metric we recommend reading the work by Anderson et. al [1]. To compute the SPICE scores for our model we used the implementation provided by Anderson et al., which can be found here: `http://panderson.me/spice`.

3. Technical Approach

In this section we will elaborate on the assembly of the network model and accordingly discuss the ways of training and testing it.

3.1. Model

The model can be summarized with the following flow for inference as shown in Figure 1:

- A picture of a certain size will be given to an encoder (the VGGNet)
- A linear layer is used to compress the VGGNet output to a hidden state and this is passed on to the Transformer decoder
- The decoder will be given an embedded <START> token
- Based on the embedding and the VGG output, the next word in the sentence will be guessed.
- The word will be appended and passed onto the bottom of the Transformer decoder and the process will be repeated until <STOP> has been guessed.

The training flow is very similar and will be elaborated on in subsection 3.2.

The proposed network as shown in Figure 1 is inspired by the caption transformer network used by Zhu et al. [7]. Although their network did work adequately, we argue that there are some improvements to be made. Sections below describe our adaptations, their underlying ideas and our implementation.

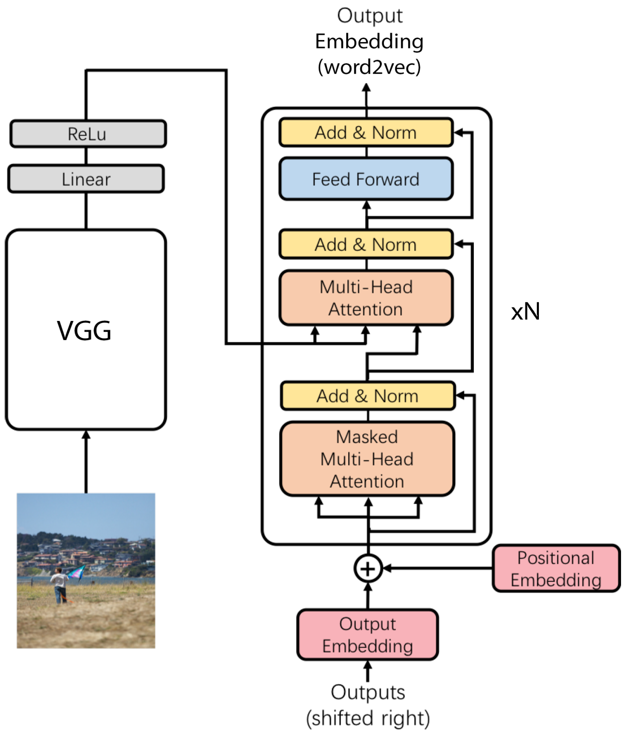

Figure 1. The proposed network inspired by the network of Zhu et al [7]. The VGG will output an array size 512×196. The fully connected layer will reshape the image as input for the transformer. The Transformer initiates with the <START> vector and keeps predicting the next word until it guesses <STOP>.

3.1.1 Word embedding with word2vec

Different solutions for word embeddings were tried. Initially, we tried to encode words with a one-hot encoding using SentencePiece [3] as a tokenizer. SentencePiece is a unsupervised text tokenizer that tokenizes captions using parts of words and these parts are determined using an Expectation-Maximization algorithm. For example *science* and *experience* are split into *sc_*, *exper_* and *ience*. Using SentencePiece, the vocabulary size can be set as a hyperparameter. However, we found that a one-hot encoding of the tokenized captions would be too memory expensive. Therefore, another word embedding was chosen, namely a word2vec model.

The word embedding is one of the big network adaptations in our work. The original Transformer decoder proposed by Vaswani et al. [5] trained a word embedding: first a dictionary was made and one-hot encoded, which was given to a fully connected layer before passing it on to the Transformer. In this work, a pre-trained word2vec embedding outputting vectors of length 50 is used. This net has been trained on the GloVe dataset. The main reasons are:

- The resulting network could be better generalizable, since the vocabulary is not restricted to the Flickr8k dataset;
- Fewer parameters have to be trained, which will result in less computation time.

The latter is of great importance due to the \$ 250 Google Cloud Platform budget constraint. Since we do not train our own embedding, it is possible some words from the train set are not present in the embedding. This was checked in preprocessing steps and it was found that all correctly spelled words in the training set were also present in the embedding.

The pre-trained space does not contain the <START>, <STOP> and <PAD>-tokens. The <PAD>-tokens in particular are needed to generate sequences of equal lengths in order to effectively compute the loss for the model. To overcome this problem, the vector space is extended with two dimensions. A regular vector will be extended with two zeros in the last two dimensions. The tokens will be given the values:

1. <START> as $[\mathbf{1}_{50}, 1, 0]$;
2. <STOP> as $[\mathbf{1}_{50}, 0, 1]$;
3. <PAD> as $[\mathbf{1}_{50}, 1, 1]$.

where $\mathbf{1}_{50}$ is a vector with ones of size 50. This method sets the first hyperparameter in the Transformer model, the dimensions to $d_{model} = 52$.

3.1.2 VGG encoder

The next step is the picture encoding. We follow the approach by Xu et al. [6], in order to remain as close as possible to their image-to-caption network with the LSTM. Hence, we will also use a pre-trained VGGNet as shown in Figure 2. This gives us again the advantage of fewer parameters to tune. This network takes a 224×224 image as input and outputs 512 feature maps of size 14×14. The linear layer resizes the features to $512 \times d_{model}$, which was determined by the choice of word embedding.

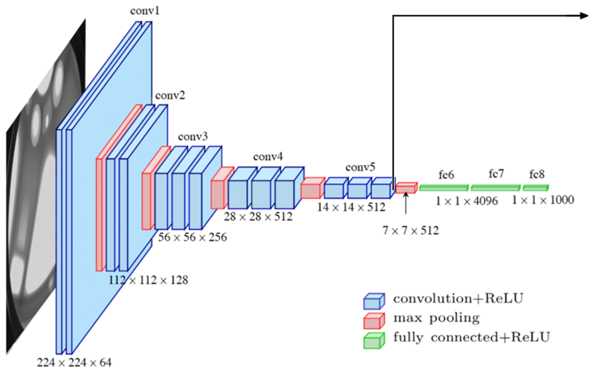

Figure 2. Image of the pre-trained VGGNet. The feature maps obtained after the 5th convolutional layer will be flattened to get the shape 512×196 used as the VGG output of the proposed network.

3.1.3 Transformer decoder

The Transformer decoder from *Attention is all you need* [5] was used. An open source TensorFlow implementation of the Transformer was retrieved from `https://github.com/Kyubyong/transformer`. For our use, the model was slightly adjusted. The linear layer at the end of the model, and the softmax were dropped. The linear layer was necessary in the original implementation to convert the decoder output back to the dimensions of the one-hot encoded words. These two layers can safely be removed from the network as a different approach is used for the text embedding. Positional sine encoding [5] is applied to preserve the order of the target sequences, allowing the model to make use of semantic order information.

3.2. Method

This section contains all information on our methods. First, we tried to **overfit** the network on a **mini-dev** set to test **trainability** and **compare** on a **full** set for **performance** testing.

3.2.1 Data

The Flickr8k dataset consists of 8000 images. The authors of the dataset provided splits into training, testing and development sets containing 6000/1000/1000 images respectively. The first objective presented in section 2 is to implement a trainable Transformer based network for the image-to-caption task. To see if the network is trainable, it was decided to initially train the network on a smaller train set. A mini-dev set was created, containing 100 randomly selected images from the Flickr8k developer set. This mini-dev set also allowed us to quickly test the effectiveness of changes to the model.

If the loss was decreasing and the results were promising, the train set from Flickr8k will be used to train the network properly. The test set will be used for evaluation of the results.

3.2.2 Training

Training of the network was done using the Google Cloud Platform (GCP). The budget available for training the network was \$250, which was enough to train our network for about 48 to 72 hours with the following GCP specifications:

- 2 vCPUs with 7.5GB memory
- NVIDIA Tesla V100 GPU
- 500GB disk space

During training, optimization and regularization techniques were added to the network. For optimization of the learning rate, Adam was used with a Noah scheme. This optimization method was used to mimic *Attention is all you need* [5], which used the same optimization technique. For regularization, dropout was used with a dropout rate of 0.3.

As mentioned before, training is done on the lemmatized captions of the Flickr8k dataset. In the lemmatized captions, the words *a* and *an* are still present. However, since these words are often present in captions we decided to remove these from the training captions. It was expected that this should improve the convergences rate of the model.

The loss function used in the model is the cosine distance loss function. The cosine distance loss is defined as

$$\mathcal{L} = 1 - \frac{y \cdot \hat{y}}{\|y\| \cdot \|\hat{y}\|}$$

The cosine loss function was used due to the fact that a word2vec embedding was used. The idea of this embedding is that the angle of the vector determines the meaning of word, while the length of the vector has no meaning. So to find a similar word in this embedding, the cosine similarity is also used. Now with the cosine loss, we can easily define how much the orientation of predicted word differs from the true word in this 52-dimensional space. Our initial assumption was that orientation was enough due to similar words being clustered together in the word2vec model. We expect the difference in magnitude between two word vectors being less important, since similar words have similar orientations. Therefore, our design only included the cosine distance loss, focusing on the difference in orientation.

Finally, the loss function is calculated over the entire predicted sequence. This results in the fact that predicted <PAD> tokens will also be used in calculating the loss. We decided to calculate loss over both predicted words as tokens, since no quick, parallelizable solutions exist for this. Our idea was that including <PAD> tokens in calculating the loss would not have a large influence on the results and the speedup obtained was preferable.

3.2.3 Validation

As stated in subsection 2.3, two metrics will be used to compare predicted captions with an images true captions. As mentioned there, both BLEU and SPICE scores will be calculated if the network converges and sensible results are produced. However, as will be explained in section 4 the network did not converge into sensible results. Therefore, time was devoted to fix problems in the network rather than evaluating retrieved scores. Still, this section will explain how evaluation would have been performed in case research on this subject is continued.

The network will be evaluated using the test set from the Flickr8k dataset, a split provided by the authors of the dataset. Both scores can be calculated using multiple reference captions. Therefore, for each of the images in the test set we will use all five true captions as reference. For both BLEU and SPICE Python libraries exist, which should make it easy to implement this in our implementation.

BLEU For BLEU, we will calculate the BLEU-1 score, which calculates the score based on 1-grams. These results can be directly compared with the results from Xu et al.[6], which used the same BLEU-1 score for their VGGNet with an LSTM for the image-to-caption task. However there exists a small difference, as the presented network is trained using lemmatized captions, whereas the implementation from Xu et al. is trained using true captions. However, we think that comparing the BLEU-1 scores for both implementations will still give insights in the performance gain or loss when using a Transformer based model.

SPICE The SPICE score will be calculated but cannot be compared with the implementation from Xu et al. However, since SPICE is specially created for image captioning we believe including SPICE scores is a great addition for future research. Since SPICE is specialized for image captioning we believe it might be used more often in the future. By including SPICE in our research, future image-to-caption models can easily be compared with the presented Transformer based model.

3.3. Network adaptations

The model is built upon the open source implementation[1], which is used as the decoder in our network. To combine this with the encoder, that encodes the VGGNet's intermediate predictions, the output of the encoder should match the size of the word embedding and two extra dimensions, as explained in subsection 3.1. To achieve this, the encoder is extended with a fully connected feed-forward layer and a ReLU to a hidden state of size d_{model}, which by matching our word embedding results in $d_{model} = 52$.

Furthermore, to deal with word2vec embedding instead of one-hot encoding, the loss function in the used implementation is changed to the cosine distance loss. The loss function is implemented using TensorFlow functions, guaranteeing that the loss can be calculated in parallel.

The maximum length of captions was determined in preprocessing steps and implemented in a way that the model will never predict captions longer than the maximum length found in the train set. If an <END> is predicted on any position smaller than the maximum caption length, prediction will terminate early and the caption will be padded until it equals the maximum length.

4. Results

The presented model was run two times on different training sets using different word embedding. The first run was performed on the Flickr8k dev set and converged to a loss of about 0.02, which can be seen in Figure 3. Two positions in the sequence were inspected further. The first point checked is at $k = 2024$, which takes place slightly before the large decrease in loss seen in Figure 3. A random sample at this point can be seen in Figure 5. Other caption predictions around this point have similar results. The second point we checked is at $k = 2676$ and can be found in Figure 6. From here on, all predictions made are the same as they all predict the <START> token repeating 50 times.

After some changes to the model, which are discussed in depth in section 5, a second run was performed on the mini-dev set. As visible in Figure 4, this converged to a loss of about 0.025. When looking at intermediate results obtained with this model, it becomes visible that instead of predicting the <START> token, the model has taught itself to predict the <PAD> token. Since we remove these tokens from each predicted caption, this results in empty sentences. Another striking fact is the appearance of the dot character in the predicted captions, even though these are not present in the train captions. An example of two predicted captions is shown below:

- $\boxed{k = 305}$ <PAD> (repeats 50 times)
- $\boxed{k = 353}$ one (repeats 7 times) . . . . one (repeats 16 times) . . . . . . one (repeats 3 times)

Since the network did not converge to correct predictions, no evaluation using BLEU and SPICE is performed. Time saved was spend in improving the model, but due to budget restrictions we were not able to obtain more results.

[1]https://github.com/Kyubyong/transformer

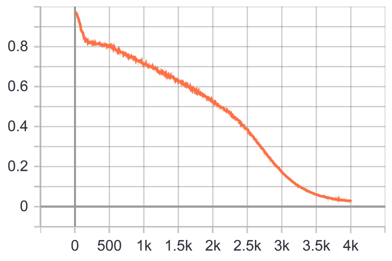

Figure 3. The cosine distance loss for the Flickr8k dev set.

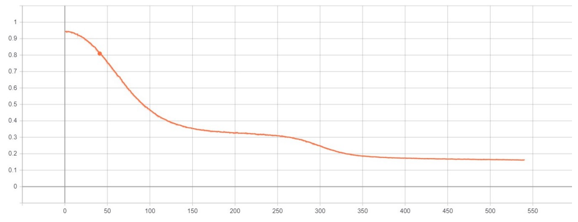

Figure 4. The cosine distance loss for the mini-dev set.

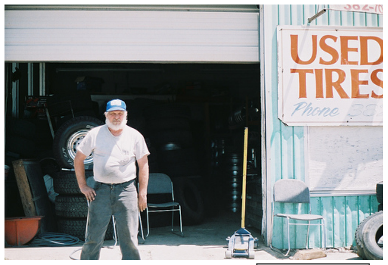

Figure 5. True and predicted caption at $\boxed{k = 2024}$
Real: START man at use tire shop have white beard END
Pred: pail man boy boy girl girl girl girl girl boy boy boy boy boy boy boy boy boy boy boy boy baby baby my you you you you pack pack walk walk walk walk walk walk walk walk walk sit sit sit sit sit sit sit sit sit sit

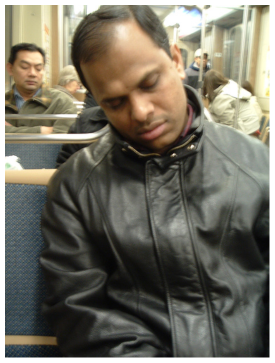
Figure 6. True and predicted caption at $\boxed{k = 2676}$
Real START man in black leather jacket be sleep in subway car END
Pred: <START> (repeats 50 times)

5. Discussion

In this section we will discuss and give recommendations on our work. First, we will show why the first run did not give the expected results and which improvements were made to the model in response. Secondly, a small reflection on the second run will be given. Finally, we give some recommendations on how to improve the models loss function for future work.

5.1. First run reflection

The presented network in the first run converged as seen in Figure 4 from around $k > 2500$. However, as an example in Figure 6 shows, the predicted captions all contain the <START> repeated fifty times. The root of the problem lays with the loss function. We know that all captions are rather short and are padded to the longest caption in the train set. This longest caption consists of 34 words, resulting in the fact that all shorter captions are padded till they reach this length of 34 tokens. The implemented values for <PAD> is $[\mathbf{1}_{50}, 1, 1]$, while the <START> token is implemented as $[\mathbf{1}_{50}, 0, 1]$. We found that these chosen values are too similar using a cosine distance loss function. When predicting a caption, the loss when predicting the <START> token while the true caption contains a <PAD> token, gives a loss of $1 - 51/52 \approx 0.02$. Combined with the fact that most captions are short and thus contain a lot of <PAD> tokens, we know that the loss on an entire caption when predicting only <START> will also approximate 0.02. When looking at Figure 4, we can see that the loss approximately converges to this value, explaining why only start tokens are predicted.

To solve this problem for the second run on the mini-dev set a different embedding for the <START>, <END> and <PAD> tokens was used. This embedding further promoted dissimilarity between the three tokens, they should not be predicted anymore. An example of tokens which are more dissimilar than in our initial solution would be:

1. <START> as $[\mathbf{rand}([-1, 1])_{50}, 1, 0]$
2. <STOP> as $[\mathbf{rand}([-1, 1])_{50}, 0, 1]$
3. <PAD> as $[\mathbf{rand}([-1, 1])_{50}, -1, -1]$

5.2. Second run reflection

As shown in section 4, the second run predicted captions containing dot characters. However, lemmatized captions were used for training which do not contain any dots. Therefore, we suspect problems arise from both the loss function and the word2vec embedding. This section will discuss problems on the word2vec space. The next section will go deeper into recommended alterations to the loss function. We expect that the word2vec space, which is GloVe pre-trained on an internet corpus, gives problems for our model. The GloVe embedding used is GloVe.6B, containing 6 billion tokens spanning a vocabulary of 400.000 words. Upon inspecting the results of the second run, we expect this to be a problem. The vocabulary of the train set of Flickr8k contains an vocabulary of about 20.000 words. Thus, the pre-trained word2vec model spans much more words, making it possible to predict words never seen before in the train set. This results in for example making it possible to predict a dot character. While it was expected that predicting words outside of the vocabulary could be useful, it actually worked against the network. Therefore, a suggestion for future work is to create a word2vec embedding that is trained on the words appearing in the Flickr8k train set. This should prevent the model from predicting dots and other words not present in the vocabulary. This should also prevent the model from predicting words which are not likely to occur in captions.

5.3. Loss functions recommendations

For future runs adjustments can be made to the loss function to improve results. These adjustments might further improve the predicting results of the model. Since the cosine distance loss only looks at the difference in orientation between two vectors in the embedding space, the model currently ignores the magnitude of these vectors. Another loss function that measures this magnitude in the form of a norm is needed. The L2 norm loss function tries to minimize the squared distance between the predicted and true captions. By combining the cosine distance and L2 norm functions, a loss function is created that looks at the difference of both orientation and magnitude between two words in the word2vec space.

The second proposed adjustment is to calculate the loss of predictions only on existing words and not on special tokens. Currently we also calculate the loss over predicted padding tokens as explained in subsection 3.2. Results have shown that including <PAD> in predictions might have more influence than we initially thought. Therefore, it might be feasible to adjust the loss function to only include words instead of all tokens. During training, predicted captions longer or shorter than the real caption, can receive a penalty on the loss. This should prevent the model from predicting too long captions. However, implementing a loss function which only looks at predicted words will increase training time of the network, due to difficulty of paralellizing the removal of padding tokens.

6. Conclusion

We implemented a trainable Transformer-VGGNet combination. The major drawback is probably the choice for embedding the special tokens <START>, <STOP> and <PAD> in a word2vec space. We proposed and tested a solution to this problem, which gave meaningful insights into the difficulty of this problem but did not yield the expected results so far. We suspect further adjustments to the loss function to better fit the embedding space could improve our model and we suggested several of such adjustments to be made in future work.

References

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

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[3] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018. Source code available on GitHub.

[4] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, 2002.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[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. In *International conference on machine learning*, pages 2048–2057, 2015.

[7] Xinxin Zhu, Lixiang Li, Jing Liu, Haipeng Peng, and Xinxin Niu. Captioning transformer with stacked attention modules. *Applied Sciences*, 8(5):739, 2018.