Image Caption

Huan KUANG, University of Florida, huan2015@ufl.edu

"dog is running through the grass"

"two girls play soccer on the beach"

"the man is walking down the street"

***Abstract*—In this project, I presented an application of a Long short-term memory (LSTM) model for describing the content of a given image. There are two neural network models in order to generate the captions of a given image. Specifically, a Convolutional Neural Networks (CNN) model for extracting the features from input images and an LSTM for generating natural language captions. The CNN extracts the features from input images and produces these features to a fixed-length vector by embedding them. It was based on a pre-trained model *VGG16* [1]. The LSTM generates the descriptions and was implemented with *Keras*. Evaluations on the test showed a relatively adequate performance result. I got an average BLEU-1 value of 0.52 on the test set.**

I. INTRODUCTION

Image caption refers to that computers automatically identify the image content and output English sentences that were structured like human language. It had made several crucial contributions to society. For example, image caption enables visually impaired people to get information from images on the web. It also helped in the fields such as early childhood education. Image caption has been used to help children to read and describe pictures [2]. Interacting with the image caption product, children could learn to identify the objects in the image and build their language skills. This type of interaction could also potentially cultivate children's logical thinking.

However, generating high-quality captions was not an easy problem. First of all, the image caption models need to be powerful enough to determine the key objects that are in an image and the relationships among them. Secondly, these models must also be capable of expressing a large amount of important visual information in a natural language. Recently, researchers could overcome these two problems due to the development of deep learning algorithms in computer vision and natural language programming. Those deep learning neural networks are able to imitate extraordinary human abilities in processing the information between images and sentences. The goal of this project was to predict natural language descriptions of input images using deep learning neural networks.

II. BACKGROUND

There are many models are available for solving image caption problems. In general, four different approaches, namely separated models [3, 4], joint models [5], reinforcement learning based models [6], and Generative Adversarial Nets (GAN) based models [7] had been used to solve image captioning tasks.

Vinyals et al. [3] proposed a neural and probabilistic framework, named "*Neural Image Caption*" (NIC), to generate descriptions from images. Within the NIC, there is one CNN-based image classification and one LSTM-based sentence generator. The two models in NIC were trained separately, and the NIC was based on statistical machine translation, which directly maximizes the probability of the correct translation given an input sentence as follows:

$$\theta^{\star} = \arg\max_{\theta} \sum_{(I,S)} \log p(S|I;\theta)$$

where θ are the parameters of our model, I is an image, and S is its correct transcription.

Based on sequence-to-sequence training with neural machine translation, Donahue et al. [5] jointly trained two models to learn temporal dynamics and convolutional perceptual representations. They used a long-term recurrent convolutional network (LRCN) model to extract the visual features from each input image. The input for this model is pixels and the output is image features. Then they directly connected the LRCN with an LSTM model to generator description. The image features and the previous word are the inputs at each time step in LSTM to learn the dynamics of the time-varying output sequence. Chen et al. [7] proposed a conditional-generative adversarial-nets-based image captioning framework. They used the "discriminator" networks to automatically and progressively determine whether the generated caption is human described or machine-generated. Moreover, they treated the well-trained discriminators as objective image captioning evaluators.

III. MODEL

Due to computational power, this project adopted separated models, namely, train a neural network for capturing the image features and a neural network for generating captions separately. I also tried GANs, but they were hard to converge. The overall procedure was shown in Fig.1.

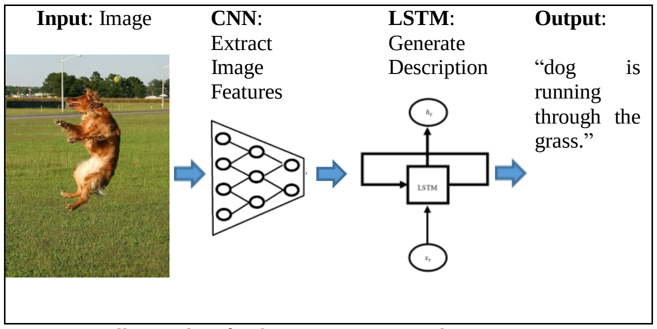

Input: Image
CNN: Extract Image Features
LSTM: Generate Description
Output: "dog is running through the grass."
h_t
LSTM
x_t

Fig. 1. Overall procedure for this image caption task

A. *Extract Image Features*

CNN becomes a popular method for image classification. A pre-trained CNN model, *VGG16*[1] from *Keras*, was used to extract image features. The input of *VGG16* are images and the output are image features in a fixed-length vector. I decided to adopt *VGG16* for two reasons. Firstly, *VGG16* was one of the famous models submitted to the Large-Scale Visual Recognition Challenge 2014 (ILSVRC-2014). It achieved 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1,,000 classes. If I train a CNN by myself, the performance of the trained model may not be comparable to *VGG16*. Secondly, it will be time-consuming if I train a CNN which has similar architecture as *VGG16*, since *VGG16* was trained for weeks and was using NVIDIA Titan Black GPUs.

Recently, the most popular pre-trained CNN architectures are *GoogLeNe*t, *AlexNets*, and *ResNet50*. VGG16 improves the *AlexNets* by replacing large kernel-sized filters (11 in the 1st convolutional layer and 5 in the second convolutional layer) with multiple 3×3 kernel-sized filters one after another. 3×3 was proposed because it is the smallest size to capture the notion of left/right, up/down, and center of a local region.

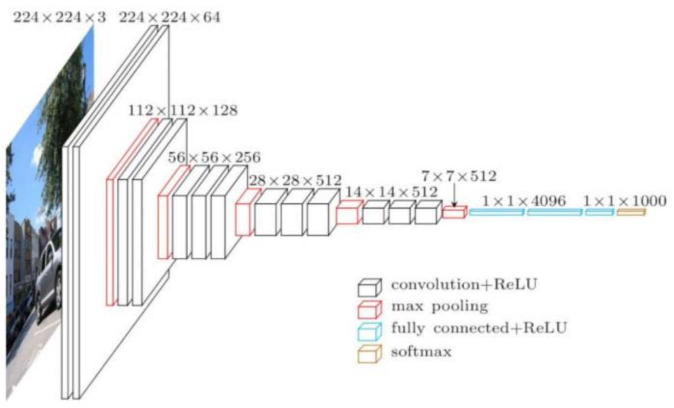


Fig. 2. VGG16 Architecture (Source from: https://neurohive.io/en/popular-networks/vgg16/)

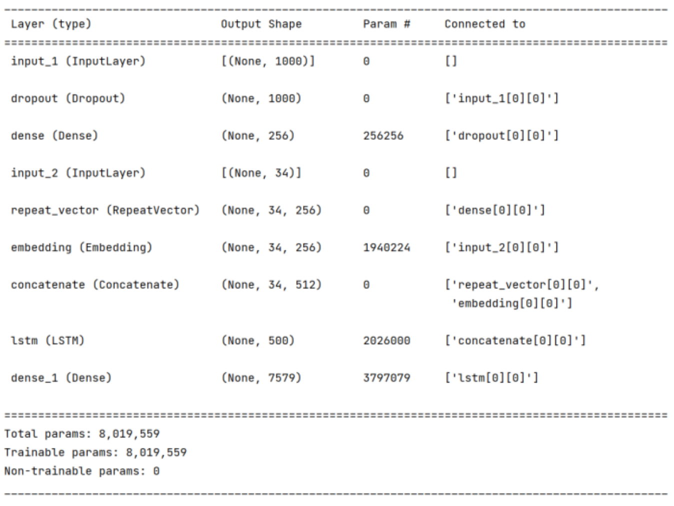
The architecture of VGG16 was presented in Fig.2. There are 16 layers in the VGG16 model. There are 13 convolutional layers (including the max-pooling) and 3 fully connected layers. Not all the convolutional layers are followed by max-pooling. Five max-pooling layers follow the 2nd, 4th, 7th, 10th, and 13th convolutional layers. The activation function for each hidden layer was rectification (ReLU). The input to the 1st convolutional layers was an RGB image with a fixed size at 224 x 224. Then, the input image passed through a stack of convolutional layers, where the filter was 3×3. The convolution stride is fixed to 1 pixel. The padding is 1 pixel for 3×3 convolutional layers to preserve the spatial resolution. Max-pooling is performed over a 2×2-pixel window, with stride 2. Three fully connected layers followed the 13 convolutional layers. The first two fully connected layers have 4,096 channels each, and the third fully connected layer contains 1,000 channels corresponding to the 1,000 classes in the ImageNet. The final layer is the soft-max layer.

B. *Generate Image Caption*

In this section, I will explain the specific implementation details for the model that I used to generate descriptions. The caption generator was a Long short-term memory (LSTM) model. I decided to use LSTM because it can directly map variable-length inputs to variable-length outputs (i.e., natural language text) and could model complex temporal dynamics.

The LSTM architecture (see Table.1) of this project contained three parts and nine layers in total. The first part processed the image feature as input, and it had four layers. Namely, one input layer, one dropout layer (with dropout rate = 0.3), one fully connected layer with embedding (ReLU as activation function), and one repeat vector layer. I added a dropout layer after the input layer, which is the output from the CNN model for extracting image features, to prevent potential over-fitting. Dropout was a regularization technique that imitated training several models that had "different architectures or be trained with different data [8]". Dropping certain neurons could help with reducing redundancy in the weight matrix in the training process since neurons were mutually dependent on each other in an FC layer [9]. I set the probability of dropout as 0.3. It means that some of the input neurons will be randomly zeroed during training with a probability of 0.3 using samples from a Bernoulli distribution in *Keras*.

Table 1. LSTM Architecture



```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 1000)]       0           []

 dropout (Dropout)              (None, 1000)         0           ['input_1[0][0]']

 dense (Dense)                  (None, 256)          256256      ['dropout[0][0]']

 input_2 (InputLayer)           [(None, 34)]         0           []

 repeat_vector (RepeatVector)   (None, 34, 256)      0           ['dense[0][0]']

 embedding (Embedding)          (None, 34, 256)      1940224     ['input_2[0][0]']

 concatenate (Concatenate)      (None, 34, 512)      0           ['repeat_vector[0][0]',
                                                                  'embedding[0][0]']

 lstm (LSTM)                    (None, 500)          2026000     ['concatenate[0][0]']

 dense_1 (Dense)                (None, 7579)         3797079     ['lstm[0][0]']

==================================================================================================
Total params: 8,019,559
Trainable params: 8,019,559
Non-trainable params: 0
__________________________________________________________________________________________________
```

The second part processed the description as input, and it had two layers. Namely, one input layer and one embedding layer. The purpose of having the embedding layer was to create an equal length vector to represent the words of the descriptions.

The third part was the main part of this model which corresponds to mapping the image features with the words in the descriptions. It had three layers, one concatenate layer to combine the two sources of inputs; one LSTM layer with 500 units; and one fully connected layer with SoftMax as an activation function. SoftMax generated a 7,579-dimensional vector, which corresponds to the 7,579 words in all descriptions for the 6,000 training images. A probability between 0 and 1 was associated with each dimension of the vector. The word (dimension of the vector) which corresponded to the highest probability will be assigned as the description of an input image.

This LSTM model was implemented using the *Keras* library, and the learning rate is set to 0.005 together with epoch as 1,00. The learning rate was set by trying 11 values among 0.000001, 0.000005, 0.00005, 0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1. Regarding the time spent on the training process and the accuracy, 0.005 was the best learning rate for this project. Adam was the optimizer for this model. I decided to use Adam because it combines the best properties of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also uses the average of the second moments of the gradients (the variance). Therefore, it could handle sparse gradients on noisy problems. Categorical cross-entropy was used as the loss function for this LSTM model. It was chosen because it was designed to quantify the difference between two probability distributions. To train this LSTM model, I used an NVIDIA GeForce RTX 2080 GPU with 8GB of memory. The purpose of using GPU was to improve the efficiency of the training process. Average GPU utilization when training the CNN is about 45%, and it took about 30 hours to complete the model training.

IV. ANALYSIS AND RESULTS

A. Data

The data used in this project was the Flickr8k dataset [10]. I decided to use Flickr8k rather than Microsoft's Common Objects in Context dataset (COCO) dataset for two reasons. The first one was Flickr8k is small. So, I can train the model with my desktop. Secondly, Flickr8k is a high-quality dataset. Five captions are provided for each image. The dataset was properly labeled and documented.

Both images and captions were provided. In terms of images, Flickr 8k dataset provides a total of 8,092 images in JPEG format with different shapes and sizes. The dataset also did not specify the number of different objects. Among those images, 6,000 were used for training, 1,000 for tests, and the rest 1,000 were used for development. In this project, I used *VGG16* to extract the features from the 6,000 training images. The features and pre-processed words from captions were used to train the LSTM model. The 1,000 test images and corresponding captions were used to evaluate and improve the performance of the trained LSTM mode. The 1,000 images for the development were never used during the model training process (training weights or cross-validation). The final result for this project was based on the development set.

For the images. I normalized the pixel values to the range from 0 to 1 by calculating the mean value per image and subtracting the mean value from the pixel values prior to putting it into the *VGG16*. The primary reason for having image normalization was to avoid the potential gradients exploding and vanishing to increase the convergence speed. I also re-sized the RGB images to a fixed size at 224 x 224 as required for running the *VGG16*.

In terms of the captions, there were five captions for each image and 40,460 captions in total for the 8,092 images (8,082*5 = 40,460). I did the following steps to pre-process the captions: firstly, tokenization by separating the captions into words; secondly, normalizing the cases of the words by transferring all characters to lowercase; thirdly, removing the punctuation and stop words; fourthly, reducing a word to its stem by removing the prefix and suffix. Finally, the size of the training vocabulary is 7,371.

B. *Loss and Accuracy on Training and Testing Sets*

Several LSTM models were trained, the best performance model had a categorical cross-entropy equal to 2.858 on training sets and 3.114 on the test set. The corresponding accuracy was 44% on the training set and 41% on the test set. This model was used as the final model to generate the captions for the images from the development set.

C. *Bilingual Evaluation Understudy (BLEU)*

BLEU claims a high correlation with human judgments of quality and is one of the most popular automated and inexpensive metrics [12]. Commonly, *BLEU* is used to evaluate the quality of text which has been machine-translated from one natural language to another. *BLEU* scores were computed for individual translated segments by comparing them with a set of good quality reference translations. Those scores were then averaged over the whole corpus to reach an estimate of the translation's overall quality. The average *BLEU* score indicates the similarity between the predicted text and the reference texts. The range of *BLEU* scores was between 0 and 1, where 1 indicated a high similarity.

In this project, the generated captions can be seen as "translated" from images. Therefore, I also computed the *BLEU*. The LSTM model achieved an average BLEU-1 value of 0.52 on the test set (with 1,000 images).

D. *Examples of Generated Captions*

In this section, I will present three images from the development set to show the generated captions and the original descriptions provided by the dataset. For each image, five captions will be generated from the LSTM model. The three images were randomly selected. There were:

- 214501174_6db1f4d69c.jpg (Fig.3a),
- 1440024115_129212c988.jpg (Fig.3b),
- and 2950637275_98f1e30cca.jpg (Fig.3c).

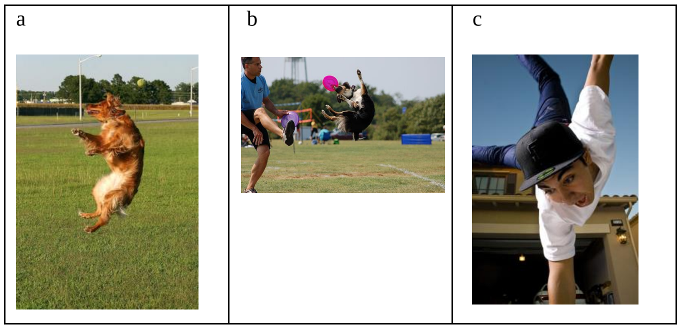


Fig.3. Images from the development set

a. *<u>The 1st Image</u>*

- The five captions generated from the trained LSTM model were:

```
dog is running through the grass [log prob: -7.83]
brown dog is running through the grass [log prob: -8.56]
brown dog is running through the water [log prob: -8.99]
black and white dog is running through field [log prob: -9.40]
black and white dog is running through the grass [log prob: -9.49]
```

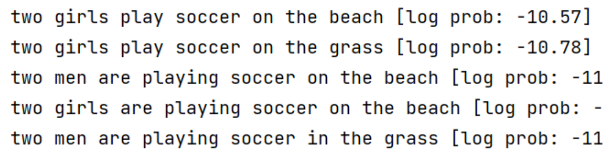
- The five original captions in the Flickr8k database were:

  #1: A dog jumping in midair above grass.
  #2: A dog jumps up to catch a tennis ball in a grassy field.
  #3: A golden dog is jumping in an attempt to catch a ball.
  #4: dog in air
  #5: The dog is in the air above the grass.

- The BLEU value for this image is 0.75.

b. *<u>The 2nd Image</u>*

- The five captions generated from the trained LSTM model were:

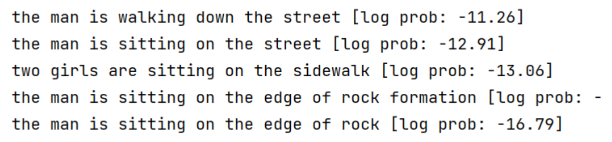


```
two girls play soccer on the beach [log prob: -10.57]
two girls play soccer on the grass [log prob: -10.78]
two men are playing soccer on the beach [log prob: -11
two girls are playing soccer on the beach [log prob: -
two men are playing soccer in the grass [log prob: -11
```

- The five original captions in the Flickr8k database were:
  - #1: A man plays Frisbee with his dog.
  - #2: A man with a Frisbee and a dog in the air with a Frisbee in his mouth.
  - #3: A tri-colored dog jumps and catches a pink frisbeen that a man in shorts has thrown.
  - #4: Small dog catching a Frisbee.
  - #5: The man and dog, which is leaping into the air, are playing Frisbee.
- The BLEU value for this image is 0.33.

c. *The 3rd Image*

- The five captions generated from the trained LSTM model were:



```
the man is walking down the street [log prob: -11.26]
the man is sitting on the street [log prob: -12.91]
two girls are sitting on the sidewalk [log prob: -13.06]
the man is sitting on the edge of rock formation [log prob: -
the man is sitting on the edge of rock [log prob: -16.79]
```

- The five original captions in the Flickr8k database were:
  - #1: A breakdancer is standing on one hand with a look of excitement
  - #2: A man doing a handstand outside of a garage.
  - #3: A person balances on one hand.
  - #4: The boy in the black hat balanced his body on one hand.
  - #5: The man is attempting a one-handed handstand in front of a building.
- The BLEU value for this image is 0.667.

V. CONCLUSION

In conclusion, I trained and evaluated an LSTM with two sources of input for generating the image caption. The average BLEU-1 value of was 0.52 on the test set based on the trained LSTM model. In general, the results showed that a trained model could generate the caption for a given image. It indicates that LSTM was an appropriate model for generating image captions. Most of the sentences are human-readable, however, the generated captions were not very accurate. For example, the one dog and one man in the second image were detected as two girls or two men. In the future, I plan to improve this project by training a deeper LSTM model and also trying GANs to improve the accuracy problem in the captions.

VI. ACKNOWLEDGEMENT

I would like to thank Dr. Corey Toler-Franklin for useful guidance on the ideas in the paper, and the encouragement.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *Int. Conf. on Learning Representations*, 2015.

[2] H. Li, X. Li, and W. Wang, “Research on Image Caption of Children's Image Based on Attention Mechanism.” *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. 2021.

[3] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. “Show and tell: A neural image caption generator,” Proceedings of the IEEE *Conference on Computer Vision and Pattern Recognition*, 2015.

[4] I. S. Alex Krizhevsky and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko, “Long-term recurrent convolutional networks for visual recognition and description,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] S. Yan, F. Wu, J. S. Smith, W. Lu, and B. Zhang, “Image captioning using adversarial networks and reinforcement learning,” *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018.

[7] C. Chen, S. Mu, W. Xiao, Z. Ye, L. Wu, and Q. Ju, “Improving image captioning with conditional generative adversarial nets,” *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting”, *The journal of machine learning research*, vol. 15, no. 1, pp 1929-1958, 2014.

[9] D. Simard, P.Y. Steinkraus and J.C. Platt, "Best Practices for Convolutional Neural Networks", *Proc. Seventh Int'l Conf. Document Analysis and Recognition*, 2003.

[10] M. Hodosh, P. Young and J. Hockenmaier, "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", *Journal of Artifical Intellegence Research*, 2013. Available: http://www.jair.org/papers/paper3994.html

[11] J. Brownlee, “How to automatically generate textual descriptions for photographs with deep learning,” *Machine Learning Mastery*, 07-Aug-2019. [Online]. Available: https://machinelearningmastery.com/how-to-caption-photos-with-deep-learning/. [Accessed: 20-Oct-2021].

[12] C. Callison-Burch, M. Osborne, and P. Koehn, “Re-evaluating the Role of BLEU in Machine Translation Research”, 1*1th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2006.