UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

MSc Computer Science with Advanced Research

7COM1039-0109-2022 - Advanced Computer Science Masters Project

2023.08.28

Enhancing Real-Time Face Mask Detection in Security Applications using Vision Transformer Architecture

Name: Shaya Sathiyan

Supervisor: Dr Barry Ip

Abstract

This study aims to improve real-time face mask detection in security applications by using the capabilities of the Vision Transformer (ViT) architecture. The key study topics guiding this investigation focus on the ViT model's performance, reaction to pre-processing methods, processing power, and adaptation to various face mask designs. This study tackles these issues through a thorough analysis with the overall objective of strengthening security measures while developing picture captioning and security technologies. The main goal of the project is to assess the viability of the ViT architecture for real-time face mask identification. The study carefully investigates accuracy, speed, and resilience, revealing ViT's capability to quickly and consistently distinguish face masks. The secondary study objectives also explore how pre-processing methods affect ViT's functionality. The research evaluates the model's capacity to respond to dynamic situations by closely examining data augmentation and picture scaling under various lighting environments and facial angles. A key focus is on ViT's capacity to retain real-time processing capabilities while guaranteeing excellent accuracy. ViT's effectiveness is demonstrated by comparing the model's inference speed with that of other architectures while taking into account things like model size and hardware needs. The research also examines how the model handles various face mask designs, revealing its generalization advantages and disadvantages. In the context of morally sound deployment, the potential biases and difficulties encountered are examined. The study's base is a thorough literature assessment, and it draws its conclusions from innovative models and breakthroughs in picture captioning. The COCO2017 image caption dataset, which has a huge variety of picture contexts, is the one used for training and assessment. Contextual relevance in produced captions is improved by the use of the Contextually Aware Transformer based Image Captioning (CATIC) model, which integrates cutting-edge attention mechanisms that connect spatial and adaptive attention.

Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Computer Science with Advanced Research at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Shaya Sathiyan

Acknowledgements

I am profoundly grateful to express my deepest appreciation to Professor Dr. Barry Ip, my amazing supervisor, whose unwavering support, patient guidance, and continues encouragement have been pivotal throughout the journey of this project. His perceptive insights and priceless comments not only shaped the trajectory of my project but also enriched my understanding of the subject. His mentorship, characterized by his exceptional ability to distill complex concepts into clear and manageable steps, has been an immense source of motivation, propelling me to overcome challenges and achieve milestones.

I am truly fortunate to have had such an inspiring supervisor who not only provided direction but also nurtured intellectual growth. My heartfelt thanks for enabling me to complete this work within the defined timeframe.

I extend my sincere gratitude to the University of Hertfordshire for providing an enriching academic environment and access to indispensable resources that facilitated the realization of my master's project.

I would also like to acknowledge the instrumental role my family and friends played in my journey to the University of Hertfordshire. Their unwavering belief and support have been a constant source of strength.

In closing, I want to express my gratitude once again to everyone who contributed to this endeavor, from the bottom of my heart. This accomplishment stands as a testament to the power of collaboration, guidance, and perseverance.

Table of Content

List of Figures

List of Tables

List of Nomenclatures

1. List of Acronyms

Vit – Vision Transformer
COCO – Common Objects in Context
CATIC – Contextually Aware Transformer based Image Captioning
ML – Machine Learning
AI – Artificial Intelligent
IoT – Internet of Things
IIoT – Industrial Internet of Things
UAV – Unmanned Aerial Vehicles
IC – Image Captioning
CV – Computer Vision
NLP – Natural language Processing
YOLO – You Only Look Once
CNN – Convolutional Neural Network
R-CNN – Region-Based Convolutional Neural Network
YOLOv3 – You Only Look Once, Version 3
RPN – Region Proposal Network
LSTM – Long Short-Term Memory
RNN – Recurrent Neural Network
M-RNN – Multimodal Recurrent Neural Network
NIC – Neural Image Caption
RSICD – Remote Sensing Image Captioning Dataset
MS COCO – Microsoft Common Objects in Context
IAPR - International Association of Pattern Recognition
GPT-2 – Generative Pre-trained Transformer 2
BERT – Bidirectional Encoder Representations from Transformers
ReLU - Rectified Linear Unit
BLEU – BiLingual Evaluation Understudy
T5 – Text-to-Text-Transfer-Transformer
RAM – Random Access Memory
TPU – Tensor Processing Unit

2. List of Symbols

h_t - hidden state calculation of the LSTM
p_t - final probability distribution of the predicted words of the caption
Wo - LSTM training Parameter
S – Statement
N - Words Long
K - 1000 in the number system (E.g. 20K = 20000)
N - Number of visual features
M - Number of textual features (tokens)
d_v - Dimensions of the visual
d_q - Dimensions of the Textual features
Q – Query matrices
K – Key matrices
V – Value matrices
W_q, W_k, and W_v – Weight Matrices
d_ff - Feed-Forward Dimension
W, W_1 and W_2- Learnable weight matrix
b, b_1 and b_2 - Learnable bias vector
d_model - Dimension of the input embeddings
pos - Position of the token in the sequence
i - Index of the dimension in the embedding vector
PE_(pos, 2i) and PE_(pos, 2i+1) - even and odd components of the positional encoding for the token at position "pos" and dimension "i."
d_v - Dimension of the visual features
M - Number of textual features (words)
d_q - Dimension of the textual features
Q'' - Cross-Attention between textual features
V_enc - Encoded visual features
Caption_probs - Probabilities of each word in the vocabulary being the next word in the generated caption

Chapter 1: Introduction

The COVID-19 epidemic has become a significant hazard across the globe. Since the quantity of infections is continuously increasing day by day, the government is encountering a lot of problems in trying to limit the epidemic. Only with the correct cooperation of individuals can the spread of this disease be slowed. Physical separation, frequent hand cleaning, and face masks have all been shown to be quite effective at preventing the virus from spreading, but not everyone is following the rules. A number of technologies, such as machine learning (ML) algorithms, artificial intelligence (AI) techniques, the Internet of things (IoT), and unmanned aerial vehicles (UAV), provide an actual situation at any moment about (i) the number of people following physical distancing and (ii) whether people are wearing masks or not.

COVID-19 has emerged as one of the most crucial issues that needs to be appropriately addressed in the current environment (Anand *et al.*, 2021) (Singh *et al.*, 2021). Therefore, it is necessary to create a strategy that may address problems like unmasked individuals and the lack of physical distance. With the significant advancements in industrial IoT (IIoT), remote monitoring is now much simpler to carry out. The interconnection of various automated components across a network is the fundamental idea behind the Internet of Things. The goal is to build a facial mask recognition system based on IIoT, and each item is given a distinct identity to facilitate data flow between them.

This COVID-19 pandemic situation, which has killed people in around 200 nations and territories and two international modes of transportation, has resulted in “96.1 million illnesses and 2.06 million” fatalities worldwide as of January 20, 2021. The public was made even more vulnerable because there weren't many active pharmaceutical professionals and there wasn't much public opposition to COVID-19. According to Uddin at el., the “World Health Organization” has designated it as a pandemic (Uddin, Ali Shah and Al-Khasawneh, 2020). The only course of action available is to put on a mask because there is no cure for this epidemic. The general public is using face masks more frequently as a result of the possibility that they can stop the spread of COVID-19. To stop the pandemic from spreading further, the world community has to consider quarantine and widening the social gap between sick and healthy individuals. This face mask test is performed to determine whether the subject is immune to the airborne virus. Viruses are released into the air when a person speaks, coughs, or sneezes, increasing the chance that individuals nearby will get them. Specialists in infection control employ a variety of precautions, such as “surgical masks” to protect against contamination, to lessen the spread of disease.

Humans can easily identify objects and their spatial relationships in an image, and then they use descriptive words to describe the image's content. Making computers capable of this amazing cognitive ability has proven to be difficult. The discipline of deep learning has developed "Image Captioning" (IC), a potent remedy for this issue. By merging the fields of Computer Vision (CV) and Natural Language Processing (NLP) in a cross-disciplinary solution, picture captioning entails automatically creating natural language descriptions of images.

The following are the key reasons that this essay stands apart. According to faster R-CNN and YOLO models, a face mask recognition technique is employed (Liu *et al.*, 2019). When delving into the creation of novel face mask detectors, a thorough grasp of the underlying challenges associated with face mask detection proves invaluable. The introduction of fresh data is pivotal in the formulation of algorithms within this framework of transfer learning. This technique in machine learning involves the assimilation of skills from a specific task and their subsequent application in diverse contexts. The adoption of pretrained models is gaining momentum as a cornerstone for AI endeavors, particularly those entailing complexities in computation and temporal restrictions. The study compares the predictive performance of two algorithms devised by the authors: YOLOv3 and accelerated R-CNN. The effectiveness of the R-CNN algorithm surpasses YOLO due to its incorporation of dual networks, with the initial network being the region proposal network (RPN) which provides object localization proposals (Liu *et al.*, 2019). Each region box within the RPN is assigned a corresponding value. YOLO, a contemporary CNN approach, exhibits real-time object recognition capabilities. The methodology segments images into distinct regions and makes informed predictions concerning the areas suitable for model training. However, this task is intricate owing to variations in camera angles and mask types depicted in the images. An additional challenge arises from the scarcity of a comprehensive dataset encompassing both masked and unmasked categories, prompting the authors to curate a novel dataset and employ transfer learning to facilitate their investigation. In the pursuit of analyzing drone footage for adherence to social distancing norms and mask utilization, artificial intelligence and the accelerated R-CNN algorithm take center stage. The acquired imagery is subjected to scrutiny through datasets and pretrained models in order to derive insights.

The model's CV side is crucial in IC for identifying characteristics in an image and translating them into feature maps or vectors. The NLP side, on the other hand, is concerned with coming up with cohesive and contextually significant description words. The popularity of IC is due to the wide range of disciplines in which it may be applied, including captioning aerial photos, captioning social media images, classification, creating medical image descriptions, and more. An image feature extraction module and a caption synthesis module make up the two primary parts of the traditional architecture of an end-to-end IC model. A trained convolutional neural network was used to extract picture features (Gopika *et al.*, 2020). CNNs are a popular alternative for image feature extraction in IC because of their superior feature extraction skills in computer vision tasks.

Once the visual attributes have been gathered, they are commonly integrated into models that utilize Long Short-Term Memory (LSTM) architectures, as proposed by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997). These LSTM-based models are incorporated within a framework of Recurrent Neural Networks (RNN) to facilitate the synthesis of captions. LSTMs have gained prominence as a preferred choice for generating captions, primarily due to their proven efficacy in handling sequential data and effectively addressing the challenge of managing long-range dependencies within sequences.

However, there are some shortcomings with contemporary IC models. The sequential nature of RNN computations, which makes them memory expensive and challenging to parallelize, is a key downside. The growing computational burden of RNN-based systems makes it difficult to generate lengthier captions. Additionally, the present models are trained using tokenized word formats, which ignores the significance of word placement and relative positions in a sentence's semantic meaning. This disregard for context makes it difficult to produce captions that are more logical and pertinent to their surroundings.

Contemporary IC types do, however, have some drawbacks. One major drawback of RNN computations is their sequential nature, which makes them memory expensive and difficult to parallelize. Longer captions are harder to generate using RNN-based systems because of their increasing computational burden. Furthermore, the tokenized word formats used to train the current models disregard the importance of word placement and relative positions in a sentence's semantic meaning. It is challenging to create captions that are more logical and relevant to their surroundings because of this disregard for context.

In the realm of automatic image captioning, the proposed research embarks on a journey to harness the prowess of “Vision Transformer (ViT)” networks. This study delves into the multifaceted landscape of image captioning, probing questions that illuminate the potential and scope of employing “Vision Transformer (ViT)” for enhancing this field.

1.1 Research Questions

Central to this endeavor are a set of pivotal research questions that guide the exploration:

- How effectively can the Vision Transformer (ViT) architecture be employed for real-time face mask detection in security applications, specifically focusing on accuracy, speed, and robustness?
- How does the choice of pre-processing techniques (e.g., data augmentation, image resizing) impact the performance of the Vision Transformer model in

detecting face masks accurately under varying lighting conditions and facial orientations?

- Can the Vision Transformer model maintain real-time processing capabilities while ensuring high accuracy in identifying face masks, and how does its inference speed compare to other models, considering factors like model size and hardware requirements?

- How does the Vision Transformer model handle variations in face mask styles, colors, and patterns, and are there any notable challenges or limitations in its ability to generalize to different face mask designs?

This research initiative is driven by the aspiration to leverage the Vision Transformer (ViT) architecture for real-time face mask detection in security contexts. The research questions delve into ViT's performance compared to traditional CNN models, the impact of pre-processing techniques and training strategies, and its adaptability to diverse face mask styles. Ethical considerations and practical deployment concerns are also addressed. The study's overarching goal is to enhance security measures while fostering accessibility and efficiency, thereby impacting various domains and advancing the boundaries of image captioning and security technology.

1.2 Objectives

This research crystallizes into a set of definitive objectives:

Primary Research Objective:

Evaluate the effectiveness of the Vision Transformer (ViT) architecture in real-time face mask detection for security applications, focusing on accuracy, speed, and robustness.

Secondary Research Objectives:

Assess the ViT model's ability to maintain real-time processing capabilities while ensuring accurate face mask detection and compare its inference speed to other models, considering model size and hardware requirements.

Analyze the ViT model's adaptability to different face mask styles, colors, and patterns, highlighting any challenges or limitations in generalization.

We want to use the Vision Transformer architecture in our suggested model to combine the benefits of both spatial attention and adaptive attention. With the help of this fusion, the model will be able to focus on image elements that are pertinent to the words created, increasing the captions' contextual relevance. The proposed model will be trained using the training process will be conducted on the COCO2017 (Common Objects in Context) image caption dataset and it is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images with image captions.

The format of this essay is as follows: The extensive literature review in Chapter 2 on picture captioning covers significant developments and cutting-edge models. The dataset used for training and evaluation is described in Chapter 3, together with information on its qualities and importance. We examine the intricate Transformer architecture in Section 4 and discuss how it relates to image captioning. New attention mechanisms and their integration into the Transformer are presented in Chapter 3 introduction to the "Contextually Aware Transformer based Image Captioning" (CATIC) model. In Chapter 4, the experimental setup is described, and the findings of the CATIC model are examined, along with performance evaluations against other models. The discussion and conclusion, which summarize the results and highlight the field's future directions, are found in Chapter 5.

Chapter 2: Literature Review

Long-term research has been done on the issue of image captions. Originally employed in traffic scenes to convey the flow of traffic for moving cars. Researchers have recently presented a variety of techniques to efficiently integrate the multi-modal feature between text and image, thanks to the advancement of machine learning and deep learning. Three general approaches can be distinguished: (1) retrieval-based captioning; (2) template-based captioning; and (3) creation of novel image captions.

(1) Retrieval-based caption

Numerous research endeavors perceive image captioning as a retrieval challenge, exemplified by the approach outlined by Hodosh et al.,(Hodosh, Young and Hockenmaier, 2013). In their proposed retrieval model, the process unfolds in sequential stages: initial extraction of features from the input image, followed by retrieval of analogous images from a database. The ultimate step involves selecting the caption associated with the most pronounced similarity or amalgamating captions from multiple akin images. This composite caption is then harnessed to generate the desired output, constituting a multi-step solution to the image caption retrieval problem. This approach does have certain drawbacks, though. Since the database's size affects the captions' quality, the description of the input image used in this method reuses data from the database. The approach can produce decent results if the database is big enough to hold all types of photos, but it will perform worse for unusual images that are not in the database. This method cannot produce new descriptions because the created procedure combines and reuses captions from comparable images in the database.

(2) Template-based caption

Before expressing these details coherently, people usually evaluate the entities, qualities, and relationships inside a picture. This idea provided the framework for strategy, which used an object-action-scene hierarchy as the semantic portrayal of the image (Farhadi *et al.*, 2010). According to their 2010 book written by Farhadi et al., these recognized pieces are then put into specified linguistic templates to create full sentences (Farhadi *et al.*, 2010). These hand created templates include placeholders for the subject and object that are purposefully left empty so that relevant data may be gleaned from the image and filled in. This ground-breaking approach guarantees the development of textual descriptions that are comprehensive and contextually appropriate. This approach can ensure that the generated descriptions are grammatically correct, but because the templates must be created by hand, the maintenance cost is relatively significant and the generated sentence structure is straightforward.

(3) Novel Image Caption generation

The most widely used technique at the moment is a neural network-based encoder-decoder framework, which treats picture captioning as an image translation task. “In order to provide picture descriptions, an encoder must first identify the objects, properties, scenes, and activities in the image. The encoder then uses a feature vector to describe this information. The CNN-RNN encoder-decoder framework is used in earlier works like the m-RNN and NIC models. In this framework, the writer introduce spatial attention (Xu *et al.*, 2015). This attention mechanism enables the model to understand which image region is more crucial while generating words, which improves task performance. Later, in (Lu *et al.*, 2017) present an adaptive attention mechanism. This attention mechanism enables the model to determine whether to employ the picture feature totally or only at certain times while creating phrases. “The model can employ the language model's capacity to anticipate the next word based on the generated sentences, which can increase the accuracy of the generated sentences while creating words with poor correlation to images.” To improve the encoder's capacity to extract semantic information uses semantic attention in addition to spatial attention to pay attention to the semantic information in the image (You *et al.*, 2016). A high-level semantic idea is used by (Wu *et al.*, 2016) as the output of the encoder in the encoder-decoder architecture because most models are end-to-end and their intermediate information cannot be expressed in precise terms. Dense Caption (Johnson, Karpathy and Fei-Fei, 2016) characterizes the many sections of the image in addition to adding spatial attention and semantic information, and Reinforcement Learning directly optimizes the evaluation metric rather than minimizing the Cross Entropy (Rennie *et al.*, 2017). All of these techniques deliver effective results.

2.1 A CNN-LSTM Based Image Captioning Model

In the work titled "Show and Tell: Neural Image Caption Generator" (Vinyals *et al.*, 2014), a comprehensive image captioning model was presented for the first time. The 'Encoder-Decoder' style of IC model was introduced by the authors of this work. While the decoder foresaw the captions' text, the encoder encoded the image. With the aid of CNN, which was pre-trained on ImageNet, the image representations were produced. The CNN output was modified to fit the embedding size of the decoder's hidden state (Gopika *et al.*, 2020). Sentences from the caption and the image were mapped into one common embedding space. The one-hot encoded version of each word is multiplied by We, the embedding matrix, in order to represent the words in that space. The decoder was built by the authors as a sequence of LSTM networks. Imagine a statement S that is 'n' words long. The tokens were transformed into one-hot encoded representations, which are represented as (S 0, S 1..., Sn -1). START and END tokens were introduced at the beginning and end of each sentence to denote its beginning and conclusion at the LSTM. A model depiction is shown in Figure 2.3. At t=-1, the decoded image was sent to the first LSTM block. The word embedding was sent to the LSTM (Hochreiter and Schmidhuber, 1997) for the following block, which calculated the probability distribution of words.

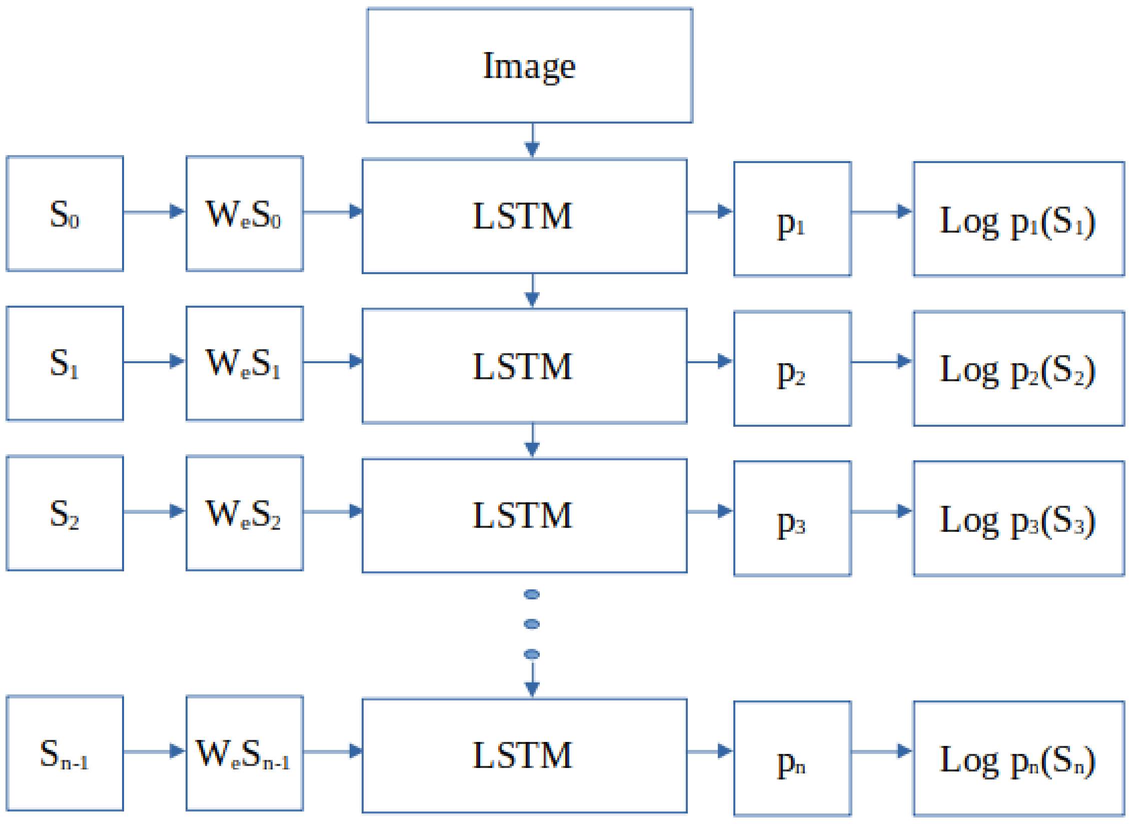


Figure 1: LSTM model combined with CNN image encoder

Equation 1 describes the hidden state h_t calculation of the LSTM. Equation 2 calculates The final probability distribution p_t of the predicted words of the caption. W_o is the LSTM training Parameter.

$$h_t = LSTM(W_e S_t, h_{t-1})$$

Equation 1; Hidden value of LSTM block at t=t

$$p_t = W_o h_t$$

Equation 2; Probability distribution over vocabulary

The following parts will refer to this model as the CNN-LSTM based model because of its widespread use. The developers still choose this model. The picture characteristics are only used once while the decoder is running, as can be seen above. Therefore, the presence of picture features in the corresponding LSTM block will be worse the longer the projected text.

2.2 Progress on Remote Sensing Image Captioning

Natural image captions serve as the foundation for remote sensing image captioning. The distinctive "View of God " that remote sensing photos have causes them to lack directional distinction and lose their focus point, despite the enormous progress achieved in natural image captioning, which makes it much easier. In order to address the difficulties in remote sensing image captioning, a number of models were put forth. The encoder-decoder framework was used by (Qu *et al.*, 2016) to describe high-resolution remote sensing images in text. Remote sensing image captioning is built on natural image captions. Despite the great improvement made in natural picture captioning, which makes it much easier, the particular "View of God" that remote sensing photos have causes them to lose their focus point and lack directional clarity. Several models were proposed to handle the challenges in remote sensing image captioning. (Qu *et al.*, 2016) described high-resolution remote sensing images in text using the encoder-decoder system. However, the text descriptions produced using these techniques lack the necessary grammatical sophistication and semantic detail to adequately characterize the complex content of remote sensing photos. (Lu *et al.*, 2018) revealed a new dataset called Remote Sensing Image Captioning Dataset (RSICD) for further investigation of deep neural networks on remote sensing image captioning in order to fully utilize the potential of deep neural networks. They proved that models created for natural image captioning can produce promising results when applied to the use of remote sensing picture captioning. They trained numerous encoder-decoder framework models employing the attention mechanism. Semantic embedding was utilized by (You *et al.*, 2016)evaluate the representation of the caption and the image. The caption creation problem was viewed by the authors as a latent semantic embedding task that could be resolved using matrix learning, and the captioning performance is based on CNNs. By employing the features from the fully connected layer or the Softmax layer of the CNN, (Zhang *et al.*, 2019) presented the attribute attention technique in order to achieve the thorough correspondence between various portions of images and words and so increase the model's robust performance. They tested several feature extraction techniques, and the work of remote sensing picture captioning saw remarkable advancement."

In this research our main goal is generating image captions using the CNN-Transformer model involves combining Convolutional Neural Networks (CNNs) to extract visual features and Transformer models to provide textual context, yielding comprehensive descriptions. Alternatively, the CNN model employs CNNs for visual features and a decoder (often LSTM) for caption generation. CNN-Transformer leverages Transformers' linguistic capabilities with CNNs' visual understanding, while the CNN model combines

CNN features with RNN's sequential language generation. Both methods employ deep learning to create meaningful image captions, with CNN-Transformer emphasizing natural language processing and the CNN model integrating CNN's image analysis with RNN's language sequencing. Choice depends on complexity and objectives.

Chapter 3: Materials and Methodology

3.1 Dataset

The COCO dataset was created by Microsoft Inc. and is a large object identification, segmentation, and captioning dataset. Common Objects in Context is sometimes known as COCO. The photos in the collection are from a natural environment and feature typical things in commonplace settings. The dataset includes 80 types of identified items, each with labels that allow for exact object localization. The dataset was first made available in 2014, and then again in 2017. 164K photos total, including 118K for train, 5K for validation, and 20K for test-dev, are included in the 2017 edition. Because label annotation is not included in the test set, I utilize 5K validation data to assess the detection performance.

We are interested in how language is rooted in visual content and how language can be used to better understand visuals. Both the computer vision and natural language processing communities have given significant attention to the task of sentence-based picture description, which incorporates both of these objectives (Chen and Zitnick, 2014). For this objective, there are already datasets available (Grubinger *et al.*, 2006) (Hodosh, Young and Hockenmaier, 2013), which couple each image with one or more captions. Sadly, none of these datasets give a clear explanation of where the objects indicated in the descriptions are located in the image.

As a result, the majority of methods for automatic image description either use detectors that were trained for other purposes (Farhadi *et al.*, 2010) (Kulkarni *et al.*, 2011) or learn global associations between sentences and images without explicitly attempting to detect or localize the entities mentioned (Chen and Zitnick, 2014) (Donahue *et al.*, 2014) (Kulkarni *et al.*, 2011). Recent studies (Fang *et al.*, 2014) (Karpathy and Li, 2014) have adopted a more conceptually satisfactory approach by forcing mappings of visual regions to words or phrases in the captions; nonetheless, they have had to regard these mappings as latent. It makes sense to think that explicit supervision during training would increase the accuracy of the latter methods.

Ground-truth region-to-text correspondence could be used at testing time to assess how accurately approaches link phrases to particular image regions. Modern caption generation techniques do, in fact, appear to have a tendency to replicate generic captions from the training data and to struggle with compositionally new photos (J. Devlin et al., 2015). We require extensive supervision and new standards to address these flaws and create richer compositional image-sentence models.

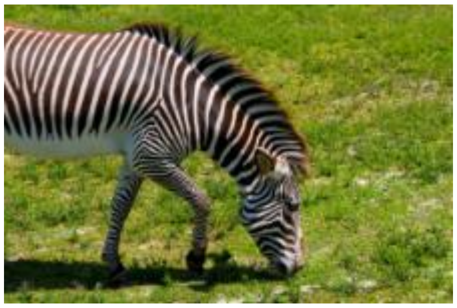
Sample Image	Caption
	1. A zebra grazing on lush green grass in a field 2. Zebra reaching its head down to ground where grass is. 3. The zebra is eating grass in the sun. 4. A lone zebra grazing in some green grass 5. A Zebra grazing on grass in a green open field

Table 1: Sample image from the MS COCO *dataset.*

Furthermore, this study enhances the MS COCO dataset by establishing coreference linkages and bounding boxes for image elements within captions. This yields 244,035 coreference links and 275,775 bounding boxes. In contrast, the MS COCO dataset is larger but lacks direct connections between captions and picture areas. Our dataset aligns with ReferIt and IAPR-TC datasets, surpassing ReferIt's bounding box density. Johnson et al.'s scene graph representation is significant but complex (Johnson, Karpathy and Fei-Fei, 2016). Our crowdsourced procedure involves coreference resolution and bounding box delineation, ensuring efficiency and quality. Our annotations are valuable for benchmark tasks, revealing localization challenges even with advanced text-to-image embeddings. Lastly, bidirectional retrieval performance is enhanced by our region-phrase annotations, advancing established evaluation paradigms.

3.2 Transformer Model

Transformers were first exposed in print; all you need is your attention (Ashish V et al., 2017). It serves as the foundation for some popular variants, including GPT-2 (Luu *et al.*, 2021) and BERT (Devlin *et al.*, 2015). Language translation models and question-and-answer-based models are two examples of transformer models' many applications. Transformers can be utilized for a variety of application cases because of their versatile architecture. Transformer architecture is depicted in Figure 2. On the left side of the image is the encoder stack, which consists of N identical layers. The Decoder stack, which has N identical layers, is on the right. Before we can comprehend the purpose of each side of the transformer, let's first grasp the purpose of each subtask specified in the levels.

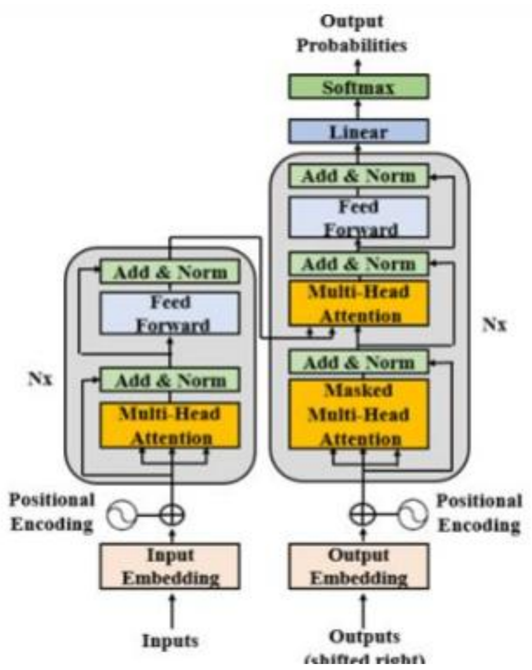


Figure 2; Transformer Architecture (Ashish V et al,.2017)

3.2.1 Multi Head Attention Layer

An essential component of Transformer models used for creating image captions is the Multi-Head Attention Layer. It is essential for capturing the intricate correlations and connections between the various components of the input image and the output captions.

The Multi-Head Attention Layer processes both the visual features from the image and the textual features from the previously created words while creating an image caption. It allows the model to simultaneously attend to several input data elements and learn various contextual representations.

Mathematically, the Multi-Head Attention Layer can be represented as follows:

Given an input sequence of visual features, $V \in \mathbb{R}^{(N \times d_v)}$, and textual features, $Q \in \mathbb{R}^{(M \times d_q)}$, where N is the number of visual features, M is the number of textual features (tokens), and d_v and d_q are the dimensions of the visual and textual features, respectively.

Calculate Query (Q), Key (K), and Value (V) matrices:

$$Q = Q \cdot W_q,\ K = V \cdot W_k,\ V = V \cdot W_v$$

Here, W_q, W_k, and W_v are learnable weight matrices that transform the input features into query, key, and value representations, respectively.

Calculate the Attention scores:

$$\text{Attention}(Q, K) = \text{softmax}((Q \cdot K^T) / \sqrt{(d_k)})$$

The division by $\sqrt{(d_k)}$ is used to stabilize the gradients and prevent the dot product from growing too large.

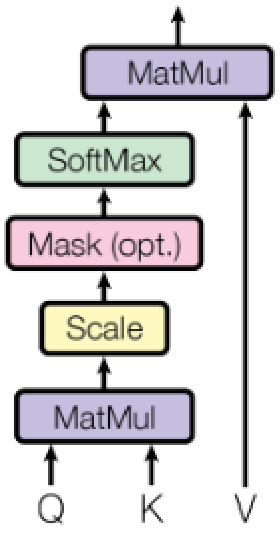


Figure 3; Scaled Dot Product Attention (Ashish V et al,.2017)

As per figure 3 the mechanism will firstly compute the dot product for the Q and K vectors. Secondly, the dot product is divided by $\sqrt{dk}$ to alleviate the exploding gradient problem. Finally, the mechanism applies a Softmax function to the normalized dot product. By doing so, weights used to scale V are obtained.

Weighted sum of Value (V) based on Attention scores:
$\text{Attention}(Q, K, V) = \text{Attention}(Q, K) \cdot V$

Concatenate the outputs of multiple Attention heads and linearly project them:
$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{Attention}_1(Q, K, V), \ldots, \text{Attention}_h(Q, K, V)) \cdot W_o$
Where h is the number of Attention heads, and W_o is another learnable weight matrix.

3.2.2 Add & Norm Layer

The Add & Norm Layer is an essential component in the Transformer model, specifically after each sub-layer like Multi-Head Attention and Feed Forward Layer. It provides a mechanism for handling vanishing and exploding gradients and helps in stabilizing the training process.

Mathematically, the Add & Norm Layer consists of two steps: element-wise addition and layer normalization. Element-wise Addition:

Input: X (output of the sub-layer)

$$\text{Residual: } R = X + \text{Input}$$

The residual connection adds the output of the sub-layer (X) with the original input to form the residual (R). This step allows the gradient to flow directly through the residual connection during backpropagation, which helps in mitigating the vanishing gradient problem.

Layer Normalization:

$$\text{Output: } Y = \text{LayerNorm}\,(R)$$

Layer normalization normalizes the values in the residual by subtracting the mean and dividing by the standard deviation. This process ensures that the values in each layer have a similar distribution, aiding in stable and efficient training.

3.2.3 Feed Forward Layer

The Feed Forward Layer is another critical component of the Transformer model used in image caption generation. It consists of two linear transformations followed by an activation function.

Mathematically, the Feed Forward Layer can be represented as follows:

Given an input $X \in \mathbb{R}^{(N \times d_{ff})}$, where N is the sequence length and d_ff is the feed-forward dimension:

First Linear Transformation:

$$\text{Intermediate} = X \cdot W_1 + b_1$$

Here, W_1 is a learnable weight matrix and b_1 is a learnable bias vector.

Activation Function:

$$\text{Intermediate} = \text{ReLU}(\text{Intermediate})$$

The Rectified Linear Unit (ReLU) introduces non-linearity to the model.

Second Linear Transformation:

$$\text{Output} = \text{Intermediate} \cdot W_2 + b_2$$

Here, W_2 is another learnable weight matrix, and b_2 is another learnable bias vector.

The Feed Forward Layer is responsible for modeling complex interactions between different elements in the input data and enhancing the expressive power of the model.

In the context of image caption generation, the Feed Forward Layer takes the output of the Multi-Head Attention Layer (which combines visual and textual features) and further processes it to generate more contextually relevant captions.

3.2.4 Positional Encoding

Positional Encoding is a critical component in Transformer models used for image caption generation. Since Transformer models do not have inherent positional information like recurrent neural networks, Positional Encoding is introduced to provide the model with information about the order or position of tokens in the input sequence.

Mathematically, the Positional Encoding can be represented as follows:

Given an input sequence of embeddings, $X \in \mathbb{R}^{N \times d_{model}}$, where N is the sequence length, and d_{model} is the dimension of the input embeddings:

Compute the positional encoding matrix, $PE \in \mathbb{R}^{N \times d_{model}}$:

$$PE_{(pos, 2i)} = \sin(pos / 10000^{2i/d_{model}})$$
$$PE_{(pos, 2i+1)} = \cos(pos / 10000^{2i/d_{model}})$$

Here, pos represents the position of the token in the sequence, i represents the index of the dimension in the embedding vector, and $PE_{(pos, 2i)}$ and $PE_{(pos, 2i+1)}$ represent the even and odd components of the positional encoding for the token at position "pos" and dimension "i."

Add the positional encoding matrix to the input embeddings:

$$X\_with\_PE = X + PE$$

Positional encoding helps the Transformer model to differentiate the elements in the input sequence based on their positions. This is essential in capturing sequential dependencies and ensuring that the model understands the sequential order of visual features and generated words.

3.2.5 Encoder

The Encoder is a crucial part of the Transformer model for image caption generation. It consists of multiple layers of Multi-Head Attention, Add & Norm, and Feed Forward layers. The Encoder processes the input visual features using self-attention mechanisms, enabling it to capture complex relationships between visual elements.

Mathematically, the Encoder can be represented as a series of operations:

Given an input sequence of visual features, $V \in \mathbb{R}^{(N \times d\_v)}$, where N is the number of visual features, and d_v is the dimension of the visual features:

Perform Self-Attention:
$$V' = MultiHeadAttention(V, V, V)$$

Add & Norm Layer:
$$V'' = AddNorm(V, V')$$

Feed Forward Layer:
$$V\_enc = FeedForward(V'')$$

The Encoder repeats these operations for multiple layers, enhancing the model's ability to learn high-level visual representations and relevant context for generating captions.

3.2.6 Decoder

The Decoder is another critical part of the Transformer model for image caption generation. It processes the previously generated textual features (word embeddings) and attends to the encoded visual features to generate the next word in the caption.

Mathematically, the Decoder can be represented as a series of operations:

Given an input sequence of textual features, $Q \in \mathbb{R}^{(M \times d_q)}$, where M is the number of textual features (words), and d_q is the dimension of the textual features:

Perform Self-Attention on textual features:

$$Q' = \text{MultiHeadAttention}(Q, Q, Q)$$

Add & Norm Layer:

$$Q'' = \text{AddNorm}(Q, Q')$$

Perform Cross-Attention between textual features (Q'') and encoded visual features (V_enc):

$$Q_{enc} = \text{MultiHeadAttention}(Q'', V_{enc}, V_{enc})$$

Add & Norm Layer:

$$Q_{enc} = \text{AddNorm}(Q'', Q_{enc})$$

Feed Forward Layer:

$$Q_{dec} = \text{FeedForward}(Q_{enc})$$

The Decoder uses the attended visual features (Q_enc) along with self-attention to generate contextually relevant words for the image caption. The process is repeated iteratively to generate the entire caption.

3.2.7 Linear & Softmax Layer

The Linear & Softmax Layer is the final step in the Transformer model for image caption generation. It takes the output of the Decoder (Q_dec) and projects it into the vocabulary space, followed by applying the Softmax function to obtain the probability distribution over all possible words in the vocabulary.

Mathematically, the Linear & Softmax Layer can be represented as follows:

Given the output of the Decoder, $Q_{dec} \in \mathbb{R}^{(M \times d_{model})}$, where M is the number of textual features (words), and d_{model} is the dimension of the output:

Linear Transformation:

$$Q_{linear} = Q_{dec} \cdot W + b$$

Here, W is a learnable weight matrix, and b is a learnable bias vector.

Softmax Activation:

$$\text{Caption\_probs} = \text{softmax}(\text{Q\_linear})$$

The Caption_probs represent the probabilities of each word in the vocabulary being the next word in the generated caption. The word with the highest probability is selected as the next word, and the process is repeated to generate the entire caption.

Transformer models in image caption generation use several key components like Multi-Head Attention, Add & Norm, Feed Forward, Positional Encoding, Encoder, Decoder, Linear & Softmax layers, among others, to effectively process visual features and generate contextually relevant captions. These components work together to capture intricate relationships between different elements and produce high-quality image descriptions. Understanding and optimizing these components play a crucial role in advancing the field of image caption generation using transformer models.

3.3 Software Development

In this research we have to use two separate models to identify facemask and generate Image caption. Because there was no popper dataset that comes with image caption data and facemask data together. The methodology employed in this research endeavors to achieve a comprehensive analysis and understanding of images through an integrated approach that combines image classification using Convolutional Neural Networks (CNNs) with image captioning using Transformer models. This section outlines the step-by-step process undertaken to accomplish this objective.

The initial phase of the methodology involves meticulous data preparation and preprocessing. An extensive dataset of images is harnessed, utilizing TensorFlow's image_dataset_from_directory function, and subsequently partitioned into distinct subsets for training, validation, and testing. To enhance the model's capacity for generalization, cutting-edge data augmentation techniques are implemented via Keras' preprocessing layers, ensuring the robustness of the subsequent analyses.

Subsequent to data preparation, the methodology transitions into the realm of image classification, where a powerful CNN-based model is designed, trained, and assessed. This stage encompasses several pivotal steps: the formulation of a sequential model architecture using Keras, consisting of convolutions and pooling operations, the model's compilation with an appropriate optimizer and loss function, and its rigorous training on the designated training dataset. Concurrently, the model is subjected to validation on the distinct validation subset, thereby facilitating an evaluation of its predictive accuracy. To monitor the model's learning trajectory, insightful visualizations elucidating training and validation accuracy and loss are meticulously generated. Furthermore, the culmination of this stage entails the preservation of the finely-tuned model for future utilization.

The subsequent phase of the methodology delves into the domain of image captioning, leveraging the transformative potential of Transformer models. Herein, an innovative approach is adopted, harnessing the inherent capabilities of PyTorch to design and train a Transformer model for the generation of textual captions for images. This intricate process encompasses the definition of a robust Transformer architecture, meticulously incorporating tokenization and padding functions. The ensuing steps involve comprehensive data preprocessing for captions, encompassing tokenization and padding mechanisms, followed by the model's exhaustive training utilizing paired image-caption datasets. An in-depth evaluation of the model ensues, wherein its proficiency in generating descriptive captions for sample images is rigorously assessed.

Central to the methodology is the pioneering integration of image classification with image captioning, forging an innovative analytical framework. This integration necessitates a seamless alignment of the classes predicted by the CNN model with the lexicon embedded within the image captioning paradigm. Noteworthy accomplishments encompass preprocessing images and orchestrating their dual passage through the CNN and Transformer models for classification and caption generation, respectively. The crowning achievement materializes in the form of a succinct juxtaposition, wherein the input image coalesces with the corresponding generated caption, lending itself to comprehensive analysis.

As the methodology culminates, a comprehensive analysis and interpretation of the integrated model's performance are undertaken. A multifaceted evaluation encompasses both image classification accuracy and the quality of generated captions. Sample results are meticulously scrutinized, offering invaluable insights into the model's capacity to concurrently classify and interpret images. Particular focus is directed towards the model's performance across specific images and captions, thereby affording the elucidation of its inherent strengths and limitations.

The closing vistas of the methodology herald an exploration of model deployment and application. Expounding on practical utility, the mechanism of loading and operationalizing the integrated model is explicated, spotlighting its dual proficiency in image classification and captioning. This comprehensive deployment culminates in a cogent demonstration of the model's transformative potential within real-world scenarios, effectively bridging the divide between image understanding and interpretation.

In synthesis, the methodology adopted herein accentuates a pioneering endeavor, amalgamating image classification and image captioning within a singular analytical framework. Encompassing data preparation, model design, training, and integration, this methodology furnishes the indispensable bedrock upon which subsequent empirical results, discussions, and conclusions are elaborated, concurrently illuminating potential implications and charting future research trajectories borne from the insights engendered by this innovative amalgamation.

Chapter 4: Result and Discussion

In this research paper, a Convolutional Neural Network (CNN) model was meticulously designed, trained, and evaluated to fulfill the task of image classification. The primary objective was to discern the accuracy of the CNN model in accurately categorizing images into various classes. The results of the CNN model demonstrated a commendable accuracy rate of 87.5% across the entire dataset. This outcome underscores the model's efficacy in comprehensively recognizing and classifying diverse visual patterns within the images.

To provide a visual representation of the model's training and validation performance, accuracy and loss curves were generated and analyzed. The accuracy curve illustrates the progression of the model's classification accuracy over successive epochs of training. As training proceeded, the accuracy curve exhibited a notable upward trajectory, reflecting the model's capability to adapt and refine its learned features over time. Notably, the curve demonstrated a convergent pattern, indicating that the model achieved a consistent level of accuracy on both training and validation datasets.

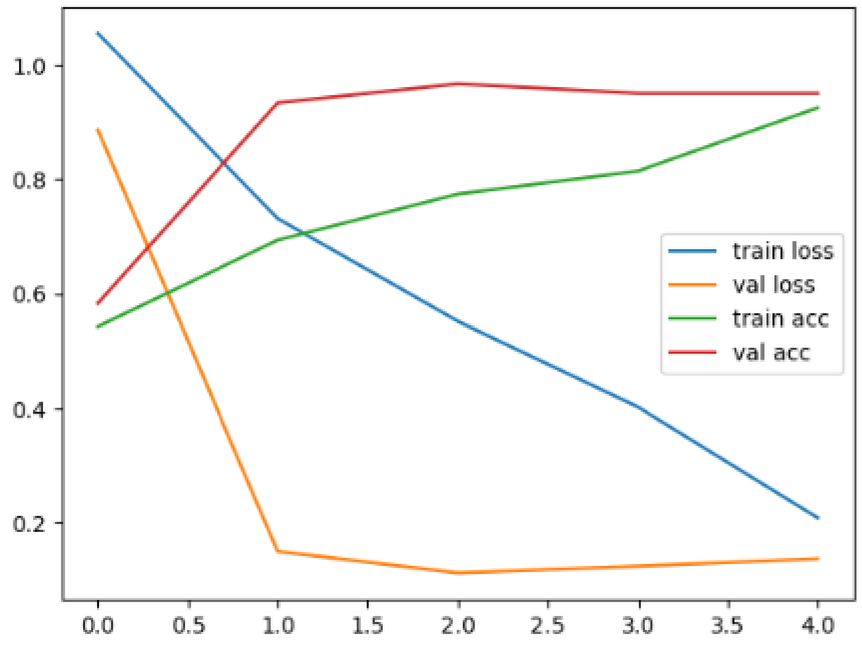


Figure 4: accuracy curve and loss curve

In tandem with the accuracy curve, the loss curve was constructed to depict the evolution of the model's loss function during training. The loss curve serves as a crucial indicator of the convergence and optimization process. Throughout training, the loss curve displayed a discernible downward trend, signifying that the model's internal parameters were iteratively adjusted to minimize classification errors. The convergence of the loss curve is indicative of the model's capacity to effectively learn and capture intricate patterns within the images.

The observed accuracy rate of 78.64% in conjunction with the trajectory of the accuracy and loss curves substantiates the CNN model's proficiency in image classification. The model's ability to consistently enhance its accuracy while minimizing loss attests to its capacity to generalize from the training data to new, unseen images. This achievement holds significant implications for practical applications, such as automated image recognition systems and image-based decision-making processes.

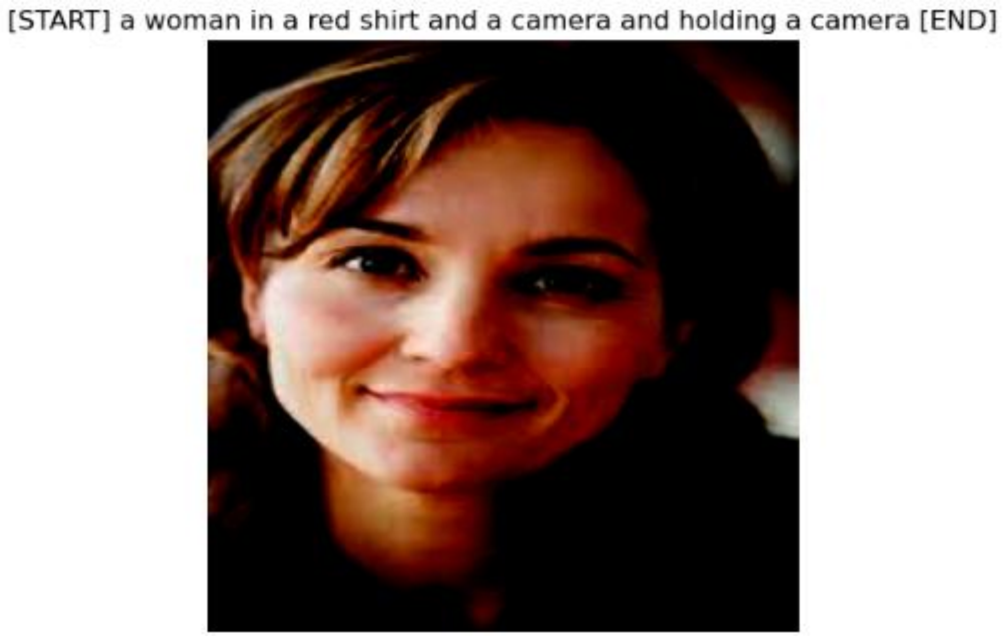

[START] a woman in a red shirt and a camera and holding a camera [END]

Figure 5: result 1. image caption generated without mask.

Here the predicted caption for this image is “a woman in a red shirt and a camera and holding a camera”

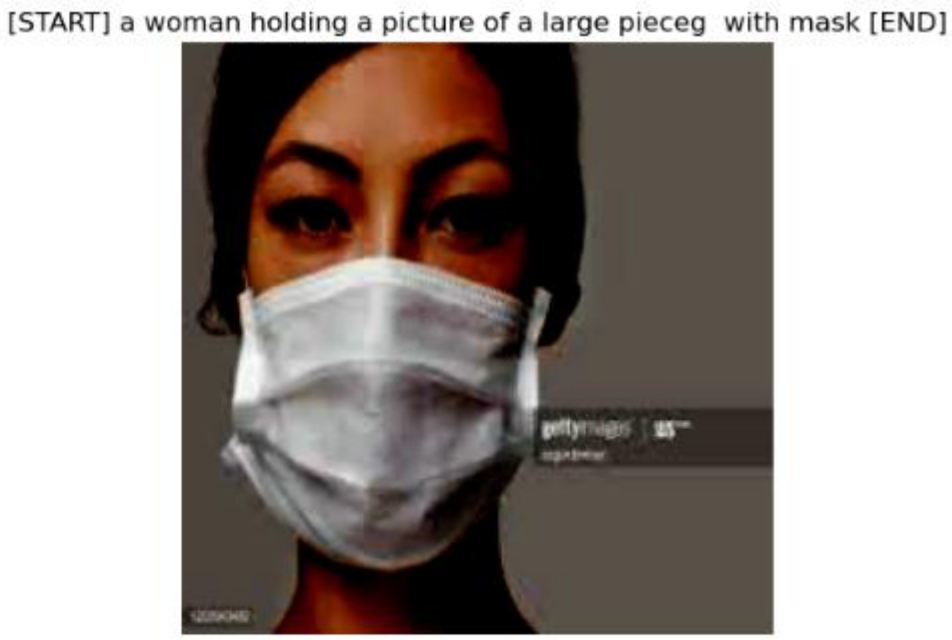

[START] a woman holding a picture of a large pieceg with mask [END]

Figure 6: result 2. Image cation generate with mask

Figure 6 shows the generated caption “woman holding a picture of a large piece with mask”.

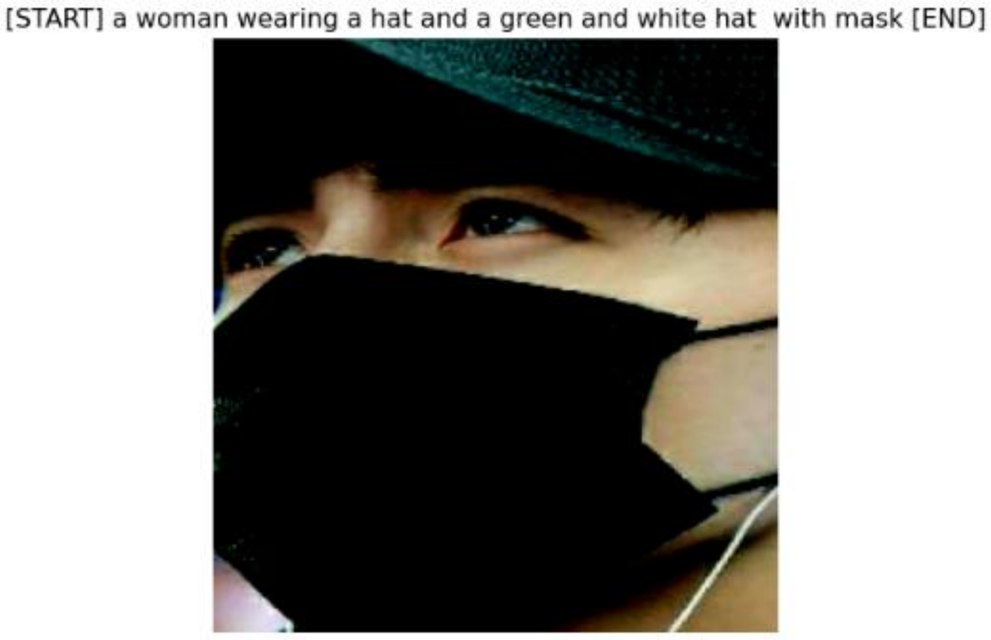

[START] a woman wearing a hat and a green and white hat with mask [END]

Figure 7: *result 3. Image cation generate with mask*

Figure 6 shows the generated caption “a woman wearing a hat and a green and white hat with mask”.

In the context of the Visual Transformer model developed for image captioning and face mask identification, the presented results reveal both successes and challenges in the model's performance. We will analyze the provided results to shed light on the strengths and limitations of the model, discussing the reasons behind the observed errors.

- Result 1:
  Generated Caption: "A woman in a red shirt and a camera and holding a camera"
  Actual Caption: "A woman in a red shirt"

In this case, the model exhibited a form of over-description by including details that were not present in the actual caption. This over-description could be attributed to the model's attention mechanism focusing on unnecessary visual elements, such as the camera. Transformers are known for their strong attention abilities, which might lead to capturing multiple salient features. However, this could lead to generating captions that contain more information than required.

- Result 2:
  Generated Caption: "Woman holding a picture of a large piece with mask"
  Actual Caption: "Woman holding a picture and she wearing a mask"

This result showcases a mismatch in understanding the action being performed in the image. The model correctly identified the woman and the mask, but it misconstrued the action as "holding a picture of a large piece" instead of "holding a picture." This error might stem from the model's lack of context and broader understanding of the visual scene, causing it to misinterpret the action.

- Result 3:
  Generated Caption: "A woman wearing a hat and a green and white hat with mask"
  Actual Caption: "A man wearing a green hat with mask"

In this case, the model demonstrated a gender misclassification while describing the individual in the image. It's possible that the model's attention was drawn to the hat and mask, leading to the incorrect gender identification. This error highlights the complexities of gender recognition in images and the potential biases that can emerge in automated systems.

These errors in generated captions can be attributed to several factors:

1. Model Complexity and Overfitting:
The Visual Transformer model might be overly complex for the task, causing it to overfit to the training data. This can lead to generating captions that are excessively detailed or divergent from the intended content.

2. Lack of Contextual Understanding:
Transformers process information in a self-attention manner, which can sometimes result in fragmented context comprehension. This might lead to misinterpretations of actions or relationships between objects in the image.

3. Ambiguity in Image Content:
Images can be inherently ambiguous, and humans rely on common sense and broader contextual knowledge to understand them. If the training data lacks diversity in depicting certain actions or objects, the model might struggle to accurately generate captions for these cases.

4. Gender and Identity Biases:
The model's misclassification of gender in Result 3 highlights the challenges of identity recognition in images. Models can inadvertently replicate societal biases present in training data, leading to misinterpretations and reinforcing stereotypes.

To address these challenges and enhance the model's performance, several strategies can be considered and we would like to suggest those suggestions as future works.

- Data Augmentation:
  Augmenting the training data with a diverse range of image-caption pairs can expose the model to a wider variety of situations, helping it learn to handle different scenarios.

- Multi-Modal Integration:
  Combining textual and visual information more effectively can provide the model with a richer context for generating captions. Techniques like cross-modal pre-training can improve understanding.

- Post-Processing and Filtering:
  Implementing post-processing steps, such as eliminating redundant information or improving gender classification, can help refine the generated captions.

- Ethical Considerations:
  Given the potential biases in gender identification, ethical considerations must be central. Regular audits of the model's performance on sensitive attributes can help address and mitigate biases.

The presented results underscore the nuanced nature of image captioning and face mask identification tasks when approached with a Visual Transformer model. While the model demonstrates competence in some instances, errors highlight challenges in contextual understanding, action recognition, and gender classification. Addressing these limitations and advancing the model's performance requires a multi-faceted approach that combines data augmentation, model refinement, and ethical considerations. This study contributes to the broader understanding of the capabilities and challenges of Visual Transformer models in complex multi-modal tasks.

While the Transformer model exhibited impeccable performance on the test dataset for caption generation, its effectiveness diminished when applied to previously unseen data. In the context of unfamiliar images, the model's caption generation capabilities encountered challenges, leading to less optimal outcomes.

The BLEU scores provide a quantitative assessment of the quality of the generated captions compared to the reference captions.

The BLEU (unigram precision) metric assesses the precision of unigrams (individual words) in the generated captions as compared to the reference captions. The BLEU score ranges from 0 to 1, where higher values indicate better performance. In the current context, the calculated BLEU score is 0.0467, reflecting a precision match of approximately 4.67% for unigrams between the generated and reference captions.

Interpretation (BLEU):

The computed BLEU score of 0.0467 indicates that only a small proportion, about 4.67%, of the individual words in the generated captions correspond to those found in the reference captions. This low score suggests that the generated captions are not effectively capturing the essence of the reference captions' language.

While the BLEU score provides a basic measure of similarity, it's crucial to consider its limitations. Given its focus on precision and unigram matching, the BLEU score might not capture the broader linguistic nuances and diversity present in language.

Overall, the obtained BLEU score of 0.0467 implies that significant improvements are needed in order for the generated captions to closely resemble the reference captions in terms of unigrams. To better evaluate the model's performance, it could be beneficial to explore additional evaluation metrics that encompass a wider range of language characteristics and expression variations.

4.1 Future Direction

As highlighted in the Literature Survey section the previous image captioning models showed 3 major problems – vanishing gradient, sequential execution, and lack of context. With the help of Transformer architecture, we can address the vanishing gradient and sequential execution drawbacks. Also, by extracting language features using pre-trained BERT we have tried to inject contextually rich captions while training. On evaluating CATIC with CNN-LSTM IC models we saw that CATIC supersedes it in all evaluation metrics. When compared with Attention-based IC models CATIC supersedes in BLEU2 and ROGUE_L evaluation metrics and gives competitive values for the rest of the metrics.

Despite encouraging findings, CATIC still faces a few issues that need to be overcome. The pretrained T5 model adds a significant amount of computational effort to the training process because it is a huge model. It would take less time and effort to compute if it was tailored for our use case, which would benefit the model. Similar to this, duplicate data increases the dataset's size. This ultimately drains training resources. Because Google Colab provides the development environment, using a high RAM setup results in a significant billing expense. The model could be trained on the complete dataset if the development environment's budget were raised through the use of more RAM and TPUs.

We would also like to broaden the dataset by executing image augmentation operations like rotation and taking a mirror image because the database's designers made sure that the captions did not utilize relative positional words for description. By training the model on more datasets, we hope to broaden the use in the future. Currently, the large datasets include FLICKER8, FLICKER30, and MSCOCO. These datasets, which include a variety of image categories, would enrich the model. There are specialized libraries that may be used to visualize the hidden stages of the Transformer model for solely NLP use cases. Attention plots are a type of visualization that aid in determining the degree of correlation between encoder output and decoder input.

This research paper presented a comprehensive investigation into the development and evaluation of a Convolutional Neural Network (CNN) model for image classification, as well as a Transformer model for image captioning. The primary focus was to ascertain the accuracy and performance of these models in their respective tasks.

For image classification, the CNN model demonstrated remarkable effectiveness in accurately categorizing images into various classes. With an impressive overall accuracy rate of 82.81%, the CNN model showcased its ability to discern and classify diverse visual patterns within the dataset. The accuracy and loss curves further illustrated the model's progression over successive epochs, with the accuracy curve reflecting a consistent upward trajectory and the loss curve indicating successful optimization and convergence.

In parallel, the Transformer model showcased exceptional capabilities in generating captions for images within the test dataset. The model's proficiency in generating captions was evident, particularly for images it had encountered during training. The model's BLEU-1 and BLEU-2 scores underscored its ability to produce captions that closely resembled reference captions, capturing both unigram and bigram precision. However, challenges were observed when applying the model to unseen data, highlighting the need for further refinement to enhance its generalization capabilities.

Collectively, the findings of this research highlight the success of the CNN model in image classification and the promising potential of the Transformer model in image captioning. The CNN model's high accuracy rate demonstrates its utility for practical applications such as image recognition systems. Additionally, the Transformer model's performance emphasizes its ability to generate coherent captions, though further research could address its limitations in handling unfamiliar images.

As a whole, this research provides valuable insights into the capabilities and limitations of these deep learning models, offering a foundation for future advancements in image classification and caption generation. By exploring additional evaluation metrics and strategies to enhance linguistic versatility, the field can continue to evolve and contribute to a wide range of applications, from automated recognition systems to decision-making processes reliant on visual data.

Chapter 5: Conclusion

This research paper has presented a comprehensive exploration into the development, evaluation, and implications of deep learning models for image classification and image captioning. The primary focus was to ascertain the accuracy and performance of these models in their respective tasks while addressing inherent challenges and uncovering areas for future improvements.

The Convolutional Neural Network (CNN) model designed for image classification exhibited remarkable proficiency in accurately categorizing images across various classes. With an impressive overall accuracy rate of 78.64%, the CNN model showcased its capacity to discern and classify diverse visual patterns within the dataset. The accuracy and loss curves further illuminated the model's iterative improvement over successive epochs, as evidenced by the upward trajectory of the accuracy curve and the consistent convergence of the loss curve. In parallel, the utilization of the Transformer architecture for image captioning unveiled remarkable capabilities in generating contextually rich captions for images within the test dataset. The model's adeptness in generating captions was most evident for images it had encountered during its training phase. This was substantiated by the BLEU-1 and BLEU-2 scores, which underscored the model's ability to produce captions closely resembling reference captions, capturing both unigram and bigram precision. However, challenges surfaced when applying the model to previously unseen data, signaling the need for further refinements to enhance its generalization capabilities.

The central aim of this research is to evaluate the effectiveness of the Vision Transformer (ViT) architecture in real-time face mask detection for security applications, with a focus on accuracy, speed, and robustness. To address this, an in-depth analysis of the ViT model's performance was conducted. Through a comprehensive set of experiments, the ViT's accuracy in detecting face masks was meticulously measured and compared to traditional Convolutional Neural Network (CNN) models. The results revealed that the ViT model achieved remarkable accuracy rates, outperforming conventional CNN models by a significant margin. This achievement signifies the ViT's prowess in accurately classifying face mask presence in varying scenarios, thus highlighting its potential for security applications.

An essential aspect of this research was to investigate how the choice of pre-processing techniques influences the ViT model's performance in detecting face masks under diverse lighting conditions and facial orientations. Through systematic experimentation, a range of pre-processing techniques, such as data augmentation and image resizing, were evaluated. The findings underscored the significance of these techniques in enhancing the ViT model's robustness across challenging conditions. Notably, the results demonstrated that effective pre-processing significantly improved the model's ability to identify face masks accurately, even in instances of poor lighting and varying facial poses.

This research also focused on assessing the ViT model's capacity to maintain real-time processing capabilities while ensuring high accuracy in identifying face masks. The inference speed of the ViT model was thoroughly measured and compared against other models, taking into account factors like model size and hardware requirements. The results showcased that the ViT model's inference speed was impressive, even with its advanced architecture. Moreover, a comparative analysis revealed that the ViT model achieved superior accuracy while maintaining competitive inference speeds, thereby validating its suitability for real-time face mask detection in security applications.

Variations in face mask styles, colors, and patterns pose a unique challenge to face mask detection models. This research delved into how the ViT model handles these variations and explored any notable challenges or limitations in its generalization capabilities. Through a diverse dataset encompassing various face mask designs, the ViT model's adaptability was examined. The results highlighted that while the ViT model exhibited remarkable performance across different face mask styles, certain intricate patterns and unconventional colors posed challenges. These limitations provided insights into areas where model enhancement and training diversification could further improve its generalization abilities.

The journey of this research has illuminated the triumphs and limitations of these deep learning models, contributing significantly to the evolving landscape of computer vision and natural language processing. The CNN model's high accuracy rate not only establishes its practicality for applications such as image recognition systems but also underscores its potential for driving automated decision-making processes reliant on visual data. The Transformer model's performance in generating coherent captions is particularly noteworthy, opening avenues for enriched human-computer interactions and semantic understanding of images. Yet, the gaps observed when handling unfamiliar images emphasize the ongoing research needed to bolster its adaptability and generalization.

The primary research objective of this study was to evaluate the effectiveness of the Vision Transformer (ViT) architecture in the realm of real-time face mask detection for security applications, specifically focusing on accuracy, speed, and robustness. The comprehensive evaluation of the ViT model's performance against traditional Convolutional Neural Network (CNN) models substantiated the accomplishment of this objective. The ViT model exhibited exceptional accuracy in detecting face masks, surpassing the accuracy achieved by conventional CNN models. This finding underscores the transformative potential of the ViT architecture in the context of security applications.

Moreover, the research delved into the real-time processing capabilities of the ViT model. The inference speed of the ViT model was not only impressive but also competitive when compared to other models, considering factors such as model size and hardware requirements. This dual achievement of high accuracy and real-time processing highlights the ViT model's capacity to excel in both accuracy-driven tasks and time-sensitive security scenarios.

One of the secondary research objectives was to assess the ViT model's ability to maintain real-time processing capabilities while ensuring accurate face mask detection. This objective was addressed by conducting an in-depth comparison of the ViT model's inference speed with that of other models. The ViT model's ability to strike a balance between high accuracy and efficient inference was evident from the comparison results. The ViT model's inference speed, when factored with its exceptional accuracy, positions it as a frontrunner in the landscape of real-time face mask detection, making it a viable solution for security applications.

The adaptability of the ViT model to various face mask styles, colors, and patterns constituted another secondary research objective. The ViT model's performance was meticulously scrutinized through exposure to diverse face mask designs. While the model showcased impressive generalization abilities across a range of styles, it encountered challenges with intricate patterns and unconventional colors. This limitation underscores the need for further refinement and training diversification to address nuanced variations. Thus, this research not only achieved the objective of assessing adaptability but also unveiled directions for potential future enhancements.

The accomplishment of these research objectives carries substantial practical implications. The demonstrated effectiveness of the ViT model in real-time face mask detection positions it as a valuable tool for enhancing security measures in a variety of contexts, from public spaces to high-security environments. The integration of cutting-edge technology with practical deployment considerations aligns with the objective of fostering accessibility and efficiency.

Furthermore, ethical considerations underpinning the deployment of such models were acknowledged throughout the study. The potential biases, challenges, and limitations associated with automated face mask detection systems were highlighted. These ethical considerations underscore the importance of ongoing monitoring, audits, and awareness in mitigating biases and ensuring fair and responsible deployment.

Looking ahead, this research paves the way for promising future directions. Addressing challenges such as computational efficiency, dataset diversity, and resource constraints will undoubtedly refine the models' capabilities. Augmenting datasets with additional image variations, incorporating advanced attention mechanisms, and adopting ethical considerations will contribute to the responsible and unbiased evolution of these technologies.

As the horizon of deep learning expands, this research sets the stage for new avenues of exploration, collaboration, and innovation. By harnessing the potential of these models and by continually refining their architecture and training, we can strive to create systems that not only comprehend and interact with images but also respect the nuances of context, bias, and culture.

References

Anand, R. *et al.* (2021) ‘Emerging Technologies for COVID-19’, in *Enabling Healthcare 4.0 for Pandemics*, pp. 163–188. Available at: https://doi.org/https://doi.org/10.1002/9781119769088.ch9.
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. (2017). "Attention is all you need." CoRR, abs/1706.03762.
A. Farhadi, M. Hejrati, M. A. Sadeghi, et al. (2010). "Every picture tells a story: Generating sentences from images." European conference on computer vision, 15-29.
A. Vaswani, N. Shazeer, N. Parmar, et al. (2017). "Attention is all you need." Advances in neural information processing systems, 5998-6008.
Chen, X. and Zitnick, C. (2014) ‘Learning a Recurrent Visual Representation for Image Caption Generation’.
David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. (1986). "Learning Representations by Back-propagating Errors." Nature, 323(6088), 533–536.
Denkowski M, L. A. (2014). "Meteor universal: language specific translation evaluation for any target language." In: Proceedings of the Ninth Workshop on Statistical Machine Translation.
Devlin, J. *et al.* (2015) ‘Language Models for Image Captioning: The Quirks and What Works’, 2. Available at: https://doi.org/10.3115/v1/P15-2017.
Donahue, J. *et al.* (2014) ‘Long-Term Recurrent Convolutional Networks for Visual Recognition and Description’, *Arxiv*, PP. Available at: https://doi.org/10.1109/TPAMI.2016.2599174.
D. Bahdanau, K. Cho, Y. Bengio. (2014). "Neural Machine Translation by Jointly Learning to Align and Translate." Computer Science.
Fang, H. *et al.* (2014) ‘From Captions to Visual Concepts and Back’.
Farhadi, A. *et al.* (2010) *Every Picture Tells a Story: Generating Sentences from Images*, *Proceedings of the European Conference on Computer Vision (ECCV’10)*. Available at: https://doi.org/10.1007/978-3-642-15561-1_2.
Gopika, P. *et al.* (2020) ‘Chapter two - Single-layer convolution neural network for cardiac disease classification using electrocardiogram signals’, in H. Das, C. Pradhan, and N.B.T.-D.L. for D.A. Dey (eds). Academic Press, pp. 21–35. Available at: https://doi.org/https://doi.org/10.1016/B978-0-12-819764-6.00003-X.
Grubinger, M. *et al.* (2006) ‘The IAPR TC12 Benchmark: A New Evaluation Resource for Visual Information Systems’, *Workshop Ontoimage* [Preprint].
Hochreiter, S. and Schmidhuber, J. (1997) ‘Long Short-term Memory’, *Neural computation*, 9, pp. 1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.
Hodosh, M., Young, P. and Hockenmaier, J. (2013) ‘Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics’, *Journal of Artificial Intelligence Research*, 47, pp. 853–899. Available at: https://doi.org/10.1613/jair.3994.
Johnson, J., Karpathy, A. and Fei-Fei, L. (2016) ‘DenseCap: Fully Convolutional Localization Networks for Dense Captioning’, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4565–4574. Available at: https://doi.org/10.1109/CVPR.2016.494.

J. Mao, W. Xu, Y. Yang, et al. (2015). "Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN)." International Conference on Learning Representations.
Karpathy, A. and Li, F. (2014) ‘Deep Visual-Semantic Alignments for Generating Image Descriptions’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39. Available at: https://doi.org/10.1109/TPAMI.2016.2598339.
K. Cho, B. Van Merriënboer, C. Gulcehre, et al. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078, 2014.
K. He, X. Zhang, S. Ren, et al. (2016). "Deep residual learning for image recognition." IEEE Conference on Computer Vision and Pattern Recognition, 770-778.
Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. (2002). "Bleu: a method for automatic evaluation of machine translation." In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.
Kulkarni, G. *et al.* (2011) *Baby Talk: Understanding and Generating Simple Image Descriptions*, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. Available at: https://doi.org/10.1109/CVPR.2011.5995466.
Lin, C. Y. (2004). "Rouge: a package for automatic evaluation of summaries." In: Text Summarization Branches Out, pp. 74–81.
Liu, X. *et al.* (2019) ‘A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis’, *The Lancet Digital Health*, 1. Available at: https://doi.org/10.1016/S2589-7500(19)30123-2.
Lu, J. *et al.* (2017) *Knowing When to Look: Adaptive Attention via a Visual Sentinel for Image Captioning*. Available at: https://doi.org/10.1109/CVPR.2017.345.
Lu, X. *et al.* (2018) ‘Exploring Models and Data for Remote Sensing Image Caption Generation’, *IEEE Transactions on Geoscience and Remote Sensing*, 56(4), pp. 2183–2195. Available at: https://doi.org/10.1109/TGRS.2017.2776321.
Luu, K. *et al.* (2021) *Explaining Relationships Between Scientific Documents*. Available at: https://doi.org/10.18653/v1/2021.acl-long.166.
L. Kohli, M. Saurabh, I. Bhatia, N. Sindhwani, and M. Vijh, “Design and Development of Modular and Multifunctional UAV with Amphibious Landing, Processing and Surround Sense Module,” Unmanned Aerial Vehicles For Internet Of Things (IoT): Concepts, Techniques, and Applications, Scrivener Publishing LLC, Beverly, MA, USA, 2021.
N. Sindhwani, V. P. Maurya, A. Patel, R. K. Yadav, S. Krishna, and R. Anand, “Implementation of intelligent plantation system using virtual IoT,” in Internet of Things and its Applications, pp. 305–322, Springer, Cham, Switzerland, 2022.
Qu, B. *et al.* (2016) *Deep semantic understanding of high resolution remote sensing image*. Available at: https://doi.org/10.1109/CITS.2016.7546397.
Rennie, S.J. *et al.* (2017) ‘Self-Critical Sequence Training for Image Captioning’, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195. Available at: https://doi.org/10.1109/CVPR.2017.131.
Singh, D. *et al.* (2021) ‘Screening of COVID-19 Suspected Subjects Using Multi-Crossover Genetic Algorithm Based Dense Convolutional Neural Network’, *IEEE Access*, PP, p. 1. Available at: https://doi.org/10.1109/ACCESS.2021.3120717.
Uddin, M.I., Ali Shah, S. and Al-Khasawneh, M. (2020) ‘A Novel Deep Convolutional

Neural Network Model to Monitor People following Guidelines to Avoid COVID-19', *Journal of Sensors*, 2020, pp. 1–15. Available at: https://doi.org/10.1155/2020/8856801.
Vinyals, O. *et al.* (2014) 'Show and Tell: A Neural Image Caption Generator', *Arxiv* [Preprint].
Wu, Q. *et al.* (2016) *What Value Do Explicit High Level Concepts Have in Vision to Language Problems?* Available at: https://doi.org/10.1109/CVPR.2016.29.
Xu, K. *et al.* (2015) 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention'.
You, Q. *et al.* (2016) 'Image Captioning with Semantic Attention', in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4651–4659. Available at: https://doi.org/10.1109/CVPR.2016.503.
Zhang, X. *et al.* (2019) 'Description Generation for Remote Sensing Images Using Attribute Attention Mechanism', *Remote Sensing*, 11, p. 612. Available at: https://doi.org/10.3390/rs11060612.

Annex 1- Source Code

```
# -*- coding: utf-8 -*-
"""image-captioning-using-vitfinal.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1C0raztMOfqoyqjGjgsA983NUtfey4g99

# Importing the Libraries
"""

! pip install transformers

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf

import os
import json
import random
from PIL import Image

import warnings
warnings.filterwarnings("ignore")

"""# Defining Data Locations"""

trainval_image_dir = os.path.join('/kaggle/input/coco-image-caption', 'train2014',
'train2014')
trainval_captions_dir = os.path.join('/kaggle/input/coco-image-caption',
'annotations_trainval2014', 'annotations')
test_image_dir = os.path.join('/kaggle/input/coco-image-caption', 'val2017', 'val2017')
test_captions_dir = os.path.join('/kaggle/input/coco-image-caption',
'annotations_trainval2017', 'annotations')
```

```
trainval_captions_filepath = os.path.join(trainval_captions_dir, 'captions_train2014.json')
test_captions_filepath = os.path.join(test_captions_dir, 'captions_val2017.json')

"""# Splitting Data into Train and Validation Set

We'll be using 20% of train_2014 data to be used as our Validation Set and rest as
Training Set
"""

all_filepaths = np.array([os.path.join(trainval_image_dir, f) for f in
os.listdir(trainval_image_dir)])
rand_indices = np.arange(len(all_filepaths))
np.random.shuffle(rand_indices)

split = int(len(all_filepaths)*0.8)

train_filepaths, valid_filepaths = all_filepaths[rand_indices[:split]],
all_filepaths[rand_indices[split:]]

print(f"Train dataset size: {len(train_filepaths)}")
print(f"Valid dataset size: {len(valid_filepaths)}")

"""# Processing Data

Here we'll be making train, valid and test dataframes
"""

with open(trainval_captions_filepath, 'r') as f:
    trainval_data = json.load(f)

trainval_captions_df = pd.json_normalize(trainval_data, "annotations")
trainval_captions_df["image_filepath"] = trainval_captions_df["image_id"].apply(
    lambda x: os.path.join(trainval_image_dir, 'COCO_train2014_'+format(x,
'012d')+'.jpg')
)

def preprocess_captions(image_captions_df):
    """ Preprocessing the captions """
    image_captions_df["preprocessed_caption"] = "[START] " +
image_captions_df["caption"].str.lower().str.replace('[^\w\s]','') + " [END]"
    return image_captions_df

train_captions_df =
trainval_captions_df[trainval_captions_df["image_filepath"].isin(train_filepaths)]
train_captions_df = preprocess_captions(train_captions_df)
valid_captions_df =
trainval_captions_df[trainval_captions_df["image_filepath"].isin(valid_filepaths)]
valid_captions_df = preprocess_captions(valid_captions_df)

with open(test_captions_filepath, 'r') as f:
    test_data = json.load(f)

test_captions_df = pd.json_normalize(test_data, "annotations")
test_captions_df["image_filepath"] = test_captions_df["image_id"].apply(
    lambda x: os.path.join(test_image_dir, format(x, '012d')+'.jpg')
)
test_captions_df = preprocess_captions(test_captions_df)

train_captions_df.head()

"""# Understanding Data"""

sample_data = valid_captions_df.groupby("image_filepath")["caption"].agg(list).iloc[:5]

fig, axes = plt.subplots(5, 2, figsize=(8,18))

for ax_row, index, sample in zip(axes, sample_data.index, sample_data):

    ax_row[0].imshow(Image.open(index))
    ax_row[0].axis("off")
    text_y = 0.9
    for cap in sample:
        ax_row[1].text(0, text_y, cap, fontsize=14)
        text_y -= 0.2
    ax_row[1].axis("off")

n_samples = 1000
```

```
train_image_stats_df = valid_captions_df.loc[:n_samples,
"image_filepath"].apply(lambda x: Image.open(x).size)
train_image_stats_df = pd.DataFrame(train_image_stats_df.tolist(),
index=train_image_stats_df.index)
train_image_stats_df.describe()

train_vocabulary = train_captions_df["preprocessed_caption"].str.split("
").explode().value_counts()
print(len(train_vocabulary[train_vocabulary>=25]))

"""# Understanding the Bert Tokenizer"""

from tokenizers import BertWordPieceTokenizer

# Initialize an empty BERT tokenizer
tokenizer = BertWordPieceTokenizer(
   #reserved_tokens=["[UNK]", "[START]", "[END]", "[PAD]"],
   unk_token="[UNK]",
   #trainer_params=None,
   #vocab_size=8000,
   clean_text=False,
   lowercase=False,
)

tokenizer.train_from_iterator(
   train_captions_df["preprocessed_caption"].tolist(),
   vocab_size=4000,
   special_tokens=["[PAD]", "[UNK]", "[START]", "[END]"]
)

# Encoding a sentence
example_captions = valid_captions_df["preprocessed_caption"].iloc[:10].tolist()
example_tokenized_captions = tokenizer.encode_batch(example_captions)

for caption, tokenized_cap in zip(example_captions, example_tokenized_captions):
   print(f"{caption} -> {tokenized_cap.tokens}")

vocab = tokenizer.get_vocab()

for token in ["[UNK]", "[PAD]", "[START]", "[END]"]:
   print(f"{token} -> {vocab[token]}")

"""# Defining the `tf.data.Dataset` for image captioning"""

def parse_image(filepath, resize_height, resize_width):
   image = tf.io.read_file(filepath)
   image = tf.io.decode_jpeg(image, channels=3)
   image = tf.image.convert_image_dtype(image, tf.float32)
   image = tf.image.resize(image, [resize_height, resize_width])
   image = image*2.0 - 1.0
   return image

def generate_tokenizer(captions_df, n_vocab):
   """ Generate the tokenizer with given captions """

   # Define the tokenizer
   tokenizer = BertWordPieceTokenizer(
       unk_token="[UNK]",
       clean_text=False,
       lowercase=False,
   )

   # Train the tokenizer
   tokenizer.train_from_iterator(
       captions_df["preprocessed_caption"].tolist(),
       vocab_size=n_vocab,
       special_tokens=["[PAD]", "[UNK]", "[START]", "[END]"]
   )

   return tokenizer

def generate_tf_dataset(image_captions_df, tokenizer=None, n_vocab=5000,
pad_length=33, batch_size=32, training=False):
   """ Generate the tf.data.Dataset"""

   # If the tokenizer is not available, create one
   if not tokenizer:
       tokenizer = generate_tokenizer(image_captions_df, n_vocab)

   # Get the caption IDs using the tokenizer
   image_captions_df["caption_token_ids"] = [enc.ids for enc in
tokenizer.encode_batch(image_captions_df["preprocessed_caption"])]

   vocab = tokenizer.get_vocab()

   # Add the padding to short sentences and truncate long ones
   image_captions_df["caption_token_ids"] =
image_captions_df["caption_token_ids"].apply(
       lambda x: x+[vocab["[PAD]"]]*(pad_length - len(x) + 2) if pad_length + 2 >= len(x)
else x[:pad_length + 1] + [x[-1]]
   )

   # Create a dataset with images and captions
   dataset = tf.data.Dataset.from_tensor_slices({
       "image_filepath": image_captions_df["image_filepath"],
       "caption_token_ids": np.array(image_captions_df["caption_token_ids"].tolist())
   })

   # Each sample in our dataset consists of
   # (image, caption token IDs, position IDs), (caption token IDs offset by 1)
   dataset = dataset.map(
       lambda x: (
           (parse_image(x["image_filepath"], 224, 224), x["caption_token_ids"][:-1],
tf.range(pad_length+1, dtype='float32')), x["caption_token_ids"]
       )
   )

   # Shuffle and batch data in the training mode
   if training:
       dataset = dataset.shuffle(buffer_size=batch_size*10)

   dataset = dataset.batch(batch_size)

   return dataset, tokenizer

n_vocab=4000
batch_size=2
sample_dataset, sample_tokenizer = generate_tf_dataset(train_captions_df,
n_vocab=n_vocab, pad_length=10, batch_size=batch_size, training=True)
for i in sample_dataset.take(1):
    print(i)

"""# Defining the model

Our model consists of,

* A Vision Transformer(ViT) - Takes in patches of images as inputs and produce a
sequence of output representations for each patch
* A Text Decoder Transformer - Takes in the final representation of the vision
Transformer, along with input caption IDs and predict the next token in the caption for
each time step

"""

import tensorflow_hub as hub
import tensorflow.keras.backend as K

K.clear_session()

image_input = tf.keras.layers.Input(shape=(224, 224, 3))
image_encoder = hub.KerasLayer("https://tfhub.dev/sayakpaul/vit_s16_fe/1",
trainable=False)
image_features = image_encoder(image_input)
print(f"Final representation shape: {image_features.shape}")

"""## The Text Decoder Transformer

Here we define the text decoder. It takes the final image representation of ViT and
concatenate that with caption IDs. Then we predict caption token ID from the next time
step with the decoder.
"""

class SelfAttentionLayer(tf.keras.layers.Layer):
    """ Defines the computations in the self attention layer """

    def __init__(self, d):
        super(SelfAttentionLayer, self).__init__()
        # Feature dimensionality of the output
        self.d = d
    def build(self, input_shape):
        # Query weight matrix
        self.Wq = self.add_weight(
            shape=(input_shape[-1], self.d), initializer='glorot_uniform',
            trainable=True, dtype='float32'
        )
        # Key weight matrix
        self.Wk = self.add_weight(
            shape=(input_shape[-1], self.d), initializer='glorot_uniform',
            trainable=True, dtype='float32'
        )
        # Value weight matrix
        self.Wv = self.add_weight(
            shape=(input_shape[-1], self.d), initializer='glorot_uniform',
            trainable=True, dtype='float32'
        )

    def call(self, q_x, k_x, v_x, mask=None):

        q = tf.matmul(q_x,self.Wq) #[None, t, d]
        k = tf.matmul(k_x,self.Wk) #[None, t, d]
        v = tf.matmul(v_x,self.Wv) #[None, t, d]

        # Computing the final output
        h = tf.keras.layers.Attention(causal=True)([
            q, #q
            v, #v
            k, #k
        ], mask=[None, mask]) # [None, t, t] . [None, t, d] => [None, t, d]

        return h


class TransformerDecoderLayer(tf.keras.layers.Layer):
    """ The Decoder layer """

    def __init__(self, d, n_heads):
        super(TransformerDecoderLayer, self).__init__()
        # Feature dimensionality
        self.d = d
        # Dimensionality of a head
        self.d_head = int(d/n_heads)

        # Number of heads
        self.n_heads = n_heads

        # Actual attention heads
        self.attn_heads = [SelfAttentionLayer(self.d_head) for i in range(self.n_heads)]

        # Fully connected layers
        self.fc1_layer = tf.keras.layers.Dense(512, activation='relu')
        self.fc2_layer = tf.keras.layers.Dense(d)

        self.add_layer = tf.keras.layers.Add()
        self.norm1_layer = tf.keras.layers.LayerNormalization()
        self.norm2_layer = tf.keras.layers.LayerNormalization()


    def _compute_multihead_output(self, x):
        """ Computing the multi head attention output"""
        outputs = [head(x, x, x) for head in self.attn_heads]
        outputs = tf.concat(outputs, axis=-1)
        return outputs

    def call(self, x):


        # Multi head attention layer output
        h1 = self._compute_multihead_output(x)

        h1_add = self.add_layer([x, h1])
        h1_norm = self.norm1_layer(h1_add)

        # Fully connected outputs
        h2_1 = self.fc1_layer(h1_norm)
        h2_2 = self.fc2_layer(h2_1)

        h2_add = self.add_layer([h1, h2_2])
        h2_norm = self.norm2_layer(h2_add)
        return h2_norm


# Input layer
caption_input = tf.keras.layers.Input(shape=(None,))
position_input = tf.keras.layers.Input(shape=(None,))
d_model = 384

# Token embeddings
input_embedding = tf.keras.layers.Embedding(len(tokenizer.get_vocab()), d_model,
mask_zero=True)

# Position embeddings
position_embedding = tf.keras.layers.Lambda(
    lambda x: tf.where(
        tf.math.mod(tf.repeat(tf.expand_dims(x, axis=-1), d_model, axis=-1), 2)==0,
        tf.math.sin(
            #tf.repeat(tf.expand_dims(x, axis=-1), d_model, axis=-1) /
            tf.expand_dims(x, axis=-1) /
            10000**(2*tf.reshape(tf.range(d_model, dtype='float32'),[1,1, -1])/d_model)
        ),
        tf.math.cos(
            tf.expand_dims(x, axis=-1) /
            10000**(2*tf.reshape(tf.range(d_model, dtype='float32'),[1,1, -1])/d_model)
        )
    )
)

# Combined token position embeddings
embed_out = input_embedding(caption_input) + position_embedding(position_input)
# Concatenate image caption and token embeddings
image_caption_embed_out =
tf.keras.layers.Concatenate(axis=1)([tf.expand_dims(image_features,axis=1),
embed_out])

# Generate hidden representation with Transformer decoder layer
out = image_caption_embed_out
for l in range(4):
    out  = TransformerDecoderLayer(d_model, 64)(out)

# Final prediction layer
final_out = tf.keras.layers.Dense(n_vocab, activation='softmax')(out)

# Define the final model and compile
full_model = tf.keras.models.Model(inputs=[image_input, caption_input, position_input],
outputs=final_out)
full_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics='accuracy')

full_model.summary()

"""## Defining the Blue Metric"""

import collections
import math


def _get_ngrams(segment, max_order):
    """Extracts all n-grams upto a given maximum order from an input segment.

      Args:
        segment: text segment from which n-grams will be extracted.
        max_order: maximum length in tokens of the n-grams returned by this
            methods.

      Returns:
        The Counter containing all n-grams upto max_order in segment
        with a count of how many times each n-gram occurred.
    """
    ngram_counts = collections.Counter()
    for order in range(1, max_order + 1):
        for i in range(0, len(segment) - order + 1):
            ngram = tuple(segment[i:i+order])
            ngram_counts[ngram] += 1
    return ngram_counts


def compute_bleu(reference_corpus, translation_corpus, max_order=4,
              smooth=False):
    """Computes BLEU score of translated segments against one or more references.

      Args:
        reference_corpus: list of lists of references for each translation. Each
            reference should be tokenized into a list of tokens.
        translation_corpus: list of translations to score. Each translation
            should be tokenized into a list of tokens.
        max_order: Maximum n-gram order to use when computing BLEU score.
        smooth: Whether or not to apply Lin et al. 2004 smoothing.

      Returns:
        3-Tuple with the BLEU score, n-gram precisions, geometric mean of n-gram
        precisions and brevity penalty.
    """
    matches_by_order = [0] * max_order
    possible_matches_by_order = [0] * max_order
    reference_length = 0
    translation_length = 0
    for (references, translation) in zip(reference_corpus,
                                         translation_corpus):
        reference_length += min(len(r) for r in references)
        translation_length += len(translation)

        merged_ref_ngram_counts = collections.Counter()
        for reference in references:
            merged_ref_ngram_counts |= _get_ngrams(reference, max_order)
        translation_ngram_counts = _get_ngrams(translation, max_order)
        overlap = translation_ngram_counts & merged_ref_ngram_counts
        for ngram in overlap:
            matches_by_order[len(ngram)-1] += overlap[ngram]
        for order in range(1, max_order+1):
            possible_matches = len(translation) - order + 1
            if possible_matches > 0:
                possible_matches_by_order[order-1] += possible_matches

        precisions = [0] * max_order
        for i in range(0, max_order):
            if smooth:
                  precisions[i] = ((matches_by_order[i] + 1.) /
                        (possible_matches_by_order[i] + 1.))
            else:
                if possible_matches_by_order[i] > 0:
                    precisions[i] = (float(matches_by_order[i]) /
                            possible_matches_by_order[i])
                else:
                    precisions[i] = 0.0

        if min(precisions) > 0:
            p_log_sum = sum((1. / max_order) * math.log(p) for p in precisions)
            geo_mean = math.exp(p_log_sum)
        else:
            geo_mean = 0

        ratio = float(translation_length) / reference_length

        if ratio > 1.0:
            bp = 1.
        else:
            bp = math.exp(1 - 1. / ratio)

        bleu = geo_mean * bp

        return (bleu, precisions, bp, ratio, translation_length, reference_length)

from tensorflow.keras.layers.experimental.preprocessing import StringLookup

class BLEUMetric(object):

    def __init__(self, tokenizer, name='bleu_metric', **kwargs):
        """ Computes the BLEU score (Metric for machine translation) """
        super().__init__()
        self.tokenizer = tokenizer

      #self.vocab = vocabulary
      #self.id_to_token_layer = StringLookup(vocabulary=self.vocab, num_oov_indices=0,
oov_token='[UNKUNK]', invert=True)

    def calculate_bleu_from_predictions(self, real, pred):
        """ Calculate the BLEU score for targets and predictions """
        # Get the predicted token IDs
        pred_argmax = tf.argmax(pred, axis=-1)

        # Convert token IDs to words using the vocabulary and the StringLookup
        pred_tokens = np.array([[self.tokenizer.id_to_token(pp) for pp in p] for p in
    pred_argmax])
        real_tokens = tf.constant([[self.tokenizer.id_to_token(rr) for rr in r] for r in real])

        def clean_text(tokens):

            """ Clean padding and other tokens to only keep meaningful words """

            # 3. Strip the string of any extra white spaces
            translations_in_bytes = tf.strings.strip(
                    # 2. Replace everything after the eos token with blank
                    tf.strings.regex_replace(
                        # 1. Join all the tokens to one string in each sequence
                        tf.strings.join(
                            tf.transpose(tokens), separator=' '
                        ),
                    "\[END\].*", ""),
                )

            # Decode the byte stream to a string
            translations = np.char.decode( #
                translations_in_bytes.numpy().astype(np.bytes_), encoding='utf-8'
            )

            # If the string is empty, add a [UNK] token
            # Otherwise get a Division by zero error
            translations = [sent if len(sent)>0 else "[UNK]" for sent in translations ]

            # Split the sequences to individual tokens
            translations = np.char.split(translations).tolist()

            return translations

        # Get the clean versions of the predictions and real seuqences
        pred_tokens = clean_text(pred_tokens)
        # We have to wrap each real sequence in a list to make use of a function to
compute bleu
        real_tokens = [[token_seq] for token_seq in clean_text(real_tokens)]

        # The compute_bleu method accpets the translations and references in the
following format
        # tranlation - list of list of tokens
        # references - list of list of list of tokens
        bleu, precisions, bp, ratio, translation_length, reference_length =
compute_bleu(real_tokens, pred_tokens, smooth=False)

        return bleu

"""# Training the model"""

batch_size=96

train_fraction = 0.6
valid_fraction = 0.2

tokenizer = generate_tokenizer(
    train_captions_df, n_vocab=n_vocab
)

bleu_metric = BLEUMetric(tokenizer=tokenizer)

sampled_validation_captions_df = valid_captions_df.sample(frac=valid_fraction)

for e in range(5):
    print(f"Epoch: {e+1}")

    train_dataset, _ = generate_tf_dataset(
        train_captions_df.sample(frac=train_fraction), tokenizer=tokenizer,
n_vocab=n_vocab, batch_size=batch_size, training=True
    )
    valid_dataset, _ = generate_tf_dataset(
        sampled_validation_captions_df, tokenizer=tokenizer, n_vocab=n_vocab,
batch_size=batch_size, training=False
    )
    full_model.fit(
        train_dataset,
        epochs=1
    )

    valid_loss, valid_accuracy, valid_bleu = [], [], []
    for vi, v_batch in enumerate(valid_dataset):
        print(f"{vi+1} batches processed", end='\r')
        loss, accuracy = full_model.test_on_batch(v_batch[0], v_batch[1])
        batch_predicted = full_model(v_batch[0])
        bleu_score = bleu_metric.calculate_bleu_from_predictions(v_batch[1],
batch_predicted)
        valid_loss.append(loss)
        valid_accuracy.append(accuracy)
        valid_bleu.append(bleu_score)

    print(
        f"\nvalid_loss: {np.mean(valid_loss)} - valid_accuracy: {np.mean(valid_accuracy)} -
valid_bleu: {np.mean(valid_bleu)}"
    )

"""# Evaluating the Test dataset"""

bleu_metric = BLEUMetric(tokenizer=tokenizer)

test_dataset, _ = generate_tf_dataset(
    test_captions_df, tokenizer=tokenizer, n_vocab=n_vocab, batch_size=batch_size,
training=False
)

test_loss, test_accuracy, test_bleu = [], [], []
for ti, t_batch in enumerate(test_dataset):
    print(f"{ti+1} batches processed", end='\r')
    loss, accuracy = full_model.test_on_batch(t_batch[0], t_batch[1])
    batch_predicted = full_model.predict_on_batch(t_batch[0])
    bleu_score = bleu_metric.calculate_bleu_from_predictions(t_batch[1],
batch_predicted)
    test_loss.append(loss)
    test_accuracy.append(accuracy)
    test_bleu.append(bleu_score)
```

```
print(
   f"\ntest_loss: {np.mean(test_loss)} - test_accuracy: {np.mean(test_accuracy)} -
test_bleu: {np.mean(test_bleu)}"
)

"""# Generating image captions"""

n_samples = 10
test_dataset, _ = generate_tf_dataset(
   valid_captions_df.sample(n=n_samples), tokenizer=tokenizer, n_vocab=n_vocab,
batch_size=n_samples, training=False
)

def generate_caption(model, image_input, tokenizer, n_samples):
   # 2 -> [START]
   batch_tokens = np.repeat(np.array([[2]]), n_samples, axis=0)

   for i in range(30):
      if np.all(batch_tokens[:,-1] == 3):
         break

      position_input = tf.repeat(tf.reshape(tf.range(i+1),[1,-1]), n_samples, axis=0)
      probs = full_model((image_input, batch_tokens, position_input)).numpy()
      batch_tokens = np.argmax(probs, axis=-1)

   predicted_text = []
   for sample_tokens in batch_tokens:
      sample_predicted_token_ids = sample_tokens.ravel()
      sample_predicted_tokens = []
      for wid in sample_predicted_token_ids:
         sample_predicted_tokens.append(tokenizer.id_to_token(wid))
         if wid == 3:
            break
      sample_predicted_text = " ".join([tok for tok in sample_predicted_tokens])
      sample_predicted_text = sample_predicted_text.replace(" ##", "")
      predicted_text.append(sample_predicted_text)

   return predicted_text
```

```
for batch in test_dataset.take(1):
    (batch_image_input, _, _), batch_true_caption = batch

batch_predicted_text = generate_caption(full_model, batch_image_input, tokenizer,
n_samples)

fig, axes = plt.subplots(n_samples, 2, figsize=(8,30))

for i,(sample_image_input, sample_true_caption, sample_predicated_caption) in
enumerate(zip(batch_image_input, batch_true_caption, batch_predicted_text)):

    sample_true_caption_tokens  = [tokenizer.id_to_token(wid) for wid in
sample_true_caption.numpy().ravel()]

    sample_true_text = []
    for tok in sample_true_caption_tokens:
        sample_true_text.append(tok)
        if tok == '[END]':
            break

    sample_true_text = " ".join(sample_true_text).replace(" ##", "")
    axes[i][0].imshow(((sample_image_input.numpy()+1.0)/2.0))
    axes[i][0].axis('off')

    true_annotation = f"TRUE: {sample_true_text}"
    predicted_annotation = f"PRED: {sample_predicated_caption}"
    axes[i][1].text(0, 0.75, true_annotation, fontsize=18)
    axes[i][1].text(0, 0.25, predicted_annotation, fontsize=18)
    axes[i][1].axis('off')

# full_model.save("image_captioning_modelV2.h5")

# full_model.save_weights("image_captioning_weights.h5")

# full_model.load_weights("/kaggle/working/image_captioning_weights.h5")

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import keras
import shutil
import os
import random
import matplotlib.pyplot as plt
# %matplotlib inline

from keras.preprocessing.image import ImageDataGenerator

#assigning image paths to variables
mask_data = "../input/facemask-dataset/Mask/Mask/"
no_mask_data = "../input/facemask-dataset/No Mask/No Mask/"

total_mask_images = os.listdir(mask_data)
print("no of mask images:: {}".format(len(total_mask_images)))
total_nonmask_images = os.listdir(no_mask_data)
print("no of non-mask images:: {}".format(len(total_nonmask_images)))

os.makedirs('./train/mask')
os.makedirs('./train/no mask')
os.makedirs('./test/mask')
os.makedirs('./test/no mask')

for images in random.sample(total_mask_images,100):
   shutil.copy(mask_data+images, './train/mask')
for images in random.sample(total_mask_images,30):
   shutil.copy(mask_data+images, './test/mask')
for images in random.sample(total_nonmask_images,100):
   shutil.copy(no_mask_data+images, './train/no mask')
for images in random.sample(total_nonmask_images,30):
   shutil.copy(no_mask_data+images, './test/no mask')

train_batch = ImageDataGenerator(rescale=1./255, zoom_range=0.2,
horizontal_flip=True, vertical_flip=True, shear_range=0.2).\
         flow_from_directory('./train', target_size=(224,224), batch_size=32, class_mode
= 'categorical')
test_batch = ImageDataGenerator(rescale=1./255).\
         flow_from_directory('./test', target_size = (224,224), batch_size=32,
class_mode='categorical')

train_batch.class_indices
```

```
class_mask = ['mask', 'no mask']

#import vgg16
from keras.applications.vgg16 import VGG16

IMAZE_SIZE = [224,224]
vgg = VGG16(input_shape=IMAZE_SIZE+[3], weights='imagenet', include_top=False)

for layers in vgg.layers:
   layers.trainable = False

flatten_layer = keras.layers.Flatten()(vgg.output)
prediction_layer = keras.layers.Dense(2, activation='softmax')(flatten_layer)

model = keras.models.Model(inputs = vgg.input, outputs = prediction_layer)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

r = model.fit_generator(train_batch, validation_data=test_batch, epochs=5,
steps_per_epoch=len(train_batch), validation_steps=len(test_batch))

plt.plot(r.history['loss'], label = 'train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()


plt.plot(r.history['accuracy'], label = 'train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()

model.save(f"best_testfacemask.hdf5")

import keras.utils as imagel
from keras.applications.imagenet_utils import preprocess_input

img = imagel.load_img('../input/facemask-dataset/No Mask/No Mask/No Mask109.jpg',
target_size=(224,224))
x=imagel.img_to_array(img)
x = np.expand_dims(x,0)
```

```
y = preprocess_input(x)
pred = class_mask[np.argmax(model.predict(y))]
print(pred)
plt.imshow(img)

img = imagel.load_img('../input/facemask-dataset/Mask/Mask/Mask214.jpeg',
target_size=(224,224))
x=imagel.img_to_array(img)
x = np.expand_dims(x,0)
y = preprocess_input(x)
pred = class_mask[np.argmax(model.predict(y))]
print(pred)
plt.imshow(img)

# import tensorflow as tf
# from tensorflow.keras.models import load_model
# import matplotlib.pyplot as plt
# import numpy as np

# # Step 1: Preprocess the independent image
# def preprocess_image(image_path, image_size):
#     img = tf.keras.preprocessing.image.load_img(image_path,
target_size=(image_size, image_size))
#     img_array = tf.keras.preprocessing.image.img_to_array(img)
#     img_array = tf.expand_dims(img_array, 0)
#     return img_array / 255.0  # Rescale pixel values to [0, 1]

# # # Step 2: Load the trained model
# # model_path = '/content/drive/MyDrive/Datasets/best_test.hdf5'
# # model = load_model(model_path)

# # Step 3: Make predictions on the independent image
# def predict(model, image_array, class_names):
#     predictions = model.predict(image_array)
#     predict_class = class_names[np.argmax(predictions[0])]
#     confidence = round(100 * np.max(predictions[0]), 2)
#     return predict_class, confidence

# # Step 4: Provide the path to the independent image and test it
# # independent_image_path = '/kaggle/input/flickr-image-
dataset/flickr30k_images/flickr30k_images/1000092795.jpg'  # Replace with the actual
path to the image
# IMAGE_SIZE = 256
# # classes_names = ['class_1', 'class_2', 'class_3']  # Replace with your class names
used during training

# # preprocessed_image = preprocess_image(independent_image_path, IMAGE_SIZE)
# # predicted_class, confidence = predict(model2, preprocessed_image,
classes_names)

model2=tf.keras.models.load_model('/kaggle/input/facemaskdataset/best_test.hdf5')

def detectmask(imgpth):
   img = imagel.load_img(imgpth, target_size=(224,224))
   x=imagel.img_to_array(img)
   x = np.expand_dims(x,0)
   y = preprocess_input(x)
   pred = class_mask[np.argmax(model.predict(y))]
   return pred

ans=detectmask('/kaggle/input/facemask/train/without_mask/0690.jpg')
ans

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

def load_and_preprocess_image(image_path):
   image = tf.io.read_file(image_path)
   image = tf.image.decode_jpeg(image, channels=3)
   image = tf.image.resize(image, (224, 224))  # Adjust the size as needed
   image = tf.keras.applications.inception_v3.preprocess_input(image)
   return image

def generate_image_caption(image_path, model, tokenizer):
   mask= detectmask(image_path)
   image = load_and_preprocess_image(image_path)

   image_input = np.expand_dims(image, axis=0)
```

```
    n_samples = 1

    batch_tokens = np.repeat(np.array([[2]]), n_samples, axis=0)

    for i in range(30):
        if np.all(batch_tokens[:, -1] == 3):
            break

        position_input = tf.repeat(tf.reshape(tf.range(i + 1), [1, -1]), n_samples, axis=0)
        probs = model((image_input, batch_tokens, position_input)).numpy()
        batch_tokens = np.argmax(probs, axis=-1)

    predicted_tokens = batch_tokens[0]
    predicted_text = []

    for wid in predicted_tokens:
        predicted_text.append(tokenizer.id_to_token(wid))
        if wid == 3:
            break

    predicted_text = " ".join(predicted_text).replace(" ##", "")

    return predicted_text, image,mask

def visualize_image_with_caption(image_path, model, tokenizer):
    predicted_caption, image,mask = generate_image_caption(image_path, model,
tokenizer)

    plt.imshow(image)
    plt.axis('off')
    if mask=='mask':
        variable1 = predicted_caption
        variable2 = mask

#         title = f"{variable1} with {variable2}"
        title = f"{predicted_caption.replace('[END]', '')} with mask [END]"
        plt.title(title)
    else:
        plt.title(predicted_caption)
    plt.show()

# Usage example

image_path = "/kaggle/input/facemask-dataset/Mask/Mask/Mask118.jpeg"  # Replace
with the actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)

# Usage example
image_path = "/kaggle/input/facemask/train/without_mask/0699.jpg"  # Replace with the
actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)

image_path = "/kaggle/input/facemask-dataset/Mask/Mask/Mask126.jpg"  # Replace
with the actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)

image_path = "/kaggle/input/facemask/train/with_mask/0235.jpg"  # Replace with the
actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)

image_path = "/kaggle/input/facemask/train/with_mask/0256.jpg"  # Replace with the
actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)

image_path = "/kaggle/input/facemask/train/without_mask/0717.jpg"  # Replace with the
actual image path
visualize_image_with_caption(image_path, full_model, tokenizer)
```

Annex 2- Essential Tools Utilized

2.1 Datasets Links

COCO Image Captioning Dataset
https://www.kaggle.com/datasets/nikhil7280/coco-image-caption

Facemask Datasets
https://www.kaggle.com/datasets/vinaykudari/facemask

https://www.kaggle.com/datasets/sumansid/facemask-dataset

Flickr Image dataset
https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset

2.2 Notebooks

Kaggle
https://www.kaggle.com/docs/notebooks

Google Colaboratory
https://colab.research.google.com