International Journal of
Image Processing and
Pattern Recognition

ISSN: 2456-6985
Volume 10, Issue 2, 2024
July–December
DOI (Journal): 10.37628/IJIPPR

https://journalspub.com/journal/ijippr

Review IJIPPR

Image Caption Generator Implementation

Sejal Jain[1], Saloni Agarwal[2], Sonam Gour[3,*], Sanjivani Sharma[4], Shrishti Agarwal[5]

Abstract

Image caption generation is a software technology that takes an image as an input and produces a descriptive caption in text form. In the modern era, this technology finds application in various fields. For instance, automatically generating captions for medical images aids in diagnosis and enhances reporting efficiency, helping healthcare professionals to quickly interpret complex visuals. In the realm of autonomous vehicles, image captioning enables these vehicles to understand and communicate about their surroundings, thereby improving safety and navigation. Furthermore, in journalism, generating captions for news images can enhance comprehension and engagement for readers. This paper will provide an overview of the technologies that can be used to develop an image caption generator using the Flickr8K dataset from Kaggle. The implementation includes various tools like OpenCV, which is widely utilized by leading tech companies, such as Google and Microsoft. The paper also includes snapshots of the generated outputs to illustrate the model's effectiveness. The primary aim of this implementation is to gain insights into the practical use of these tools and technologies in real-world projects.

Keywords: Image caption, Python, neural networks, ResNet, OpenCV, long short-term memory (LSTM), Keras, deep learning etc.

INTRODUCTION

The aim is to provide a brief idea about how an image caption generator can be implemented using tools like Python, Keras, TensorFlow ResNet50 model, long short-term memory (LSTM), and OpenCV [1]. The literature review consists of tools and technology description we identified most relevant for building the model and then making a framework to showcase the practical implementation of the model using flask.

LITERATURE REVIEW

We reviewed various research papers from different publishers such as IEEE, Springer, EDP Sciences, etc. This curated selection serves as the foundation for our investigation into the ever-changing landscape of image caption generation [2].

***Author for Correspondence**
Sonam Gour
E-mail: sonam.gaur@poornima.org

[1–2,4–5]Student, Computer Science and Engineering, Poornima College of Engineering, Jaipur, Rajasthan, India.
[3]Assistant Professor, Computer Science and Engineering, Poornima College of Engineering, Jaipur, Rajasthan, India.

Received Date: May 06, 2024
Accepted Date: June 22, 2024
Published Date: October 08, 2024

Citation: Sejal Jain, Saloni Agarwal, Sonam Gour, Sanjivani Sharma, Shrishti Agarwal. Image Caption Generator Implementation. International Journal of Image Processing and Pattern Recognition. 2024; 10(2): 18–23p.

LSTM is a type of recurrent neural network architecture that excels at capturing long-term patterns and is ideal for sequential prediction tasks. It uses the tanh activation function. It is designed to take individual elements of an information as an input and then pass it to the other convolutional layer, but it stores the keywords in the form of a constant and while making prediction in the future it looks back to the keywords it has stored and then predicts the next value [3]. One of the most important applications of the LSTM model is the image caption generation in which it takes the image as an input and provides a one liner sentence for the

description of the image provided as input. Text generation involves the computation of words when a sequence of words is fed as input [4]. Language models can be operated. Although LSTM has numerous applications, including machine translation, text summarization, sentiment analysis, question answering, and chatbots, it also has some drawbacks. For instance, LSTM requires more time to train the model to achieve higher accuracy, lacks true long-term memory, making it less favored by developers, and can lead to overfitting due to its potential for high accuracy [5].

Residual Network 50 (ResNet 50), which is used for image caption generation is a variant of the ResNet model. It is a classic neural network commonly used as the backbone for many computer vision tasks. It features 48 convolution layers, along with one max pool and one average pool layer [6]. ResNet model is important because it solves the issue of finishing gradient problem with most of the neural network models. It employs skip connections to address the issue of vanishing gradients. ResNet finds its application in: performing image classification; object detection; semantic segmentation, while KERAS is an API for deep learning of high level. Python language was used for writing this API for neural networks [7]. Keras is widely used nowadays because it provides support to various neural network computations which are performed at back and it makes it easy to implement neural networks. Keras is an easy and very powerful open-source API for developing complex deep-learning neural network models.

It consists of efficient computational libraries that allow us to define and train neural networks in the minimum lines of code. It uses the concept of distributed training of deep learning models [8].

TensorFlow is also an open-source library which was created by Google so that implementation of large-scale machine learning models can be done. It was initially used for the purpose of optimizing the results obtained by a search to improve customer experience on the browser [9]. Tensors are basically containers that are used to hold data in the form of matrix. Just like matrix, which can be of one dimension, two dimension, or three-dimension, tensor can also be used to hold data in a 3D space. It gives fast and accurate results for dot products and cross products, which are intensively required for higher and scientifically complex calculations.

Tensor flow does all the calculation using some graphs that are known as computational graphs. Computational graphs have two main elements: operators and tensors [10]. We can also choose to do a minor calculation, that is, to execute a part of a graph rather than executing a complete graph this can be done by creating a session of the graph. TensorFlow utilizes Python as its primary front-end API for developing applications with the framework, but it also includes wrappers for several other languages, such as C^{++} and Java (Figure 1).

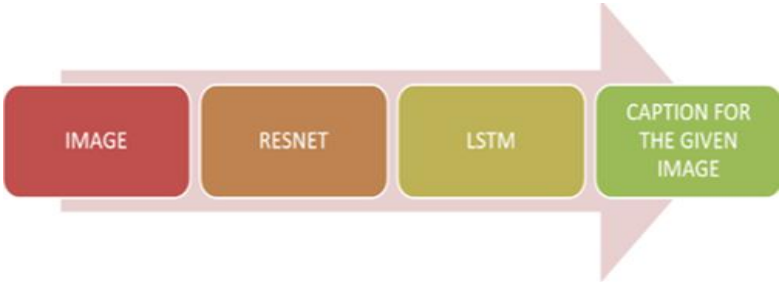

Figure 1. Architecture of ResNet – LSTM model.

METHODOLOGY

The process of creation can be broadly classified into 5 categories (Figure 2). They are data collection, image processing, model development, testing, and web framework [11].

Data Collection

The Flickr8k dataset is a widely used dataset for image captioning tasks.

Size
8000 images with 5 captions per image.

Source
The images are sourced from the Flickr photo-sharing website.

Annotations
Each image is associated with multiple captions, providing diverse descriptions for each image.

Usage
Frequently employed in tasks like image captioning, where models are trained to create descriptive captions for provided images.

Image Processing
Steps involved in image processing are loading an image, displaying images, converting color spaces (from RGB to grayscale or vice versa), image manipulation (cropping), image filtering (blur or sharp), object detection, saving images [12].

Model Development
Clearly define the problem and find sources to collect data relevant to the problem statement [13]. Perform data preprocessing by addressing missing values and managing outliers.

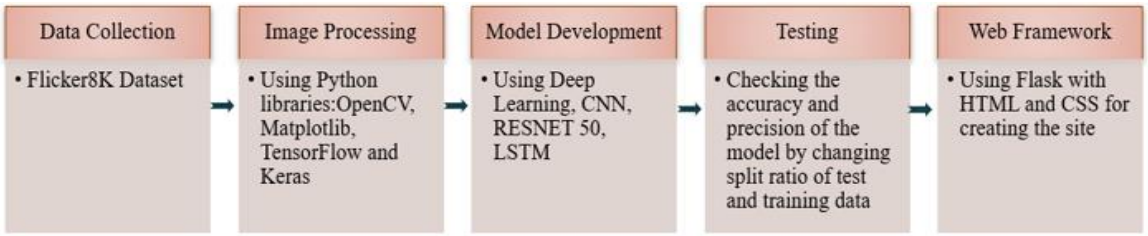

Figure 2. Step-wise representation of solution designing.

Split the given data points into training, validation, and test data points. Evaluate different model configurations. Finalize the model and train the selected model using optimization algorithm [14].

Select suitable evaluation metrics based on the problem type – accuracy, precision, recall, and F1-score for classification; mean-squared error and R-squared for regression. Assess the model's performance, then deploy the trained model into a production environment to make predictions on new data points [15]. Continuously monitor the deployed model's performance in real-world scenarios and gather feedback. Periodically retrain the model on updated data to check its performance and find scope of improvement.

Testing
Testing is done keeping some points as standards, such as evaluation matrices, cross-validation, feature analysis, error analysis, and evaluate model performance across different groups. Model robustness against noisy or outlier data points [16].

Web Framework
Python provides a web framework named 'Flask', which can be used to create web pages in Python using HTML and CSS, thus providing a better user-friendly interface [17].

IMPLEMENTATION OUTPUT
When discussing the 'implementation output' for an image caption generator, we refer to the results produced after implementing and executing the image captioning system [18, 19]. This includes the captions generated for images and various metrics to evaluate the performance of the system as shown in Figures 3–9 and Table 1.


```
captions_dict = {}
for i in captions:
    try:
        img_name = i.split('\t')[0][:-2]
        caption = i.split('\t')[1]
        if img_name in images_features:
            if img_name not in captions_dict:
                captions_dict[img_name] = [caption]

            else:
                captions_dict[img_name].append(caption)

    except:
        pass
```

Figure 3. Snapshot of the code.

```
import matplotlib.pyplot as plt

for k in images_features.keys():
    plt.figure()

    img_name = '../input/flickr8k-sau/Flickr_Data/Images/' + k


    img = cv2.imread(img_name)

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.xlabel(captions_dict[img_name.split('/')[-1]])
    plt.imshow(img)

    break
```

Figure 4. Snapshot of the code.

Figure 5. Snapshot of the predicted caption.

Predicted caption for the given image is.. three boys playing in the snow . . . .

Figure 6. Snapshot of the output.

Figure 7. Snapshot of the output.

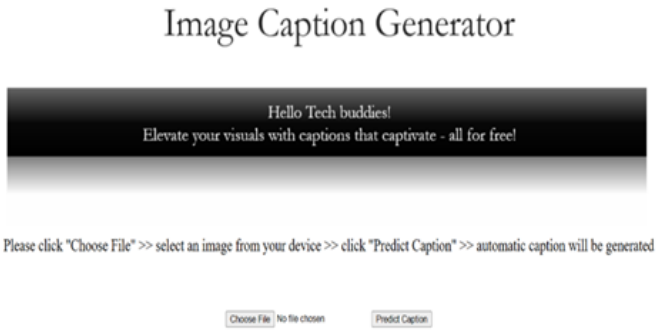

Figure 8. Snapshot of the output.

Figure 9. Snapshot of the model accuracy.

Table 1. Overall analysis of the model.

Total images	8000
Total captions	40,000
Test and training data split	75:25
Batch size	512
Epochs	50
Accuracy %	73.28

CONCLUSION

In this paper, the introduction to image caption generator and the applications of image caption generator in modern era are explained. Introduction to modern tools have been provided to practically implement the model. Paper talks about the systematic approach to build a model. Step-by-step approach is mentioned from data collection sources to model deployment framework using flask. Along with applications of neural networks, ResNet 50, OpenCV, LSTM, and Keras, the limitation of each tool is also described, so that problems of overfitting and underfitting can be avoided. Snapshots of the designed web page are attached for the reference; the flask framework is used with HTML and CSS for improving front-end design of the web page. User needs to upload an image by clicking upload image button and then click on predict caption button. After clicking on predict caption button the user will be directed to next web page, where output caption will be displayed along with the input image.

REFERENCES

1. Liu S, Bai L, Hu Y, Wang H. Image captioning based on deep neural networks. In: MATEC web of conferences. 2018. pp. 01052). EDP Sciences.
2. Farhadi, A. et al. (2010). Every Picture Tells a Story: Generating Sentences from Images. In: Daniilidis, K., Maragos, P., Paragios, N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Heidelberg: Springer, Berlin; pp. 15–29. https://doi.org/10.1007/978-3-642-15561-1_2
3. Feng Y, Lapata M. Automatic caption generation for news images. IEEE Trans Patt Anal Machine Intell. 2012;35(4):797–812.
4. Chen J, Dong W, Li M. Image caption generator based on deep neural networks. 2014. Available from: https://www.cs.ubc.ca/~carenini/teaching/cpsc503-19/final-projects-2016/image_caption_generator_final_report.pdf
5. 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. 2015. pp. 3156–3164.
6. Liu C, Wang C, Sun F, Rui Y. Image2Text: a multimodal image captioner. In: Proceedings of the 24th ACM international conference on Multimedia (MM '16). Association for Computing Machinery. New York, NY. 746–748. https://doi.org/10.1145/2964284.2973831
7. Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncert Fuzz Knowledge-Based Sys. 1998;6(02):107–16.
8. Tanti M, Gatt A, Camilleri KP. Where to put the image in an image caption generator. Nat Lang Eng. 2018;24(3):467–89.
9. Venugopalan S, Rohrbach M, Donahue J, Mooney R, Darrell T, Saenko K. Sequence to sequence-video to text. In: Proceedings of the IEEE international conference on computer vision. 2015. pp. 4534–4542.
10. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078. 2014 Jun 3.
11. Yang L, Hu H. TVPRNN for image caption generation. Electron Lett. 2017;53(22):1471–3.
12. Blandfort P, Karayil T, Borth D, Dengel A. Image captioning in the wild: how people caption images on Flickr. In: Proceedings of the workshop on multimodal understanding of social, affective and subjective attributes. 2017 Oct 27. pp. 21–29.
13. Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002 July. pp. 311–318.
14. 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: International conference on machine learning. 2015 Jun 1. pp. 2048–2057. PMLR.
15. Papineni K. BLEU: a method for automatic evaluation of MT. Research Report, Computer Science RC22176 (W0109-022). 2001.
16. 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. 2005 June. pp. 65–72.
17. Lin CY. Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. 2004 July. pp. 74–81.
18. 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. 2015. pp. 4566–4575.
19. Anderson P, Fernando B, Johnson M, Gould S. Spice: Semantic propositional image caption evaluation. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam: Springer International Publishing;. October 11–14, 2016. pp. 382–398).

