International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

Transforming Image Captioning: Refining Models with Advanced Encoder-Decoder Architecture and Attention Mechanism

Vikash Kumar Singh
Department of Computer Engineering
Parul University
Vadodara, Gujarat
vik439@gmail.com

Ankita Gandhi
Department of Computer Engineering
Parul University
Vadodara, Gujarat
ankita.gandhi@paruluniversity.ac.in

Brijesh Vala
Department of Computer Engineering
Parul University
Vadodara, Gujarat
brijesh.vala@paruluniversity.ac.in

Abstract—Image captioning involves the generation of textual descriptions that describe the content within an image. This process finds extensive utility in diverse applications, including the analysis of large, unlabeled image datasets, uncovering concealed patterns to facilitate machine learning applications, guiding self-driving vehicles, and developing software solutions to aid visually impaired individuals. The implementation of image captioning relies heavily on deep learning models, a technological frontier that has simplified the task of generating captions for images. This paper focuses on the utilization of encoder-decoder model with attention mechanism for image captioning. In classic image captioning model, the words usually describe only a part of the image, however with attention mechanism special attention is given to the low level and high-level features of the image. With the use of stable dataset and improvised encoder – decoder modelling, it is possible to generate captions having an accurate description of image with CIDEr score more by 16.52% of established models.

Keywords-component Image Captioning system; Deep Learning; encoder-decoder architecture; attention mechanism; COCO datasets.

I. INTRODUCTION

Image captioning is an evolving research domain that utilizes computer vision and natural language processing [5], for the generation of human-like textual descriptions for images. The objective is to use the potential of both these two fields, for developing models that are proficient enough in comprehending visual content and conveying the understanding through natural language.

This digital era has brought in never-seen-before entry of visual data, and images have become an integral part of our daily interactions. As the volume of images shared and stored online continues to grow exponentially, the need for advanced methods to extract meaningful information from these images has never been more demanding. Image captioning, which involves the generation of descriptive text to a given set of images, represents an application that combines natural language processing (NLP) with computer vision. The system at present has the potential to comprehend visual content and communicate it in a human-readable form, offering immense usability for different types of applications, including image retrieval, content recommendation, accessibility tools, and autonomous systems.

Within the scope of image captioning, attention mechanisms have emerged as a key architectural component having the potential to enhance the relevance and accuracy of generated captions. Attention mechanisms [12, 14], based on the cognitive process of visual attention in human beings, enable models to focus selectively on different areas of an image while generating any caption. The result is captions that are not only linguistically accurate but also semantically [1] aligned with the salient visual elements of the image. These mechanisms have gathered significant attention and have been integrated into various captioning models, yet the employability of different attention mechanisms and their impact on the accuracy of generated captions remain areas of active research.

In **figure 1**, we illustrate a common example of how an image captioning system generates textual descriptions for images fed into the model. The system comprehends the high-level features present in the image and produces suitable captions that are related to the context.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

Figure 1. An image captioning system that can generate suitable captions for images given as input to the model.

A notable challenge arises as the majority of image captioning models predominantly rests on the visual features guiding the encoder [14, 15], while the decoder heavily relies on textual information derived from the training dataset. This approach can pose challenges in accurately identifying objects within an image. In scenarios where multiple objects coexist in an image, the model may encounter difficulties recognizing them all and may struggle to comprehend the relationships among these objects. These limitations can lead descriptions of relationship between objects, or generating text that doesn't represent the image correctly. These issues directly impact the quality of final image caption. This indicates that it could be essential to (a) use a stable data source, (b) perform a reliable feature extraction method, (c) The encoder - decoder model should incorporate self - attention and semantic understanding during the model's training to improve its understanding of context and enhance the accuracy of caption generation.

The scope of this study leverages an improved version of attention mechanism and adjustment of hyperparameter to tune to an optimum point within the context of image captioning. It involves empirical evaluations, data analysis, and comparisons of different approaches, providing valuable insights into their application and utility.

Including a structured approach in the model would be necessary to address complex multi-modal image captioning challenges. By enhancing the source image and annotations and using a proven encoder - decoder architecture our goal is to enhance both the precision and depth of image captions, ultimately elevating the overall quality of the image captioning process.

The study holds significance in its potential to make further advancements in this field by offering a nuanced understanding of the role of attention mechanisms. The insights gained from this research can inform the development of more accurate, contextually meaningful, and interpretable image captioning models, which have far-reaching implications for both the academic and industrial communities. Furthermore, the study contributes to the broader narrative of how the integration of visual and linguistic modalities in deep learning can enhance the capabilities of machines in understanding and generating human-like descriptions.

II. Literature Review

A. Image Captioning: Early Approaches

Image captioning is a technique which is used to link visual information with textual interpretation, facilitated by machine learning methodologies [2,3]. Although the researches in image captioning spans a mere decade, the field has witnessed remarkable evolution. Kiros et al. [4] introduced a pivotal approach that used neural networks to address the image captioning issues. Their pioneering work in deep visual models and long short-term memory (LSTM) not only helped in retrieval ranking but also revolutionized the image captioning process. Subsequently, the community delved into the exploration of more deep neural networks [5] and various network architectures, including gated recurrent units (GRU), long short-term memory units (LSTM) [6] and their variants such as bidirectional LSTM [7], and phi-LSTM [8], all aiming to further enhance image captioning.

In yet another novel approach, Yao et al. [9] put forward the Long Short-term Memory network (LSTM-A) that was based on attributes, which enhanced the traditional CNN and LSTM model with advanced attribute features. Furthermore, they employed a multi-instance learning strategy which was weakly supervised to enhance the image captioning task. In the last two years it was seen that attention mechanism was incorporated in image captioning [10–12]. Cornia et al. [10] who conceptualized the M2 Transformer model, brought in more features in the system and has been a milestone achievement. To extract features from both image and text, this model leveraged a multi-layer attention mechanism that harnesses mesh connections to establish interdependencies between these two models. Simultaneously, innovative decision-making techniques, including reinforcement learning and generative adversarial networks [13], integrated into image captioning studies have enriched the naturalness and diversity of the generated textual descriptions.

B. Attention Mechanism

The idea of an attention mechanism initially came from intrinsic need of human cognition to navigate vast volumes of information efficiently. Humans are skilled to detect the focal points in their visual field, directing a sizeable number of cognitive resources to the subjects of interest.

This concept was initially applied into computer vision [14], with Xu et al. [15] being the pioneer who applied this in relation to image captioning. The core framework consists of an encoder-decoder architecture, which additionally keeps the essence of processing textual input by the analysis of visual content within the encoder. This processed visual information is subsequently used as an input for the decoder, generating natural language text.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
***Article Received:** 25 November 2023 **Revised:** 12 December 2023 **Accepted:** 30 January 2024*

Models founded on attention mechanisms have the unique capacity to not only record the spatial relationships among information but also gauge the significance of individual features based on their weighted importance. By dynamically adjusting these weight parameters, the model emphasizes crucial information and suppresses irrelevant details. This approach enhances the efficiency of algorithm used in deep learning, also addressing shortcomings associated with deep learning methods. The channel-based or spatial-based attention mechanism, are fundamentally used in visual attention methods. However, depending only on visual attention will not be able to retrieve the most pertinent features, leading to inaccuracies and verbosity in the generated textual descriptions.

C. Transformers

In 2017, Transformers as an architecture was introduced by Vaswani et al. [16] and was integrated into machine translation tasks. The Transformer also consists of two fundamental features: encoder and a decoder. The input content sequence is linked to a hidden layer through the encoder, which is then followed by positional encoding. Subsequently, the decoder connects the hidden layer to the output sequencing. Both the encoder and decoder mechanisms are composed of multiple Transformer blocks that share a consistent structure. Each Transformer block includes a multi-headed attention layer, a feedforward neural network, residual connections, and layer normalization.

The initial breakthroughs in the field of Transformer architecture were first observed in the context of natural language processing (NLP) [16]. Following this, Bidirectional Encoder Representations from Transformers (BERT) was introduced by Devlin et al. [17] . BERT is a model pre-trained on text which are not annotated but considers the contextual information of each word. Brown and his team [18] subsequently pioneered the Generative Pre-trained Transformer 3, a model consisting of 175 billion parameters, that was trained on an extensive 45TB of data in compressed and plaintext form. Notably, this model showcased outstanding performance without requiring additional fine-tuning. The Transformer-based architecture have transformed the relevance of NLP, primarily due to their robust representational capabilities.

Aided by the capacity of robust representation of the Transformer model, a large number of applications in a range of computer vision domains [19, 20], encompassing areas such as image processing [21], semantic segmentation [22], video captioning [23], and even image classification were made. Dosovitskiy and his team introduced the Vision Transformer (ViT) [24], a model that directly employs a pure Transformer architecture to process image patches and achieve reliably accurate results in various image recognition configurations. In a similar vein, the Globally Enhanced Transformer was presented by Ji et al. [25] that harnesses the power of Transformers to tackle image description tasks and producing promising results.

The Transformer architecture addresses a myriad of challenges through the utilization of its encoder-decoder structure, the attention mechanism, feedforward neural network and layer normalization. Nevertheless, certain shortcomings exist, such as the Transformer's efficiency in processing shorter sequences, which remains an area for improvement. In the context of image captioning, the Transformer processes object regions, converting them into vector representations that serve as guidance for generating descriptions. This approach excels at capturing regional-level details but may not holistically encapsulate global image information.

D. Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) was considered to be a notable breakthrough in the realm of image captioning. GANs introduce an innovative method for generating image captions by framing it as a generative context. This framework involves a pair of neural networks—a generator and a discriminator—participating in a competitive interplay. The generator strives to create convincing captions for a provided image, while the discriminator aims to distinguish between genuine and generated captions. Through this adversarial training, the process encourages the generation of cohesive and contextually relevant descriptions.

Chen et al. [26] introduced a GAN-based model for image captioning, showcasing outstanding performance. Their approach combines the generator's ability to produce diverse captions and the discriminator's capacity to provide feedback. The interplay between these networks yields captions that are both informative and contextually consistent.

Over the past few years, the realm of image captioning has seen continuous progress in the evolution of Generative Adversarial Networks (GANs). Diverse enhancements in architecture and training methodologies have been investigated to enhance the quality and diversity of the captions generated [27]. These advancements highlight the persistent effort to create increasingly refined, coherent, and contextually aware textual descriptions for images.

III. METHODOLOGY

The project employs a detailed methodology for image captioning using the power of deep learning to create detailed and contextually meaningful captions for images. At its core, the approach involves the meticulous processing of image-caption datasets, ensuring seamless integration with the chosen model architecture. In **figure 2** we have shown the workflow used in this project. The various functional blocks used in the project is as given below.

A. Datasets

In this research paper, we have utilized the COCO dataset as a foundational component of our study on enhanced image captioning with deep learning. The diverse nature of the dataset allows us to train and evaluate our proposed model across a wide range of real-world scenarios, providing a robust assessment of its performance and generalization capabilities.

The COCO dataset includes a diverse range of object categories, comprising 82,783 images for training, 40,504 for validation, and 40,775 for testing (approximately half for training, one-fourth for validation, and one-fourth for testing). In the 2014 train+val data alone, there are almost 270,000 segmented people and a total of 886,000 segmented object instances.

The training set, serves as the foundation for training and fine-tuning deep learning models. The validation set, with carefully curated annotations, facilitates model evaluation

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

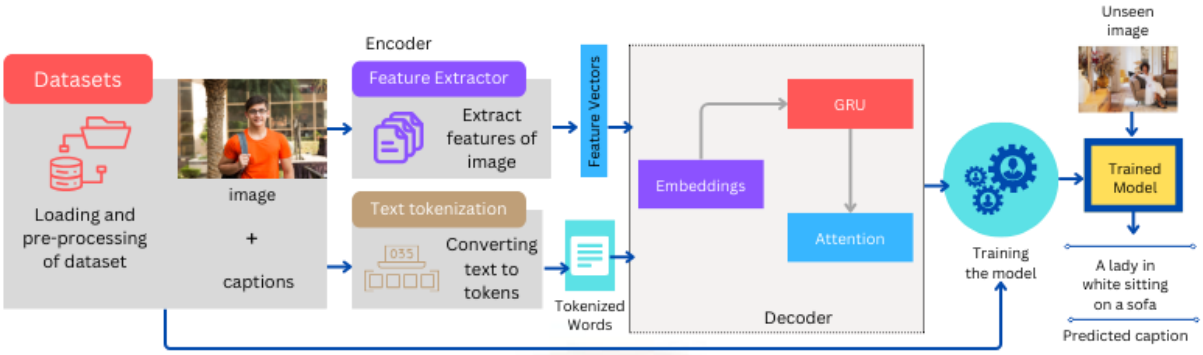

Figure 2. The Methodology adopted to generate detailed and contextually meaningful captions for images

during development. The test set, often used for benchmarking, is intentionally withheld during training and is employed to assess the generalization capabilities of models to unseen data.

What distinguishes COCO from other datasets is its focus on capturing the contextual relationships between objects within an image. Each image is annotated with object segmentations, object keypoints, and per-pixel object and stuff labels, allowing for a nuanced understanding of scene composition.

B. Loading and Pre-processing Datasets

In this stage, the COCO captions dataset is made ready to train the image captioning model. The COCO dataset's massive collection of images, each paired with descriptive captions, is loaded for processing. For each image-caption pair, we first focus on the caption – specifically, we extract the first one. Simultaneously, we resize the corresponding image to ensure consistent dimensions for streamlined processing.

The next stage is image normalization in which we divide each pixel value by 255, for creating a uniform representation across all images. This normalization step is akin to creating a common language for our diverse set of visuals. Next, we introduce a bit of randomness in the order of our image-caption pairs. This prevents the model from picking up any patterns based on the order of examples during training. Finally, prefetching optimizes the data-loading process. It's like having the next set of images and captions ready in the background while the model is busy processing the current batch. This way, we keep the training pipeline flowing smoothly.

C. Text Tokenization

Tokenization involves breaking down a text sequence into smaller units, referred to as tokens. In our context, this process is crucial as it aids the model in understanding and generating captions for images in a more organized and manageable manner.

Suppose in a sentence, "A joyful dog is playing in the park." Tokenization would transform this sentence into individual units or tokens: ["A", "joyful", "dog", "is", "playing", "in", "the", "park"]. Each word becomes a distinct token, making it easier for the model to process and analyze the linguistic information associated with an image.

Now, let's consider the specific example in our code. The TextVectorization tool is employed to tokenize and pre-process the captions associated with our images. It takes care of various aspects, such as lowercasing the text and removing punctuation. Additionally, it ensures that special tokens like "<start>" and "<end>" are preserved, as these play a vital role in our image captioning framework.

By tokenizing the captions, we convert the textual information into a format that the model can understand and learn from during training. This process of transforming words into numerical representations is fundamental for training the model to generate coherent and contextually relevant captions.

Furthermore, tokenization aids in constructing a vocabulary – a dictionary-like structure that maps each unique word to a numerical index. This index-based representation is crucial for the model to efficiently process and generate captions. For example, the word "dog" might be mapped to index 42 in our vocabulary.

We have applied a TextVectorization tool that applies a standardization process, ensuring uniformity in the text data. It lowers the case of all characters and removes any unwanted characters like punctuation. This not only simplifies the learning process for the model but also ensures consistency in understanding words that might be typed in different cases or contain extraneous characters.

Within the framework of our image captioning model and the application of TextVectorization in the provided code, the equation conceptualized is as follows:

$$Tokens_{captions} : TextVectorization = (Captions_{original})$$

Here, $Captions_{original}$ represents the original textual descriptions associated with images, and $Tokens_{captions}$ is the tokenized representation obtained through the TextVectorization process. In our code we created a vocabulary, the tool adapts to the captions in our dataset. It selectively retains the most frequent words, forming the foundation of our vocabulary. This tailored vocabulary is then utilized to tokenize and represent the entire dataset consistently.

Tokenization is not just a pre-processing step; it's an important component that enables the model to visualize language models. As our image captioning model learns to associate images with corresponding captions, the tokenized representation becomes the shared language between the visual and textual domains, facilitating a seamless integration of these two modalities in order to produce meaningful and captions relevant to context for our images. Simultaneously, we resize the corresponding image to ensure consistent dimensions for streamlined processing.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

D. Creating String Lookup Tables

In the context of the image captioning model, the string look up tables play a crucial role in ensuring a seamless transition between textual data and numerical representations.

Why Numeric Representation? Machine learning models, including neural networks, require numerical data as input. The string lookup tables efficiently convert words into numerical indices. This process establishes a numeric representation for words, making them suitable for processing by the model. For instance, the word "dog" might be represented by the index 42.

$$[Word \rightarrow Index: \quad word\_to\_index(dog) \rightarrow 42]$$

String lookup tables also contribute to managing the vocabulary by creating a reference for the model. The vocabulary ensures consistent processing of recognizable tokens during training, a crucial factor for both tokenization and decoding processes.

$$[Vocabulary: \quad \{dog: 42, cat: 17, \ldots\}].$$

During tokenization, where words are converted to indices, the string lookup table efficiently maps each word to its corresponding numerical representation. This operation enables the model to process input data in a numerical form.

$$[Tokenization: \quad word\_to\_index(dog) \rightarrow 42]$$

On the other hand, for tasks like decoding, where the model generates textual outputs, the reverse lookup is crucial. The model predicts numerical indices, and the string lookup table facilitates the conversion back to words.

$$[Decoding: \quad index\_to\_word(42) \rightarrow dog]$$

Masking and Padding: String lookup tables often incorporate functionalities for handling special tokens, such as padding tokens used for sequences of varying lengths or mask tokens to indicate areas where the model should ignore. These tokens contribute to the overall robustness of the model during training.

$$[Special\ Tokens: \quad \{< pad >: 0, < mask >: 1, \ldots\}]$$

These tables ensure consistency between training and inference processes. During training, the model needs to map words to indices, and during inference, it requires the reverse mapping. String lookup tables facilitate this bidirectional consistency.

$$[Training: \quad word\_to\_index(dog) \rightarrow 42]$$
$$[Inference: \quad index\_to\_word(42) \rightarrow dog]$$

Handling Out-of-Vocabulary (OOV) Words: String lookup tables can be configured to handle out-of-vocabulary words gracefully. This is crucial in scenarios where the model encounters words during inference that were not present in the training dataset.

$$[Handling\ OOV: \quad word\_to\_index(< unk >) \rightarrow <OOV\_index>]$$

In the provided code, two components, `word_to_index` and `index_to_word`, represent the string lookup tables. These tables serve as essential components in the tokenization and decoding processes, enabling the model to seamlessly navigate between textual and numerical representations.

E. Define Feature Extractor

In this project we have utilized the InceptionResNetV2 architecture, and also have introduced a custom encoder.

The InceptionResNetV2 model is a pre-trained deep neural network initially designed for image classification tasks. In this context, it serves as a feature extractor for images. Feature extraction entails capturing abstract and high-level representations from input images, which can then be employed for image captioning task. The InceptionResNetV2 architecture is chosen for its ability to capture intricate features in images and its pre-trained weights, which can be leveraged to enhance the model's understanding of visual data.

$$[InceptionResNetV2: \quad FEATURE\_EXTRACTOR(image\_input) \rightarrow image\_features]$$

Here, (image_input) represents the input image tensor, and (image_features) signifies the extracted features in a structured manner. This extraction process is crucial for transforming raw pixel values into meaningful representations that the model can comprehend.

A custom encoder is introduced to further process the features extracted by InceptionResNetV2. This encoder reshapes the features and applies a dense layer at the end to enhance the selectivity. The encoder's purpose is to filter the relevant information from the extracted features, reducing their dimensionality while preserving the essential characteristics for the subsequent stages of the model.

$$[encoder\_output = Dense(ATTENTION\_DIM, activation=relu)(x)]$$

In these equations, (x) represents the reshaped features, and (encoder_output) is the output after passing through a dense layer. The "relu" (Rectified Linear Unit) is an activation function that is employed to introduce non-linearity, capturing complex relationships in the data. In **figure 3** we see the layers used in the model of a feature extractor for getting the features of the image.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 299, 299, 3)]     0

 inception_resnet_v2 (Functi  (None, None, None, 1536)  54336736
 onal)

 reshape (Reshape)           (None, 64, 1536)          0

 dense (Dense)               (None, 64, 512)           786944

=================================================================
Total params: 55,123,680
Trainable params: 786,944
Non-trainable params: 54,336,736
_________________________________________________________________
```

Figure 3. The feature extractor, inceptionResNetV2 being used for fetching the features of image

For extracting the features of image, we have used InceptionResNetV2. Subsequently, we have used a custom encoder that helps in model's ability to understand and interpret visual information. The pre-trained weights of InceptionResNetV2, helps in understanding the knowledge acquired during its original training on large-scale image

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

datasets. The custom encoder makes the extracted features to align with the specific requirements of the image captioning task, enhancing the model's overall learning capability.

F. Define Decoder

The decoder is responsible for generating meaningful captions based on the extracted features from the images. **Figure 4** refers to a typical working of a decoder used for this project. The different components of the decoder and their roles in the captioning process are as given below.

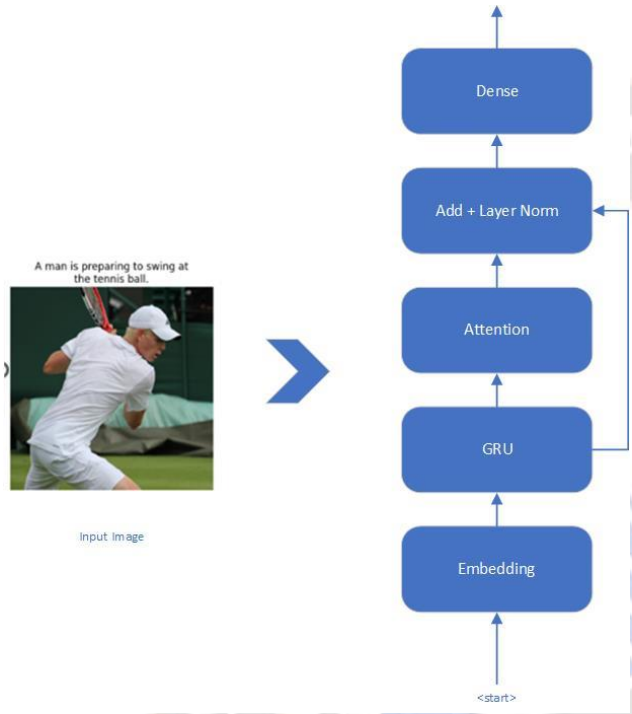

Figure 4. Working of a Decoder

Embedding Layer: The decoder begins with an Embedding layer. This layer is used to convert the numerical indices of words into dense vectors, effectively representing each word as a continuous-valued vector. This transformation is crucial as it allows the model to learn relationships and semantics between words during training.

[*Embedding Layer: embed_x = Embedding(VOCAB_SIZE, ATTENTION_DIM)(word_input)*]

Here, (VOCAB_SIZE) represents the vocabulary size, and (ATTENTION_DIM) gives the dimensionality of the embedded space. The resulting (embed_x) captures the semantic information of the words, creating a distributed representation that the model can use.

GRU Layer: Following the Embedding layer is a Gated Recurrent Unit (GRU) layer. The GRU, a type of recurrent neural network (RNN) layer, processes sequences of input data, capturing temporal dependencies. The GRU takes the embedded words as input and recurrently updates its internal state, effectively understanding the sequential nature of language.

[*GRU Layer: decoder_gru = GRU(ATTENTION_DIM, return_sequences=True, return_state=True)(embed_x)*]

Here, (ATTENTION_DIM) denotes the dimensionality of the GRU layer. The (return_sequences=True) argument ensures that the GRU layer outputs the full sequence of hidden states, while (return_state=True) returns the final hidden state. These hidden states capture the evolving context and information as the decoder processes each word. **Figure 5** shows a typical functioning of a GRU.

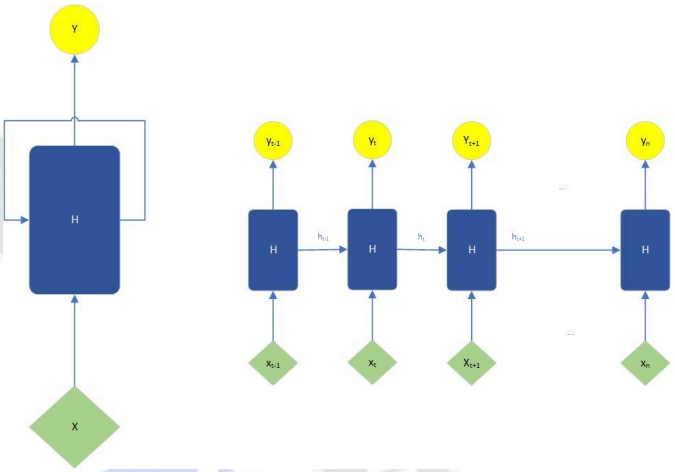

Figure 5. Working of a GRU

Attention Layer: Incorporating an Attention layer into the decoder directs focus to specific segments of the input sequence (image features) during the generation of each word in the output sequence (caption). The utilization of attention mechanisms empowers the model to assign distinct weights to various segments of the input, accentuating more pertinent information.

[*Attention Layer: decoder_attention = Attention()*]
[*context_vector = decoder_attention([decoder_gru, encoder_output])*]

Here, (decoder_gru) represents the hidden states from the GRU layer, and(encoder_output) is the output obtained from feature extraction process. The resulting (context_vector) captures the weighted information from the image features, dynamically adjusting attention during caption generation.

Layer Normalization and Dense Layer: To further enhance the model's learning and stability, Layer Normalization is added to the GRU output and the context vector. This normalization normalizes the values in the tensor across the features dimension, contributing to smoother training.

[*Layer Normalization: layer_norm_out = layer_norm(addition)*]

Finally, a Dense layer is employed which gives the final output vocabulary distribution. This layer projects the normalized and attended features into the vocabulary space, allowing the model to make prediction for the next word in that sequence.

[*Dense Layer: decoder_output_dense = Dense(VOCAB_SIZE)*]
[*decoder_output = decoder_output_dense(layer_norm_out)*]

Here, (VOCAB_SIZE) represents the size of the vocabulary. The output of the decoder represents a probability distribution

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
***Article Received:** 25 November 2023 **Revised:** 12 December 2023 **Accepted:** 30 January 2024*

on top of the vocabulary, indicating the likelihood of each word being the next word in the caption.

In short, the Embedding layer transforms words into continuous vectors, the GRU layer captures sequential dependencies, the Attention layer focuses on relevant image features, and subsequent layers contribute to stable and effective caption generation. These components work in tandem to produce coherent and contextually relevant captions for the input images.

G. Compile and Train the Model

The first step is to compile the model, where the optimizer and loss function are specified. In this case, the Adam optimizer is employed. Adam is an optimization algorithm that combines ideas from both Momentum and RMSprop, providing efficient gradient-based optimization. The learning rate of Adam is adaptive, adjusting during training based on the observed gradients.

[*Compile the Model:*
image_caption_train_model.compile(optimizer=adam, loss=loss_function)]

Here, $image\_caption\_train\_model$ represents the entire image captioning model. The $optimizer$ argument specifies the optimization algorithm, and $loss$ indicates the loss function that is used during training.

Sparse Categorical Crossentropy Loss: The choice of loss function is crucial in guiding the training process. The sparse categorical crossentropy loss is suitable for scenarios where the target values are integers, representing the class indices. In image captioning, the task is essentially a sequence generation problem where words are predicted sequentially. Sparse categorical crossentropy is well-suited for this scenario, as it handles the conversion of integer labels into one-hot encoded vectors internally.

Training the Model: The model is further trained on the dataset using compiled configuration. The training process involves iteratively feeding batches of data into the model, computing gradients, and updating the model's parameters in order to minimize the given loss function.

[*Training the Model: history =*
image_caption_train_model.fit(batched_ds, epochs=50)]

The fit method is called on the model, specifying the training dataset ($batched\_ds$) and the number of training epochs. An epoch represents one pass through the entire training dataset. The fit method manages the optimization process, adjusting the model's weights to minimize the specified loss. The model was trained for 50 epochs.

Impact of Adam Optimizer: Adam optimizer combines the benefits of both momentum and adaptive learning rates, making it suitable for a wide range of deep learning tasks. It helps convergence and handles variations in the learning rates of different parameters, contributing to stable and efficient training process.

In this step, we prepare the model for training by selecting Adam optimizer along with sparse categorical cross entropy loss. This training involves iteratively updating the model's parameters in order to minimize the chosen loss function. This combination of optimizer and loss function is essential for the successful training of the image captioning model, enabling it to produce accurate and captions that are relevant to the context.

H. Stateful Decoder for Prediction

This section focuses on adapting the decoder for generating captions at inference time. Inference time refers to the period when a trained model is used to make predictions based on new and unseen data. This happens to be a crucial step as it involves reusing the trained decoder in a stateful manner, allowing it to maintain context across predictions, preserving the sequential nature of language generation.

Stateful Decoder Architecture: In the context of sequence generation tasks, maintaining state across predictions becomes essential. At inference time, when generating captions for images, the model needs to remember its internal state from one time step to the next. The stateful version of the decoder facilitates this continuity, ensuring that the hidden state is preserved between predictions. The decoder is set up to receive inputs not only for the current time step but also from the hidden state obtained from the previous time step. This enables the model to consider the context of previously generated words while predicting the next word in the sequence.

Reuse of Trained Components: The section outlines the reuse of several components from the training decoder, including the GRU layer, Attention layer, and other associated layers. These components are preserved with their trained weights, ensuring that the model utilize the knowledge acquired during the training phase.

State Input and Output: A stateful GRU layer is introduced to the decoder, allowing the model to receive and update its hidden state at the time of generation of each word. The GRU state input ((gru_state_input)) represents the hidden state obtained from the previous time step, and state output ((gru_state)) signifies the updated hidden state after processing the current input.

[*Stateful GRU: gru_output, gru_state =*
decoder_gru(embed_x, initial_state=gru_state_input)]

Here, (embed_x) is the embedded representation of the current input word, and (initial_state=gru_state_input) indicates the incorporation of the previous hidden state.

Attention Mechanism and Context Vector: Attention mechanism remains a crucial aspect of the stateful decoder, enabling the model to concentrate on specific parts of the image features during generation of each word. The context vector ((context_vector)) captures the weighted information from the image features, dynamically adjusting attention during the caption generation process.

[*Attention Layer: context_vector =*
decoder_attention([gru_output, encoder_output])]

The (decoder_attention) layer considers both the current hidden state ((gru_output)) and the image features ((encoder_output)) producing a context vector that guides the generation of the next word.

Dense Layer and Output: Similar to the training decoder, a Dense layer is employed to produce the final output vocabulary distribution. This layer projects the normalized and attended

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
***Article Received:** 25 November 2023 **Revised:** 12 December 2023 **Accepted:** 30 January 2024*

features into the vocabulary space, facilitating the predicting next word that would come in the sequence.

[*Dense Layer: decoder_output = decoder_output_dense(layer_norm_out)*]

The (layer_norm_out) represents the normalized and attended features, and (decoder_output) is the final output, representing the probability distribution over the vocabulary.

By introducing stateful components and reusing trained layers, the decoder retains context across predictions, allowing for coherent and contextually relevant caption generation for input images. The stateful decoder reflects the intricacies of sequential language generation during inference, ensuring a smooth and context-aware captioning process.

I. Prediction Function

This section is crucial for leveraging the trained image captioning model for generation of captions for new and unseen images. The prediction function is designed to take a file path representing an image as input. This file is typically a JPEG image file. The function is designed to perform the necessary pre-processing steps to make the image compatible with the model's input requirements.

Image Pre-processing: The first step in the function is to decode the JPEG image file, resizing it to match the dimensions expected by the image captioning model. The pixel values are then normalized to be in the range [0, 1], ensuring consistency with the training data. These pre-processing steps matches the input image with the format expected during training, enabling the model to generate captions which are based on the processed image.

The pre-processed image is then given as input into the encoder, which extracts high-level, distinct features from the image. The encoder uses the InceptionResNetV2 architecture, preserving the spatial and contextual information of the input image.

The resulting features represent a condensed and meaningful representation of the input image, capturing essential information for generating captions.

The prediction function includes a loop to iteratively build each word in the caption sequence. The start token ("<start>") is the initial input to the decoder, and the loop continues end token ("<end>") is encountered by the model or until a specified caption length is reached.

Here, `decoder_pred_model` is the stateful version of the decoder used for predictions. The loop iterates through each time sequence, updating hidden state and generating the next word in the sequence.

Probabilistic Prediction: The predictions from the model represent a probability distribution on top of the vocabulary. A probabilistic approach is employed to select the next word. The model's predictions are sampled, introducing randomness into the generation process. This sampling mechanism ensures diversity in the generated captions, as the model has a chance to explore different possibilities.

Result Accumulation: The generated word is then added to the result, and the loop continues until the end token surfaces or a given caption length is reached. The function returns the processed image and the accumulated result, representing the generated caption.

A prediction function encapsulates the entire process of predicting captions for a given image using the trained image captioning model. It handles image pre-processing, feature extraction, and sequential caption generation, demonstrating how the model can be applied to generate contextually relevant captions for new and unseen images.

J. Generate and Display Captions

In this section the previously defined prediction function is utilized to generate captions for a sample image, and the results are displayed using the Matplotlib library.

The prediction function, which was defined earlier to generate captions for a given image file on the trained model, is employed here. The chosen image file for caption generation is specified, and the prediction function is called with this input.

Here, the prediction function is invoked for the image, and the captions generated by the model is printed. The loop allows for multiple captions to be generated, providing some diversity in the generated outputs. Matplotlib is then utilized to display the sample image along with its generated captions.

Caption Rendering: For each generated caption, line breaks are applied to format the text, and the captions are set as titles for the subplots. Finally, the entire plot with the image and its generated captions is displayed.

In summary, the "Generate and Display Captions" section demonstrates the practical application of the prediction function on a sample image. The generated captions are printed, and Matplotlib is employed to visually showcase the image alongside its diverse set of captions. This section allows for a qualitative assessment of the model's performance in generating contextually relevant and varied captions for the given image.

IV. Observations, Results And Discussion

The project is coded in Python using TensorFlow. The feature extractor used as encoder is InceptionResNetV2 and MSCOCO dataset is primarily used for the training and captioning task. In this project, we have used three methods to evaluate our model: i) graphical analysis, ii) evaluation metrics, and iii) manual evaluation.

Graphical Analysis: **Fig. 6** shows some of the metrics represented graphically using Tensor Board. The Epoch Accuracy refers to the accuracy of the model's prediction on the training data for a single pass through the entire dataset, which is the epoch.

In the figure we can see that the accuracy is stable with values at 0.95 clocking to 0.97 showing that the model responds with accuracy in predicting captions. Epoch loss is the average loss or error incurred by the model on the training dataset during one epoch.The epoch loss here is 0.04 which shows how the model's prediction are closer to the ground truth captions.

The Evaluation Accuracy vs Iterations plots the evaluation accuracy of model against number of iterations during training. This plot helps in monitoring the training process and determining when to stop the training to prevent overfitting. Loss graph measures the dissimilarity between predicted captions and the ground truth captions. In this project we have used sparse categorical cross entropy loss. The loss value

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
Article Received: *25 November 2023* ***Revised:*** *12 December 2023* ***Accepted:*** *30 January 2024*

averages out to 0.15, which shows that the error associated in training is very less and the model can definitely predict the caption with accuracy.

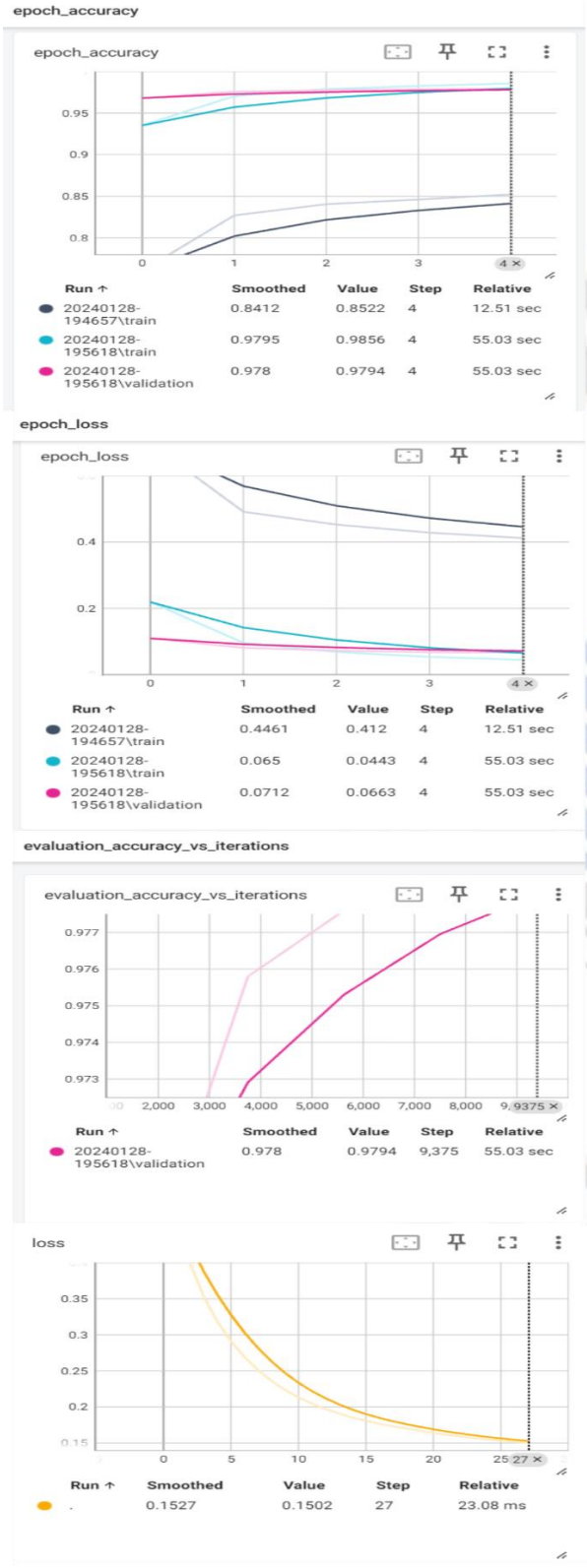

Figure 6. Tensor Board graphical analysis of the model using the metrics (top to bottom) epoch_accuracy, epoch_loss, evaluation_accuracy_vs_iterations and loss.

Through the graphs we have seen how convergence has taken place so smoothly. The graph has captured various metrics like loss and accuracy with epoch loss and epoch accuracy. It also shows how the evaluation accuracy and loss has performed over the iterations. All in all, the result shows how well the model has learned to perform the task of image captioning by itself.

Evaluation Metrics: Evaluation metrics are used in image captioning to assess the effectiveness of the predicted captions in image captioning system. In this model we have used three such metrics: i) BLEU, ii) METEOR, and iii) CIDEr. BLEU metrics are used to carry out a quantitative analysis on the correspondence between the predicted caption as output and the human translated captions. METEOR metrics uses the harmonic mean of unigram precision and recall, for all the recall having more weight than the precision.

$$Meteor = [10 * Precision * Recall / (Recall + 9 * Precision)]$$

The CIDEr is used to assess the semantic similarity and diversity. It provides more comprehensive assessment than BLEU and METEOR. **Table 1** gives us the result of comparison of our model with the baseline model of MS COCO Karpathy [3] split.

Metrics	**Baseline Model (MS COCO Karpathy split)**	**Our Model (With InceptionResNetV2 and attention mechanism)**
BLEU-1	0.463	0.496
BLEU-2	0.273	0.289
BLEU-3	0.156	0.177
BLEU-4	0.087	0.092
METEOR	0.157	0.167
CIDEr	0.339	0.395

Table 1: Result of superimposing our model to the baseline model used on the MSCOCO Karpathy split

In this table we can see how well the evaluation scores have increased after using our improved model consisting of improved and stable feature extractor InceptionResNetV2 and enhanced attention mechanism features. We can see that the CIDEr score has increased by 16.52 % which shows that the model has good improvement over the human judgement showcasing a boost in grammatical integrity and semantic similarity.

Manual Evaluation: **Figure 7** depicts some of the images with their captions generated by the trained model. By analyzing various image samples, it has been observed that the model consistently provides accurate and contextually relevant descriptions of the scenes depicted in the images. This thorough examination involved comparing the generated captions with the actual content of the images.

Impressively, the model demonstrated a keen ability to capture the semantic context of the visual elements, accurately identifying and describing objects, actions, and overall contexts within the images. The reliability and precision exhibited by the model in consistently producing meaningful and coherent captions reflect its effectiveness in understanding and interpreting diverse visual content. These findings underscore the model's proficiency in the task of image captioning and its

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 12 Issue: 2
***Article Received:** 25 November 2023 **Revised:** 12 December 2023 **Accepted:** 30 January 2024*

potential for generating informative and contextually rich descriptions across a range of images.

Figure 7. Each of the images were randomly selected and were given for caption generation. The captions have been found to generate accurate and contextually relevant content.

V. Conclusion And Summary

In this research, we have tried to explore and implement an automated image captioning system that leverages leading deep learning techniques. The primary objective was to develop a model capable of generating coherent and contextually relevant captions for images, thereby addressing the contextual gap between visual content and natural language.

We wanted to begin with a huge dataset base and for us, the selection of the COCO dataset, a rich source of diverse images paired with descriptive captions, facilitated the training of a robust image captioning model. TensorFlow and its associated libraries, including TensorFlow Datasets and TensorFlow Hub, was used throughout in the implementation and experimentation phases.

For our deep learning model, we used a pre-trained InceptionResNetV2 as the feature extractor. It was particularly helpful in not only quick convergence but also brought a good amount of diversity captured by the ImageNet weights. In the subsequent sections, we went on with data pre-processing, tokenization, and the design of the image captioning architecture.

Key components of our model architecture included an attention mechanism, GRU (Gated Recurrent Unit) layers, and embedding layers. These elements worked together perfectly to capture salient features from the images and generate meaningful captions through recurrent processing. The attention mechanism, inspired by Bahdanau-style attention, enabled the model to concentrate on relevant parts of the image while generating each word in the caption.

The incorporation of StringLookup and TextVectorization layers facilitated efficient tokenization and vocabulary management. We introduced a start and end token to distinguish between the start and end of each caption, enhancing the model's understanding of sentence structure.

The training process involved careful implementation of data pipeline operations, such as shuffling, prefetching, and batching, to ensure optimal utilization of computational resources and accelerated convergence. The model's performance was evaluated using the Sparse Categorical Crossentropy loss function, providing valuable insights into its ability to learn from the dataset.

A noteworthy aspect of our approach was the introduction of a probabilistic prediction model during inference. This model, utilizing the trained decoder and attention mechanisms, exhibited the capability to generate diverse and contextually appropriate captions for a given image.

Despite the achievements, our exploration encountered challenges, including dataset management complexities and fine-tuning nuances. However, through meticulous problem-solving and continuous refinement, we successfully navigated these hurdles, contributing to the robustness and effectiveness of our image captioning system.

In conclusion, our research represents a significant stride towards the realization of automated image captioning systems with a focus on interpretability and context-awareness. The integration of attention mechanisms, deep learning architectures, and tokenization strategies has yielded a model that not only learns from data but also exhibits creativity in generating descriptive captions.

The potential applications of our work extend to diverse domains, including accessibility for visually impaired individuals, content indexing, and human-computer interaction. As we reflect on our journey, we seek to establish a future where

automated image captioning becomes an integral component of multimedia understanding, enhancing our interaction with visual content in the digital landscape.

VI. REFERENCES

[1] X. Jia, Y. Wang, Y. Peng, and S. Chen, "Semantic association enhancement transformer with relative position for image captioning," Multimedia Tools and Applications, vol. 81, no. 15, pp. 21 349–21 367, 2022.

[2] Reichert, D.P.; Series, P.; Storkey, A.J. "A hierarchical generative model of recurrent object-based attention in the visual cortex." In Proceedings of the International Conference on Artifical Neural Networks, Berlin, Heidelberg, Germany, 14–17 June 2011; pp 18–25.

[3] Karpathy, A.; Fei, L. "Deep visual-semantic alignments for generating image descriptions." IEEE Trans Pattern Anal. Mach. Intell. 2017, 39, 664–676. [CrossRef] [PubMed]

[4] Kiros, R.; Salakhutdinov, R.; Zemel, R. "Multimodal neural language models". In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014.

[5] Kuznetsova, P.; Ordonez, V.; Berg, A.C.; Berg, T.; Choi, Y. "Collective generation of natural image descriptions". In Proceedings of the Meeting of the Association for Computational Linguistics: Long Papers, Jeju Island, Korea, 8–14 July 2012.

[6] Hochreiter, S.; Schmidhuber, J. "Long short-term memory. Neural Comput." 1997, 9, 1735–1780. [CrossRef] [PubMed]

[7] Wang, C.; Yang, H.; Bartz, C.; Meinel, C. "Image captioning with deep bidirectional LSTMs." In Proceedings of the ACM on Multimedia Conference, Amsterdam, The Netherlands, 15–19 October 2016.

[8] Tan, Y.H.; Chan, C.S. phi-LSTM: "A phrase-based hierarchical LSTM model for image captioning." In Proceedings of the Asian Conference on Computer Vision, Taipei, Taiwan, 21–23 November 2016.

[9] Yao, T.; Pan, Y.; Li, Y.; Qiu, Z.; Mei, T. "Boosting image captioning with attributes." In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.

[10] Cornia, M.; Stefanini, M.; Baraldi, L.; Cucchiara, L. "Meshed-memory transformer for image captioning." In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, DC, USA, 14–19 June 2020.

[11] Chen, S.; Jin, Q.; Wang, P.; Wu, Q. "Say as you wish: Fine-grained control of image caption generation with abstract scene graphs." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, DC, USA, 14–19 June 2020.

[12] Huang, L.; Wang, W.; Chen, J.; Wei, X. "Attention on attention for image captioning". In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019.

[13] Zhou, Y.; Wang, M.; Liu, D.; Hu, Z.; Zhang, H. "More grounded image captioning by distilling image-text matching model." In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, DC, USA, 14–19 June 2020.

[14] Luong, M.T.; Pham, H.; Manning, C.D. "Effective Approaches to Attention-based Neural Machine Translation." In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015.

[15] Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. "Show, attend and tell: Neural image caption generation with visual attention." In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6 July 2015; pp. 2048–2057.Electronics 2022, 11, 1397

[16] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. "Attention Is All You Need." In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4 December2017; pp. 5998–6008.

[17] Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. Bert: "Pre-training of deep bidirectional transformers for language understanding." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 4 June 2019; pp. 4171–4186.

[18] Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. "Language models are few-shot learners." In Proceedings of the Neural Information Processing Systems, Vancouver, BC, Canada, 5–12 December 2020.

[19] Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. "End-to-end object detection with transformers." In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020.

[20] Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. "Deformable detr: Deformable transformers for end-to-end object detection." In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 3–7 May 2021.

[21] Chen, H.; Wang, Y.; Guo, T.; Xu, C.; Deng, Y.; Liu, Z.; Ma, S.; Xu, C.; Xu, C.; Gao, W. "Pre-trained image processing transformer." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021.

[22] Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P.H.S.; et al. "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021.

[23] Zhou, L.; Zhou, Y.; Corso, J.J.; Socher, R.; Xiong, C. "End-to-end dense video captioning with masked transformer." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8739–8748.

[24] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. "An image is worth 16x16 words: Transformers for image recognition at scale." In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 3–7 May 2021.

[25] Ji, J.; Luo, Y.; Sun, X.; Chen, F.; Luo, G.; Wu, Y.; Gao, Y.; Ji, R. "Improving Image Captioning by Leveraging Intra and Inter-layer Global Representation in Transformer Network." In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.

[26] Shi, X.; Hu, H.; Che, W.; Sun, Z.; Liu, T.; Huang, J. "Understanding Medical Conversations with Scattered Keyword Attention and Weak Supervision from Responses." In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.

[27] Liu, W.; Chen, S.; Guo, L.; Zhu, X.; Liu, J. "CPTR: Full Transformer Network for Image Captioning." arXiv 2021, arXiv:2101.10804.

[28] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3156–3164.

[29] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6077–6086.

[30] Zanyar Zohourianshahzadi · Jugal K. Kalita, "Neural Attention for Image Captioning: Review of Outstanding Methods" arXiv:2111.1501v1 [cs.CV] 29 Nov 2021