AI-Powered Image Captioning and Title Generation System (Report)

Asad Irfan

2022120

AI-351

Instructor: Muhammad Talha

ABSTRACT

This project is solely based on an Image captioning and title generation using AI solutions. The aim was to use an approach that would help to understand images and produce captions and titles which would reflect the image’s content. The project thus relies on deep learning methodologies, where the image features are extracted using Convolutional Neural Network (EfficientNetB0) while sequential predictive uses the Transformer model. These data (image vectors and captions) is feed to an encoder-decoder Transformer framework which contains multi-head attention layers and position encoding for caption generation. Considerable volume of images together with their captions are fed to the transformer based model ensuring that the system establishes the relationship between the image content and the natural language descriptions. For assessment, BLEU Scores were used. In the training phase, the model passes through multiple captions for a single image adjusting weights using the computed loss and accuracy. The outcomes of the project show that CNN can successfully be used for feature extraction and Encoder-Decoder Transformers are suitable for sequential prediction. Experimental results show that the system can provide meaningful and logical captions for images that are quite accurate to the content of what is captured in the image. The project can be utilized to solve some real-life problems, for instance, automatic image captioning, generation of content for the social media, and for helping visually impaired people. The future work will involve enhancing the proposed model’s output by creating more meaningful captions by training the model on larger datasets.

INTRODUCTION

Background
Image caption generation and title generation are complex tasks in both computer vision and natural language processing (NLP). These tasks include changing images into meaningful captions through different approaches of NLP and computer vision. The advancement in the deep learning has contributed a lot in the field of computer vision and NLP. The Flickr dataset is applied in this project which describes images by using captions. Image captioning and title generators can be used in content generation, image descriptions for visually impaired people and hashtags and captions in social networks.

Problem Statement
The proposed idea of this project is to build an image captioning model, which would produce useful descriptions of images. The challenge is to develop a model that captures the semantics of an image. A model which can understand the context of the image and can describe the scenario in that image in a meaningful contextually appropriate language.

Objective
The aim is to obtain a state-of-the-art deep learning model which can predict an image caption and title using a CNN feature extraction layer and a Transformer based encoder-decoder structure for enhancing the performance of the model.

Scope
In this project, source captions are derived from CNNs and Transformers which are trained with images and their corresponding captions. Some of its limitations are as follows: first, one must rely on the quality of the received dataset, second, the model is trained to work with images of the type used in the project and may exhibit unexpected behavior when presented with different images.

Motivation
The project's objectives include helping the visually impaired by generating descriptions for images, increasing the relevance of the posted information, and optimizing creative work in sectors such as social networks and marketing etc.

Literature Review

[1] In this paper, the RNN Encoder-Decoder framework was introduced by Cho et al . This architecture forms the base for sequence-to-sequence models which exist in applications such as image captioning by relating input image features to textual descriptions.

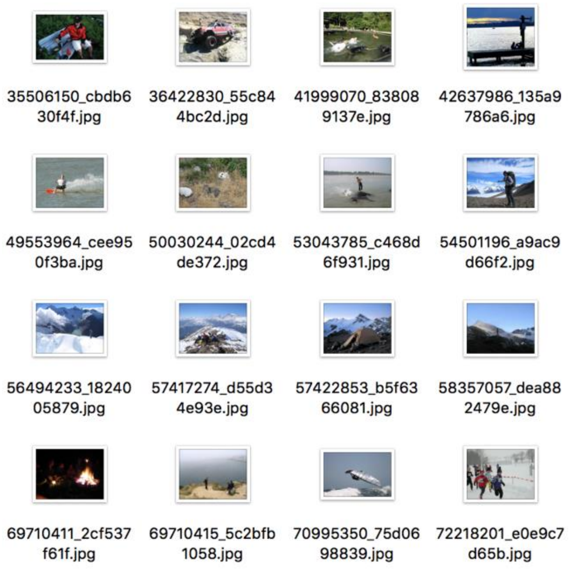
[2] Deep Visual-Semantic Alignment for Generating Image Descriptions by Karpathy and Li in 2015 describes a model uses both the CNN feature maps with the RNN for caption generation to enhance image descriptions and utilizes the corresponding visual-semantic alignment.

[3] Explaining Images with Multimodal Recurrent Neural Networks These methods were proposed by Mao et al. in 2014 and aimed at the application of the multimodal RNN for image captioning for visual and textual inputs to improve the quality of generated captions as the model can produce contextually accurate descriptions by interacting between two inputs.

[4] The authors Ranzato, Marc'Aurelio, et al. present Sequence level training for RNNs and the idea was to enhance the quality of the captions generated by directly minimizing the error at the sequence level rather than the token level and as a result it would improve the coherency of the captions generated for the images.

Data-Set Description

DATASET LINK: Flickr Dataset Link



Flickr dataset was used for training the transformer architecture which consists of 30000+ images in jpg format and their corresponding captions. The captions were in a csv format which consists of three columns one for the image, second for the comment and third for the caption for the image.

METHODOLOGY

Implementation

The main programming language used is python. Tensor-Flow/keras environment was used during the implementation of the project. The overall Project consist of following milestones

- Pre-processing
- Model Building and Training
- Title Generation
- Results and Evaluation of the model

PRE-PROCESSING

1. Loading Data and Preprocessing Captions
The initial process involves loading images and their respective captions from the Flickr dataset which will be used to train the model for a captioning and title generation task. The captions is in a CSV file format. This function takes the CSV described above and reads the necessary data – particulars of images and their captions and

stores them in the dictionary. The captions are then split into words and preprocessed which involves removing extra-spaces, non-alphanumeric characters and adding “start” and “end” special token to tell the start and the end of the captions. Any image having a caption which is either too short, below 4 words, or lengthy, above a certain number of words are rejected.

2. Train, Validation, and Test Split
Once the data is loaded and cleaned, it is split into three sets: For the training and testing, data has been splitted into three sets of training, validation, and test. This is usually done to evaluate the performance of the model and to check whether the model is doing its work properly or not. The data is shuffled and then splitted into training data (for training the model), testing data (for testing the model) and validation (for tuning the model hyper-parameters). Data is splitted into 20 percent for validation and 2 percent for the testing.

3. Text Vectorization
Text vectorization is an essential process that aims to transform human understandable captions in to a format which can be understood by the transformer model. A Text Vectorization layer is used to turn captions into “int” sequences where every word in the caption is remapped to an index in a vocabulary. The captions are first normalized (all letters are put in lower case and special characters are eliminated). The vectorization process adapts to the vocabulary size and sequence length (vocabulary size is 13000 and sequence length is 24), creating a consistent representation of all captions.

4. Image Augmentation
To enhance the generality of the proposed model, the images are augmented. This includes:

- Random Horizontal Flipping
- Rotation
- Random Contrast Adjustment

These augmentations assist the model to create distinct representations of the object of interest under different conditions thus enhancing the general performance and balance of the model.

5. Word Frequency Analysis
Analysis of the captions is carried out to determine the count of each word mean`s frequency of each word in the captions. Function for processing the captions is used in order to remove words such as “a”, “<start>”, and “<end>”, and then to find the frequency of the rest of the words. For most frequent words are represented with color codes and the intensity of that color represent the frequency of that word in the captions. This enables us to understand which words are more important depending upon their frequency.

6. Image Preprocessing
Every image which makes up the dataset is loaded and examined. The image is then translated into a tensor of three RGB channels, reshaped to a 299x299 dimension and float32 data type to feed in the next part to CNN based model which will be extracting features from that image which is used to focus only on the meaningful features of the image.

7. Creating the Dataset
The processed image paths and their titles are used to create a Tensor Flow dataset with use of the processed captions. This dataset is then shuffled and a mapping function is used on both the images and captions before preprocessing them. The dataset is then batched and prefetched for the efficient management of the input pipeline and the batches are of the defined batch size which is 32. This is done for both the training dataset, and the validation dataset for the trained model.

Model Building and Training

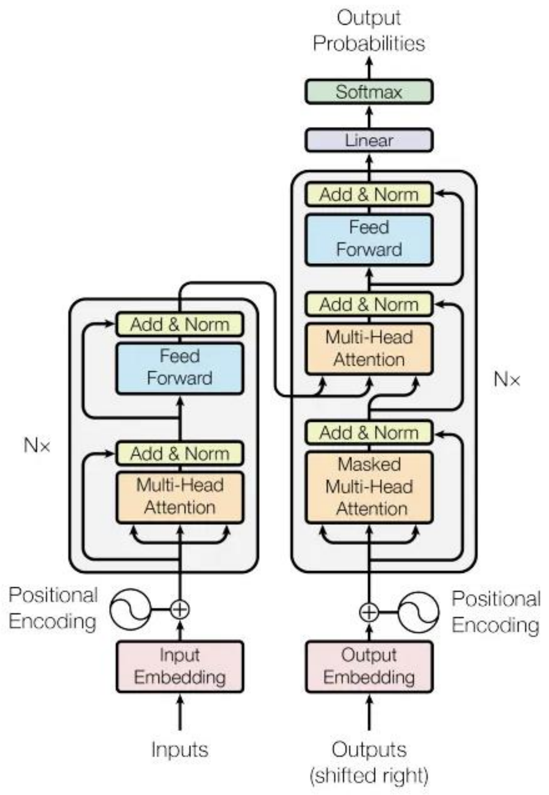


Figure.1 Transformer Architecture

1. Extracting features using CNN model.

CNN model (EfficientNetB0), a pre-trained model is used for extracting the features from the images in our dataset. The CNN model is a per-trained architecture which is trained on Image Net. Top level layers of the model are removed for concentrating solely on obtaining useful feature vectors from the given image. Since only trained weights of the EfficientNetB0 layers are to be kept, the layers of the model are freezed to prevent weights from changing during the training. The output of the cnn model is a 2D tensor which focuses on the relatively key characteristics of the input images. This process increases the model ability to extract the significant features from the data by decreasing the computational overhead.

2. Encoder

A custom encoder layer is used to provide self-attention of an input sequence along with dense transformations. It incorporates attention heads to find dependency structures in the given sequence by computing attended scores between the input features. The proposal of dense layers combined with ReLU activation improves the encoder's capability of learning higher level features. Layer normalization is used multiple times to make the training process stable and keep the features properly scaled. This certain encoder structure is capable to process and extract meaningful features from the input data by capturing local and global patterns.

3. Positional Embedding

The positional embedding layer is used to generate both token and position embedding's of the input text (captions). It produces dense representations of the tokens (words) in the caption in addition to positional information of tokens in a sequence. This is important for understanding positioning of words in the captions since context established by position is crucial in the generation of correct syntactic and semantic sequences in descriptions. The layer scales the token embeddings for better representation and appends with positional embeddings; so the model can comprehend both the word meaning and structure of the sequences. It calculates a mask to exclude the effects of padding tokens during the training so that it cannot not influence the training or inference.

4. Decoder

The Transformer-Decoder-Block is used which implements the key component of a Transformer network which is used for building captions from image embedding's. The feeding-forward component comprises of three major components:

- self-attention
- cross-attention
- feeding forward layers

Which are necessary for enabling the decoder to capture the sequential nature of text and synchronize it with encoded image features.
The block starts with token and positional embedding's which are performed by the positional embedding layer that encodes the position as well as the meaning of the words in the caption. It employs a causal mask which blocks the decoder from accessing beyond the current token during training so that the auto regressive property is maintained. The self-

attention layer enable the decoder to pay attention to some of the words that have been previously generated in the caption while the cross attention layer engages with the outputs of the encoder to enable the mapping of textual tokens to the visual features inferred from the image. Preprocessing is performed through feed-forward layer where dropout is used for regularization, and multiple normalization steps in order to stabilize the network.

All the steps in the decoder layer makes it possible for the decoder to discover meaningful associations between features in the image and word sequence representations in a manner that allows that model to generate accurate and contextually sound captions for the image at hand. The block is concluded with a softmax output layer, which predicts the next word of the sequence for the word by word generation of the captions.

4. Custom Image Captioning Model Architecture

The basic task of this part is to connect all the modules we defined such as CNN for feature extraction encoder and decoder network defined above. Loss function to produce captions for images. This model first extracts image features from a pre-trained CNN which are then passed through and encoder over the image embeddings. The decoder incorporates such embeddings together with other previously emitted tokens to forecast the next token in the caption sequence using both self-attention and cross-attention. From the model, one can obtain several captions per image, which is important for creating various textual descriptions. The *train_step* method, is used to calculate the loss and accuracy of each batch of the images and captions and perform backpropagation to update the model weight. The loss is computed from a function with a mask to focus on non-padded tokens and the accuracy has been calculated on the basis of tokens. While testing, the *test_step* calculates and stores the loss/accuracy of the model without updating the model's weights. Moreover a function for augmenting the image is implemented for the purpose to diversify the training inputs. During the training process the model tracks the loss and the accuracy providing valuable insights into the model's performance and convergence. This class combines all the required modules for training and testing in the image captioning and title generation model, enabling the model to create accurate captions from the images fed into it.

Custom property metrics that returns the loss and the accuracy counter of the model that is, *self.loss_tracker* and *self.acc_tracker*. These trackers are further used to track the performance of the model at the time of training and evaluation.

The Caption Model is trained using the Adam optimizer, learning rate of 0.0001 is used, cross entropy loss function is used for training as the image captioning problem falls under multi class classification. Last step after compiling the model is to use the fit method which trains the model for 20 epochs using training while the results are validated with validation dataset. Training history is stored in history and will include information regarding loss and accuracy on each epoch of the entire training process for both the training and the validation datasets. This configuration is to enhance the model performance so that it can generate good and meaning captions for an image fed to the model.

Title Generator

1. Model and Tokenizer Loading:

The T5Tokenizer and T5ForConditionalGeneration available in the Hugging Face website are used in the title generation module. The tokenizer converts strings to tokens whereas the model produces the output title of the text given as input.

2. Input Preparation:

The caption (text) is modified with a prompt: Finally, tweeting simply write "summarize in three words only: {text}." This assists the model in understanding the task of generating a title against a caption generated.

3. Tokenization:

The input text is converted to token IDs with *tokenizer.Encode()* method from the tokenizer.

4. Title Generation:

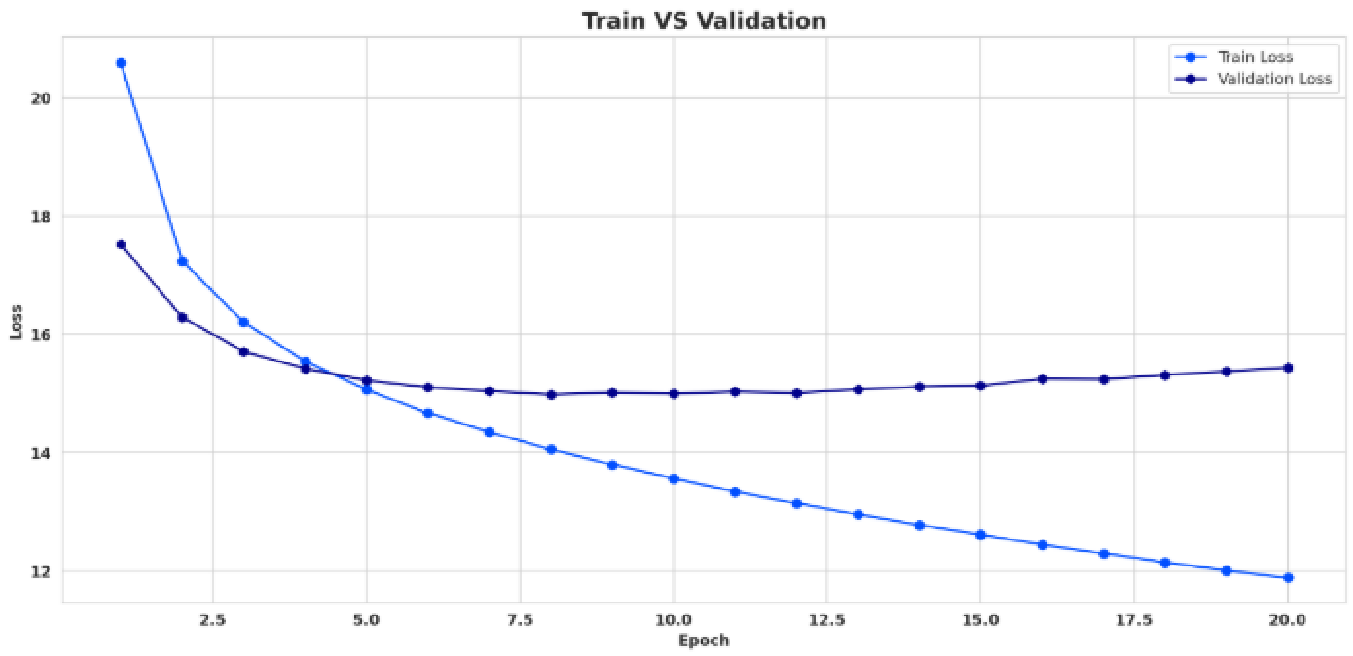
The generate() function of the model produces a title. Parameters such as max_length=8 confines the title to 8 tokens while num_beams=5 provides high-quality output using the beam search.

5. Decoding:
Finally, by using tokenizer.decode(), the output token IDs are resolved back into text to get the final title.

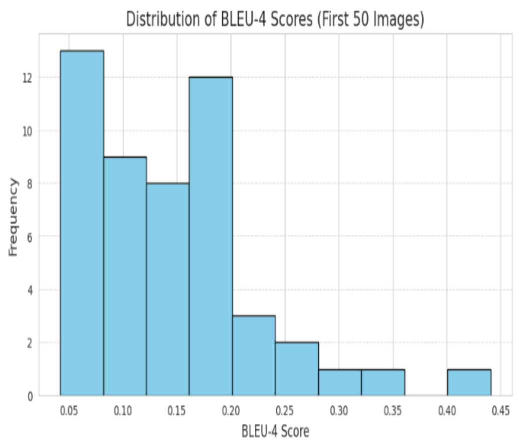
6. Output:
The function returns the generated three-word title.

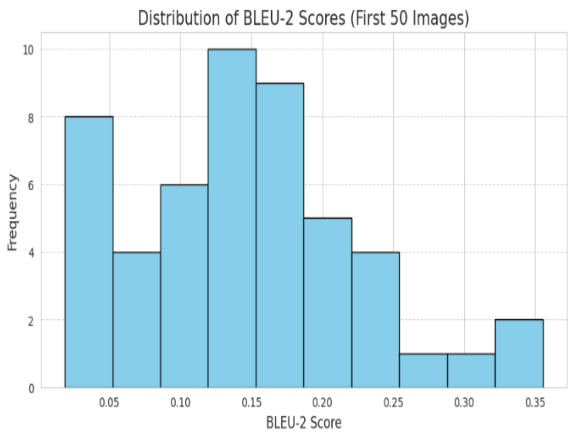
Testing and Collecting the Results

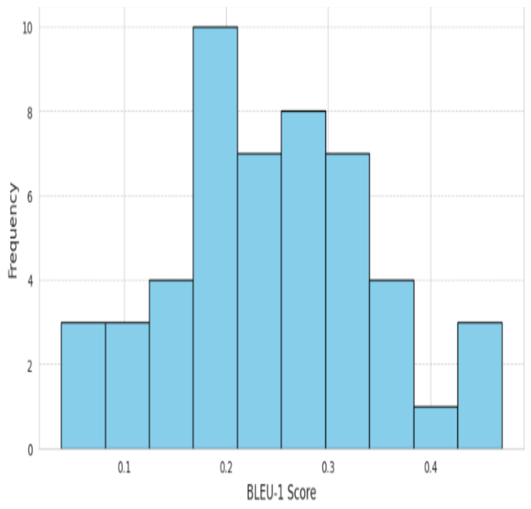
The loss for the captioning continuously decreases. The model was also tested on the test dataset.
For the evaluation of image captioning model BLEU scores were used. Through BLEU scores, the quantitative analysis was made to compare on the basis of the overlap of n-grams between the generated and reference captions. For instance, BLEU typically uses a modified precision of n-grams (e.g., unigrams, bigrams, trigrams) and incorporates a brevity penalty to penalize very short captions that may match many reference captions but lack completeness. These scores are especially useful in understanding how well or badly the generated captions fit human annotations. The result of the scores are below.

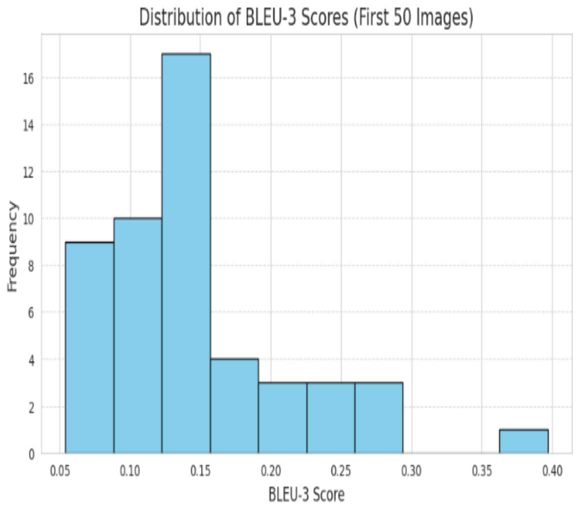


Training and validation loss on the training and validation dataset.









References

1. *Cho, Kyunghyun, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." Computer Science (2014) [1]*

2. *Karpathy, Andrej, and F. F. Li. "Deep visual-semantic alignments for generating image descriptions." Computer Vision and Pattern Recognition IEEE, 3128-3137. (2015) [2]*

3. *Mao, Junhua, et al. "Explain Images with Multimodal Recurrent Neural Networks." Computer Science (2014) [3]*

4. *Ranzato, Marc'Aurelio, et al. "Sequence Level Training with Recurrent Neural Networks." Computer Science (2015) [4]*