Image Captioning

A computer vision and NLP project

Done by: William Stefan (2040797), Don Nimesh (2060579), Nikil Jayasooriya (2056118), Favour Akubuo(2014972)

Abstract

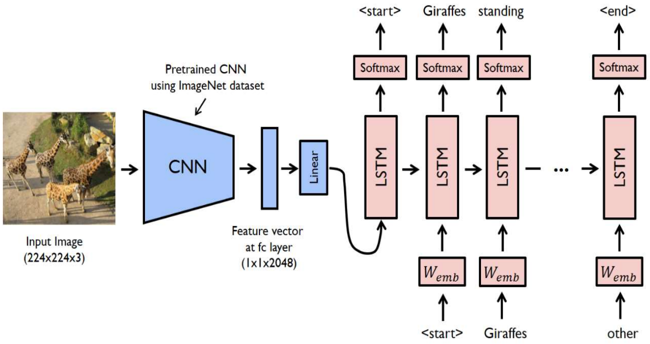
Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. Inspired by advancements in Artificial Intelligence, our objective was to develop a system that can generate descriptive captions for images, thereby bridging the gap between visual content and natural language processing. While existing solutions often relied on simple template-based approaches, we aimed for a more sophisticated and accurate method.

Introduction

Automatically generating descriptive captions for images using natural language is a complex task that has significant potential, such as aiding visually impaired individuals in understanding web content. This task goes beyond traditional image classification or object recognition by requiring the model to not only identify objects but also describe their relationships, attributes, and activities in coherent sentences. This project leverages a combination of deep convolutional neural networks (CNNs) for visual feature extraction and recurrent neural networks (RNNs) for language generation to create an end-to-end image captioning system. By training the model to maximize the likelihood of generating accurate captions for given images, we aim to enhance the practical utility and performance of image captioning.

Related Work

The task of generating natural language descriptions from images has evolved significantly, leveraging advances in both object recognition and natural language processing. Early approaches, such as those by Farhadi et al. (2010) and Kulkarni et al. (2011), used template-based methods to convert detected objects and their relationships into text. These systems, while pioneering, were heavily hand-designed and limited in flexibility. Recent advancements have seen the integration of deep learning models, notably the combination of convolutional neural networks (CNNs) for image feature extraction and recurrent neural networks (RNNs) for sequence generation, as demonstrated by Vinyals et al. (2015) and Xu et al. (2016). These end-to-end models have significantly improved the accuracy and coherence of generated captions. Additionally, approaches like Karpathy and Fei-Fei (2015) utilized neural networks to co-embed images and text, enhancing the ability to generate novel descriptions by leveraging multimodal embeddings. Our work builds on these foundations by utilizing VGG16 for feature extraction and an LSTM-based RNN for caption generation, aiming to further improve caption quality and relevance.



Methodology

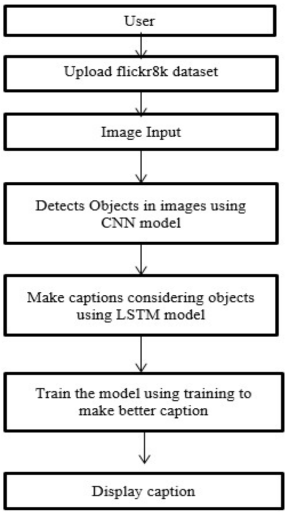
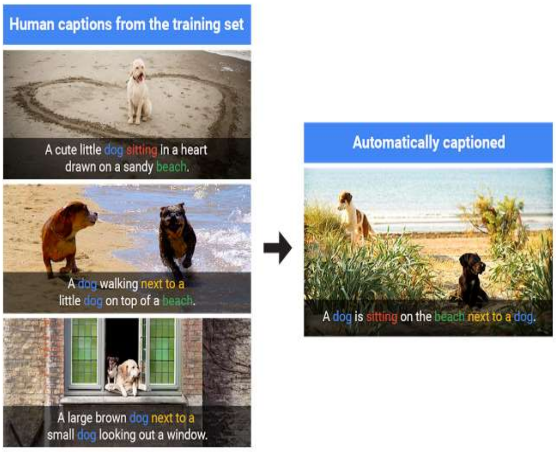

User
Upload flickr8k dataset
Image Input
Detects Objects in images using CNN model
Make captions considering objects using LSTM model
Train the model using training to make better caption
Display caption

Figure: Flow Chart


Human captions from the training set
A cute little dog sitting in a heart drawn on a sandy beach.
A dog walking next to a little dog on top of a beach.
A large brown dog next to a small dog looking out a window.
Automatically captioned
A dog is sitting on the beach next to a dog.

Code Overview

Libraries Used: TensorFlow, Keras, Flask, PIL, numpy, matplotlib, tqdm.

Model Architecture

Image Feature Extraction: A pre-trained model called VGG16 which is a convolutional neural network architecture that is pre-trained on ImageNet dataset was used to extract the image features. The model architecture is modified to exclude the final classification layer, retaining the second-to-last layer which provides a 4096-dimensional feature vector.

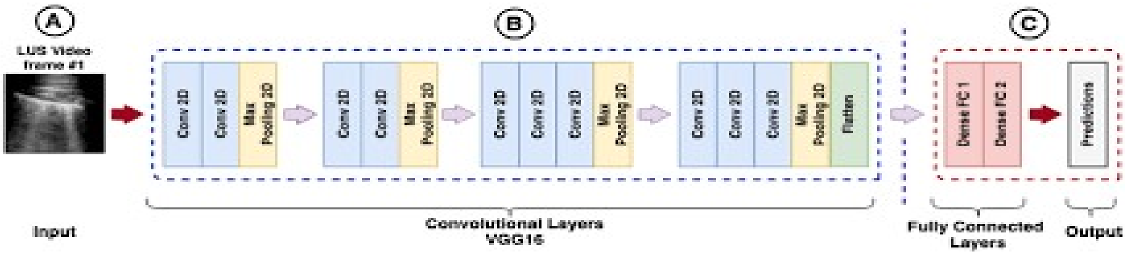


Image preprocessing

- Input image is resized to 224x224 pixels.
- Image is preprocessed to match the input format required by VGG16.

Text Preprocessing

- **Captions Cleaning:** Captions are converted to lowercase, non-alphabet characters are removed, extra spaces are trimmed, and start and end tokens ('startseq' and 'endseq') are added.
- **Tokenization:** The cleaned captions are tokenized using Keras's Tokenizer to convert words into integers.
- **Padding:** Captions are padded to ensure they all have the same length for input into the LSTM.

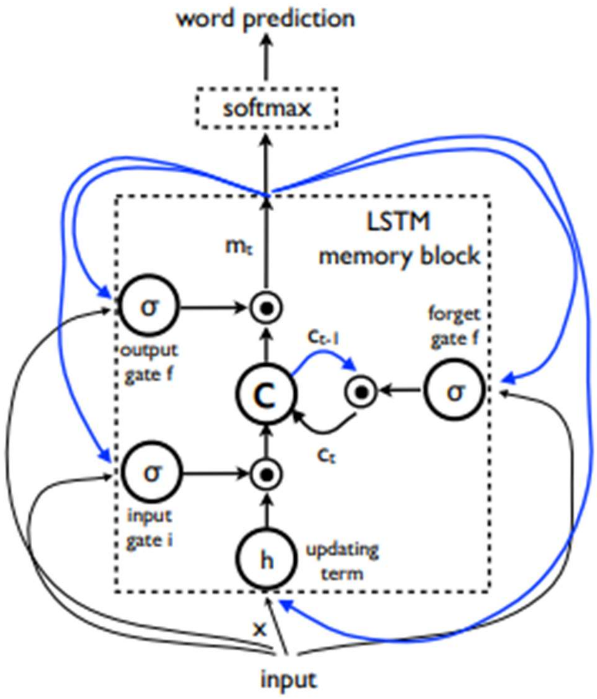
Model Layers:

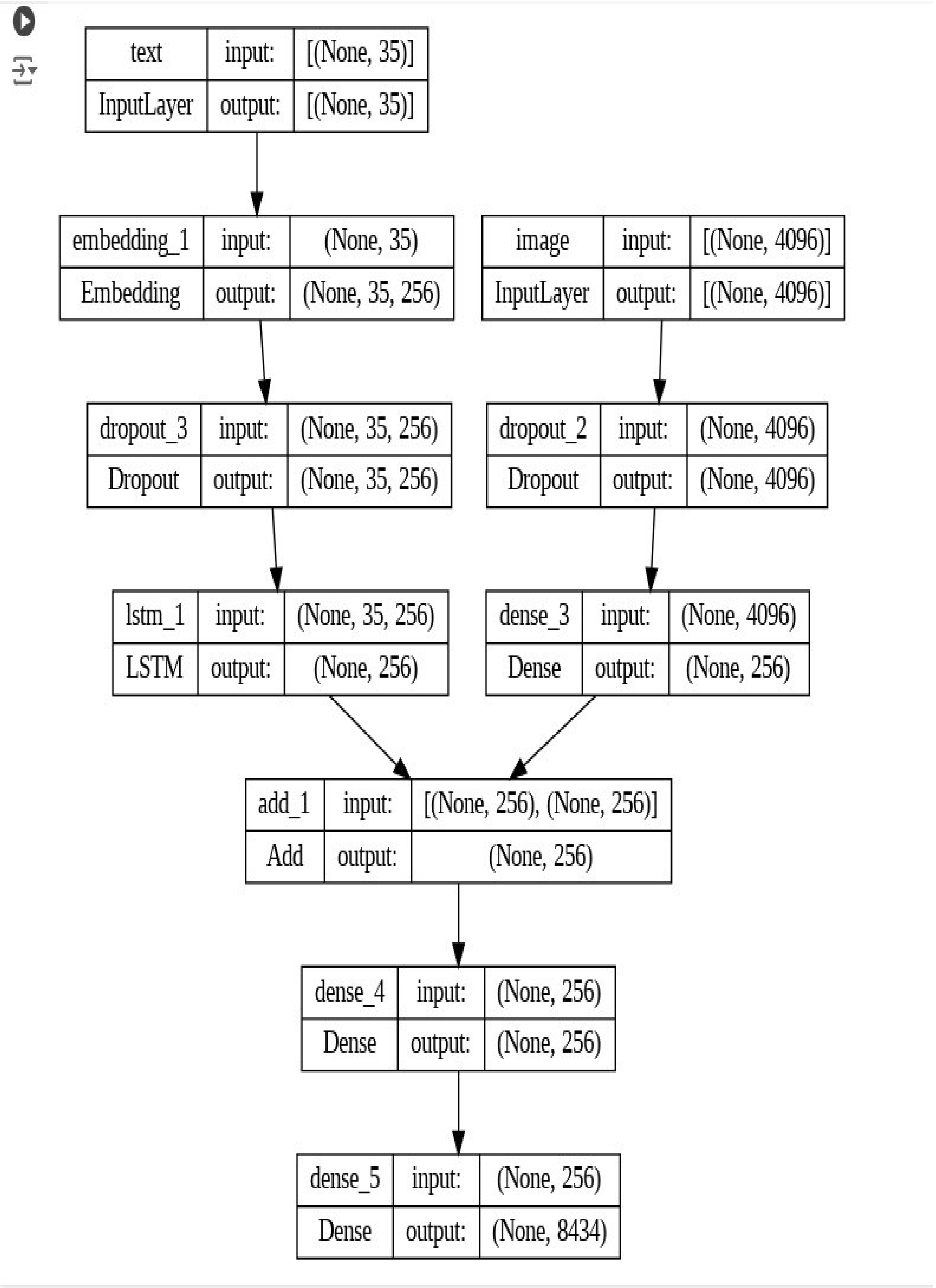
- ### Image Feature Processing:

  - Input Layer (Image): Takes the 4096-dimensional feature vector.
  - Dropout Layer: Applies dropout with a rate of 0.4 to reduce overfitting.
  - Dense Layer: Fully connected layer with 256 units and ReLU activation.

- **Text Processing:**
  - Input Layer (Text): Takes the padded integer-encoded captions.
  - Embedding Layer: Embeds the input text sequence into a dense vector of fixed size (256 dimensions).
  - Dropout Layer: Applies dropout with a rate of 0.4.
  - LSTM Layer: Processes the sequence and outputs a fixed-length vector of 256 dimensions.

- **Decoder:**
  - Add Layer: Combines the outputs from the image processing and text processing pathways.
  - Dense Layer: Fully connected layer with 256 units and ReLU activation.
  - Output Layer: Fully connected layer with SoftMax activation, producing a probability distribution over the vocabulary for the next word prediction.




text | input: | [(None, 35)]
InputLayer | output: | [(None, 35)]
embedding_1 | input: | (None, 35)
Embedding | output: | (None, 35, 256)
image | input: | [(None, 4096)]
InputLayer | output: | [(None, 4096)]
dropout_3 | input: | (None, 35, 256)
Dropout | output: | (None, 35, 256)
dropout_2 | input: | (None, 4096)
Dropout | output: | (None, 4096)
lstm_1 | input: | (None, 35, 256)
LSTM | output: | (None, 256)
dense_3 | input: | (None, 4096)
Dense | output: | (None, 256)
add_1 | input: | [(None, 256), (None, 256)]
Add | output: | (None, 256)
dense_4 | input: | (None, 256)
Dense | output: | (None, 256)
dense_5 | input: | (None, 256)
Dense | output: | (None, 8434)

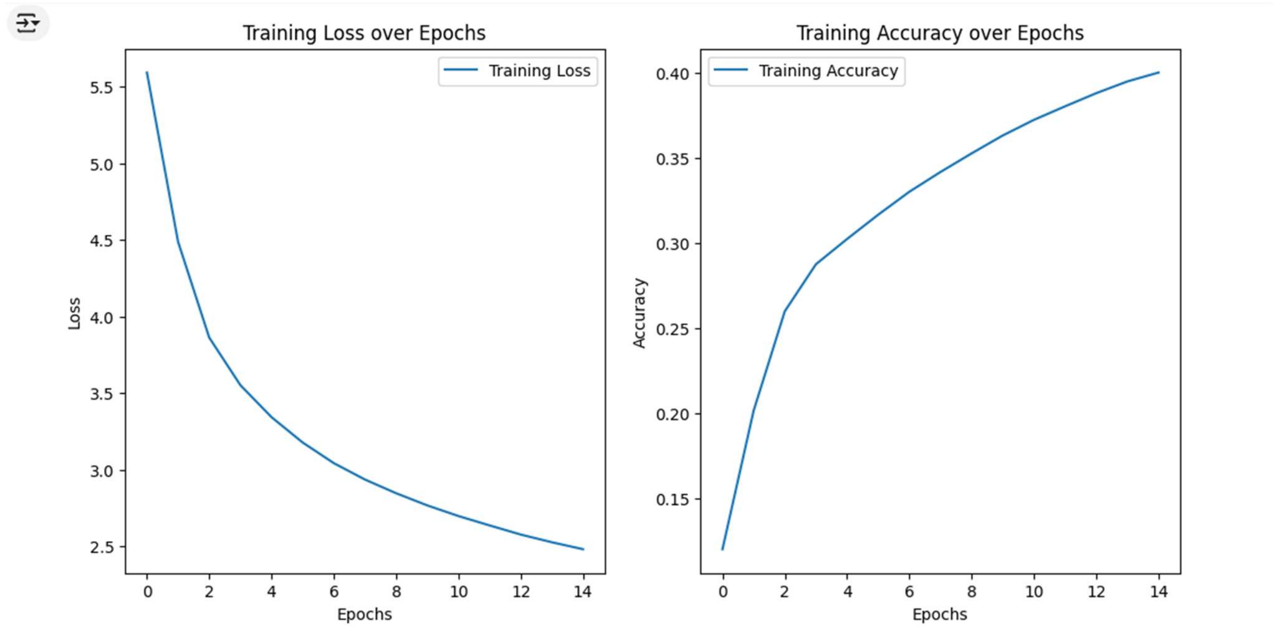
Training and Hyperparameters

Epoch = 15

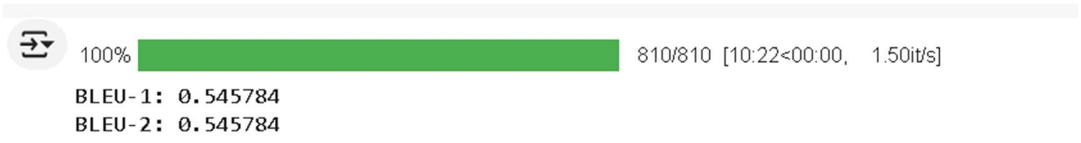
Batch size = 64

Results

Evaluation Metrics



BLEU Score: The BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of machine-generated text by comparing it to one or more reference texts, focusing on the precision of n-gram matches.



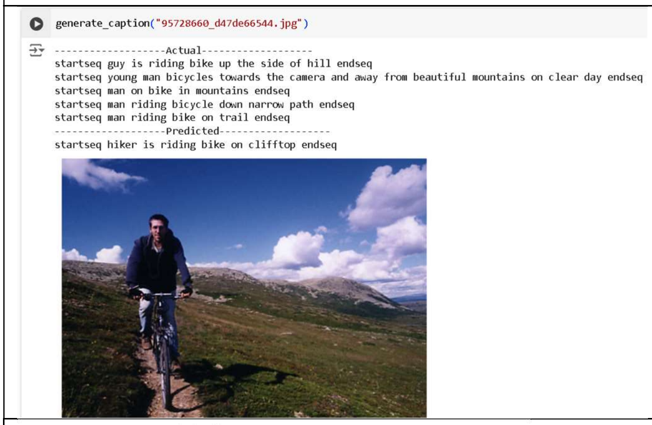
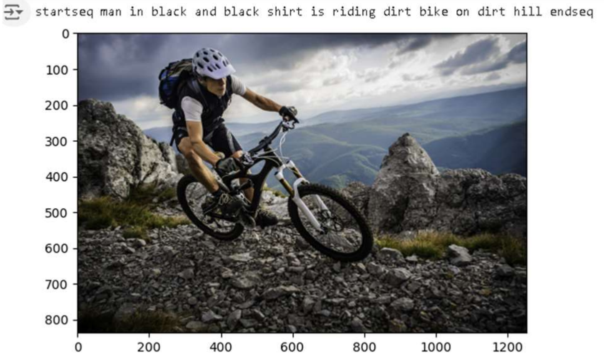
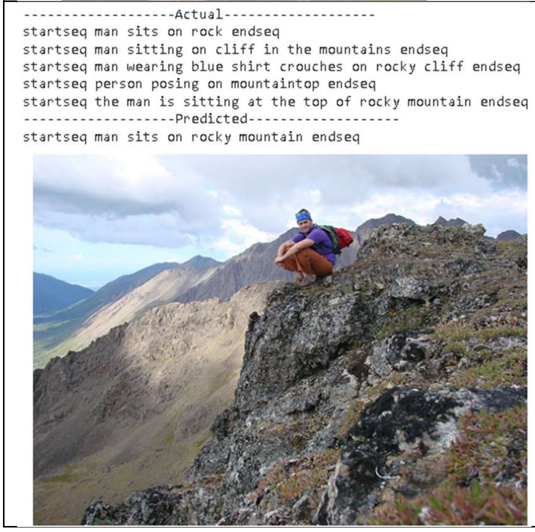
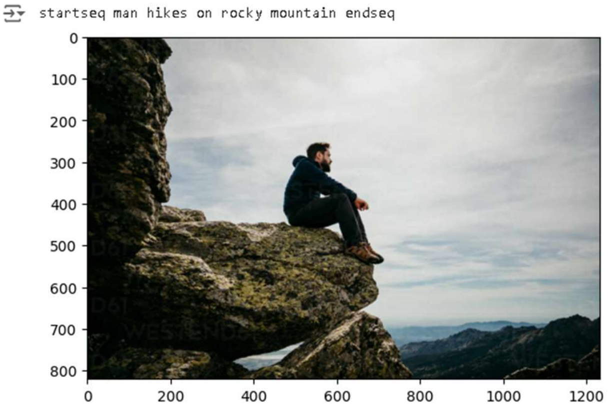
```
100% 810/810 [10:22<00:00,  1.50it/s]
BLEU-1: 0.545784
BLEU-2: 0.545784
```

Observations

Training Accuracy: The model successfully learned to generate captions based on the training dataset, showing progressive improvement in loss over the training epochs.

Accuracy in Common Scenarios: The model performed well on images depicting common scenarios such as people, animals, and indoor scenes.

Qualitative Analysis: Visual inspection of generated captions for a diverse set of images

Test Data	New Data
 generate_caption("95728660_d47de66544.jpg") ------------------Actual------------------ startseq guy is riding bike up the side of hill endseq startseq young man bicycles towards the camera and away from beautiful mountains on clear day endseq startseq man on bike in mountains endseq startseq man riding bicycle down narrow path endseq startseq man riding bike on trail endseq ------------------Predicted------------------ startseq hiker is riding bike on clifftop endseq	 startseq man in black and black shirt is riding dirt bike on dirt hill endseq 0 100 200 300 400 500 600 700 800 0 200 400 600 800 1000 1200
 -------------------Actual------------------- startseq man sits on rock endseq startseq man sitting on cliff in the mountains endseq startseq man wearing blue shirt crouches on rocky cliff endseq startseq person posing on mountaintop endseq startseq the man is sitting at the top of rocky mountain endseq -------------------Predicted------------------- startseq man sits on rocky mountain endseq	 startseq man hikes on rocky mountain endseq 0 100 200 300 400 500 600 700 800 0 200 400 600 800 1000 1200

Challenges and Solutions

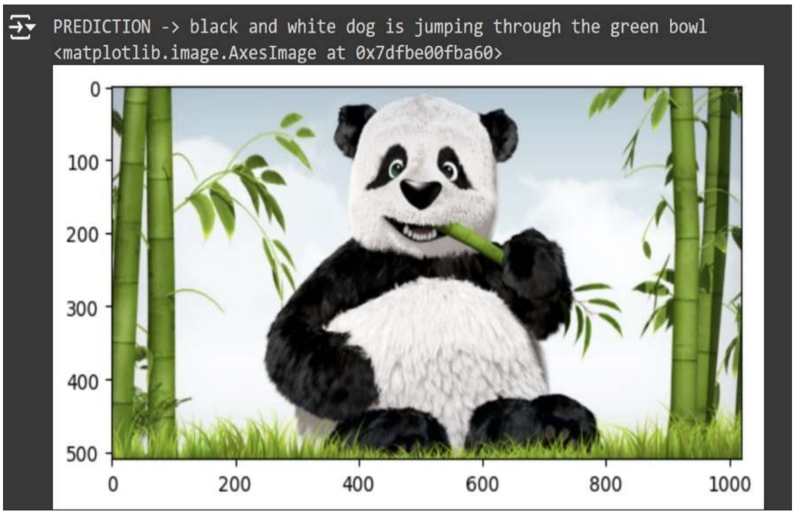
Handling Large Datasets: Utilized Google Colab and Google Drive for storage and computation.

Balancing Accuracy and Performance: Experimented with different model configurations and hyperparameters.

Limitations

Domain-Specific Bias: The model performs poorly on images from domains not covered in the training dataset (e.g., medical images, underwater scenes), limiting its applicability in different contexts.

Limited Dataset Size: The size of the dataset used is not large enough to capture the full variability of image content and language structure.
Class Bias: The model is biased towards the classes it was trained in. For instance, when presented with an image of a panda, the model inaccurately generates a caption such as "a black and white dog," due to its training on a dataset where dogs are more common.



```
PREDICTION -> black and white dog is jumping through the green bowl
<matplotlib.image.AxesImage at 0x7dfbe00fba60>
```

Conclusion and Future work

This project successfully developed an image captioning model capable of generating descriptive captions for various images.

Future work includes ,

- **Attention Layers:** Integrate attention mechanisms to allow the model to focus on different parts of the image while generating each word, improving the relevance of the generated captions.

- **Transformer Models:** Utilize transformer architectures, which have built-in attention mechanisms, for both image and text processing.
- **Data Augmentation:** Apply data augmentation techniques to increase the variety of training images, helping the model generalize better.
- **Real-time Applications:**
    1. Edge Computing: Optimize models for deployment on edge devices to enable real-time captioning for applications like assistive technology for the visually impaired.
    2. Mobile Deployment: Develop lightweight models suitable for mobile devices, making image captioning more accessible.

References

1. Reference Articles and Papers

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and Tell: A Neural Image Caption Generator. Available at: https://arxiv.org/abs/1411.4555

2. Tutorials and Blog Posts

Brownlee, J., 2019. How to Develop an Image Captioning Model with Keras. Machine Learning Mastery. Available at: https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

3. Books

Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd edn. Sebastopol, CA: O'Reilly Media. https://powerunit-ju.com/wp-content/uploads/2021/04/Aurelien-Geron-Hands-On-Machine-Learning-with-Scikit-Learn-

Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd edn. Sebastopol, CA: O'Reilly Media. https://powerunit-ju.com/wp-content/uploads/2021/04/Aurelien-Geron-Hands-On-Machine-Learning-with-Scikit-Learn-

4. Datasets

Adityajn105, 2020. Flickr8k Dataset. Kaggle. Available at: https://www.kaggle.com/datasets/adityajn105/flickr8k