Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/48/20241175

Image Caption using VGG model and LSTM

Yuepeng Li

School of International Education, Nanjing Institute of Technology, Jiangning, 21116, China

X0020220123@njit.edu.cn

Abstract. Deep convolutional networks and recurrent neural networks have gained significant popularity in the field of image captioning tasks in recent times. As we all know the performance and the architecture of models are still eternal topic. We constructed the model using a new method to enhance its performance and accuracy. In our model, we make use of pretrained CNN model VGG (Visual Geometry Group) to extract image features, and learn caption sentence features using bidirectional LSTM(Long-Short-Term-Memory) which can better understand the meaning of sentences in the text. Then we combine the image features and caption features to predict captions for images. The dataset Flickr8K is used to train and test the model. Additionally, the model has the ability to produce captions that are shorter than a specified caption length. We evaluated our model with Bilingual Evaluation Understudy (BLEU) score which measures the similarity of predicted text and to the real text. After evaluation and comparison, our model is proved to be well-done on some conditions.

Keywords: VGG, LSTM, BLEU, Caption Generation.

1. Introduction

Image Caption Generation is a research field that combines computer vision and natural language processing. Its goal is to utilize deep learning models to automatically produce precise and coherent plain language descriptions of visual information. It is very meaningful that it can provide assistance for visually impaired people, enabling them to understand the content of images through textual descriptions [1].

Computer vision, machine translation, and object detection are dynamic domains in the realm of machine learning, witnessing significant growth in the past decade [2]. This surge has led to the emergence of diverse frameworks facilitating the implementation of caption generation.

Traditional methods mainly rely on hand-designed feature extractors and rule-based language models, but the effect is limited [1]. Additionally, techniques based on deep learning, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (Recurrent Neural Networks, RNN), particularly Long Short-Term Memory (LSTM), have had considerable success in the production of visual descriptions. Through end-to-end training, these methods are able to extract semantic information from images and generate natural language descriptions related to image content.

The three main steps in creating the image captioning model were extracting image and caption features to support the model, utilizing these features to train the model, and using the trained model to produce captions based on input picture attributes. These facets were attained by combining two different approaches: feature extraction was handled by a pretrained CNN model called the Visual Geometry Group Neural Network (VGG), and caption text generation was handled by a Recurrent Neural Network (RNN).

As a result, we used the Bilingual Evaluation Understudy (BLEU) score as a benchmark for comparison in order to assess the model's effectiveness. The results showcased a notable enhancement in BLEU score accuracy compared to the baseline approach. Furthermore, our model possesses the unique capability of genuinely crafting image captions, setting it apart from the baseline method that merely borrowed existing captions.

2. Main Method

In our approach, we train our model using the Flickr8K picture dataset [3]. For prediction, a Long Short-Term Memory (LSTM) neural network architecture is employed, along with a combination of Flickr8K dataset captions and image attributes extracted using VGG. The core aspects of our image prediction process involve VGG feature extraction and utilizing the trained LSTM model for caption generation. As you can see from the Figure1 to figure out the our model frame.

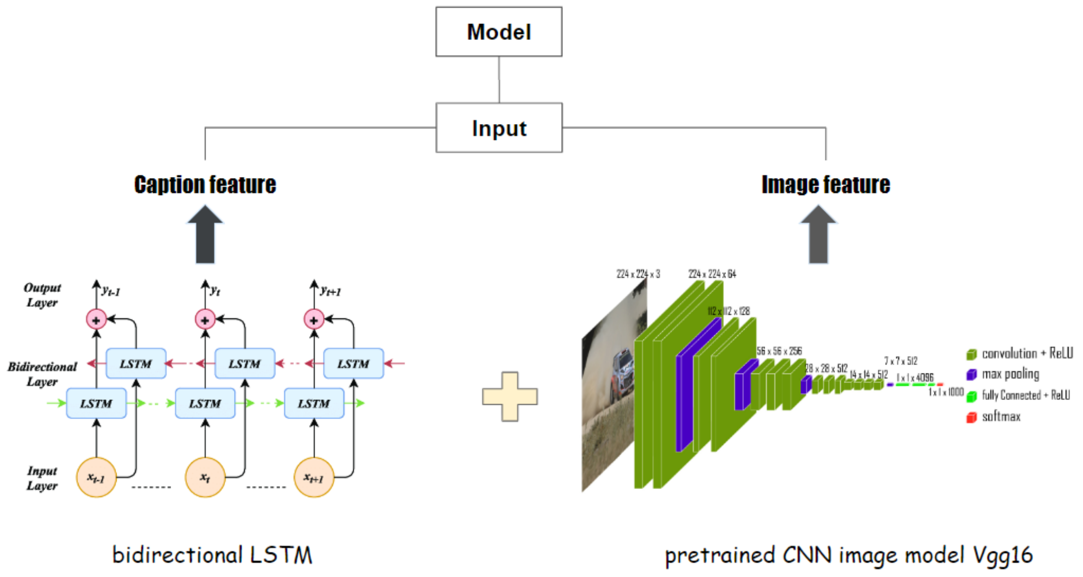


Figure 1. Brief overview of the model.

2.1. Caption Features Extraction with LSTM

In our model, we firstly preprocess the captions. We extract the captions for images and clean the data, then we will build a mapping dictionary from image_id to image_captions, which will be very useful in generating training dataset and generating captions for images. Then we train Keras Tokenizer on the captions and obtain a tokenizer for our task [4]. In the part of caption feature extraction, we had set embedding layer, dropout layer, three LSTM layers, two Resnet layers, two add layers and a concatenate layer the architecture for this part can be seen as Figure 2.

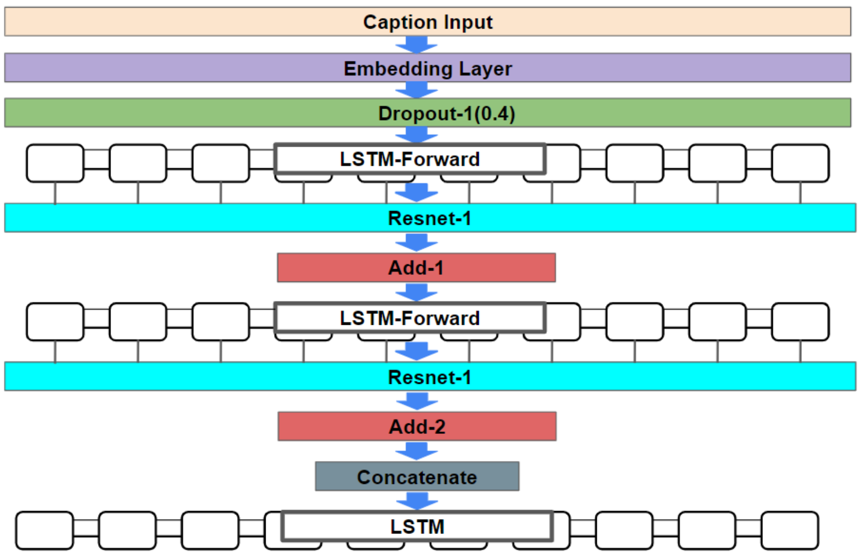


Figure 2. Architecture of caption feature.

First, caption input will be processed by a word embedding layer. The caption embedding is then placed in a bidirectional LSTM layer [5].

We create the embeddings to pass through the LSTM layer both forward and backward in the bidirectional LSTM layer and construct hidden states with a dimension that is half that of the embeddings.

LSTM is an improved version of RNN that addresses issues like gradient disappearance, gradient explosion, and difficulty in capturing long-range dependencies. The hidden layer h incorporates three gate structures, namely the forget gate, input gate, and output gate. These gates enable better information flow control, selective memory management, and effective capture of long-term dependencies. In the function below, f_t, i_t, and o_t are the above three gates at time t in turn., respectively, and a_t reflects the value of h_t and preliminary feature extraction of x_t[6]:

$$\begin{aligned} f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\ i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\ a_t &= tanh\,(W_a h_{t-1} + U_a x_t + b_a) \\ o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \end{aligned} \tag{1}$$

The hidden layer's state value at time t-1 is represented by h_{t-1}, whereas the input at time t is represented by x_t;

In the forgetting gate, input gate, output gate, and feature extraction processes, the weight coefficients associated with the previous hidden state, ht-1, are represented by *Wf*, *Wi*, *Wo*, and *Wa*, respectively. These weight coefficients control the flow of information and determine the impact of the previous hidden state on each process.

Similarly, in the same processes, the weight coefficients related to the current input, *xt*, are denoted by *Uf*, *Ui*, *Uo*, and *Ua*. These weight coefficients govern how the current input influences the respective processes, such as controlling the input information, regulating the output, and extracting relevant features.

The bias values are represented by *bf*, *bi*, *bo*, and *ba*, respectively, in the three-gate and feature extraction processes. The sigmoid activation function is denoted byσ, and the tangent hyperbolic function is denoted by tanh. Here are the two functions:

$$tanh\ x = \frac{sinh\ x}{cosh\ x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{2}$$

$$\sigma_x = \frac{1}{1+e^{-x}} \tag{3}$$

The results of the forget gate and input gate calculations interact with the previous cell state, c_{t-1}, to form the cell state at time t, c_t. This relationship can be represented by the following formula:

$$c_t = c_{t-1} \odot f_t + i_t \odot a_t \tag{4}$$

The hidden layer state, h(t), at time t is obtained by the Hadamard product (element-wise multiplication) of the output gate, o(t), and the current cell state, c(t), and can be expressed as:

$$h_t = o_t \odot tanh\,(c_t) \tag{5}$$

To avoid vanishing gradient, we add ResNet skip way to the output of the bidirectional layer and the formula is [7]:

$$y_{residual} = f(x) + g(f(x)) \tag{6}$$

We achieve this by making the embeddings to pass through a Dense layer without bias to get the protected weight and add the protected weight to the output of LSTM layer. We do this for both the forward and backward LSTM process. After get the residual weight for bidirectional LSTM layer, then we concatenate then together to get the real output of the first bidirectional LSTM layer [8]:

$$Z_{concat} = \sum_{i=1}^{c} X_i * K_i + \sum_{i=1}^{c} Y_i * K_{i+c} \tag{7}$$

Now, the real output is of the same size of the embedding dimension after concatenation. Since the real output is obtained from bidirectional LSTM layer, we believe it now captures contextual information from both past and future sequences, thus can better understand the sentence. Afterwards, we pass the real output of the first bidirectional LSTM layer into another LSTM layer with 256 nodes to learn the sentence meaning and context in a more advanced level. The feature of the caption is then taken from the output of the second LSTM layer, which is of shape (1, 256).

2.2. *Image Features Extraction with VGG*

Our second step is to extract feature from images. Because of that we use a pre-trained CNN image model Vgg16 for learning the images contents and take the output of the layer before the last softmax layer as the input of images, which is of shape 1x1x4096.We make it go through a Dense layer with 256 nodes to extract image features, and at the same time, adjust the dimension of image feature to be of shape (1, 256). The architecture of this part can be seen as Figure 3.

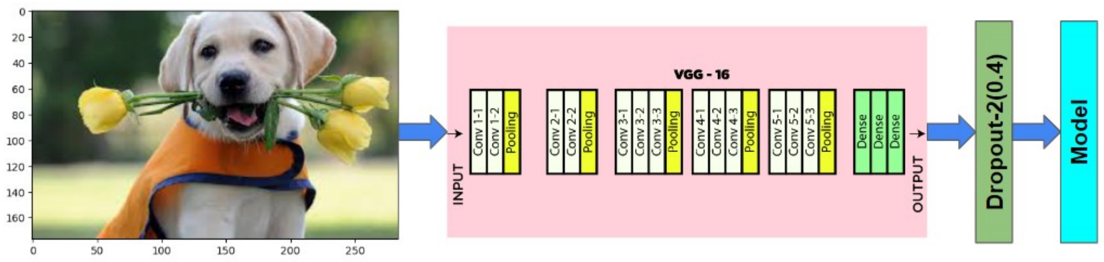


Figure 3. Architecture of image feature.

Notably, there exist two exist versions of the VGG network: a 16-layer variant and a 19-layer variant. For our model, we focused primarily on the 16-layer version, known as VGG16.VGG16 has 21 layers in total—13 convolutional layers, 5 Max Pooling layers, 3 Dense layers—but only 16 of them are weight layers, or parameter-learning layers.[9]. Through our work, we experimented with both the VGG16 and VGG19 versions .Actually, there is no big difference between the two models. Ultimately, we selected VGG16, which best aligned with our final model.

Actually, VGG model was initially designed to classify image [9]. Because of that the output of the last layer is the classification of objects within an image, which is not suitable to out context. To apply this pre-trained model to help us extract the image feature. The second fully connected layer, which contains the image's feature data, was used instead of the last output layer, which we removed (Figure 4).

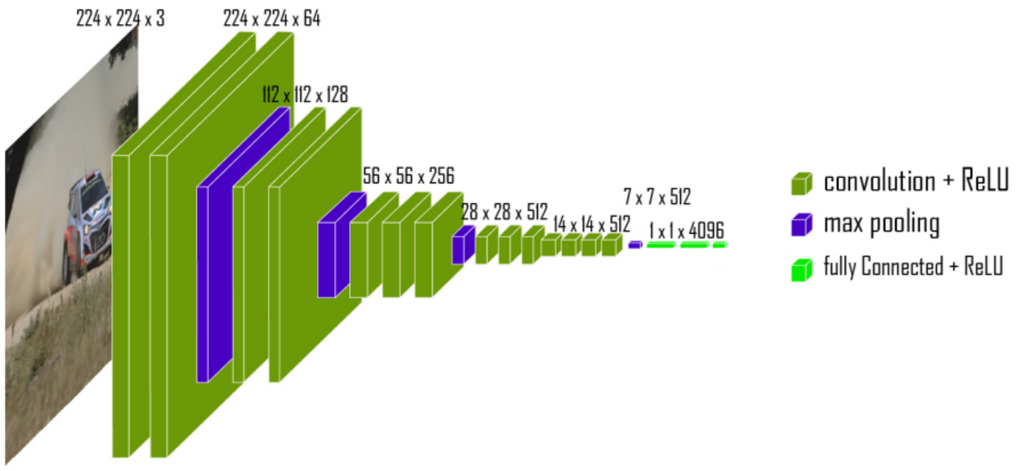


Figure 4. VGG without last output layer.

2.3. *Combine the two parts*

Now that we have get the image features and caption features of the same size, we add them together into a combined feature of shape (1, 256) [10]:

$$Z_{add} = \sum_{i=1}^{c} (X_i + Y_i) * K_i \tag{8}$$

Then we make the combined feature to go through a Dense layer with activation "ReLu" to learn and summarize the relationship between the two features [11]:

$$f_{Relu}(x) = max(0, x) \tag{9}$$

Then, the output of the Dense layer will be passed into another Dense layer with activation "Softmax", and vocab size nodes to get result which is the probability distribution of the prediction for the next word [11]:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{C}^{c=1} e^{z_i}} \tag{10}$$

The output value of each node is referred to as 'i' in this formula, and the number of output nodes is referred to as 'C'.

The whole architecture can be seen as Figure 5.

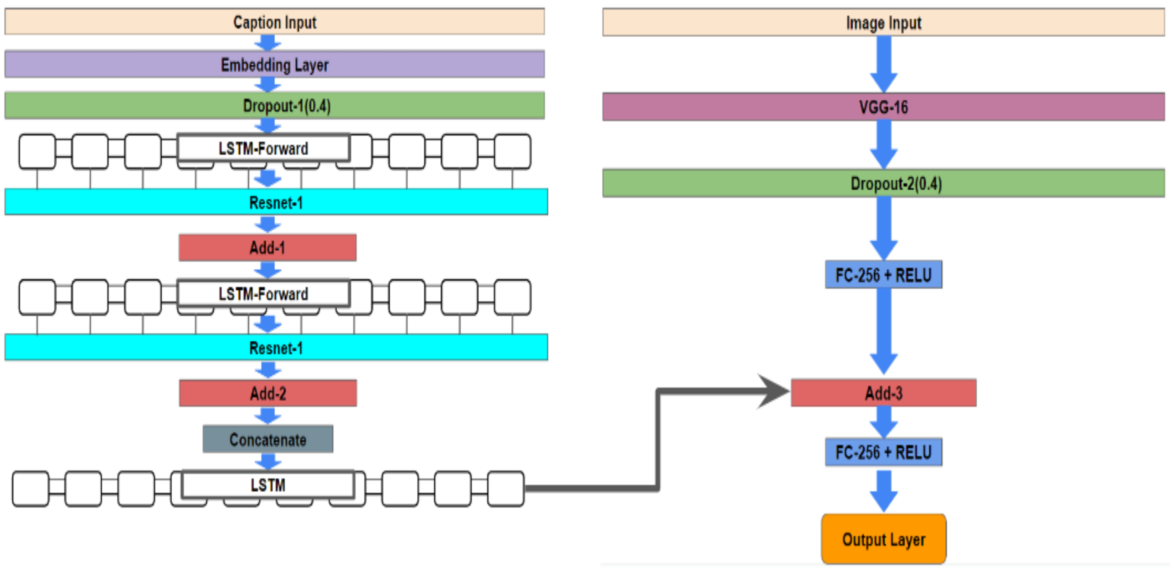


Figure 5. Architecture of the whole model.

To prevent the model from overfitting, we add a Dropout layer beneath the embedding layer and the Image Input layer in addition to the fundamental structure of the model that we previously described. During the training process, the working mechanism of dropout is: a Bernoulli distribution with a probability of p randomly generates 0, 1 values with the same number of nodes, and some nodes are blocked after multiplying these values with the input [12]. Use these node values for subsequent calculations. The specific formula is as follows [12]:

$$\begin{gathered} r_j^l \sim Bernoulli(p) \\ \tilde{y}^l = r^l * y^l \\ z_i^{l+1} = w_i^{l+1}\tilde{y}^l + b_i^{l+1} \\ y_i^{l+1} = f(z_i^{l+1}) \end{gathered} \tag{11}$$

In our model we set the probability p to 0.4. As we all know 0.5 may be better theoretically, but 0.4 performs better than 0.5 in our model.

For the loss function and optimizer, we chose the “categorical_crossentropy” and “adam”, most people will choose these two to train their model. For multi-class classification problems where the output has multiple categories, Categorical Cross-entropy is a loss function that is frequently employed [11]:

$$Loss = -\sum_{i=1}^{outputsize} y_i \cdot log\ \hat{y}_i \tag{12}$$

Adam is an algorithm for adaptive optimization that brings together the advantages of both Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [13].

The result is a vector whose entries represent the possibility of every word in the dictionary. Our current ‘best word’ would be the word with the highest probability.

We have trained our model for 40 epochs and set the batch size 32. After being trained, the loss curve is showed as figure 6:

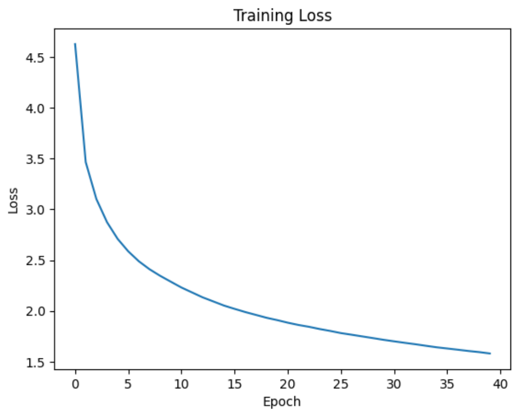


Figure 6. Loss curve.

Our final training rate is 1.4864, which proves we have built a good model.

2.4. *Generate captions for the image*

when generating the captions, we first pass the image we want to predict through the VGG16 model to extract image feature. When it comes to generating captions, we continue to utilize the LSTM. The model's initial input word is"\<start\>", followed by additional inputs derived from previous predictions. we will generate words one by one and predict the next words with all the predicted words before, which is just the same as when we do seq2seq task. When we meet "\<end\>" or the max_len is reached, we will finish the prediction.

2.5. *Results*

In the evaluation section, we use BLEU (Bilingual Evaluation Understudy) which is a score for a set of candidate predicted sentences compared to reference groundtruth sentences [14]. It assesses the similarity between the candidate translations and reference translations in terms of n-grams and is frequently used to assess the quality of machine-generated phrases [14]. Between 0 and 1 is the BLEU Score range. Higher values imply references that are more closely matched. The BLEU Score formula is:

$$BLEU = BP * exp(\frac{1}{n}\sum_{i=1}^{N} P_n) \tag{13}$$

Among them,BP stands for short penalty factor. It penalizes sentences that are too short to prevent short training outcomes, and its expression is [14]:

$$BP = \{ \begin{array}{c} 1, if MToutputlength > referenceoutputlength \\ exp(1 - MT\frac{outputlength}{referenceoutputlengh}), otherwise \end{array} \tag{14}$$

Pn is the accuracy based on n-gram, and its expression formula is:

$$P_n = \frac{\sum n{-}gram \in yCounterClip(n{-}gram)}{\sum n{-}gram \in yCounter(n{-}gram)} \tag{15}$$

For our caption generating purpose, a score above 0.6 is considered as a good result. Our average BLEU score on test data is actually over 0.6 from the (Figure 7).

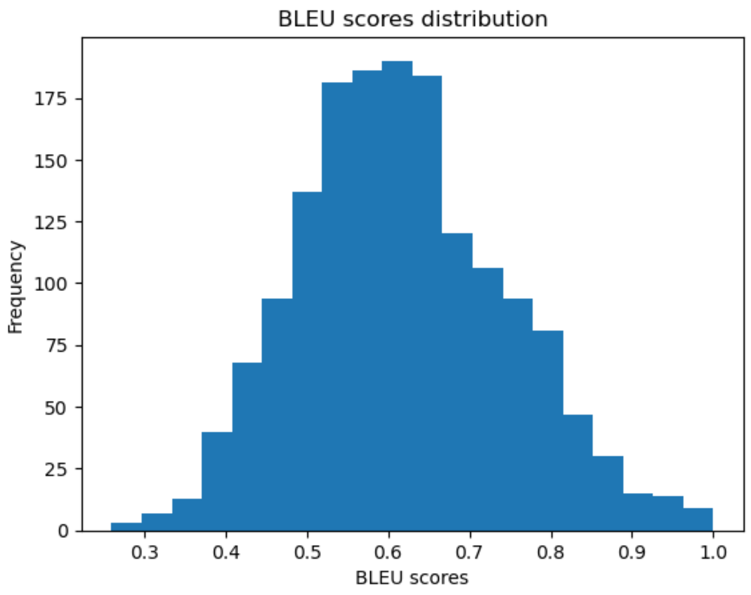


Figure 7. BLUE scores distribution.

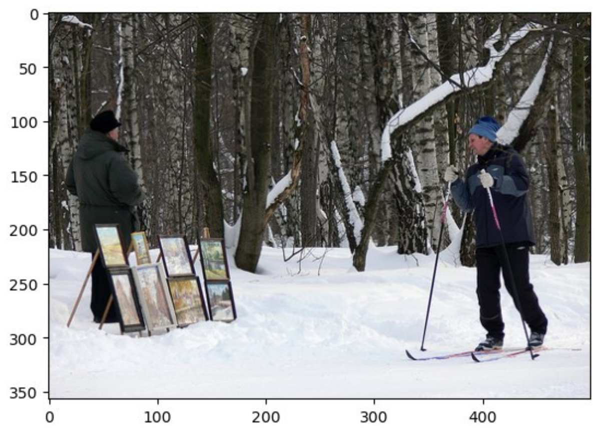
For the same dataset (Flickr8K image dataset), we compared some other models, which use different methods to build their models. We chose the models both use LSTM but with different dropout rate. The comparison is showed as below:

Table1. Compare with other models in the same dataset.

Model	BLEU-1	BLEU-2
LSTM with Dropout (0.4)	0.532588	0.311006
LSTM with Dropout (0.3)	0.421167	0.193618
Our Model (Bidirectional LSTM with Dropout (0.4))	0.561534	0.366148

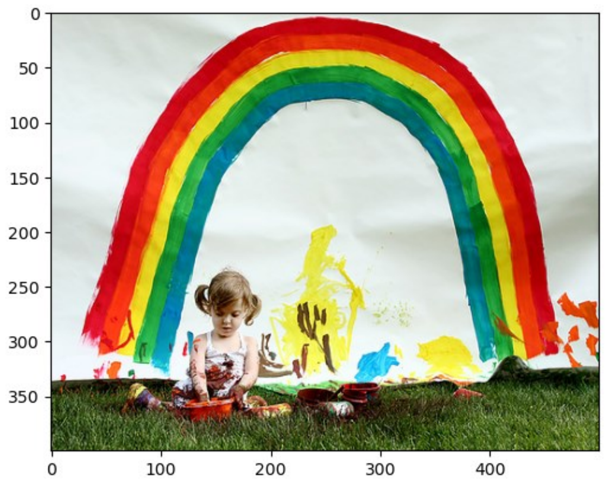
In a reference translation, BLEU-1 counts the number of times each word appears in a machine translation and normalizes it to the total number of words in the machine translation. These normalized counts are then averaged to get the BLEU-1 score. BLEU-1 only considers single word matching and does not consider phrase or sentence level matching. Therefore, it is the simplest form of the BLEU indicator and one of the most basic evaluation indicators. Similarly, BLEU-2 counts the number of occurrences of two consecutive phrases in machine translation output and reference translations. By considering two consecutive phrases, BLEU-2 is able to better capture phrase-level matching, further improving the evaluation of machine translation quality. We can find that no matter the BLEU-1 or BLEU-2, our model has a higher value, which proves our model can efficiently predict proper words regardless of the context.

When processing the same image simultaneously, we compared the output from various models, and the results are displayed below:



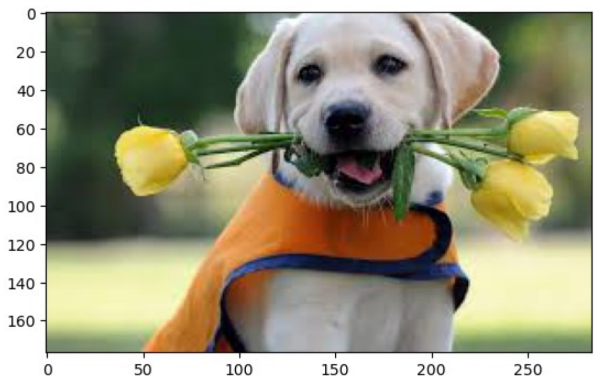
LSTM with Dropout(0.3):*startseq man on artwork displaying pictures next to skier in the snow endseq*
LSTM with Dropout(0.4):*startseq skier skis down skis endseq*
Our model(Bidirectional LSTM with Dropout(0.4)):*<start> a man in a red outfit is displaying pictures in the snow <end>*

Figure 8. Results comparison on the same picture from test data_1.



LSTM with Dropout(0.3):*startseq small girl covered her hands in fingerpaints in front of rainbow hands endseq*
LSTM with Dropout(0.4):*startseq two children in red and white play in the grass endseq*
Our model(Bidirectional LSTM with Dropout(0.4)):*<start> a girl in a red and yellow outfit is sitting in a large tent with a rainbow bear wrapped on her head <end>*

Figure 9. Results comparison on the same picture from test data_2.



<start> a dog with a toy in his mouth standing in a field <end>

Figure10. Testing of online pictures.

From the comparison (Figure 8-10), we can see that our model is overall very reasonable despite some minor errors. We also tried image from the website, our model still performs well.

3. Conclusion

In this paper, we proposed an image captioning model that combines the VGG model and LSTM to generate accurate and coherent natural language descriptions of image content. Our model utilized the VGG model to extract image features and employed bidirectional LSTM for caption sentence feature learning. By combining these features, we were able to generate captions for images.

Through the evaluation and comparison of our model using the BLEU score, we demonstrated its superior performance and accuracy compared to traditional methods and a baseline approach. Our model showcased significant improvements in generating image captions and outperformed previous models in certain conditions.

Furthermore, while our predictive model for learning caption features serves its purpose, it falls short of achieving state-of-the-art performance. In lieu of this, more advanced strategies could be explored, such as incorporating pretrained BERT for caption embeddings and adopting the Transformer architecture to enhance caption feature learning and prediction. Similarly, replacing the current image feature learning model with a more advanced CNN could yield a better grasp of image content.

References

[1] H. Wang, Y. Zhang, X. Yu, 2020 “An overview of image caption generation methods,” Computational intelligence and neuroscience, 1-11.

[2] A. Grigorev, R. Shanmugamani, A. Boschetti, et al.,2018 TensorFlow Deep Learning Projects: 10 real-world projects on computer vision, machine translation, chatbots, and reinforcement learning, Packt Publishing Ltd

[3] K. Anitha Kumari, et al.,2020 “Automated image captioning for Flickr8K Dataset,” Artificial Intelligence, Smart Grid and Smart City Applications,679-687.

[4] K. Ramasubramanian, et al.,2019 “Deep learning using keras and tensorflow,” MLUR pp 667-688

[5] A. Graves, J. Schmidhuber,2005 “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” IEEE International Joint Conference on, vol 18 nos 5-6 pp 602-610

[6] W. Fang, J. Jiang, S. Lu, et al.,2020 “A LSTM algorithm estimating pseudo measurements for aiding INS during GNSS signal outages,” Remote Sensing, 2020, 12(2):256

[7] M. Dhakal, A. Chhetri, A. K. Gupta, et al., 2022 “Automatic speech recognition for the Nepali language using CNN, bidirectional LSTM and ResNet,” International Conference on Information and Computer Technologies, pp 515-521

[8] M. S. Ko, K. Lee, J. K. Kim, et al.,2020 “Deep concatenated residual network with bidirectional LSTM for one-hour-ahead wind power forecasting,” Institute of Electrical and Electronics Engineers vol 12 no 2, 1321-1335

[9] H. Qassim, A. Verma, D. Feinzimer, 2018”Compressed residual-VGG16 CNN model for big data places image recognition,” CCWC IEEE pp 169-175

[10] X. Gao, G. Zhang, Y. Xiong,2022 “Multi-scale multi-modal fusion for object detection in autonomous driving based on selective kernel,” Measurement vol 194 p 111001

[11] H. Benradi, A. Chater, A. Lasfar,2023 “A hybrid approach for face recognition using a convolutional neural network combined with feature extraction techniques,” IAES IJAI, vol 12 no 2 p 627

[12] S. Boluki, R. Ardywibowo, S. Z. Dadaneh, et al.,2020 “Learnable Bernoulli dropout for Bayesian deep learning,” ICAI and Statistics PMLR, pp 3905-3916

[13] I. K. M. Jais, A. R. Ismail, S. Q. Nisa,2019 “Adam optimization algorithm for wide and deep neural network,”KEDS vol 2 no 1 pp 41-46

[14] O. Vinyals, A. Toshev, S. Bengio, et al.,2015 “Show and tell: A neural image caption generator,” arXiv:1411.4555 [cs.CV] 3156-3164