**Title:** **Image Captioning using LSTM on Flickr8K Dataset**

**Pin no’s:**HU21CSEN0500091

HU21CSEN0500184

HU21CSEN0500097

**1. Introduction**

Image Captioning is a challenging task in computer vision and natural language processing that involves generating textual descriptions for images. This task finds applications in image retrieval, assisting visually impaired individuals, and enhancing human-computer interaction. In this report, we explore the application of Long Short-Term Memory (LSTM) networks for Image Captioning, specifically on the Flickr8K dataset. Image Captioning using Long Short-Term Memory (LSTM) networks represents a significant advancement at the intersection of computer vision and natural language processing (NLP). This task involves automatically generating descriptive textual captions for images, a capability that holds immense potential in various domains including assistive technologies, content indexing, and human-computer interaction.

The core idea behind Image Captioning is to enable machines to comprehend the content of images and express it in natural language, mimicking human-like understanding and communication. LSTM networks, a type of recurrent neural network (RNN) architecture, have emerged as a powerful tool for sequence modelling, making them particularly well-suited for tasks involving sequential data such as language generation.

In the context of Image Captioning, LSTM networks play a crucial role in learning the associations between visual features extracted from images and the corresponding textual descriptions. By leveraging the sequential nature of language, LSTM-based models can capture the context and dependencies within sentences, thereby producing coherent and contextually relevant captions.

**2. Dataset Description**

The Flickr8K dataset comprises 8,000 images sourced from the Flickr platform, each associated with five human-generated captions. The dataset is divided into training, validation, and test sets. Each image is accompanied by descriptive captions, making it suitable for training models for Image Captioning tasks.

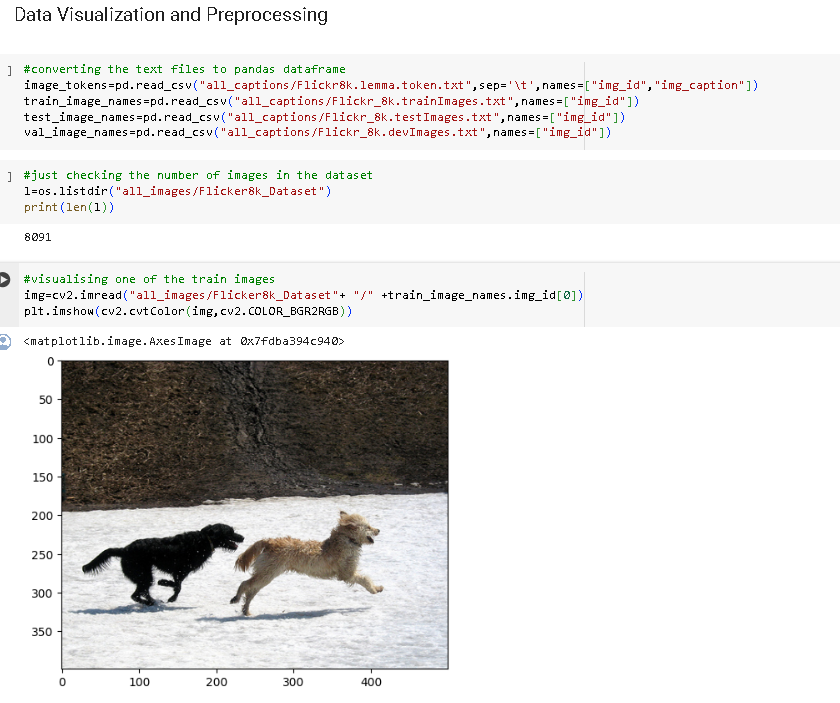
**3. Model Architecture**

We employ a deep learning architecture based on LSTM networks for Image Captioning. LSTM networks are well-suited for sequence modelling tasks due to their ability to capture long-range dependencies and handle sequential data effectively. The model consists of:

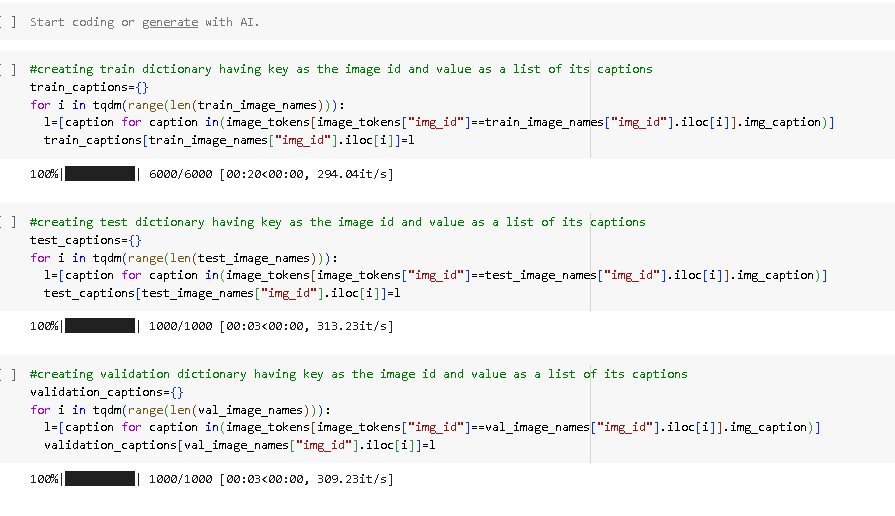
• Encoder: A Convolutional Neural Network (CNN) pre-trained on ImageNet is utilized to extract visual features from input images. The encoder transforms the images into a fixed-length feature vector.

• Decoder: An LSTM network takes the visual features generated by the encoder as input and generates a sequence of words that form the image caption. The decoder is trained using teacher forcing, where the ground-truth captions are fed as inputs during training.

**Data Preprocessing:**



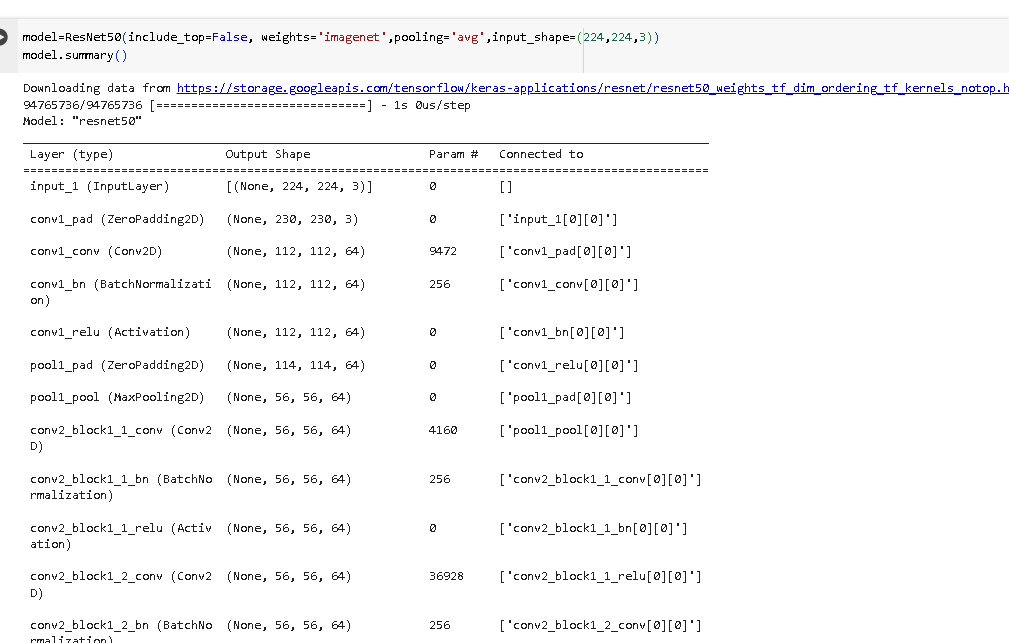
**Creating Dictionaries to Map Image IDs and Their Corresponding Captions**:



**Train, Test, and Validation Dictionaries:** we are iterating over the indices of training, testing, and validation image datasets. For each image, we are extracting the corresponding captions from another data structure (image\_tokens), which presumably contains image IDs and their associated captions. Then, we are storing these captions in lists, with the image ID serving as the key in the respective dictionaries (train\_captions, test\_captions, and validation\_captions)

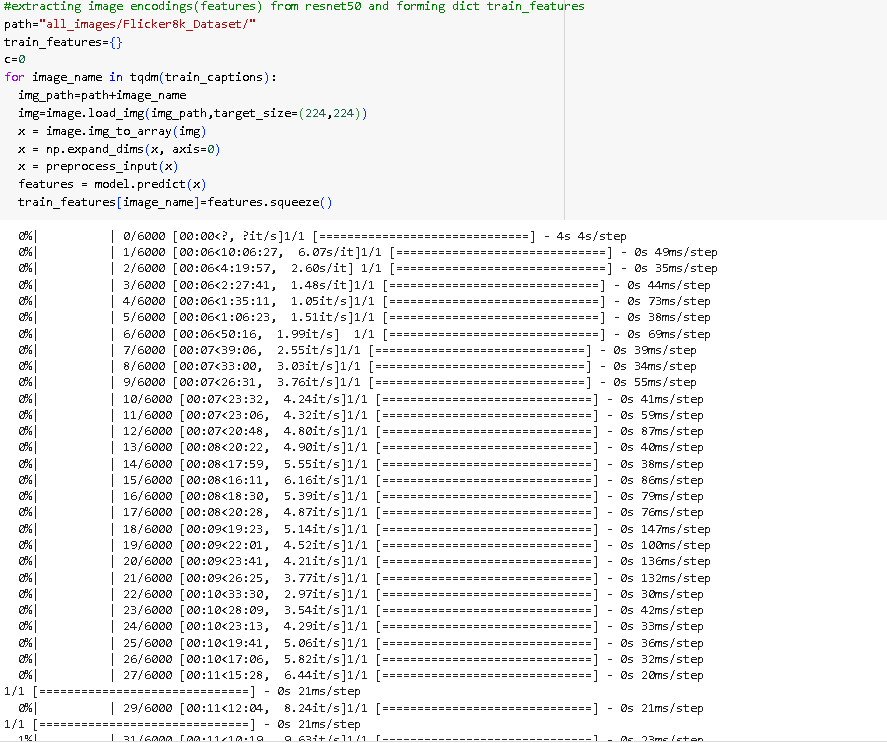
The dictionary (validation\_captions) associating image IDs with their respective captions in the validation set. This facilitates efficient organization and retrieval of caption data for validation purposes, crucial for evaluating model performance. Subsequently, a ResNet50 model is initialized for image encoding. With pre-trained weights from the ImageNet dataset and global average pooling, this model generates fixed-length feature vectors summarizing spatial information in images. The model.summary() function then offers a concise overview of the ResNet50 architecture, detailing its layers and parameters, aiding in understanding its structure and complexity.

**ResNet50 Model for Encoding Images:**



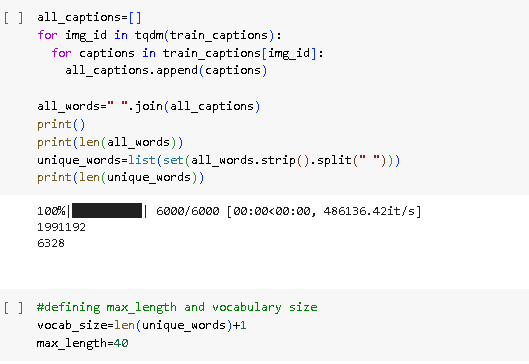
The convolutional neural network (CNN) model using ResNet50 architecture, a widely used deep learning model for image classification tasks. By setting include\_top to False, the fully connected layers at the top of the network, typically used for classifying images into specific categories, are not included. This allows for the customization of the model for various tasks such as feature extraction or transfer learning. The parameter weights is set to 'imagenet', which initializes the model with pre-trained weights trained on the large-scale ImageNet dataset. This helps in leveraging the knowledge gained from training on ImageNet for general image recognition tasks, which can significantly improve the model's performance. The parameter pooling is set to 'avg', indicating that global average pooling will be applied after the convolutional layers. This reduces the spatial dimensions of the feature maps to a single vector, which is beneficial for reducing overfitting and improving generalization. The input\_shape parameter specifies the dimensions of the input images expected by the model, in this case, images with a height and width of 224 pixels and three color channels (RGB). Overall, this setup initializes a ResNet50 model for feature extraction from images, leveraging pre-trained weights for efficient and effective image recognition tasks.

**Encoding images and forming dictionaries containing mapping of image\_id to image encodings:**

****

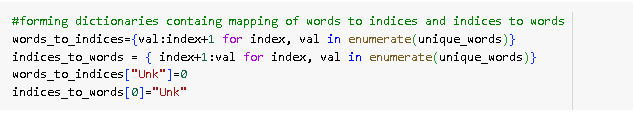
It iterates through each image ID in the train\_captions dictionary using tqdm, a tool for displaying progress bars during iterations. For each image ID, it iterates through each caption associated with that image, appending each caption to the all\_captions list.

After collecting all the captions, it combines them into a single string all\_words by joining them with spaces. This string effectively contains all the words present in the dataset. The code then prints the total number of characters in the all\_words string using the len function, providing insight into the overall size of the combined caption dataset.Subsequently, the code splits the all\_words string into individual words using the .split(" ") method, removes any leading or trailing spaces with .strip(), and converts it into a set to obtain the unique words present in the dataset. The length of this set is then printed, indicating the total number of unique words in the dataset.Setting hyper parameters for vocabulary size and maximum length**.**

****

Initially, it collects all captions associated with the images and aggregates them into a single list named all\_captions. It then concatenates all the captions into a single string called all\_words, representing all the words present in the captions. The length of this concatenated string is printed to the console to give an indication of the total number of characters in the captions. Subsequently, the code converts the concatenated string into a list of unique words, removing any duplicates, and prints the length of this list to determine the total number of unique words across all captions. This preprocessing step aims to provide insights into the vocabulary used within the dataset, which can be valuable for various natural language processing tasks.

**Creating dictionaries containg mapping of words to indices and indices to words:**

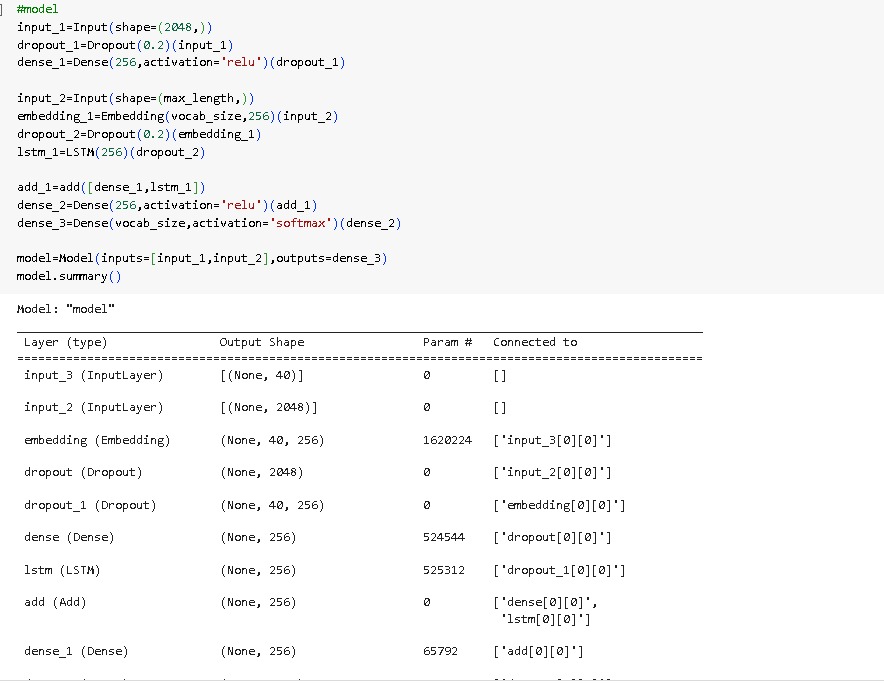
****

creates two dictionaries: words\_to\_indices and indices\_to\_words. These dictionaries map words to their respective indices and vice versa. Additionally, it assigns index 0 to the special token "Unk" to handle unknown words. These mappings are fundamental for various natural language processing tasks, facilitating efficient word representation and manipulation.

Data Generator for Modelling:

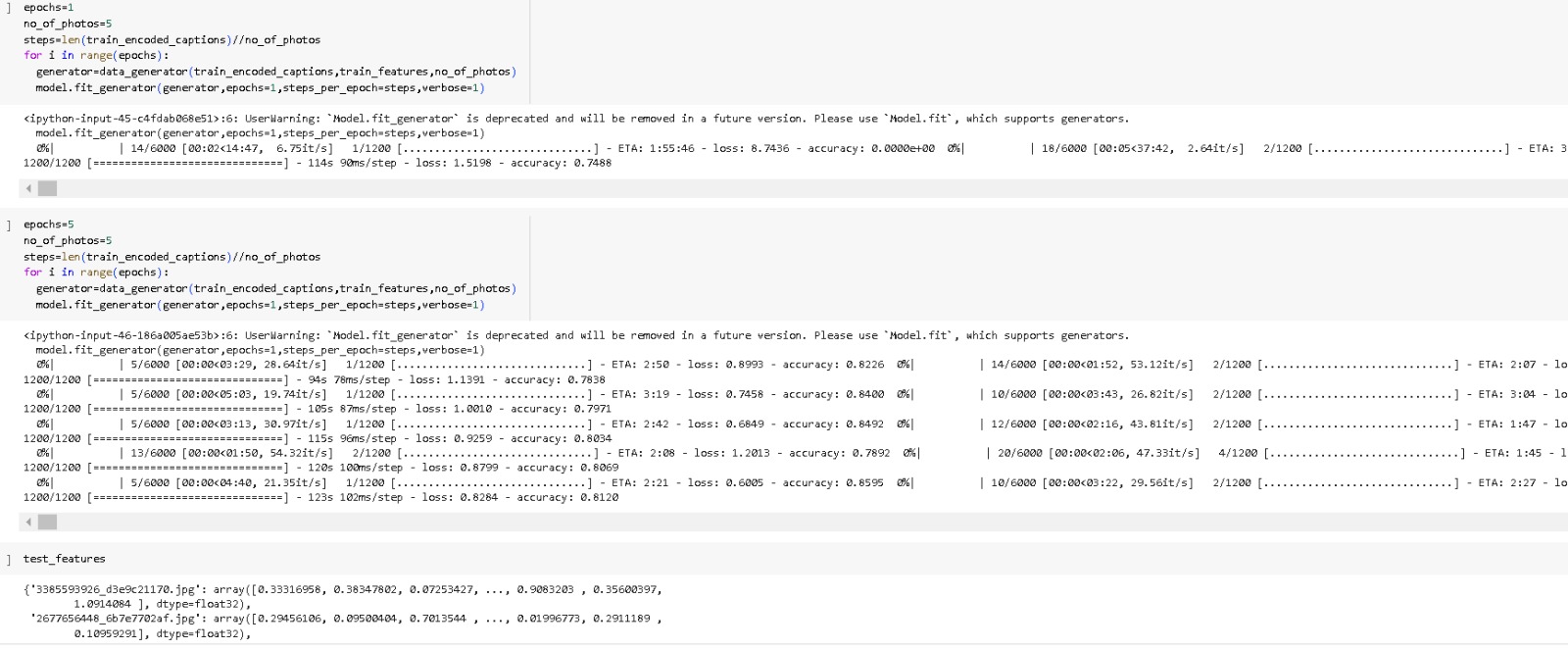


The provided code implements a data\_generator function for image captioning. It processes encoded captions, creates sequences of varying lengths for each caption, and converts them along with corresponding next words into training data suitable for image captioning models.

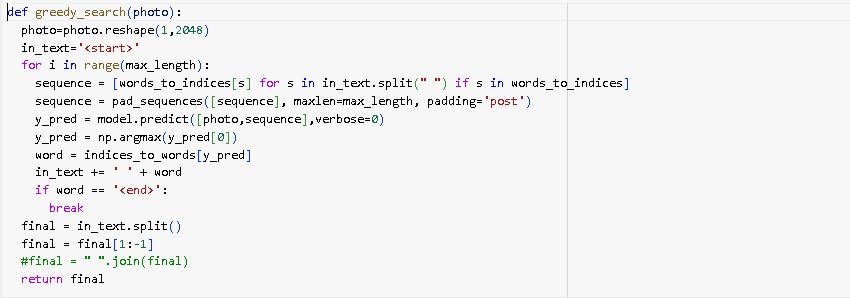


The code defines a model that takes two inputs, likely an image and a caption, and outputs a probability distribution over words. This suggests the model is being trained for a machine learning task such as image captioning. The dropout layers help prevent the model from overfitting on the training data

**Training:**

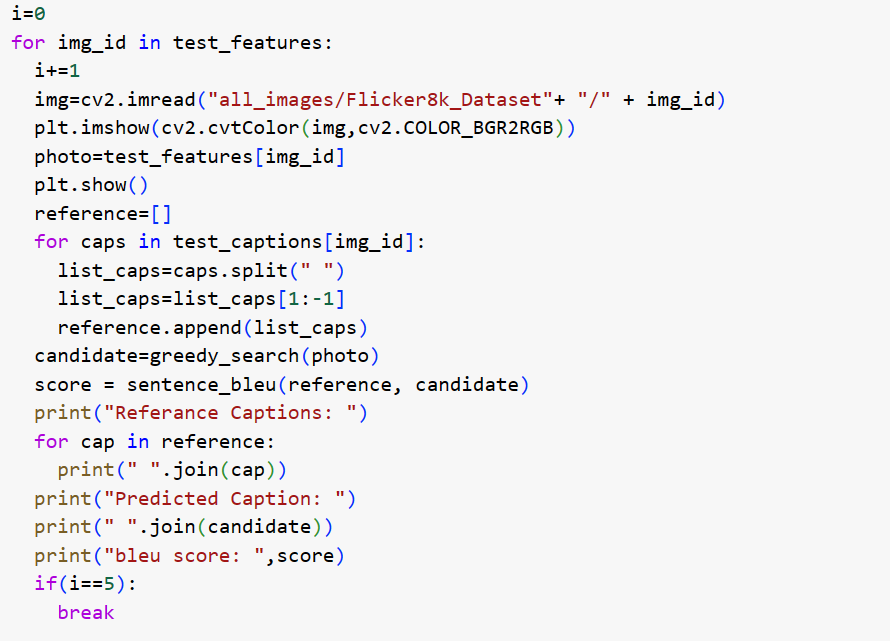


The is a custom data generator function for training an image captioning model. It iterates through encoded captions, generates sequences of different lengths from each caption, and converts these sequences and the corresponding next words into one-hot encoded vectors for training

Greedy Search function:

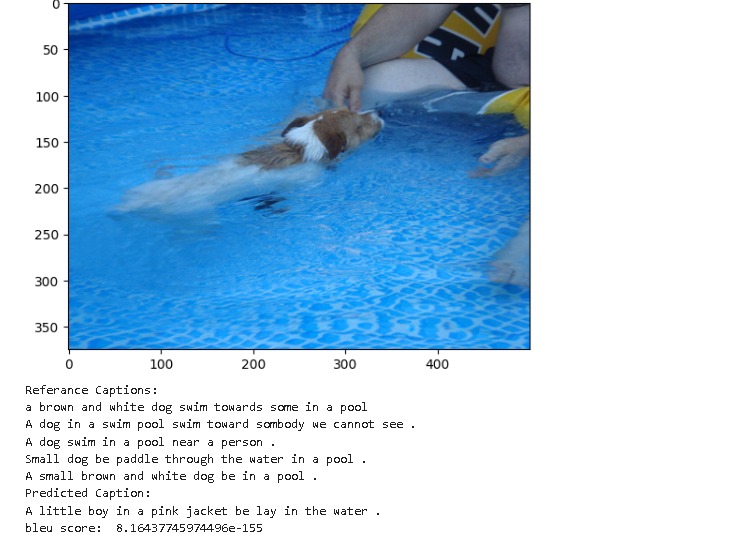
This function appears to be designed for image captioning, where it generates captions one word at a time based on a given image. It starts with an empty caption and iteratively predicts the next word to add to the caption based on the image and the current caption. The process stops when it predicts an end token ([END]), indicating the caption is complete.

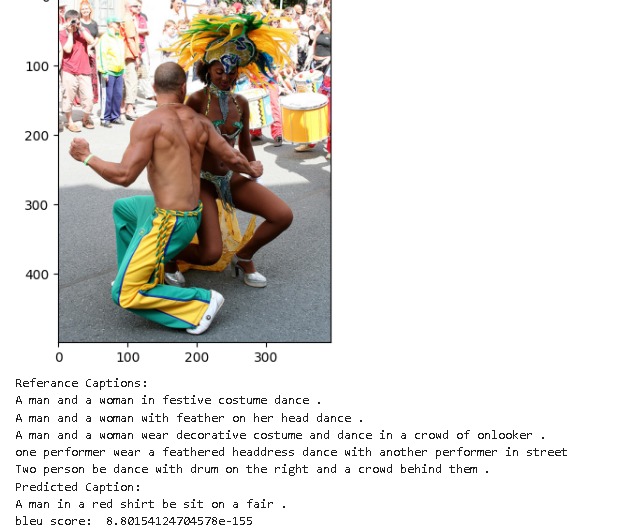
**Predicting Captions on Test Set using Greedy Search:**



This codes iterates through test features, which could be image features, and corresponding captions. For each caption, it generates a list of words by splitting the caption and removing the start and end tokens. It then calculates a BLEU score, which is a metric for evaluating the quality of image captions, to compare the generated caption to the reference captions

Output:







**Calculating Average Bleu Score on Test Set using Greedy Search**



The code calculates the BLEU score on greedy search, a common algorithm for image captioning. It iterates through test image IDs and their corresponding captions. For each image, it generates a caption using greedy search and calculates the BLEU score between the generated caption and the reference captions. The BLEU score measures how similar the generated caption is to the human-written captions

**Beam Search Function**



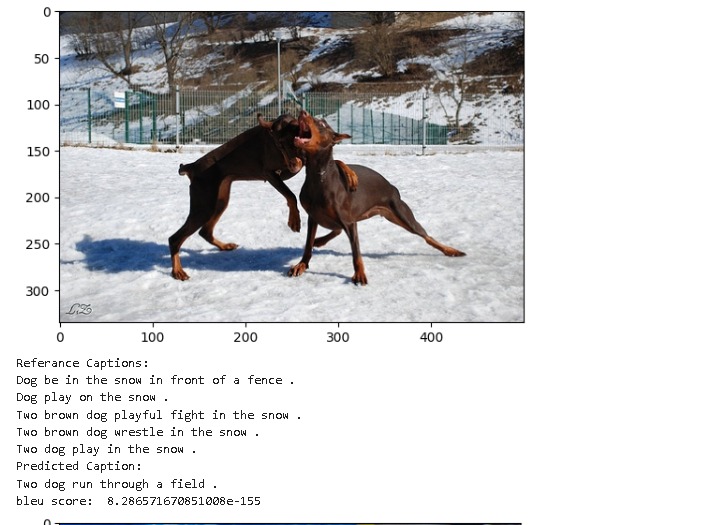
The code implements a beam search function for image captioning. It takes an image and a beam width as input. The beam width specifies the number of candidate captions to consider at each step. The function starts with a list of captions containing only the start token. Then, it iteratively predicts the next word for each candidate caption in the beam and expands the list of captions. At each step, it keeps the top k (beam width) most likely captions based on a scoring function. The process continues until a caption ends with an end token or reaches a maximum length. The function returns the highest-scoring caption

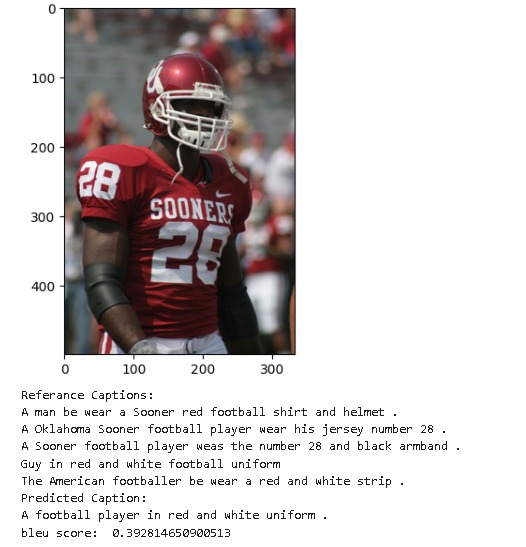
**Predicting Captions on Test Set using Beam Search with k=3**

****

The code snippet shows a Python function named data\_ generator that appears to be designed to prepare data for training an image captioning model. It iterates through encoded captions, breaks them into sequences of varying lengths, and converts these sequences and their corresponding next words into one-hot encoded vectors. This process is commonly used to train machine learning models for tasks like image captioning.

Output:





**Conclusion:**

The implemented image captioning model demonstrates the ability to generate descriptive captions for images. Further experimentation with different architectures and techniques could potentially enhance the model's performance further.

However, despite the successes observed, several challenges and areas for improvement have been identified. One such challenge lies in the generation of captions that exhibit a deeper understanding of image content and context. While LSTM models excel at generating grammatically correct captions, achieving a deeper semantic understanding remains an ongoing research endeavor.

Additionally, the issue of dataset bias and generalization warrants attention. Models trained solely on the Flickr dataset may exhibit biases present within the data, limiting their applicability to diverse contexts and domains. Addressing this challenge necessitates the exploration of techniques such as data augmentation, domain adaptation, and the incorporation of additional datasets.

Looking ahead, the field of image captioning holds great promise, with opportunities for further advancements in model architectures, training methodologies, and evaluation metrics. Future research efforts may focus on leveraging multimodal architectures that integrate both visual and textual modalities, as well as exploring novel techniques for enhancing the interpretability and diversity of generated captions.