Parallel Image Captioning Using 2D Masked Convolution

Chanrith Poleak and Jangwoo Kwon *

Department of Computer Engineering, Inha University, Incheon 402-751, Korea; poleakchanrith@gmail.com

* Correspondence: jwkwon@inha.ac.kr

Received: 10 April 2019; Accepted: 30 April 2019; Published: 7 May 2019

Featured Application: This work can be used in applications that require image-context analyzing function such as image/video indexing, content descriptor for visually impaired, and automatic image/video annotation.

Abstract: Automatically generating a novel description of an image is a challenging and important problem that brings together advanced research in both computer vision and natural language processing. In recent years, image captioning has significantly improved its performance by using long short-term memory (LSTM) as a decoder for the language model. However, despite this improvement, LSTM itself has its own shortcomings as a model because the structure is complicated and its nature is inherently sequential. This paper proposes a model using a simple convolutional network for both encoder and decoder functions of image captioning, instead of the current state-of-the-art approach. Our experiment with this model on a Microsoft Common Objects in Context (MSCOCO) captioning dataset yielded results that are competitive with the state-of-the-art image captioning model across different evaluation metrics, while having a much simpler model and enabling parallel graphics processing unit (GPU) computation during training, resulting in a faster training time.

Keywords: computer vision; convolutional networks; LSTM

1. Introduction

Image captioning (a task that allows a computer to automatically describe the content of an image in natural language) is a very challenging problem. To do this, the machine is required to not only recognize the objects contained in the image but to also understand the relationship between each object in the scene, along with certain attributes such as color, quantity, size, etc. As challenging as it seems, image captioning also holds great interest in the research community because it has many benefits, ranging from assisting the visually impaired in understanding image content to improving the image indexing and annotation tasks [1].

For many years, the main approach to image captioning has always been a combination of a convolutional neural network (CNN) as an encoder and a recurrent neural network (RNN) as a decoder. This approach has shown great results in the image captioning task, and it has seen many different adaptations and improvements from the research community. However, RNNs have known limitations, such as the parallel training problem and vanishing gradients, which result from the sequential processing nature of the model [2]. Even though long short-term memory (LSTM) [3] was later used in place of the RNN to solve the vanishing gradient problem, LSTM itself requires a large amount of memory, since it has to compute multiple linear layers per cell during its sequential learning. For this reason, many studies have been done to find a way to solve this problem [4,5].

Recent research in neural machine translation has seen many successful attempts to replace LSTM with a convolutional-based architecture [6,7]. Inspired by these studies, we also propose

an end-to-end convolutional-based image captioning approach that has a result comparable to the traditional CNN-LSTM model while also enabling parallel graphics processing unit (GPU) computing, which reduces the model training time on a Microsoft Common Objects in Context (MSCOCO) dataset from two days to less than 20 h. On the MSCOCO test set, we achieved evaluation scores of 73.6 with the BLEU-1 metric, 25.9 with METEOR, and 99.4 with CIDEr, which outperforms the conventional LSTM and non-LSTM approaches by a considerable margin, and is comparable to the state-of-the-art image captioning model [8].

This paper makes two main contributions to image captioning research. First, we created a convolutional-based image captioning model that gives state-of-the-art performance over non-LSTM based models in some evaluation metrics. Secondly, we created a model that requires fewer parameters and less training time, compared to the state-of-the-art LSTM model, while also enabling parallel GPU computation.

2. Related Work

2.1. Image Captioning

The method of describing the visual data in natural language form has been studied since the early days of deep learning. The first approach would be in the form of an image classification task [9], where a model tries to classify an image into different categories. After that, other authors [10,11] proposed models that make use of a region proposal to generate bounding boxes on different areas of the image.

Recently, interest in the field of image captioning has started to grow tremendously, where researchers have started using recurrent connection such as LSTM for their caption generation model. One of the earlier adoption of this approach was proposed by Vinyals et al. [12]. This method is a simple multimodal architecture that uses CNN to extract features of the image, which are the input for the initial state of the LSTM model. During each sequence, input at time t largely depend on the output of the LSTM at time $t - 1$ and the training process happen sequentially, one word at a time, until it reaches a special end of sentence token. This method effectively generates a meaningful caption, but lacks the understanding of spatial relationship between the image and current word token. To solve this problem, Xu et al. [13] extended the work of Vinyals et al. [12] by proposing two attention methods that make use of the feature from the last convolutional layer of CNN to calculate weighted context vector of the image, allowing the model to focus on the salient regions of the image when generating the caption. Beside from improving the architectural part of the model, Rennie et al. [14] proposed an optimization approach to improve the result of the image captioning model by using reinforcement learning to normalize the reward during the test-time inference. Using this method, the model can identify and give positive weight to the best samples and reduce the possibility of generating an inferior sample. Many attention methods in image captioning use top-down approaches, which consider the whole context of the image when generating the salient image regions, while giving little focus to the objects that appear in the image. Therefore, Anderson et al. [8] proposed a method combining both top-down and bottom-up attention mechanisms into the decoder layer of image captioning. For the bottom-up attention, the authors used Faster-RCNN [15] model to detect all the regions of objects presented in the image and used mean-pooled convolutional feature to convert these regions into image feature. This proposed method establishes state-of-the-art result by achieving first place on the current MSCOCO test server.

Despite making different contribution to the image captioning research, all the models mentioned above share the same sequential structure regarding the training process where a token at time t can be processed only after the process at time $t - 1$ is completed. Due to this sequential computation of the LSTM, the model cannot utilize the parallel feature of multi-GPUs, which can be an issue when performing on longer sequence task that require larger memory. To solve this problem, there is also a lot of research on using a convolution and self-attention model for this task. Aneja et al. [16] first

proposed a convolutional approach based on earlier work [7] by replacing LSTM cells with a CNN and an attention mechanism, while Zhu et al. [17] used the self-attention approach from Vaswani et al. [18] for the decoder part of the image captioning model.

2.2. Sequence-to-Sequence Model

Sequence-to-sequence tasks, such as neural machine translation, are dominated by the RNN-based encoder–decoder model. However, since recurrent connection in sequential learning has a lot of issues, many of which are related to the sequential process of the network, new research has focused on replacing the RNN with different approaches, such as the convolutional network and attention. Successful non-RNN work is presented in [19], where the authors used the concept of the masked convolutional network to restrict information of future tokens from the current state that is being predicted. By doing so, the model only uses information about the current and the past time steps to learn the word sequence. Later, the authors of [6,7] adopted this method, and other authors further improved the model by using gated linear units (GLUs) [20], allowing the network model to learn the long-range dependencies of the input word so that it could perform even better than most of the RNN approaches to the machine-translation task. Besides the convolutional approach, Vaswani et al. [18] proposed an attention-based model that uses positional encoding and multi-headed self-attention to learn the sequence order of the input tokens and generate targeted sentences without using any sequence-aligned recurrent or convolutional neural network.

3. Materials and Methods

The main goal of our model is to automatically generate a contextual description when receiving the input using a convolutional approach for both encoder and decoder functions of the model. To do so, we replace LSTM in the decoder with the mask convolutional approach used in [6]. Unlike the LSTM approach where caption model process one word token at a time, our model can process all tokens in the sentence at once in a feed forward manner. By doing so, the model can also take advantage of multi-GPUs computing power allowing large batch input for each training iteration and faster convergence speed. Figure 1 shows the detailed implementation of the model.

The overall process of our image caption model is as follows: during training, our CNN encoder receives input image and learns its feature representation, while our decoder takes the output feature from the encoder and the corresponding captions to establish a relationship, and then outputs a text description of the image. We discuss each part of the model and their implementation in the next sections.

Figure 1. The architectural overview of our image captioning using the 2D-mask convolutional network.

3.1. Image Encoder

Our encoder plays the role of feature extractor that learns the representative feature from the input image. In this work, we have chosen to use a variant of the CNN architecture called Resnet-101 [21], which makes use of a deep residual connection to learn the high-level information of the image. To get

the feature of the image, we use the output from the last convolutional layer of the Resnet-101 model and apply adaptive average pooling to get a 512-dimensional feature map, Q , sized at 14×14 , which we use for self attention. In addition, we apply a spatial average across all pixels to get a 2048-dimensional feature representation and a linear layer to map our image feature to a 512-dimensional vector, which is used as input for the decoder.

3.2. Decoder

The decoder in our model consists of three main components: a word-embedding layer, a 2D masked convolutional layer, and the prediction layer. First, when receiving input from the model, the word-embedding layer encodes the caption of the image into an n -dimensional vector space. Then, this output from the word-embedding layer is concatenated with the image feature from our CNN encoder, which results in a tensor that is further processed through the 2D masked convolutional layer in order to generate a meaningful output caption. Each of the layer processes are thoroughly described below.

3.2.1. Word Embedding

Before the sentence from the dataset can be input into the word-embedding layer, it needs to be converted into the $|V| \times 1$ -dimensional space using a 1-hot encoder, where $|V|$ is the size of the vocabulary dictionary that we generate using all the words existing in the dataset. Then, each word vector is input into this embedding layer, which has a size of $512 \times |V|$, outputting a 512-dimensional vector that is concatenated with the image features from the CNN encoder.

3.2.2. Positional Encoding

Besides using word embedding to embed a sentence into a word vector, we also apply positional embedding to the 1-hot encoded vector of the sentence, enabling the model to understand the positional information of the input token and to make sense of the parts of the sentence passed to the model. In this work, we use the sinusoidal version of the positional encoding from [18], because it allows the model to generalize into a longer sequence during reference time. Sinusoidal positional encoding makes use of the sine and cosine functions to generate a wavelength that represents the frequency for different word tokens. The positional encoding can be calculated with the equations below:

$$PE_{(pos, 2j)} = \sin(pos/10000^{2j/d_{model}}) \quad (1)$$

$$PE_{(pos, 2j+1)} = \cos(pos/10000^{2j/d_{model}}) \quad (2)$$

where pos is the position, j is the dimension of our input, and d_{model} represents the embedding dimension of our word vector. After applying positional encoding to our word vector, we use its output and apply summation with the output from word embedding, resulting a 512-dimensional vector, emb_i .

3.2.3. 2D Masked Convolutional Layer

Unlike the traditional image captioning approach where the input is processed through the LSTM model, we decided to use this convolutional approach as a replacement. This approach was inspired by the method of Elbayad et al. [6], which uses this masked convolutional layer for both encoder and decoder in the sequence-to-sequence problem of neural machine translation. Consistent with Elbayad et al. [6], our convolutional layer is adopted from the DenseNet architecture [22], in which each layer's input is taken from the preceding layer to produce a long-distance connection that is useful for solving the vanishing gradient problem and that improves the gradient flow. Figure 2 shows the use of this architecture.

Figure 2. The DenseNet architecture of the 2D masked convolution.

In this architecture, we first apply batch normalization and rectified linear units (ReLU) to our input from the encoded word-embedding vector. Then, we apply a 1D convolution layer to downsample this output to reduce the computational cost. This layer is followed by more batch normalization and ReLUs, and we apply the masked convolution layer, which has the kernel size $k = 3$. This convolution-layer mask allows us to mimic the LSTM approach by restricting the future information flow from our input, and, hence, only making the prediction based on past data. In addition to this, we apply the tanh activation function between the convolutional layers of our model, which gives the output of a 512-dimensional vector for each word in the sentence. For our model, we stack this 2D masked convolutional layer $l = 4$ times, which increases h_i^l , the state size of the input elements, to 9. Between each layer, we also apply GLU nonlinearity so that the model focuses only on the most important elements.

Aside from the GLU unit, a residual connection was also added to each layer of this 2D masked convolutional layer to allow the model to have a deep residual connection [21] between each of the output states. The residual connection can be expressed in the equation below:

$$h_i^l = [W^l(h_{i-k/2}^{l-1}, \dots, h_{i+k/2}^{l-1}) + b^l] + h_i^{l-1} \quad (3)$$

where l is the layer of our 2D masked convolution, i represents the input token, and b is the bias.

3.2.4. Multi-Step Attention

Attention plays a really important role in the language generation task since it allows the model to understand which part of the sentence it should focus on during each sequence. Besides the language task, the attention mechanism also proved to be effective in the image captioning model [13]. In this method, we implement the multi-step attention layer from Gehring et al. [7] that uses emb_i to help providing word context, and image feature Q and residual connections, which allow the model to attend to all parts of the image across different layers. The implementation of multi-step attention is shown in the equations below:

$$A_i^l = W_p^l h_i^l + b_p^l + emb_i \quad (4)$$

$$v_i^l = \tanh(A_i^l \cdot Q) \quad (5)$$

$$a_{ij}^l = \text{softmax}(v_i^l \cdot z_j^u) \quad (6)$$

$$c_i^l = \sum_{j=1}^m a_{ij}^l z_j^u \quad (7)$$

where A_i^l represents the decoder state summary of state i for layer l , emb_i is the output word embedding vector, v_i^l is the non-linear dot product between state summary and image feature Q , z_j^u represents the output of each 2D masked convolution, u , for sequence j , a_{ij}^l is the attention element, m is the length of

the sentence, and c_i^l is the attended vector for word j . Figure 3 shows the detailed implementation of the 2D convolution architecture of the decoder model.

Figure 3. The detailed implementation of our 2D convolution. After receiving the output from word embedding, we apply a 2D masked convolutional layer followed by a gated linear unit and an attention layer. We stack this 2D convolution four times before applying output embedding to get the output word sequences.

3.2.5. Prediction Layer

For each word, we apply max pooling to the output vector that we get from the convolutional layer. Then, we apply linear embedding with a size of $|V| \times 512$ to our pooled feature to map this output to the $|V|$ -dimensional vector. Finally, we apply Softmax to this output in order to obtain the probability distribution of the word in the dictionary.

To evaluate and learn the parameters of each iteration of our model, we use cross entropy loss as our loss function. Let us denote θ as all parameters required of our 2D convolutional model; then, our objective is to minimize the loss, as follows:

$$s_i = W_s(c_i^l + h_i^l) \quad (8)$$

$$L(S|I, \theta) = \sum_{i=0}^m \log P_i(s_i | s_{j < i}, I, \theta) \quad (9)$$

where s_i is the output token for sequence i , m is the total length of the sequence, j is word sequences before i , and $L(S|I, \theta)$ is the cross-entropy loss for our sentence.

4. Experiment

4.1. Dataset

We used the MSCOCO 2014 dataset [23] as the experimental dataset for our model. MSCOCO is a large-scale image dataset that consists of 123,287 images, each of which has five corresponding captions. We split our dataset into training/validation/testing splits using the method from [24]. The splitting resulted in 113,287 training images, 5000 validation images, and 5000 testing images. Before feeding the images into our model, we do some scaling and apply normalizing techniques in order to make the size of the images the same as the CNN model. In the training set, we also preprocessed all

the captions of each image by replacing all words with an occurrence of less than five with the <UNK> token and by appending <START> and <END> tokens to the vocabulary set, resulting in a vocabulary size of 8856 words.

4.2. Training and Evaluation

For training, we used stochastic gradient descent with an adaptive learning rate as our optimization algorithm, in which the learning rate starts at 4×10^{-4} and is scaled by a factor of 0.1 every 10 epochs. We also used the method from Vinyals et al. [25] on our image encoder model by fine-tuning the model after 15 epochs in order to reduce the noise coming from the initial gradient of the decoder model. In addition, to avoid overfitting, we also applied a dropout to each layer of our convolution, as well as the prediction layer, with a dropout rate of 0.1. For this model, we used the same embedding dimension, $d = 512$, for all the word embedding, the attention, and the convolutional model. Using a batch size of 64, our model was trained for 30 epochs on a 2 Nvidia Titan X GPU parallel graphics card for less than 20 h.

To evaluate the performance of our model, we used multiple conventional language evaluation metrics: BLEU [26], METEOR [27], and CIDEr [28]. Each metric was evaluated using the caption generated by our model and the ground truth caption in the validation and testing datasets.

During each epoch, we evaluated our model based on the BLEU-1 metric, and saved the model with the highest score.

4.3. Inference

To generate a caption sentence for the new image, first, we encoded the image into the feature space using our Resnet-101 CNN encoder. After getting the feature map, we inputted it along with the <START> token to our model in a simple feed-forward manner. Similar to the RNN/LSTM approach, the inference for our CNN approach was in a sequential manner, meaning that output from the first token was used as feed-forward pass input for the next token. We performed this operation sequentially until the model reached the <END> token or the maximum sentence length.

To improve the performance of our language model generation, we also implemented beam search, which was used during the inference/testing time of our model. In each sequence generation, beam search maintained a list of the top- k sequence tokens generated by the model, rather than selecting the highest probable token. After all the top- k sentences were generated, we chose the sentence with the best overall score that was calculated by the beam search algorithm. In this work, we experimented with different sizes, k , for the beam, ranging from $k = 1$ (greedy decoding) to $k = 5$. After comparing the result with the different beam sizes, we found that $k = 3$ gives the optimal performance, with the best overall result across all the evaluation metrics on the MSCOCO test dataset. Figure 4 shows the top three caption results generated by beam search.

<p><u>Top 3 result:</u></p> <p>a dog jumping in the air to catch a frisbee. a dog is jumping in the air to catch a frisbee. a dog is laying in the grass with a frisbee.</p>
<p>a baby sitting on a bed with a teddy bear. a baby laying on a bed with a teddy bear. a young boy sitting on a bed with a teddy bear.</p>

Figure 4. The top three sentences generated by the beam search algorithm.

5. Result and Discussion

To show how well our model performed, we compared the result of the MSCOCO testing dataset against some of the popular traditional image caption models with LSTM as well as other non-attention methods, including those in [16,17], using the above evaluation metrics. As shown in Table 1, our model obtained a better result than most of the LSTM models across all the evaluation metrics, while it still fell far behind the state-of-the-art result of the LSTM approaches in [8,14]. However, from our understanding, the significant performance increase compared to the methods in [8,14] resulted from a critical schedule sampling method as well as from REINFORCE algorithm [29] training, which reduced the variance of the gradient and increased the probability of sampling the correct caption. Thus, we believe that, by carefully fine-tuning the model, the results from our model would be improved and could reach those of the state-of-art LSTM approaches. Figure 5 shows the qualitative comparison between our generated caption and ground truth.

Table 1. The comparison of our model with other LSTM and non-LSTM models on the MSCOCO test dataset based on the Karpathy splitting method [24].

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr
NIC [12]	66.6	46.1	32.9	24.6	23.7	85.5
Attention (Soft) [13]	70.7	49.2	34.4	24.3	23.90	-
Attention (Hard) [13]	71.8	50.4	35.7	25.0	23.04	-
SCST [14]	-	-	-	33.3	26.3	111.4
Up-Down Attention [8]	79.8	-	-	36.3	27.7	120.1
ConvCap [16]	71.1	53.8	39.4	28.7	24.4	91.2
Caption Transformer [17]	73.0	56.9	43.6	33.3	-	108.1
Baseline	70.6	53.2	38.9	28.3	24.3	90.9
Baseline + Positional Embedding	71.3	54.1	40.3	30.1	24.8	94.3
Baseline + PE + Residual Connection	73.0	54.6	42.2	31.6	25.2	94.4
Baseline + PE + RC + Attention	73.1	55.1	42.4	32.5	25.7	96.2
Baseline + PE + RC + A + BeamSearch (size = 3)	73.6	56.6	44.1	33.2	25.9	99.4

<p><u>Caption:</u> a woman and a girl standing on a surfboard in the water.</p> <p><u>GT:</u></p> <ul style="list-style-type: none"> -two people on a surf board in water. -two people are balancing on a surfboard in the water. -a woman and a girl standing on a surfboard in the water on the shore. -some people standing on a surf board on the beach. -a couple of people on one big surfboard.	<p><u>Caption:</u> a man doing a trick on a skateboard.</p> <p><u>GT:</u></p> <ul style="list-style-type: none"> -a man doing a trick on a skateboard on a sidewalk. -a man in black shirt doing a trick on a skateboard. -a skateboarder flips his board near outdoor stairs. -male skateboarder doing aerial trick on bricked outdoor area. -a man jumping in the air on a skateboard.
<p><u>Caption:</u> a train is sitting on a bridge over a river in a city.</p> <p><u>GT:</u></p> <ul style="list-style-type: none"> -a train rides through a tunnel as another crosses on a bridge. -a train crossing over tracks over a river. -commercial trains passing overtop of each other in a city. -there is a train crossing a bridge over a river -a train coming out of a bridge next to a river.	<p><u>Caption:</u> a couple of people riding horses in the water.</p> <p><u>GT:</u></p> <ul style="list-style-type: none"> -two people mounted on horses drinking from a water trough. -a person holding a camera while on a horse that's drinking water. -the couple let their horses drink at a watering trough. -two people on horses with one horse drinking from a trough -two horses are standing and drinking from a water source.

Figure 5. The qualitative comparison between the captions generated by our model and the ground truth (GT) captions randomly picked from the MSCOCO dataset.

We also compared the result of our model to the non-LSTM approaches in [16,17]. Aneja et al. [16] implemented their image captioning model with a technique similar to our approach, which replaces the LSTM cells with a masked convolution network. While the architecture in [16] is conceptually simpler and uses fewer parameters, the results with our model are much better, outperforming the other approach across all evaluation metrics. As for Zhu et al. [17], their approach to the image captioning model is much different from our model, which makes use of the multi-headed self-attention module in [18] to generate the word sequence without the presence of LSTM. With this architecture, the best-performing model generated caption results that are comparable to our model across most of the evaluation metrics, but performed better based on CIDEr and worse based on BLEU-1. The reason for the better performance in CIDEr may result from the model being trained and optimized based on CIDEr cross entropy. Figure 6 shows the training progression of our model.

Figure 6. The progression of scores and cross entropy loss improvement along the epochs: (a) the BLEU-1 evaluation score, starting from 59.5 in Epoch 1 and peaking at 73.6 in Epoch 26 before it stop improving; (b) the CIDEr evaluation score, starting from 50.6 in Epoch 1 and peaking at 99.6 in Epoch 24 before no longer improving; and (c) the cross entropy loss of our model across the epoch.

Another thing to notice is the performance increase in speed during the training process. Since our model uses a convolutional approach, we can feed our words as input to the model all at once, unlike the LSTM approach, which works in a sequential process, one word at a time. By feeding all the words at once, our model also benefits from parallel GPU computing, allowing us to use a large batch size for training and to reduce the training time per epoch by 35%. Table 2 shows a performance comparison between our model and other image captioning models. Compared to other models, our model may have more parameters than the method in [12], but the training for each epoch can be completed in a comparable time while achieving a significantly better result across all metrics. This performance improvement becomes even more transparent when comparing with the method in [13], since our model can be trained twice as fast with fewer parameters and better result. The method in [8] may have a better result, but the training speed is slower than our proposed method. This also does not consider that the method in [8] uses Faster R-CNN to pre-process all their bottom-up attention feature before their input to captioning model. Using this method can significantly speed up the training since it does not need to extract the image feature during the training, as the model longer performs in an end-to-end manner. If we consider the duration spent during processing the bottom-up attention feature, the training of the model would take much longer than stated in the table. In our study, we ran up to four GPUs in parallel; however, we found that the speed-to-accuracy result was the best with two GPUs. Using more than that reduced the accuracy of our predicted sentences. Table 3 shows the performance of our image captioning model on the multi-GPU training of different GPU setups. We hypothesize that the performance obtained may be because of the information loss from data communications across GPUs during feed-forward and back-propagation in the training process, which led to this performance loss [30,31].

Table 2. Comparison of the number of parameters and the training speeds per epoch of different image captioning models. The work in [12,13] presents our implementation based on the paper using PyTorch, whereas the work in [8] is the open-source implementation using Caffe.

Model	#param	Single-GPU	Multi-GPU
NIC [12]	14.6 M	25 ± 5 min	-
Attention [13]	20.2 M	50 ± 3 min	-
Up-Down [8]	21.5 M	46 ± 4 min	37 ± 4 min
Ours	19 M	40 ± 5 min	28 ± 6 min

Table 3. Results comparison of multi-GPU training based on BLEU-1, METEOR, and CIDEr metrics.

#param	BLEU-1	METEOR	CIDEr
GPU × 1	73.3	26.2	100.8
GPU × 2	73.6	25.9	99.4
GPU × 3	71.7	23.9	95.5
GPU × 4	72.1	24.0	97.4

6. Conclusions

In this study, we successfully implemented an image captioning model using a convolutional neural network that enables parallel GPU processing for this task. Our model can achieve increasing performance in training speed, while obtaining results comparable to the traditional approaches.

From this study, we believe there is still room for improvement of our model. One of the first things we want to do is to fine-tune the hyper-parameters to find the optimal performance. Another thing that can improve this model is to use the reinforcement learning approach of Rennie et al. [14] during training, which we believe will improve the results. We also want to explore on the differences of the reinforcement technique between the LSTM and convolutional approaches and whether there would be any changes or improvement to speed and result of the model. In addition, we also want to try captioning longer-sequence tasks. All of this will be kept in mind for future work.

Author Contributions: Review previous research, C.P.; conceptualization, C.P.; funding acquisition, J.K.; investigation, C.P.; methodology, C.P.; software, C.P.; supervision, J.K.; validation, C.P.; writing—original draft preparation, C.P.; and writing—review and editing, J.K.

Funding: This research received no external funding.

Acknowledgments: This work was supported by an INHA UNIVERSITY Research Grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hossain, M.; Sohel, F.; Shiratuddin, M.F.; Laga, H. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv. (CSUR)* **2019**, *51*, 118. [\[CrossRef\]](#)
2. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1310–1318.
5. Lu, Y.; Salem, F.M. Simplified gating in long short-term memory (lstm) recurrent neural networks. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1601–1604.
6. Elbayad, M.; Besacier, L.; Verbeek, J. Pervasive Attention: {2D} Convolutional Neural Networks for Sequence-to-Sequence Prediction. In Proceedings of the 22nd Conference on Computational Natural Language Learning, Brussels, Belgium, 31 October–1 November 2018; pp. 97–107.
7. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning—Volume 70, Sydney, Australia, 6–11 August 2017; pp. 1243–1252.
8. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-up and top-down attention for image captioning and visual question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6077–6086.
9. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.

10. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)] [[PubMed](#)]
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3156–3164.
13. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2048–2057.
14. Rennie, S.J.; Marcheret, E.; Mroueh, Y.; Ross, J.; Goel, V. Self-critical sequence training for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7008–7024.
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
16. Aneja, J.; Deshpande, A.; Schwing, A.G. Convolutional image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5561–5570.
17. Zhu, X.; Li, L.; Liu, J.; Peng, H.; Niu, X. Captioning transformer with stacked attention modules. *Appl. Sci.* **2018**, *8*, 739. [[CrossRef](#)]
18. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
19. van den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv* **2016**, arXiv:1601.06759.
20. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language modeling with gated convolutional networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, Sydney, Australia, 6–11 August 2017; pp. 933–941.
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
22. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
23. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 740–755.
24. Karpathy, A.; Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3128–3137.
25. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 652–663. [[CrossRef](#)] [[PubMed](#)]
26. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
27. Banerjee, S.; Lavie, A. 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, Ann Arbor, MI, USA, 29 June 2005; pp. 65–72.
28. Vedantam, R.; Lawrence Zitnick, C.; Parikh, D. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4566–4575.

29. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [[CrossRef](#)]
30. Shi, S.; Wang, Q.; Xu, P.; Chu, X. Benchmarking state-of-the-art deep learning software tools. In Proceedings of the 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, 16–18 November 2016; pp. 99–104.
31. Kim, H.; Nam, H.; Jung, W.; Lee, J. Performance analysis of CNN frameworks for GPUs. In Proceedings of the 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Santa Rosa, CA, USA, 24–25 April 2017; pp. 55–64.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).