A Study on Image Caption Generator

Jayvardhan Singh, [1] Kaustubh Mishra [2]

New York University, [1] New York University [2]

js12919@nyu.edu, [1] km5939@nyu.edu, [2]

GitHub Repository

Abstract

The exponential surge in the volume of digital images online underscores the critical need for automated tools capable of distilling pertinent information and context from these visual assets. A significant application that meets this need is image caption generation - a field of study in which artificial intelligence systems are trained to generate natural language descriptions for images. This exciting domain has captured the attention of researchers worldwide due to its broad range of applications. Image Caption Generation can powerfully assist visually impaired individuals by translating image content into verbal descriptions, enhance the accuracy of search engine results by providing contextual information, and augment user experience on social media platforms by automatically generating relevant image captions. This project delves into the implementation and fine-tuning of such a system, exploring the intricate blend of image analysis and natural language processing.

Introduction

The Image Caption Generator employing Convolutional Neural Networks (CNN) is a project developed in Python that leverages deep learning techniques to autonomously generate captions for images. This project integrates a Convolutional Neural Network capable of extracting features from an image with a Natural Language Processing (NLP) model, designed to produce cohesive sentences describing the image's content.

Two widely recognized image captioning datasets, Flickr8k and MSCOCO, are utilized to train both the CNN and NLP models. The images in these datasets are subjected to preprocessing to standardize their size and color, while the associated textual data is converted into a format suitable for the NLP model.

The crux of the project lies in the training phase where the CNN learns to extract salient features from the images, and concurrently, the NLP model hones its ability to construct descriptive captions based on these extracted features. As a result, the integrated model can generate accurate and detailed captions for a variety of images.

The potential applications of this project are manifold, including aiding visually impaired individuals in interpreting images or offering automated caption generation for social media posts.

Problem Statement

Generating accurate and insightful image captions is a multifaceted challenge at the intersection of computer vision and natural language processing. This task necessitates not only the identification of visual elements within an image but also their translation into a meaningful, grammatically correct sentence. Historically, humans primarily undertook this function. However, the advent of deep learning technologies has introduced automation capabilities into this process. Nonetheless, to ensure the generation of accurate and coherent captions, it is essential to leverage a sophisticated neural network architecture that can effectively learn from large datasets comprising images and their associated captions.

The central objective of this project is to experiment with various combinations of pre-trained Convolutional Neural Network (CNN) models (specifically VGG16 and RESNET50) and Recurrent Neural Network (RNN) models (specifically LSTM and GRU) to generate captions for images. We aim to assess the effectiveness of these combinations based on BLEU scores, using different datasets such as Flickr8k and MSCOCO. The project involves several stages including the preprocessing of image data, training a CNN model to extract features from these images, and subsequently training an RNN model to construct captions from the features extracted by the CNN.

Literature Survey

The use of pre-trained Convolutional Neural Networks (CNNs) coupled with a Decoder Recurrent Neural Network

(RNN) for image captioning represents an important line of work within the field of computer vision and natural language processing. This approach integrates the image representation power of CNNs with the sequential data modeling capabilities of RNNs to generate descriptive sentences for a given image.

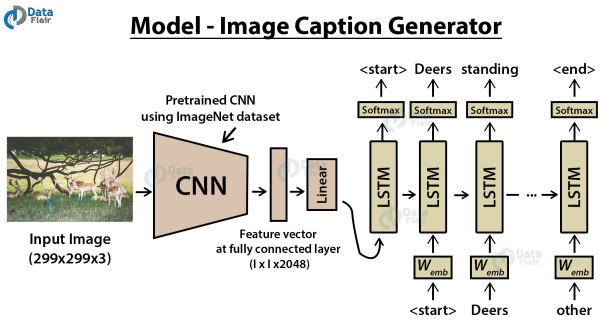
Below is a brief survey:

1. "Show and Tell: A Neural Image Caption Generator" by (Vinyals et al., 2015): This is one of the pioneering works in this field. The authors proposed an encoder-decoder framework where a CNN was used as an encoder to extract visual features from the images, and an RNN served as the decoder to generate corresponding text descriptions.

2. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" by (Xu et al., 2015): The authors introduced an attention mechanism into the image captioning task. Their model dynamically focuses on different parts of the image when generating each word in the caption, improving the relevance and accuracy of the generated captions.

3. "Visual Image Caption Generator Using Deep Learning" by Sharma et al. (2019): The authors propose an image caption generator that utilizes deep learning techniques. The paper contributes to the existing body of research on image captioning, which seeks to automate the process of generating textual descriptions for images.The authors' approach relies on the integration of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which have separately demonstrated considerable effectiveness in the realms of image understanding and sequence generation, respectively. This aligns with a significant trend in image captioning research, leveraging the power of CNNs for feature extraction from images and RNNs for subsequent generation of relevant captions.Their model employs the InceptionV3 architecture as the CNN component for encoding visual information, due to its performance and efficiency. This choice represents a common strategy in image captioning, namely, the use of pre-trained CNNs (often trained on ImageNet) to effectively extract transferable features.

Methodology



Source: DataFlair

Due to the absence of a dedicated GPU on our local machines, we elected to execute our work on Google Colab, taking advantage of the T4 GPU it provides. Our Image Caption Generator project adheres to a conventional deep learning methodology, which encompasses the subsequent steps:

A. Data Collection

We utilized 2 different datasets for our Image Caption Project, namely:

1. **Flickr8k**: The Flickr8k dataset, accessible through Kaggle, is a popular resource in the machine learning community, particularly for tasks related to computer vision and natural language processing. It consists of 8,000 images, each paired with five unique, human-annotated captions. These captions provide context and describe the content of the images, thereby transforming the dataset into over 40,000 unique image-caption pairs. The structure of the dataset is such that it can be used for training models on a task such as image captioning.

2. **MS-COCO:** The MS COCO (Microsoft Common Objects in Context) dataset, on the other hand, is available for download on the official website. It is a large-scale dataset for multiple computer vision tasks, including object detection, segmentation, and captioning. This dataset is pre-partitioned into training, validation, and testing sets, making it a robust resource for machine learning projects. To use this dataset, one can directly download the data from the MS COCO website, or clone the official COCOAPI GitHub repository to reference data from the URL. This API, created by the dataset's originators, allows for easy fetching of data, making it a convenient tool for deep learning projects.

B. Data Preparation

This phase entails the preparation of the dataset that will be input into the model. The process comprises the following steps:

1. **Pre-processing**: We examined the Flickr8k dataset during this preliminary step, partitioning it into training and testing datasets. The Flickr8k dataset comprises 8,091 images, each provided with five unique captions. This essentially translates into 40,455 combinations of images and captions. These 40,455 pairs were divided into the training and testing data with testing data consisting of 455 pairs. The MS COCO dataset, already partitioned into training, testing, and validation sets, is readily available on the official website. We employed the 2014 version of this dataset, which includes approximately 83,000 training images (14GB) and 41,000 validation and testing images. Due to practical constraints and the infeasibility of loading 14GB of data onto Colab for each new session or in response to server connection errors, we cloned the official COCOAPI Github repository. This allows us to access data from the URL. The COCO API, a publicly accessible repository created by the dataset's originators, facilitates the easy retrieval of data.

2. **Vocabulary Creation**: This step focuses on the creation of the model's "vocabulary". It involves generating tokenized training captions and establishing dictionaries to reference the numericalized tokens.

3. **Dataset Loader**: This phase involves the development of a dataset class that ingests the image and caption data. The process applies designated transformations to the image, including the adaptation of the image resolution to align with the input specifications of a particular pre-trained model and the conversion of captions into corresponding numerical values. Subsequent to this, we devised a data loader that systematically batches and prepares the data for efficient model input. This approach allows for a smooth and effective data feeding process into the model during training.

C. Feature Extraction

In the Image Caption Generator project, we utilized pre-trained CNN models to extract salient features from images. Specifically, we employed the VGG16 and Resnet50 models to carry out this feature extraction process.

The VGG16 model is a deep convolutional neural network principally designed for image classification. It comprises 16 layers, including convolutional layers, max pooling layers, and fully connected layers. It's noteworthy that the penultimate fully connected layer of VGG16 produces a 4096-dimensional vector.

Conversely, the Resnet50 model, a renowned deep convolutional neural network known for its groundbreaking architecture and superior performance in image classification tasks, operates slightly differently. The global average pooling layer that immediately follows the final convolutional block in Resnet50 generates a 2048-dimensional feature vector. [1]

We harnessed these pre-trained models to extract the aforementioned features from every image in our dataset. The resultant feature vectors from the final fully connected layer were then forwarded to a custom linear layer, specifically created with output features equivalent to the specified embed_size during layer creation.

The feature vectors obtained from this process were subsequently channeled into our RNN model for training. Notably, feature extraction plays a pivotal role in the Image Caption Generator project. This process allows for a significant reduction in the dimensionality of the images while preserving their key visual properties.

Finally, our NLP models - LSTM and GRU - were employed to generate captions that accurately depict the contents of the images. By extracting and focusing on the salient visual elements of the images, our model was able to generate meaningful and contextually relevant captions.

D. Model Training

Model training constitutes a crucial stage in the Image Caption Generator project. In this phase, Convolutional Neural Networks (CNN) are employed to extract distinctive features from images, while Natural Language Processing (NLP) models are trained to generate relevant captions for these images.

In this study, we utilized pre-trained models for the extraction of salient features from the images. Our choice of Recurrent Neural Networks (RNN) for this task included either networks equipped with Long Short-Term Memory (LSTM) cells or those with Gated Recurrent Units (GRU).

The training of the NLP model involved feeding input sequences into the model and subsequently comparing its output with the target captions for each image in the dataset. The model was optimized using the cross-entropy loss function, with the aim of minimizing the discrepancy between the model's output and the intended captions.

Our training data for the NLP model comprised sequences

of image feature vectors and their associated captions. To prevent overfitting, the dataset was partitioned into training and validation sets.

Throughout the training process, model updates were performed via backpropagation through time. This involved computing the gradients of the loss function with respect to the model parameters and making adjustments using gradient descent. Model performance was further optimized by fine-tuning hyperparameters, including the learning rate and batch size.

Adam optimizer: The Adam optimizer is a widely-used optimization algorithm in deep learning, known for its efficient handling of sparse gradients on noisy problems. It combines the benefits of two other extensions of stochastic gradient descent: AdaGrad, which works well with sparse gradients, and RMSProp, which works well in online and non-stationary settings. Adam uses adaptive learning rates and momentum, enabling it to converge rapidly and accurately.

Loss function was calculated using Cross Entropy. Cross-entropy loss, also known as log loss, is a popular loss function used in deep learning models, particularly for classification tasks. It measures the dissimilarity between the model's predicted probability distribution and the true distribution, with a lower value indicating better model performance.

E. Model Evaluation

In our training procedure, we employed the cross-entropy loss function along with the Adam and Stochastic Gradient Descent (SGD) optimizers with momentum. We harnessed the computational power of the T4 GPU on Google Colab, and each model was trained for 20 epochs with a batch size of 4 using the Flick8k dataset. Our initial experiments involved training the models for 10 and 15 epochs, but we subsequently increased the number of epochs to 20 for better performance. Our range of learning rates fell between 1e-4 and 3e-4, with 3e-4 identified as the most effective learning rate for the Adam optimizer.

We explored various combinations of pretrained architectures - VGG16 and RESNET50 - and RNN cells - LSTM and GRU. These were rigorously tested on the validation set. Our exploration also extended to various activation functions and optimization algorithms, ultimately finding that the ReLU activation function and Adam optimizer provided optimal results for our model.

Our chosen hyperparameters for the ***Flickr 8k*** dataset were as follows:

- Training batch size: 4
- Number of training epochs: 20
- Frequency threshold: 5
- Embedding size: 400
- Hidden layer size: 512
- Learning rate: [1e-4, 3e-4]
- Maximum length of target text: 20, 40
- Optimizer: Adam
- Loss function: Cross-entropy loss

For the ***MS COCO*** dataset, we utilized the following hyperparameters:

- Training batch size: 4
- Number of training epochs: 1
- Frequency threshold: 8
- Embedding size: 256
- Hidden layer size: 100
- Learning rate: [1e-4, 3e-4]
- Maximum length of target text: 20
- Optimizer: Adam
- Loss function: Cross-entropy loss

We meticulously logged the loss values at each step of the training process and visualized them using TensorBoard, the screenshots of which are included with the submitted repository. Lastly, we computed the BLEU scores (Bleu-1, Bleu-2, Bleu-3, Bleu-4) for each model using the testing data. This involved randomly selecting data from the test set and calculating the average score across ten images to report the average BLEU scores. Screenshots of these average scores are also included with the submitted code files.

Results

Table-1a: The following table gives the Average value of Bleu-Scores Generated by the models trained on Flickr 8k.

Model	Bleu-1	Bleu-2	Bleu-3	Bleu-4
Resnet+ LSTM	0.43356	0.22201	0.16845	0.08898
Resnet+	0.50620	0.25329	0.18151	0.10076

GRU				
VGG16 +LSTM	0.50437	0.24611	0.20526	0.12697
VGG16 +GRU	0.61549	0.35117	0.31879	0.23935

(Each model trained on 20 epochs and hyperparameters were kept constant.)

Table-1b: The following table gives the Average value of Bleu-Scores(10 images) generated by the models trained on MSCOCO

Model	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**
Resnet+ LSTM	0.19568	0.000	0.05918	0.02922
Resnet+ GRU	0.16413	0.000	0.05180	02507

(The models were trained for just 1 epoch.)

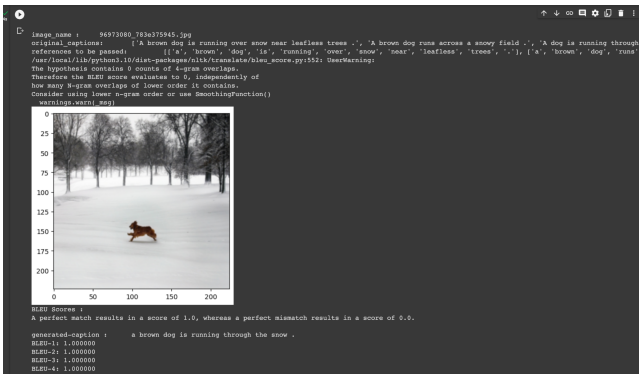
As we wanted to improve the scores of the lowest performing model Resnet+LSTM on the flickr 8k, we created another variation of the model by adding an attention layer to it. To be precise Bahdanau's Attention

Table 2- Comparison of the Upgraded Resnet+LSTM model and the original model by comparing the Average values of Bleu-Scores:

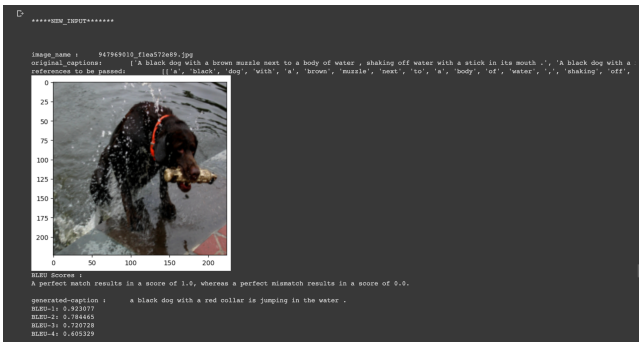
Model	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**
Resnet+ LSTM	0.43356	0.22201	0.16845	0.08898
Resnet+ LSTM + attention	0.63185	0.42221	0.30257	0.17838

Below are some of the generated images with captions using different models:

Img 1 - Image with caption generated by Resnet LSTM with Attention Layer



Img 2 - Image with caption generated by VGG 16 GRU model



Img 3 - Image generated by VGG 16 LSTM model

Observations

- Based on the results presented in Table-1a, it is clear that the Gated Recurrent Unit (GRU) outperforms the Long Short-Term Memory (LSTM) when the pre-trained model is held constant. We have chosen to omit Table-1b from this discussion due to the fact that it represents results from a single epoch, coupled with the occurrence of timeout errors during API calls for image testing.
- In the case of pre-trained models, VGG16 appears to deliver superior results compared to Resnet

models as inferred from Table-1a. Notably, the combination of Resnet and GRU yielded scores comparable to those of VGG16 and LSTM, suggesting an interplay between the type of pre-trained model and the choice of RNN cell.

- Our examination of Table-2 reveals that the integration of an attention layer, coupled with an extended training duration of 20 epochs (with all other hyperparameters remaining constant), led to a marked improvement in the average score of the previously lowest-performing model (Resnet+LSTM).

- While training models on FLickr8k, each epoch took about ~9 minutes to complete, hence each model took ~3 hrs to train on FLickr8k. Whereas due to the sheer amount of data and the API calls in between the model training on MSCOCO for a single epoch it took ~4.5 hrs. This does not include the time involved while retrieving validation and training data from the API. These values were obtained from the colab itself after the training cell finished executing.

- Lastly, it is important to note that while re-evaluating the BLEU scores may result in variations in the average values (since the average is calculated based on a sample of just ten images), the observed trend remains consistent and valid. These observations provide valuable insights into the performance of various model configurations and will guide our future work.

Conclusion

In the initial phase of our project, our objective was to construct a diverse assortment of Image Caption Generators. Our plan involved the utilization of various pre-trained models and the application of bidirectional LSTMs as an alternative to the traditional LSTM RNN model. However, we encountered resource constraints that limited our ability to execute the full range of planned experiments.

Despite these constraints, our project successfully accomplished the principal goal we set out to achieve: the development and evaluation of a suite of models for image caption generation. Through a methodical process, we constructed multiple model combinations and assessed their performance by means of the Bleu Score metric. This metric provided us with a quantitative means of comparing the quality of generated captions, allowing us to refine our models and draw conclusions about their effectiveness.

Looking ahead, our future work will further build upon these achievements. Our first objective is to extend the training duration of our models by increasing the number of epochs, with the aim of enhancing model performance.

Further, we intend to conduct a more comprehensive exploration of the MS COCO dataset, considering different combinations of model architectures tailored to this dataset. This will allow us to evaluate the robustness and versatility of our models across different data configurations.

References

[1] - "Show and Tell: A Neural Image Caption Generator"

Authors - [Oriol Vinyals. Alexander Toshev, Dumitru Erhan, Samy Bengio]
2015

[2] - "*Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*"

Authors - [Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio]
2016

[3] - "*Visual Image Caption Generator Using Deep Learning*"

Authors - [Grishma Sharma, Priyanka Kalena, Nishi Malde, Aromal Nair, Saurabh Parkar]
2019

[4] *Flickr 8k Dataset*: https://illinois.edu/fb/sec/1713398: The official request form. However as the site containing this dataset is taken down the dataset was downloaded from Kaggle.

[5] - MS Coco Official site