Image Caption Generator Report

S. Jappeswaran Balasubramanian
Sachin Krishan Thyaharajan
Atharva Kshirsagar

I. Introduction

The goal of this project is to develop an advanced image caption generator, leveraging the synergy between computer vision techniques and natural language processing. By harnessing the power of these two domains, the generator aims to automatically generate relevant and descriptive captions for a wide range of input images. The generated captions will not only accurately describe the content of the image but also strive to capture the essence of the depicted objects, actions, and relationships. To achieve this, the model will employ state-of-the-art computer vision algorithms to extract meaningful visual features from the image, enabling it to understand the visual context. These features will then be fed into a natural language processing model that will generate captions that exhibit grammatical correctness, natural language fluency, and semantic coherence. The training of the model will involve a large dataset of paired images and corresponding captions, allowing it to learn the intricate associations between visual cues and textual descriptions. The generated captions can find applications in various fields, including image indexing, content retrieval, assistive technologies for the visually impaired, and enhancing the overall user experience in image-based platforms and applications.

The image caption generator developed in this project has versatile applications. It enhances accessibility by providing textual descriptions for visually impaired individuals and improves content indexing and search capabilities. It elevates user engagement and searchability of shared content on social media platforms. Additionally, it aids in e-commerce by generating descriptive captions for product descriptions, aiding potential buyers in making informed decisions. Overall, the generator offers immense potential in various domains, benefiting accessibility, content indexing, social media, and e-commerce experiences.

II. Related Works

Image captioning has been extensively studied, with several notable contributions in the field. Here, we highlight key approaches and advancements in image captioning, specifically focusing on projects utilizing the Flickr8 dataset.

Show and Tell [1]: Vinyals et al. proposed a neural image caption generator using a CNN to extract image features and an RNN for caption generation, laying the foundation for subsequent research.

Show, Attend, and Tell [2]: Xu et al. enhanced image captioning by incorporating a soft attention mechanism and a multimodal architecture, achieving superior results on various datasets, including Flickr8.

Bottom-Up and Top-Down Attention [3]: Anderson et al. proposed a region-based model that employs bottom-up and top-down attention mechanisms, resulting in state-of-the-art performance on datasets similar to Flickr8.

Generative Adversarial Networks for Image Captioning [4]: Reed et al. integrated GANs, using a generator network to produce captions and a discriminator network to improve caption quality and diversity.

These works have significantly contributed to the field of image captioning, providing insights and techniques that have shaped the development of systems, particularly when applied to the Flickr8 dataset.

III. Data

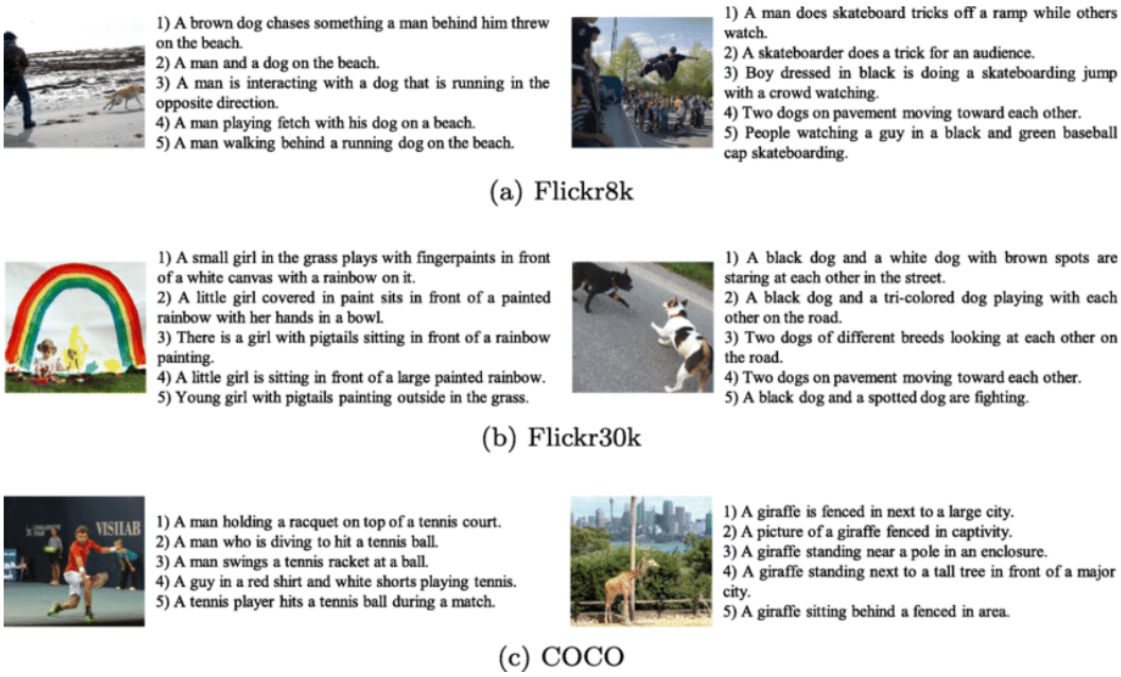

Fig 1. Datasets explored

We have used the **Flicker8k** Dataset for our project. The Flickr8K dataset is a widely used benchmark dataset in the field of image captioning. It consists of 8,000 high-quality images sourced from the photo-sharing platform Flickr, with each image accompanied by five human-generated captions. The dataset provides a diverse range of scenes, objects, and activities, making it suitable for training and evaluating image captioning models. The carefully curated captions in the dataset capture relevant details and context, allowing models to learn the associations between visual content and textual descriptions. Building on the success of Flickr8K, the Flickr30K dataset expands the collection to 31,783 images, again with five captions per image. It offers a larger and more diverse set of images, enabling more robust training and evaluation. Additionally, the COCO (Common Objects in Context) dataset is another widely used dataset in the image captioning community. It contains over 120,000 images, each with multiple human-annotated captions. Captions in Flickr8k and Flickr30k datasets are tailored for research in image captioning, whereas the COCO dataset serves a broader range of research topics including image recognition, segmentation, and captioning. In terms of training efficiency, the Flickr8k dataset is advantageous for resource-constrained environments, as it provides a smaller collection of images compared to Flickr30k, making it ideal for faster training with limited resources.

IV. Preprocessing

```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg,A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
```

Fig. 2 Captions.txt format before preprocessing

In the caption cleaning process, a mapping dictionary is created from the Captions.txt file, where each image ID is associated with a list of captions. These captions undergo several steps of cleaning, including conversion to lowercase, removal of digits, special characters, and spaces. Additionally, a start tag ('startseq') and an end tag ('endseq') are added to each caption, indicating the beginning and end of the sentence. As an example, a sample caption would look like: *"startseq child in pink dress is climbing up set of stairs in an entryway endseq"*. To prepare the captions for further processing, they are tokenized using the Tokenizer() function from the *tensorflow.keras.preprocessing.text* library. The dataset consists of a total of 40,455 captions, resulting in a vocabulary size of 8,485 distinct words.

```
[10] # before preprocess of text
     mapping['2224995194_518859d97d']

     ['The rock climbers are on their equipment in front of a cave .',
      'Two people hang from safety ropes over a giant cliff .',
      'Two people hanging on a rope over a cliff .',
      'Two people lower themselves into a cave with long ropes .',
      'two people repel down into a hole .']
```

Fig. 3 Mapping created after the preprocessing step

V. Data Generation

To prepare the data for the model, a custom data generator is implemented. It operates on the training set and follows a series of steps. For each image in the training set, the generator processes each caption from the captions list associated with the image. The caption is encoded using a tokenizer, which assigns a numerical representation to each word. Next, starting from the first word up to the length of the caption, the sentence is split into input and output

pairs. The input serves as the X or feature, while the output is the target or Y. To ensure consistent input shapes, the sequences are padded to a specific length, often set to the length of the longest caption in the dataset, such as 35 in this case. The output sequences, representing the next word in the sentence, are encoded using a binary categorical encoder. This data generation process enables the model to learn the associations between the input features and their corresponding target words, facilitating the generation of accurate and contextually relevant captions for given images.

VI. Model Architecture

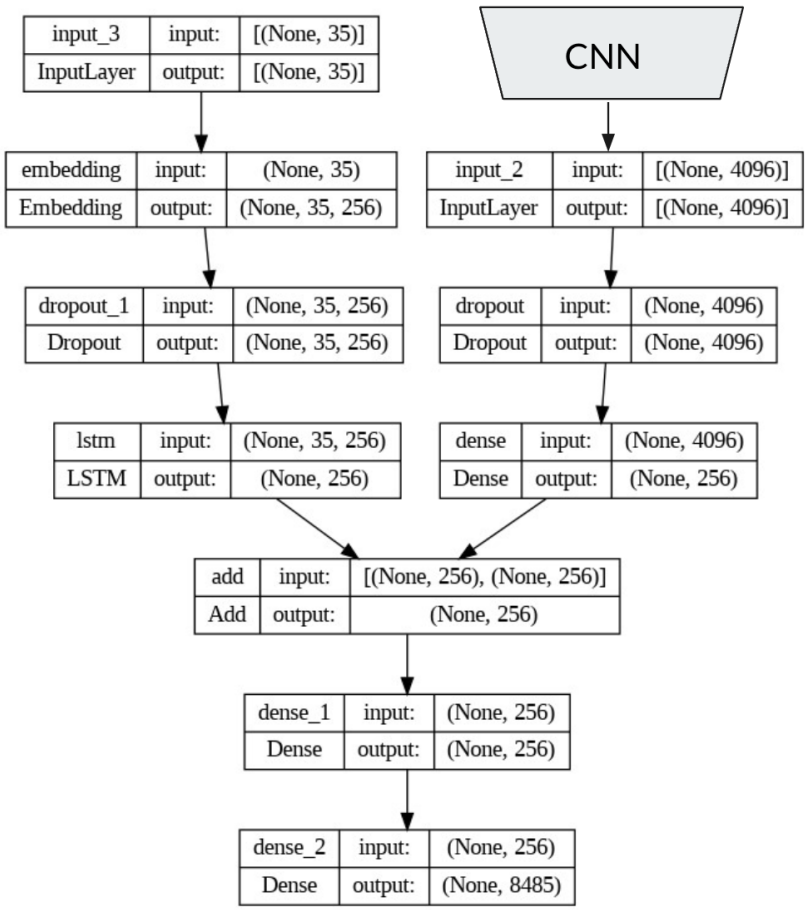

Fig 4. Model architecture

The main model consists of a CNN and a LSTM model.The input image is first passed into the CNN model to get a feature vector representing the input image in a vector format. Simultaneously, the embedding model works with the caption for that image and tokenizes the input caption into numbers. The caption is then padded to make it length 35. The captions as well as the image vector are then

passed to the *Data Generator* function to get input and output for our model. During training, this input pair (X1 - image feature vector & X2 - word by word tokenizing) is sent to the LSTM model to generate the next word in the sequence in an iterative manner until the <endseq> tag is reached which indicates the end of the example.

CNN Models:
Pre-trained state-of-the-art CNN models, including **VGG16**, **ResNet50**, and **Xception**, are utilized to encode images into feature vectors. These models capture high-level visual features, enabling the generation of descriptive and contextually relevant captions. Leveraging transfer learning, the learned representations from these models ensure rich and aligned image features for the image captioning pipeline.

LSTM Model:

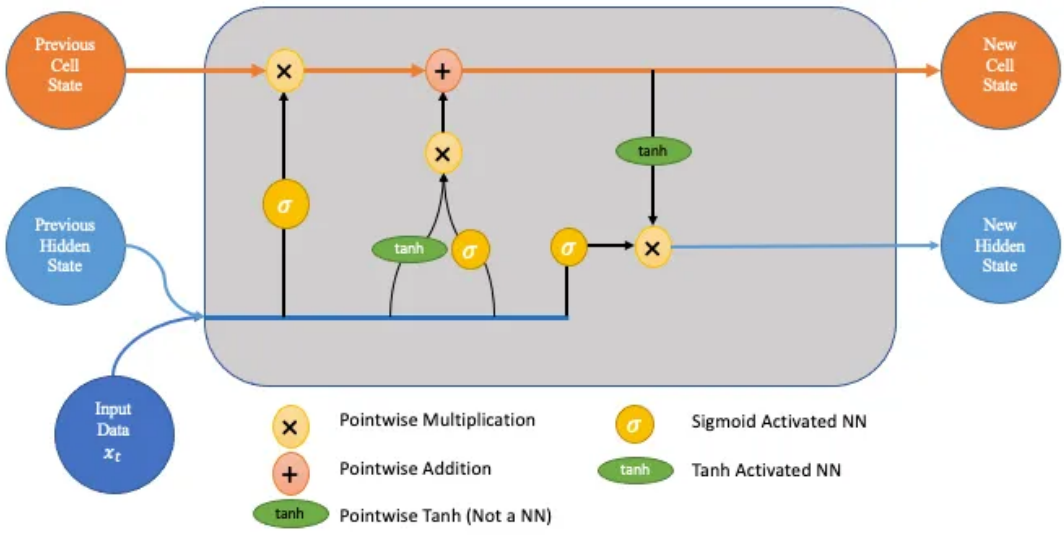

Fig 5. LSTM Model

An LSTM (Long Short-Term Memory) model is employed for the image captioning task. LSTM is a type of recurrent neural network (RNN) that is capable of capturing long-range dependencies in sequential data. In our project context, the LSTM model takes as input the encoded image features and generates a sequence of words to form the caption. It effectively learns the associations between the visual information from the images and the corresponding textual descriptions. The LSTM model's ability to capture temporal dependencies makes it suitable for generating coherent and contextually relevant captions for images.

VII. Training

We split the dataset into a training set and a test set in a 90:10 ratio, respectively. To provide for more training data for the model. Our model is designed to solve a multi-class classification problem with 8485 classes, representing the vocabulary size. We utilize the categorical cross-entropy loss function. This loss function is well-suited for multi-class classification tasks, as it measures the dissimilarity between the predicted class probabilities and the true class labels. We employed the Adam optimizer. Adam combines the benefits of both adaptive learning rate methods and momentum-based methods, resulting in efficient and effective parameter updates. We create mini-batches of size 32 from the training data. Mini-batch training allows us to process multiple samples simultaneously, improving the training and memory usage efficiency. We trained the model for 20 epochs.

The vector representations of images were generated and stored of all the three CNNs before training and stored. The vectors were read based on their image ID and fed as input to input_2 layer (fig. 4) during training.

VIII. Metrics

With applications where the result is a sentence, how good an output is more difficult as it tends to be subjective. In this situation, there may be multiple valid responses rather than just one that is always true. Two distinct translators may produce two somewhat different, yet equally accurate, translations of the same sentence, for example.

Applications where there is a wider variety of acceptable solutions, such as image captioning or text summarizing, make the issue much more difficult. We require a quantitative indicator for evaluating the validity of our model's predictions in order to assess its performance.

A variety of NLP measures have been created over time to address this issue. The BLEU Score is among the most well-known. It has numerous flaws and is far from ideal. However, it is easy to calculate, comprehend, and has a number of strong advantages. Despite having a wide range of alternatives, it remains one of the most popular measures. Its foundation is the notion that the more closely the predicted language resembles the human-generated target sentence, the more accurate it will be.

Bleu Scores range from 0 to 1. The best you can do is get a score between 0.6 and 0.7. Even two humans would probably come up with various sentence alternatives for a given issue and would infrequently find a perfect match. Because of this, a score closer to 1 is illogical in real life and serves as a warning sign that the model is overfitting.

The score is based on the n-gram precision and the brevity penalty. BLEU compares n-grams (contiguous sequences of words) in the generated translation with the reference translation. It calculates the precision of n-grams by counting how many n-grams in the generated translation also appear in the reference translation. The precision is then calculated by dividing the count of matching n-grams by the total count of n-grams in the generated translation.

The individual modified precisions for different n-gram orders (usually from 1 to 4) are combined using a weighted geometric mean. The weights can be uniform or vary depending on the order of the n-gram. Finally, the brevity penalty is applied to the combined modified precision score.

$$
\begin{aligned}
Geometric\ Average\ Precision\ (N) &= exp(\sum_{n=1}^{N} w_n\ log\ p_n) \\
&= \prod_{n=1}^{N} {p_n}^{w_n} \\
&= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}}
\end{aligned}
$$

The brevity penalty term is used to handle cases where the generated translation is shorter than the reference translation. It is calculated as the ratio of the reference length to the generated length, and a penalty term is applied to the BLEU score accordingly.

$$
Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}
$$

c is predicted length = number of words in the predicted sentence and
r is target length = number of words in the target sentence

Finally, to calculate the Bleu Score, we multiply the Brevity Penalty with the Geometric Average of the Precision Scores.

$$Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N)$$

Bleu Score can be computed for different values of N. Typically, we use N = 4.

IX. Results

Test set results:

	BLEU-1	BLEU-2
VGG16 -LSTM	0.526	0.303
ResNet-50 - LSTM	**0.562**	**0.339**
Xception - LSTM	0.556	0.335

We employed BLEU-1 and BLEU-2 scores to evaluate the models on the test set. The table provided displays the performance of different CNN architectures utilized for extracting image embeddings. Among these architectures, the **ResNet-50 - LSTM** configuration demonstrated the highest BLEU-1 and BLEU-2 scores of 0.562 and 0.339, respectively. Interestingly, despite VGG-16 being a larger model with a higher number of parameters compared to ResNet-50, it did not yield superior performance on the test set. This outcome can be attributed to the fact that ResNet-50 incorporates skip connections, which enable it to generalize better on the test data. On the other hand, the Xception-LSTM model, despite its greater complexity and fewer parameters, did not outperform the other architectures.

Sample Outputs:

- **Output 1:**

VGG16 - Little girl with pigtails playing with fingerpaints
ResNet50 - little girl in pigtails peeks out to rainbow object
Xception - girl in pink dress is covered in front of rainbow colored flag

- **Output 2:**

VGG16 - Two dogs are fighting on dirt path
Resnet50 - two dogs are playing with plastic bag
Xception - black and white dog with its mouth open and teeth standing in the snow

X. Conclusion

In this project, we explored an approach to image captioning using LSTM and CNN models. Our experiments demonstrated that the combination of these models effectively captures the visual and linguistic features necessary for generating accurate and descriptive captions. By leveraging the hierarchical structure of LSTM and the powerful visual representation capabilities of CNN, we achieved comparable results to state-of-the-art models on the Flickr 8k dataset.

Furthermore, we explored the impact of various architectural modifications using different CNN models to generate a vector representation of the images. These experiments showed that Resnet50 is the best CNN for this architecture giving a BLEU-1 and BLEU-2 scores of 0.562 and 0.339, respectively. While our approach shows promising results, there are several avenues for future improvement in this project such as, incorporating attention mechanisms into our model. This could allow it to focus on specific regions of an image while generating captions, potentially improving the alignment between the visual and textual information.

XI. Contribution of group members:

- S. Jappeswaran Balasubramanian: Sideshwar worked on the data collection, cleaning, formatting and preprocessing aspects of the project. The data was collected and visualized to do an exploratory data analysis of the images and the captions present in the dataset. He coded the functions necessary for cleaning up the data. He also wrote the function for mapping the captions with the corresponding input image. Furthermore, he wrote the function for the data generator which involves creating a pair of input and output to be passed on to the model while training so that we do not need to load the entire dataset in the memory. Here the text captions were padded to the necessary size and also tokenized.
- Sachin Krishan Thyaharajan: Sachin researched different Convolutional Neural networks and decided on the three CNNs to generate the vector representation: VGG16, Resnet50, Xception. The networks' output layers were cut and used to generate the vector representation using their pretrained weights. These vector representations were stored for training later. Sachin also collaborated with Atharva to design the model architecture and coding.

- Atharva Kshirsagar: Atharva along with Sachin worked on the Model Architecture aspect of the project. This involves researching related works on how different models are used and coding the model architecture. Atharva also performed the model training process which involves coding the model to compile and run for sets of epochs. Furthermore, Atharva also coded the function for generating outputs given the input image which involves passing the input image through the preprocessing function and then through the model to get the text prediction from the model. This process also involves passing the actual caption through the function and interpreting the output in a meaningful way given by the *.predict* function. The index with the highest probability needs to be extracted and the output needs to be converted back to word format and appended together to form a whole sentence as soon as the <*endseq*> tag is reached.

XII. References

1. Vinyals, Oriol & Toshev, Alexander & Bengio, Samy & Erhan, Dumitru. (2015). Show and tell: A neural image caption generator. 3156-3164. 10.1109/CVPR.2015.7298935.
2. Xu, Kelvin & Ba, Jimmy & Kiros, Ryan & Cho, Kyunghyun & Courville, Aaron & Salakhutdinov, Ruslan & Zemel, Richard & Bengio, Y.. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.
3. Anderson, Peter & He, Xiaodong & Buehler, Chris & Teney, Damien & Johnson, Mark & Gould, Stephen & Zhang, Lei. (2018). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. 6077-6086. 10.1109/CVPR.2018.00636.
4. Reed, Scott & Akata, Zeynep & Yan, Xinchen & Logeswaran, Lajanugen & Schiele, Bernt & Lee, Honglak. (2016). Generative Adversarial Text to Image Synthesis.
5. https://www.kaggle.com/datasets/adityajn105/flickr8k
6. https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9
7. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918
8. https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758
9. https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568
10. https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b