The University of Manchester

Image Captioning

A dissertation submitted to The University of Manchester
for the degree of BSc Hons Computer Science
by

Suphal Sharma

Supervisor : Dr Jiaoyan Chen

Academic Year : 2023-2024

Abstract

This project focuses on the development and evaluation of image captioning models using various combinations of architectures and datasets. The models are built using VGG16 and Vision Transformer (ViT) for feature extraction, combined with Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) for processing sequential data. These models are trained on different datasets, namely Flickr8k and Flickr30k. The main aim is to experiment with these different configurations and evaluate their performance in generating accurate and contextually relevant image captions. The evaluation process involves both quantitative measures using BLEU scores and qualitative assessment through peer reviews. This comprehensive approach ensures a robust evaluation of the models, taking into account both the precision of the generated captions and their relevance and descriptiveness in the context of the images. Upon experimentation, it was found that the model trained on the flick30k dataset using VGG16 as the image encoder and GRU as the language model yielded the highest BLEU-1 and peer evaluation scores out of the models created in the project. These scores were 0.558658 (out of 1) and 3.62 (out of 5) respectively.

Acknowledgements

I would like to extend my sincere appreciation towards my supervisor Dr Jiaoyan Chen. Without his guidance, feedback and encouragement this project would not have been possible. I am very grateful to all of the people that agreed to participate in the evaluation of this project. Lastly, I would like to thank my parents for their unconditional love and support .

Table of Contents

Word Count: 10169

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

The intersection of computer vision and natural language processing has paved the way for significant advancements in technology. This has enabled machines to interpret visual content with a level of proficiency that approaches human capabilities. One of the most compelling applications of this convergence is image captioning, which involves generating textual descriptions for images automatically [1]. This capability holds substantial implications across a range of areas, including content understanding, accessibility, and enhancing human-computer interaction. In particular, it has the potential to significantly improve the internet experience for visually impaired individuals by making visual content more accessible to them [2].

The motivation for undertaking this project on image captioning comes from the desire to contribute to these advancements. The project is driven by the potential of image captioning to revolutionize various fields and the intellectual challenge it presents. It offers an opportunity to explore and understand the complex relationship between visual perception and language understanding.

This report records the findings of my final year project that dives into the exploration of image captioning techniques. The project uses established deep learning models, employing recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) for processing sequential data [3]. A key aspect of the project involves the extraction of image features using different architectures. This includes the investigation of transformer-based architectures, specifically the Vision Transformer (ViT) [4], and conventional Convolutional Neural Networks (CNNs), particularly VGG16 [5].

In addition to the quantitative evaluation of the models using metrics like BLEU scores [6], the project also incorporates qualitative evaluation through peer reviews. This dual approach ensures a comprehensive assessment of the models, taking into account both the precision of the generated captions and their relevance and descriptiveness in the context of the images.

1.2 Aims and Objectives

The primary aim of this project is to gain insights into the performance and efficacy of different deep learning architectures and their combinations in the realm of image captioning. Through rigorous experimentation and analysis, the project contributes to the ongoing discourse around image captioning techniques. It explores various configurations and parameters, shedding light on their impact on the model's performance.

Furthermore, the project aims to promote advancements in the development of more reliable and

human-like image captioning systems. It does so by highlighting the nuances and finer details of these models, thereby providing a roadmap for future research in this area.

The primary objectives of this project are:

- Exploring Pretrained Feature Extractors: Investigating the use of pretrained CNNs and transformer-based models for extracting image features, understanding their impact on the performance of the caption generation models.
- Sequence Modelling with Recurrent Neural Networks: Implementing LSTM and GRU architectures to capture the sequential nature of language and generating logical textual descriptions for the extracted image features.
- Evaluation and Comparison: Comparing the effectiveness of various feature extractor and sequence model combinations in the context of image captioning tasks in order to identify the advantages and disadvantages of each.

1.3 Dissertation Outline

The report is divided in the following sections -

- Chapter 1 - This section provides an introduction about the project and the report. It discusses the objectives of the project and report structure.
- Chapter 2 - This section provides a technical background required to understand the functioning of an image captioning model. It discusses the various elements of architecture individually, which gives a better understanding of their working. The section concludes by analyzing the related works on the subject. We also take a look at some models that have made a significant impact in the field.
- Chapter 3 - This section discusses the methodology and implementation used build the image captioning model. It describes the design choices used to structure the architecture of the model. We also take a look at some important functions that were used.
- Chapter 4 - This section describes the experiments and analyses the results. This helps to explore the strengths and weaknesses of individual components of the image captioning architecture through detailed evaluation and rigorous experimentation.
- Chapter 5 - This section examines the overall functioning of the model, the progress made on the project's aims and objectives. We take a look at limitations in experiments. It concludes by exploring the prospects of future works on the subject.

Chapter 2

Technical Background

There are various methodologies and architecture which have been proposed in the field of image captioning in the last few years. Regardless of that, all image captioning models can be logically deduced to a visual encoder module which is simply in charge of processing the visual features, and a language model which is in charge of generating the actual captions [7].

2.1 Image encoding

In this work we try to study and analyse CNN (VGG16) and Vision Transformer for the task of image captioning.

2.1.1 CNN

Convolutional Neural Networks (CNNs) are skilled in automatically learning hierarchical representations of visual data because of their hierarchical topology. They consist of several layers and are highly effective at capturing spatial features at various scales in an image. They process images and represent them as numerical arrays, extracting features for analysis and prediction [8].

VGG16 is a simple yet effective model. It consists of 16 layers with learnable parameters, which are 13 convolutional layers and 3 fully connected layers. Apart from that it also has 5 max-pooling layers [9].

Primary function of the convolutional layer is to extract various high-level features like shapes or complex structures and low-level features like edges or points. It is also responsible in maintaining local connectivity of pixels.

It uses small (3x3) convolution filters throughout the entire network. These filters are small matrices of weights. The weights of the filters are learned during the training process. They help to determine the features that the network can extract from the input data. They slide over the input image one pixel at a time (a stride of 1). The smaller the stride is, the more the overlap between receptive fields. This helps in capturing more information about the input. This operation produces a feature map which allows the network to capture local patterns in the image, like edges and textures [10].

To ensure that the spatial dimensions (width and height) of the output from the convolution operation are the same as the input, a process known as ‘same’ padding is used. Padding is a technique where layers of zeros are added to the input matrix. It allows the filter to properly convolve around the edge pixels of the input image. This is particularly useful in deep networks as it allows control over the size of the output volumes (spatial dimensions of activation maps) throughout the network. With-

out padding, the size of the output volume would reduce after each convolution operation, limiting the depth of the network [10].

This architecture is followed by the max-pooling layer. Max-pooling helps to reduce the spatial dimensions of the convolved feature output. This decreases the computational complexity of the network. It extracts the most prominent features of the image and discards the less relevant information. It uses a 2x2 filter of stride 2. It reduces the spatial dimensions by half. This is done by selecting the maximum value from each block of pixels covered by the filter. The convolution and max pool layers

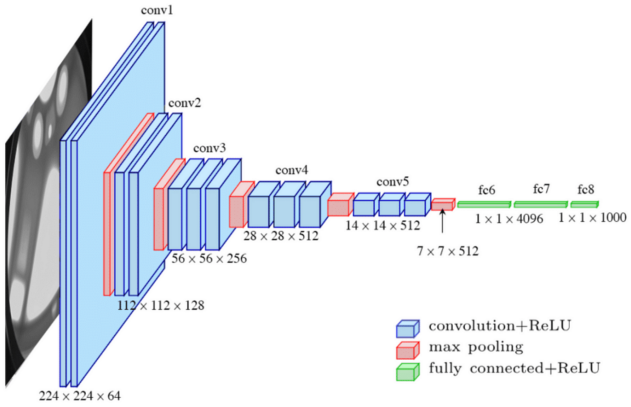


Figure 2.1: VGG16 Model Architecture

are consistently arranged throughout the whole architecture.

After the convolutional and max-pooling layers, the output is flattened into a one-dimensional vector. This flattened vector contains all the spatial and depth information learned by the previous layers. It is then passed to the fully connected layers.

The fully connected layers are dense, meaning every neuron in a layer receives input from all neurons of the previous layer. These layers perform high level reasoning by combining the features learned by the previous layers. For example, if the convolutional layers have recognized edges and textures, the fully connected layers can recognize more complex structures by combining these features [11].

There are three fully connected layers. The first two have 4096 channels each. They continue the process of combining the features learned by the convolutional layers.

The third fully connected layer has 1000 channels, which correspond to the 1000 classes of the ImageNet dataset [12]. This layer is followed by a softmax activation function, which outputs a probability distribution over the 1000 classes. The class with the highest probability is chosen as the final prediction [10].

In the context of image captioning, the output of the VGG16 can be fed into a decoder, such as an LSTM, to generate a caption for the input image. This allows the model to leverage the powerful feature extraction capabilities of the CNN architecture. Its ability to understand the dependencies between different parts of an image makes it a strong choice for tasks like image captioning [13].

2.1.2 ViT

The Vision Transformer (ViT) is a novel approach to computer vision tasks. It was developed by researchers at Google. It leverages the transformer architecture, which was originally designed for natural language processing tasks, and applies it to image classification [4].

In the context of ViT, an image is treated as a sequence of patches, similar to how text is treated as a sequence of tokens in NLP. The image is split into fixed-size patches, each of them is then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard transformer encoder [14].

The transformer encoder in ViT allows the model to understand the context and relationships between different parts of the image. This is a significant change from traditional convolutional neural networks (CNNs), which typically use filters to convolve over the image and capture local features [15].

ViT uses a 16x16 patch size, meaning the input image is divided into 16x16 pixel patches [16]. These patches are then flattened and passed through a series of transformer encoders. The transformer encoders process the patches as a sequence by applying self-attention mechanisms to understand the dependencies between different patches [17].

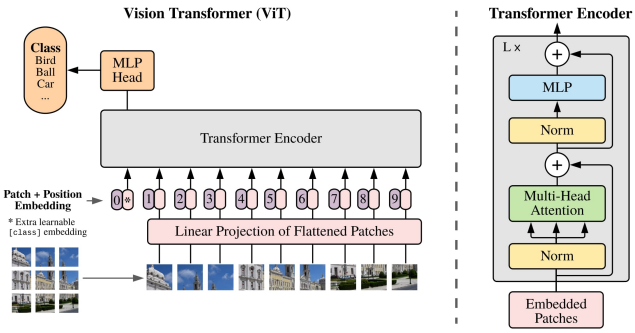


Figure 2.2: ViT Model Architecture

After the transformer encoders, the output is passed to a fully connected layer [18]. This layer takes the output of the transformer encoders and makes the final prediction.

In ViT, there is one fully connected layer that corresponds to the number of classes in the dataset. This layer is followed by a softmax activation function, which outputs a probability distribution over the classes. The class with the highest probability is chosen as the final prediction.

One of the key advantages of ViT is its ability to be pretrained on large datasets [19]. When pretrained on sufficiently large amounts of data, ViT can achieve excellent results, outperforming state-of-the-art CNNs by almost four times in terms of computational efficiency and accuracy. This makes ViT a powerful tool for a wide range of computer vision tasks [17].

In the context of image captioning, the output of the ViT can be fed into a decoder, such as an LSTM, to generate a caption for the input image. This allows the model to leverage the powerful fea-

ture extraction capabilities of the transformer architecture. Its ability to understand the dependencies between different parts of an image makes it a strong choice for tasks like image captioning.

2.2 Language model

Recurrent Neural Networks (RNNs) are a type of neural network designed specifically for handling sequential data. Unlike traditional feed-forward neural networks, RNNs have a distinctive feature: they have loops in them, allowing information to persist from one step in the sequence to the next [20]. This characteristic makes them particularly well-suited for tasks where the order of inputs carries significant information, such as language modelling, speech recognition, and, of course, image captioning [21].

In the field of image captioning, RNNs play a crucial role. The CNN part of the model takes care of understanding the image, extracting a fixed-length feature vector that captures the content of the image. This vector, which can be thought of as a condensed representation of the image, is then fed into an RNN. The RNN takes this condensed representation and generates a sequence of words, one word at a time, to form a coherent and descriptive caption of the image. The RNN is trained to predict the next word in the sequence given the current word and the image feature vector [22].

In deep learning, the gradient is a vector that contains the partial derivatives of a loss function with respect to the weights of the neural network. The gradient points in the direction of the greatest rate of increase of the loss function, and its magnitude is the rate of increase in that direction [23].

The vanishing gradient problem arises during the training of deep neural networks when these gradients become extremely small as they are propagated backward through the network. This can cause the weights in the lower layers of the network to be updated very slowly, if at all. This results in poor learning performance [24].

2.2.1 LSTM

Long Short-Term Memory (LSTM) networks are a special kind of RNNs. LSTM was introduced by Hochreiter and Schmidhuber in 1997 and was designed to combat the vanishing gradient problem. LSTMs solve this problem with a unique architectural feature: the memory cell. Each memory cell contains several components. These include a cell state, input gate, forget gate, and output gate. These gates regulate the flow of information into and out of the cell state which allows the LSTM to effectively learn and remember patterns in sequential data [24].

(i) Cell state (C_t): The cell state serves as the memory of the LSTM cell. It is passed along from one time step to the next. It allows the LSTM to retain information over long sequences. This makes it suitable for tasks that require memory of past events [25].

(ii) Forget gate (f_t): The forget gate determines which information from the previous cell state

should be discarded and which should be retained. It takes as input the previous cell state (C_{t-1}) and the current input (x_t). It passes them through a sigmoid activation function to generate a forget gate vector (f_t) between 0 and 1. The forget gate vector selectively scales the elements of the previous cell state. This helps in effectively deciding what information to forget [25].

(iii) Input Gate (i_t) and Candidate Cell State ($\tilde{C}_t$): The input gate controls the flow of new information into the cell state. It takes as input the previous cell state (C_{t-1}) and the current input (x_t), passing them through a sigmoid activation function to generate an input gate vector (i_t) between 0 and 1. Additionally, it passes the previous cell state and current input through a tanh activation function to generate a candidate cell state ($\tilde{C}_t$), which represents new candidate information to be added to the cell state [25].

(iv) Update Cell State: The forget gate and input gate vectors are used to update the cell state. The forget gate determines which information to forget from the previous cell state, while the input gate determines which new information to add. The new cell state (C_t) is computed by element-wise multiplication of the forget gate and the previous cell state, followed by element-wise addition with the product of the input gate and the candidate cell state [25].

(v) Output Gate (o_t) and Output (h_t): The output gate regulates the flow of information from the cell state to the output at the current time step. It takes as input the previous cell state (C_{t-1}) and the current input (x_t), passing them through a sigmoid activation function to generate an output gate vector (o_t) between 0 and 1. The current cell state (C_t) is passed through a tanh activation function to generate the output (h_t), which is then scaled by the output gate vector to produce the final output of the LSTM cell [25].

In the context of image captioning, LSTMs can be used to generate more accurate and contextually relevant captions. The LSTM's memory cells allow it to maintain a kind of 'context' or 'memory' about what it has generated so far, making it possible to generate captions that are not only relevant to the content of the image but also coherent and grammatically correct [26].

2.2.2 GRU

GRUs are a variant of RNNs that use gating mechanisms to control the flow of information through the network. They are simpler than LSTMs, with fewer gates and a more streamlined architecture [27]. They simplify the LSTM architecture by combining the forget and input gates into a single "update gate." They also merge the cell state and hidden state.

(i) Update Gate (z_t): The update gate determines how much of the previous memory to retain and how much of the new candidate memory to use. It takes as input the previous hidden state (h_{t-1}) and the current input (x_t), and passes them through a sigmoid activation function to generate an

update gate vector (z_t) between 0 and 1. The update gate controls the trade-off between the old memory and the new candidate memory [28].

(ii) Reset Gate (r_t): The reset gate determines how much of the previous hidden state to forget. It takes as input the previous hidden state (h_{t-1}) and the current input (x_t), and passes them through a sigmoid activation function to generate a reset gate vector (r_t) between 0 and 1. The reset gate determines which parts of the previous hidden state should be ignored when computing the new candidate hidden state [28].

(iii) Candidate Hidden State ($\tilde{h}_t$): The candidate hidden state represents the new information to be added to the hidden state. It is computed by combining the previous hidden state (h_{t-1}) and the current input (x_t), then passing them through a tanh activation function. The reset gate (r_t) controls which parts of the previous hidden state are used to compute the candidate hidden state [28].

(iv) Update Hidden State: The update gate (z_t) determines how much of the previous hidden state to retain and how much of the new candidate hidden state to use. The new hidden state (h_t) is computed as a weighted combination of the previous hidden state (h_{t-1}) and the candidate hidden state ($\tilde{h}_t$), where the weights are determined by the update gate [28].

In the context of image captioning, GRUs can be used in much the same way as LSTMs. A CNN is used to extract a feature vector from the image, and this vector is then fed into a GRU network that generates a sequence of words. The GRU's update and reset gates allow it to capture dependencies in the input sequence, leading to accurate and contextually relevant image captions [28].

RNNs and their variants, LSTMs and GRUs, have proven to be highly effective for image captioning tasks.

2.3 Related works

Image captioning has been a topic of interest in the field of computer vision and natural language processing for several years. This section provides a comprehensive overview of the technical background and related works in the field of image captioning.

2.3.1 Early Approaches

Early approaches to image captioning were largely rule-based and relied on manually designed features. These methods often involved segmenting the image into regions, extracting features from these regions, and then using these features to generate a caption [7]. The captions were typically generated using templates, where the slots in the templates were filled with words or phrases based on the extracted features [29].

One of the earliest works in this area was the Automatic Generation of Descriptions (AGD) system, which used a combination of image segmentation, object recognition, and natural language generation to produce image captions [30]. However, these early approaches were limited by their reliance on hand-crafted features and templates, which made them inflexible and unable to handle the wide variety of images and scenes found in the real world [31].

2.3.2 Deep Learning Approaches

The advent of deep learning marked a significant shift in the field of image captioning. Deep learning models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have the ability to learn complex features from data, making them well-suited for the task of image captioning [31].

One of the first deep learning-based approaches to image captioning was the work by Vinyals et al., which introduced the use of a CNN to extract features from the image and an RNN to generate the caption. This approach, often referred to as the "encoder-decoder" framework, has since become the standard approach for image captioning [32].

Following this, a number of variants and improvements to the encoder-decoder framework were proposed. These include the use of attention mechanisms, which allow the model to focus on different parts of the image at each step of the caption generation, and the use of more sophisticated RNN architectures, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, which are better able to handle long sequences of data [7].

2.3.3 Recent Advances

More recently, there has been interest in exploring alternative architectures and training methods for image captioning. One such approach is the use of Transformer models, which replace the recurrent layers in the decoder with self-attention layers. This allows the model to capture long-range dependencies in the caption, which can be beneficial for generating more coherent and grammatically correct captions [33].

Another recent trend is the use of reinforcement learning for training image captioning models. Reinforcement learning allows the model to be trained directly on the quality of the generated captions, as measured by a reward function, rather than on the prediction of the next word in the caption. This can lead to captions that are more relevant and descriptive [33].

In addition to these, there has been work on incorporating additional sources of information into the image captioning process. This includes the use of visual attention, which allows the model to focus on different parts of the image at each step of the caption generation, and the use of semantic information, which can help the model generate captions that are more relevant and informative [34].

2.3.4 Bootstrapping Language-Image Pre-training (BLIP)

A significant development in the field of image captioning is the introduction of the Bootstrapping Language-Image Pre-training (BLIP) model. Developed by Salesforce Research, BLIP represents a new approach to image captioning that leverages both vision-language understanding and generation tasks.

BLIP employs a pre-trained model that learns from noisy web data by filtering out less accurate captions and retaining the more accurate ones. This approach allows the model to generate captions for images with high accuracy, using natural language processing and computer vision techniques. The model is trained on a large-scale dataset collected from the web, which provides a diverse range of images and captions for the model to learn from. This diversity in training data enables the model to handle a wide variety of images and generate accurate captions for them.

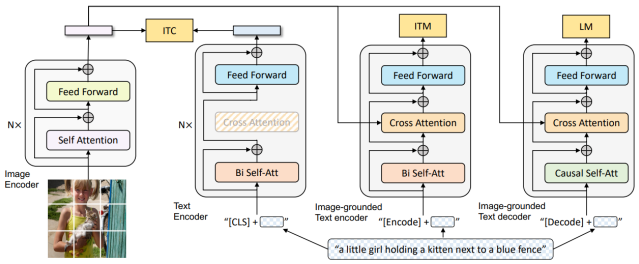


Figure 2.3: BLIP Model Architecture

One of the key features of BLIP is its ability to provide contextually relevant captions. The model is capable of understanding the relationship between objects in an image and uses spatial arrangements to generate captions. This functionality enables the creation of human-like captions that are not just generic, but also contextually relevant. For instance, if an image contains a cat sitting on a mat, the model will not only recognize the cat and the mat as separate objects, but will also understand the spatial relationship between them (i.e., the cat is sitting on the mat) and reflect this in the generated caption [35].

BLIP also supports multiple languages, making it a beneficial tool for international brands and businesses. This multilingual capability allows the model to generate captions in different languages, thereby catering to a global audience. Furthermore, it exhibits real-time processing, making it a powerful tool for image captioning. This means that the model can generate captions quickly, making it suitable for real-time applications such as live video captioning [35].

In addition to these features, BLIP also incorporates several advanced techniques to further improve the quality of the generated captions. For instance, it uses beam search and nucleus sampling during the caption generation process to ensure that the generated captions are not only accurate but also diverse [36]. It also employs various regularization techniques to prevent overfitting and ensure that the model generalizes well to new, unseen images [37].

However, like all models, it has its limitations and areas for improvement. Its adaptability to

various vision-language tasks can be challenging. The quality of web-collected training data and dependency on high-quality human annotated captions are other concerns [35]. It's also computationally expensive and requires pre-computation for new images [38]. Like other models, it has inherent risks related to bias and safety. This presents opportunities for future research and development [35].

2.3.5 Challenges

The field of image captioning has seen significant progress over the years, with advances in deep learning leading to the development of models that are capable of generating increasingly accurate and descriptive captions [39].

However, despite these remarkable advances, there remain challenges to be addressed. One such challenge is the need for more effective training methods. Current methods, while effective, can often lead to overfitting or underfitting, and may not fully exploit the potential of the deep learning models [31].

Another challenge lies in the incorporation of additional sources of information. While current models primarily rely on visual data, incorporating other sources of information, such as contextual or semantic information, could potentially enhance the quality and relevance of the generated captions [31].

Furthermore, there is a need for the development of models that are capable of handling a wider variety of images and scenes. Current models, while impressive, may struggle with complex or unfamiliar scenes, leading to inaccurate or nonsensical captions [40].

As research in this area continues, it is expected that these challenges will be addressed, leading to further improvements in the quality of the generated captions.

Chapter 3

Methodology and Implementation

To achieve the desired outcome of the project, each phase of the development and the general system architecture had to be planned individually. The framework is composed of singular components, implemented and evaluated in different combinations for the purpose of experimentation. In order to get high-quality results, every component of the framework had to produce high-quality results. Due to the large quantity of data and lots of experiments, the process of training was very time consuming. To get a better understanding on the design, we also looked at a few open sourced codes [41, 42]. This helped in development of the template of the models which were utilized in experimentation.

3.1 Tasks

The following tasks were set:

(1) Finding suitable datasets

(2) Choosing and implementing image encoders

(3) Preprocessing and cleaning datasets

(4) Choosing and implementing language models

(5) Caption generation

3.2 Datasets

In the process of training our image captioning models, we have chosen to utilize two prominent datasets: Flickr8k and Flickr30k. Each dataset was used to train models separately.

The Flickr8k dataset is a well-established benchmark in the field of image captioning. It comprises of over 8,000 images that are each paired with five different captions. These images were selected for their diversity and represent a wide range of situations and objects. The multiple captions provide a comprehensive description of each image, allowing our model to learn varied language expressions linked to the same visual content [43].

The Flickr30k dataset, an extension of Flickr8k, contains over 30,000 images, each with five associated captions. The larger volume of data in Flickr30k provides our model with a broader scope of scenarios and vocabulary, enhancing its ability to generate accurate and diverse captions [44].

By training models on these two datasets, we aim to understand the strengths and weaknesses of our image captioning models across different data volumes and diversities.

3.3 Image Encoders

In our image captioning models, we utilized two different image encoders: VGG16 and ViT (Vision Transformer). Each encoder was used to train separate models. By training models using these two image encoders, we aim to understand the strengths and weaknesses of our image captioning models across different architectural complexities and feature extraction capabilities [1].

Both VGG16 and ViT were used in their pre-trained forms, leveraging the concept of transfer learning. Transfer learning is a machine learning technique where a pre-trained model is used as the starting point for a related task. The idea behind transfer learning is that this model has already learned to extract useful features from the large dataset it was trained on, and these features can be useful for the new task, even if the new task is quite different from the original task. This is particularly useful when the dataset for the new task is relatively small, as training a deep learning model from scratch on a small dataset can lead to overfitting [1]. Using pre-trained models brings two main advantages:

(1) Efficiency: It reduces the training time as the models have already learned useful feature representations from large-scale datasets.

(2) Performance: It often leads to better performance, especially when the available data for our specific task is limited.

We chose VGG16 for its proven performance in image recognition tasks. The pre-trained VGG16 model has been trained on a vast array of images, enabling it to extract meaningful features from our dataset effectively. This leads to a more robust model that can handle a wide variety of image content [1].

In the VGG16 model, we froze some of the layers during training. This is a common practice in transfer learning where we leverage the pre-trained weights of the model. Freezing the layers helps in preserving the learned features and prevents them from being updated during the training of our specific task [1].

ViT was chosen for its ability to handle complex scenes. Unlike traditional convolutional networks, ViT can model long-range, global dependencies in the image data. This makes it particularly effective when the scene contains multiple objects that are contextually related. For both models, each image is loaded and resized to 224x224 pixels, which is the standard input size expected by these models. The image is then converted to an array and reshaped to match the input shape of the model. An in-built function, `preprocess_input`, is used to prepare the image for the respective model. For the VGG16 model, the preprocessed image is passed directly through the model to extract features. For the ViT model, the preprocessed image is further converted to a PyTorch tensor before being passed through the model. In both cases, the models process the images to extract features. These extracted features are stored in a dictionary, "mapping", with the image ID as the key. This process is repeated for all images in our dataset. This allows us to build a comprehensive set of im-

age features for training our language models. The extracted features are saved in Python pickle file format so they can be reused later when needed.

3.4 Preprocessing

Preprocessing is a crucial step in any machine learning or natural language processing task. It involves preparing and cleaning the raw data to make it suitable for a model. The goal of preprocessing is to remove any obstacles that might hinder the learning process. This helps in improving the performance of the model. It often leads to a more accurate and efficient image captioning model [45].

In the context of our task, we dealt with textual data in the form of captions. Text data is often messy and unstructured, and requires a series of preprocessing steps to make it suitable for a model. We built a function called clean. This includes the following preprocessing steps:

(1) Lowercasing: The first step in the function is to convert all the captions to lowercase. This is done to ensure that the model treats words like "Hello" and "hello" as the same word, rather than two different words.

(2) Removing special characters and digits: The next step is to remove special characters and digits from the captions. This is done to simplify the text and reduce the size of the vocabulary that the model needs to learn.

(3) Removing additional spaces: We then remove any additional spaces in the captions. This is done to ensure that the model does not treat multiple spaces as a meaningful pattern.

(4) Adding start and end tags: We add ‘startseq’ and ‘endseq’ tags to the beginning and end of each caption. This is a common practice in sequence generation tasks like image captioning or machine translation. The ‘startseq’ tag helps the model know when to start generating the caption, and the ‘endseq’ tag tells the model when to stop.

(5) Removing single-letter words: We remove any single-letter words from the captions. This is done to eliminate words that might not carry much meaning and could potentially confuse the model.

Figure 3.1: Preprocessing Example

A large white bird goes across the water A white bird is flying off the water surface . A white bird preparing to catch something in the water The large white bird's reflection shows in the water . White bird walking across wet sand .
startseq large white bird goes across the water endseq startseq white bird is flying off the water surface endseq startseq white bird preparing to catch something in the water endseq startseq the large white bird's reflection shows in the water endseq startseq white bird walking across wet sand endseq

Table 3.1: Actual Captions vs Captions After Cleaning

Following the cleaning of the captions, the next step in the preprocessing pipeline is creating a unified list of all captions and then, the tokenization of these captions.

We compiled all the cleaned captions into a single list called `all_captions`. This is done by iterating over each key (representing an image ID) in the mapping dictionary and appending each associated caption to the `all_captions` list. The main purpose of the list is to gather all captions in one place for vocabulary creation, tokenization, and potential analysis.

Next, the tokenization process is carried out. A Tokenizer object from the Keras library is instantiated. This object is then fitted on `all_captions` using the `fit_on_texts` method, which updates the tokenizer's internal vocabulary based on the list of captions. This vocabulary is used to convert text to sequences of integers. This form can be fed into the model.

Then, we have adopted a three-way split for our dataset. This includes training, validation, and testing sets. This approach is a common practice in machine learning tasks and allows for effective training and evaluation of the model. Specifically, 70% of the dataset is allocated for training the model. This is the largest portion and is used to train the model to make predictions based on this data. 10% of the dataset is set aside for validation during the training process. The validation set provides an unbiased evaluation of the model while tuning the hyperparameters, such as learning rate and epochs. The remaining 20% of the dataset is used for testing the model. The testing set provides a final evaluation of the model's performance [46].

Finally, we created function called `data_generator`. This function prepares batches of data for training the image captioning model. It is designed to be memory efficient and only loads one batch of data into memory at a time. This is particularly useful when working with large datasets that don't fit into memory. It helps to avoid session crashes during training.

To do this, it iterates over each key, which represents a unique identifier for images. It fetches image features and the corresponding captions. Note that there are 5 captions for each image. Hence, when we train using flickr8k dataset, there are 40k image captions and using flickr30k, there are over 150k image captions.

Then, for each caption of the current image, it tokenizes the caption into a sequence of integers, where each integer represents a unique word. It then creates multiple pairs of input and output sequences from this sequence. The input sequence consists of the first n words of the caption, and the output sequence is the $(n+1)$th word. This way, the model can learn to predict the next word given the previous words. For each input-output pair, it pads the input sequence to ensure all input sequences have the same length, and one-hot encodes the output sequence to match the vocabulary size. These processed sequences, along with the image features, are stored in lists.

Once it has prepared a batch of data (the size of the batch is determined by the `batch_size` parameter), it yields this batch. The batch consists of two input arrays (image features and input sequences) and one output array (output sequences). After yielding a batch, it resets the lists and continues with the next batch.

It ensures that the model receives data in the correct format and size during training, and does so in a memory-efficient manner. This completes the process of data preparation required to train the language model.

3.5 Language Models

The choice of language model played a pivotal role in the architecture of our image captioning model and its performance. There were many options which we could have explored, including traditional recurrent neural networks (RNNs), attention mechanisms, and transformer-based models. We decided to utilize Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks. This was based on several key reasons.

(1) Handling Long-Term Dependencies: Both LSTM and GRU networks are specifically designed to capture long range dependencies within sequential data effectively. This characteristic is crucial in image captioning tasks, as the generated captions often require contextual understanding of the entire image.

(2) Learning Complex Patterns: Both networks are well-suited for learning complex patterns in sequential data. This allows the models to capture subtle nuances and semantic relationships between visual elements. As a result, it produces more descriptive and contextually relevant captions.

(3) Robustness to Variability: The models offer robustness to variability in the images because of their adaptive gating mechanisms. This adaptability allows the models to effectively handle diverse image inputs and generate captions that are both accurate and diverse.

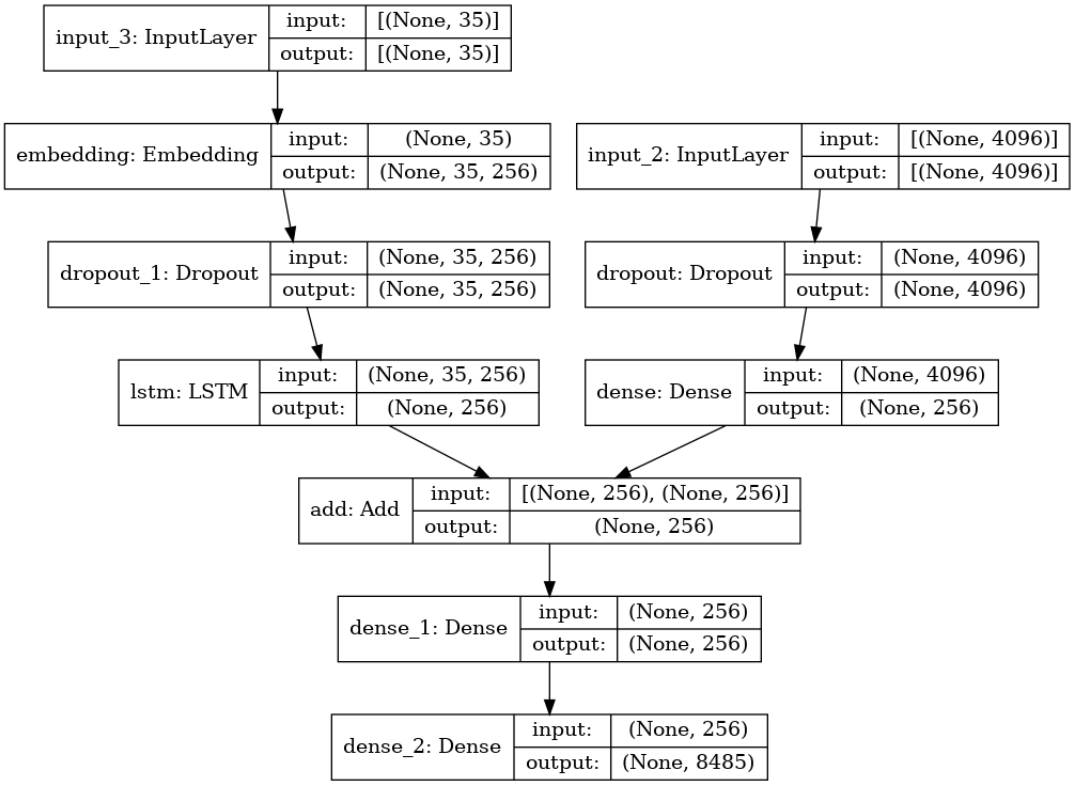
(4) Established performance: Their effectiveness has been well-documented in the fields like machine translation, sentiment analysis, and text generation. This provides a strong basis for their use in our task of image captioning.

(5) Model Complexity and Training Efficiency: While more recent architectures such as transformers have shown remarkable performance in various NLP tasks, they often come with increased computational complexity and training requirements. In contrast, GRU and LSTM networks strike a balance between model performance and computational efficiency. This makes them more practical choices for image captioning tasks, especially in scenarios with resource constraints [47].

By training separate models with these two distinct types of RNNs, we aim to understand the strengths and weaknesses of our image captioning models across different architectural complexities and sequence handling capabilities. This approach allows us to evaluate how each type of RNN contributes to the performance of the models and helps us in identifying the most effective strategies for image captioning tasks.

3.6 Decoder

The decoder is designed as a combination of feature extraction layers for the image and sequence feature layers for the text input. The image features and the text sequences are processed through separate paths and then merged together to form a unified representation. This representation is then passed through a dense layer with a relu activation function which introduces non-linearity into the model. The final Dense layer uses a softmax activation function and outputs a probability distribution over all possible words in the vocabulary.



The model is trained using the Adam optimizer and the Categorical Cross-Entropy loss function. Adam, short for Adaptive Moment Estimation, is an optimization algorithm that uses adaptive learning rates for different parameters, making it efficient for problems that are large in terms of data and/or parameters [48]. The Categorical Cross-Entropy loss function is used for multi-class classification problems, measuring the performance of the model's output in terms of probability value between 0 and 1. The closer the model's outputs are to the target values, the lower the loss.

To prevent overfitting and improve the model's generalization capability, different regularization techniques were used in separate models: Dropout and L2 Regularization (Ridge). Dropout is a technique where randomly selected neurons are ignored during training, helping the model to generalize better [49]. On the other hand, L2 regularization discourages large weights in the model by adding the squared value of the weights to the loss function, preventing the model from relying too heavily on any single feature.

In some models, an additional technique called Batch Normalization was used. Batch normalization is a technique used to improve the training of deep neural networks. It normalizes the layers' inputs by recentering and rescaling, which helps in reducing the internal covariate shift, making the training faster and more stable [50].

To sum up, the decoder takes as input the features extracted from an image and a sequence of words, and outputs a probability distribution over the next word in the caption. The model is trained to minimize the difference between its predictions and the actual next words in the captions. After training, the model can generate a caption for a new image by predicting one word at a time, using the image's features and the words it has generated so far as input. After training, we save the models for future use and analysis.

3.7 Caption Generation

To complete the last step of our task, generating captions, we built two functions: `idx_to_word` and `predict_caption`.

The `idx_to_word` function simply takes an integer and a tokenizer as input and returns the corresponding word from the tokenizer's word index.

The `predict_caption` function generates a caption for a given image. It takes as input the trained model, the image features, the tokenizer, and the maximum length of the captions. The function starts with the word 'startseq' and then enters a loop where it predicts the next word based on the current sequence of words and the image features. The current sequence is tokenized and padded to the maximum length, and then the model predicts the next word.

The prediction is a probability distribution over the vocabulary from which the word with the highest probability is selected. This word is then added to the current sequence. The loop continues until the maximum length is reached or the word 'endseq' is predicted, indicating the end of the caption.

Predicted Captions: "startseq girl in bikini jumping into the water endseq"

Chapter 4

Experimentation and Evaluation

4.1 Setting up experiments

After creating the template of the model, it was time to set up the experiments. Planning for experiments was done to ensure we get an insight of the strengths and weakness of all parts of the image captioning model.

4.1.1 VGG16

We used VGG16 as image encoder to train 12 models. 10 of these were trained on the flickr8k dataset and the other two were trained on the flickr30k dataset.

Flickr8k

- **LSTM**

  We created five different models with varying configurations to investigate the impact of different parameters and techniques on the performance of the models. Let us call them models 1, 2, 3, 4 and 5.

  Models 1, 2 and 3 consisted of a single LSTM layer and used Dropout for regularization. These models were trained for 20 epochs, 50 epochs and 100 epochs. This is mainly done to observe how the model's performance evolves over time. Model 4 had two LSTM layers, making it deeper compared to the first three models. This helps us to explore the model's capacity and complexity. Model 5 had a single LSTM layer but did not use Dropout. Instead, it used L2 regularization along with Batch Normalization. Models 4 and 5 were trained for 20 epochs.

Figure 4.1: Example Model 1 (VGG16)

Predicted captions by model 1: "startseq two dogs playing with each other endseq"

- **GRU**

  We created five different models using GRU. Let us call them models 6, 7, 8, 9 and 10.

  Models were similar to their LSTM counterparts with the only difference being these used GRU instead of LSTM. Model 6 corresponds to 1, 7 to 2, 8 to 3, 9 to 4 and 10 to 5.

Figure 4.2: Example Model 6 (VGG16)

Predicted captions by model 6: “startseq white bird flying over the water endseq"

Flickr30k

For the Flickr30k dataset, I trained two additional models using VGG16 as the image encoder. This helps us to understand the strengths and weaknesses of our image captioning models across different data volumes. Both models were trained with a batch size of 64 for 20 epochs. They used dropout for regularization. Model 11 used LSTM while 12 used GRU.

Figure 4.3: Example Model 11 (VGG16)

Predicted captions by model 11: “startseq two men are playing poker endseq"

4.1.2 ViT

In addition to the models trained using VGG16, we also experimented with the Vision Transformer (ViT) as the image encoder. The choice of LSTM and GRU was again explored in these models.

Flickr8k

Four models were trained on the Flickr8k dataset. Model 1 and Model 2 used LSTM and were trained for 20 and 50 epochs respectively. Model 3 and Model 4 used GRU and were also trained for 20 and 50 epochs respectively. All four models used dropout for regularization.

Figure 4.4: Example Model 2 (ViT)

Predicted captions by model 2: “startseq little boy in swim trunks is walking in the sand endseq"

Flickr30k

Two additional models were trained on the Flickr30k dataset. Model 5 used LSTM and model 6 used GRU. Both models were trained for 20 epochs and used dropout for regularization.

These experiments were conducted to understand the performance of the models when using ViT as the image encoder.

Figure 4.5: Example Model 6 (ViT)

Predicted captions by model 2: “startseq two men are playing soccer endseq "

4.2 Evaluation

4.2.1 Performance Metrics

For evaluating our predictions, we used a popular metric called the BLEU (Bilingual Evaluation Understudy) score. It compares the generated text with reference texts and calculates a score based on the precision of n-grams. Specifically, we used BLEU 1 and BLEU 2 [51].

By using both BLEU-1 and BLEU-2, we were able to provide a more comprehensive evaluation of our model. While BLEU-1 assessed our model’s ability to generate relevant words individually, BLEU-2 evaluated its ability to generate words in the correct order, which is crucial for producing grammatically correct and semantically coherent captions [51].

The following table 4.1 provides a straightforward comparison of our models through BLEU scores:

Table 4.1: BLEU Scores for VGG16 Models

VGG16	BLEU 1	BLEU 2
Model 1	0.530617	0.307579
Model 2	0.519077	0.293305
Model 3	0.508832	0.288978
Model 4	0.533071	0.304085
Model 5	0.530610	0.301645
Model 6	0.545339	0.312137
Model 7	0.517999	0.291111
Model 8	0.505403	0.285044
Model 9	0.528948	0.304769
Model 10	0.497757	0.273092
Model 11	0.548011	0.302180
Model 12	0.558658	0.308375

We can derive the following insights:

(1) Model Performance: Model 12, which was trained on the Flickr30k dataset using VGG16 as the image encoder and GRU as the decoder, achieved the highest BLEU-1 and BLEU-2 scores among all models. This suggests that this model was most effective at generating relevant words (as indicated by the BLEU-1 score) and arranging them in the correct order (as indicated by the BLEU-2 score).

(2) Impact of Training Dataset: Models trained on the Flickr30k dataset (Models 11 and 12) generally outperformed those trained on the Flickr8k dataset in terms of both BLEU-1 and BLEU-2 scores. This indicates that the larger volume of data in the Flickr30k dataset contributed to better model performance.

(3) Decoder Architecture: Comparing the performance of models using LSTM (Models 1-5, 11) and GRU (Models 6-10, 12) as decoders, it appears that performance of both were loosely the same. This might suggest that the choice between LSTM and GRU did not significantly impact when using VGG16 as the image encoder.

(4) Regularization Techniques: Comparing Model 5 (which used L2 regularization and Batch Normalization) with Models 1-4 (which used Dropout), it seems that the choice of regularization technique did not significantly impact the BLEU scores. This suggests that both Dropout and L2 regularization with Batch Normalization can be effective for this task.

(5) Model Complexity: Comparing Model 4/9 (which had two LSTM/GRU layers) with Models 1-3/6-8 (which had a single LSTM/GRU layer), the additional complexity did not lead to a significant improvement in BLEU scores but ended up degrading efficiency. This could indicate that a single LSTM/GRU layer is sufficient for this task, and adding more layers might not necessarily improve performance.

From the table 4.2 below, we can derive the following insights:

Table 4.2: BLEU Scores for ViT Models

ViT	BLEU 1	BLEU 2
Model 1	0.461870	0.215882
Model 2	0.438290	0.200381
Model 3	0.454991	0.214120
Model 4	0.447608	0.210303
Model 5	0.460261	0.205046
Model 6	0.468518	0.217186

(1) Model Performance: Model 6, which used GRU as the decoder, achieved the highest BLEU-1 and BLEU-2 scores among all models using ViT. This suggests that this model was most effective at generating relevant words (as indicated by the BLEU-1 score) and arranging them in the correct order (as indicated by the BLEU-2 score).

(2) Impact of Training Dataset: Models trained on the Flickr30k dataset (Models 5 and 6) generally outperformed those trained on the Flickr8k dataset in terms of both BLEU-1 and BLEU-2 scores. This could indicate that the larger volume of data in the Flickr30k dataset contributed to better model performance.

(3) Decoder Architecture: Comparing the performance of models using LSTM (Models 1, 2, 5) and GRU (Models 3, 4, 6) as decoders, it appears that performance of both were loosely the same. This might suggest that the choice between LSTM and GRU did not significantly impact when using ViT as the image encoder.

4.2.2 Peer Evaluation

It is important to note that while BLEU scores provide a quantitative measure of the model's performance, they may not always align with human judgment of the quality of the generated captions. Other factors such as the relevance and descriptiveness of the captions in the context of the images are also important for a comprehensive evaluation of an image captioning model [52].

Therefore, with the assistance of 20 academic peers, unbiased testing was conducted to get a more human standpoint of the performance. It was kept in mind that, enough test cases were provided to

assess different aspects but not so many that the process became burdensome. Each of the 20 was provided with 20 images per model and the captions predicted by each model. They were requested to rate the predicted captions on a linear scale of 1 to 5, 1 indicating that the caption is not correct at all and 5 indicating that the generated caption is fully correct for the image. Note, that there were no ethical considerations as the process of surveys was kept anonymous.

To analyse the feedback let us look at the following tables 4.3 and 4.4 for a straightforward comparison:

Table 4.3: Average Scores for VGG16 Models

VGG16	**Average Score**
Model 1	3.32
Model 2	2.84
Model 3	3.15
Model 4	2.83
Model 5	2.32
Model 6	3.39
Model 7	2.95
Model 8	2.85
Model 9	2.34
Model 10	2.42
Model 11	3.52
Model 12	3.62

Based on the peer evaluation scores for the models using VGG16, here are some insights:

(1) Model Performance: Model 12, which was trained on the Flickr30k dataset using VGG16 as the image encoder and GRU as the decoder, achieved the highest average score of 3.62. This suggests that this model was most effective at generating captions that were rated highly by the peers.

(2) Impact of Training Dataset: Similar to the BLEU scores, models trained on the Flickr30k dataset (Models 11 and 12) generally outperformed those trained on the Flickr8k dataset in terms of peer evaluation scores. This indicates that the larger volume of data in the Flickr30k dataset contributed to better model performance.

(3) Decoder Architecture: Comparing the performance of models using LSTM (Models 1-5, 11) and GRU (Models 6-10, 12) as decoders, it appears that performance of both were loosely the same. This might suggest that the choice between LSTM and GRU did not significantly impact when using VGG16 as the image encoder.

(4) Model Complexity: Comparing Model 4 (which had two LSTM layers) with Models 1-3 (which had a single LSTM layer), the additional complexity did not lead to a significant improvement

in peer evaluation scores. This could indicate that a single LSTM layer is sufficient for this task, and adding more layers might not necessarily improve performance.

(5) Regularization Techniques: Comparing Model 5 (which used L2 regularization and Batch Normalization) with Models 1-4 (which used Dropout), it seems that the choice of regularization technique made a significant impact on the peer evaluation scores. This suggests that Dropout might be more effective for this task.

These insights largely align with the findings from the BLEU scores and provide a more comprehensive understanding of the model performance from a human perspective.

Table 4.4: Average Scores for ViT Models

ViT	Average Score
Model 1	1.78
Model 2	1.68
Model 3	1.74
Model 4	1.76
Model 5	2.05
Model 6	2.08

Based on the peer evaluation scores for the models using Vision Transformer (ViT), here are some insights:

(1) Model Performance: Model 6, which used GRU as the decoder, achieved the highest average score of 2.08. This suggests that this model was most effective at generating captions that were rated highly by the peers.

(2) Impact of Training Dataset: Models trained on the Flickr30k dataset (Models 5 and 6) generally outperformed those trained on the Flickr8k dataset in terms of peer evaluation scores. This indicates that the larger volume of data in the Flickr30k dataset contributed to better model performance.

(3) Decoder Architecture: Comparing the performance of models using LSTM (Models 1, 2, 5) and GRU (Models 3, 4, 6) as decoders, it appears that performance of both were loosely the same. This might suggest that the choice between LSTM and GRU did not significantly impact when using ViT as the image encoder.

It is worth noting that the average scores for the models using ViT were significantly lower than those using VGG16. This could suggest that VGG16, being a convolutional neural network, might be better at capturing local features in the images, which could be more relevant for the task of image captioning.

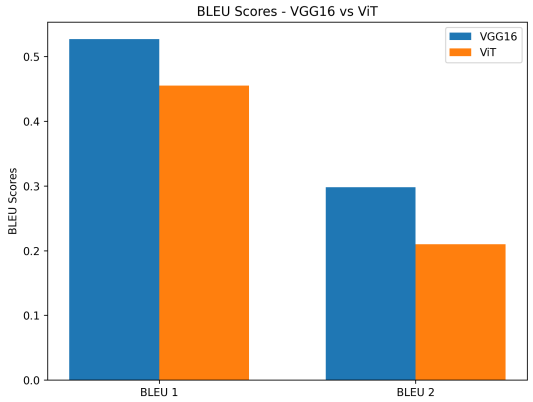


Figure 4.6: VGG16 vs ViT (BLEU)

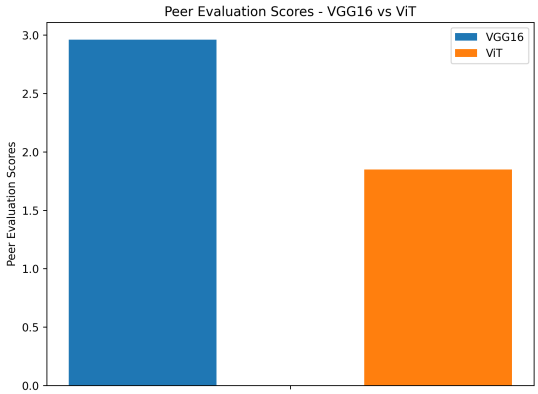


Figure 4.7: VGG16 vs ViT (Peer)

VGG16 vs ViT: Both the BLEU scores and peer evaluations suggest that models using pre-trained VGG16 as the image encoder generally outperformed those using pre-trained ViT. VGG16, being a convolutional neural network, might be better at capturing local features in the images, which could be more relevant for the task of image captioning. On the other hand, ViT, being a transformer-based model, captures global, long-range dependencies in the image, which might not be as useful for this specific task.

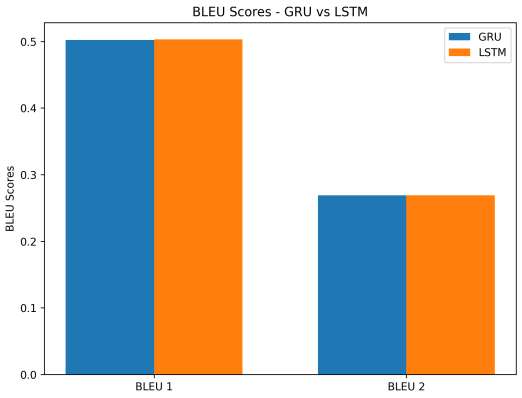


Figure 4.8: GRU vs LSTM (BLEU)

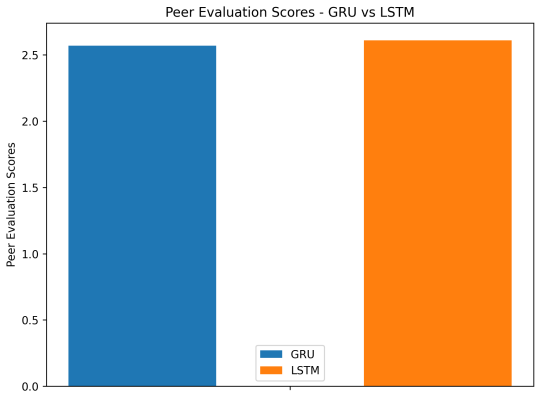


Figure 4.9: GRU vs LSTM (Peer)

GRU vs LSTM: Comparing the performance of models using LSTM and GRU as decoders, it appears that performance of both were loosely the same. This might suggest that the choice between LSTM and GRU did not significantly impact the performance of the model. Although, GRU has fewer parameters than LSTM [53] and is computationally more efficient, which might be a contributing factor. This makes GRU a more convenient choice as it takes less time to train.

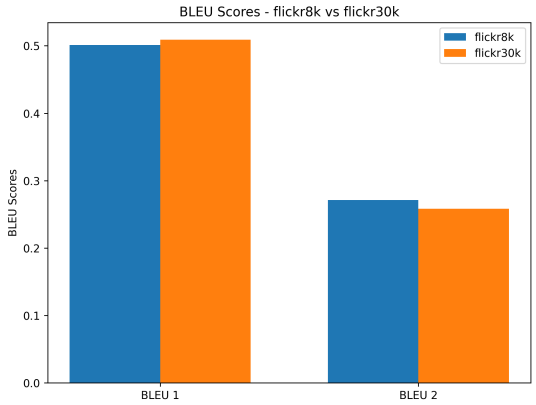


Figure 4.10: Flickr8k vs Flickr30k (BLEU)

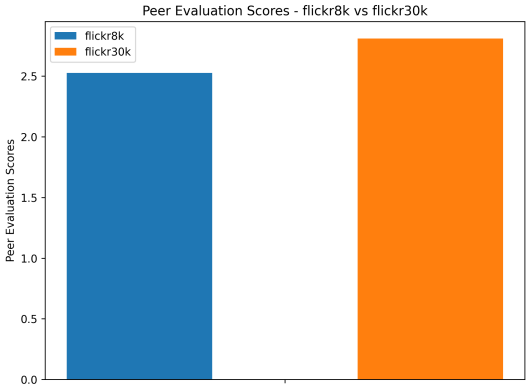


Figure 4.11: Flickr8k vs Flickr30k (Peer)

Flickr8k vs Flickr30k: Models trained on the Flickr30k dataset generally outperformed those trained on the Flickr8k dataset in terms of both BLEU-1 and BLEU-2 scores, as well as peer evaluation scores. This indicates that the larger volume of data in the Flickr30k dataset contributed to better model performance. The graph makes their performance seem closer, This is due to the fact that the ratio of models trained by VGG16 on flickr30k to models trained by ViT on flickr30k is 1:1 while the ratio of models trained by VGG16 on flickr8k to models trained by ViT on flickr8k is 5:2. This affects the graph heavily as performance of ViT was significantly lower than VGG16.

Chapter 5

Conclusion and Future Works

5.1 Results

As we saw in the examples, the models were able to produce fully corrected captions for some cases and partially correct for some. They also produced some captions which were completely unrelated to the images. This shows that the model has gaps and can be improved.

The experimentation and evaluation of the image captioning models provided valuable insights into their performance and effectiveness. The models were trained using two different image encoders, VGG16 and Vision Transformer (ViT), and two different decoders, LSTM and GRU. They were also trained on two different datasets, Flickr8k and Flickr30k.

For models using VGG16 as the image encoder:

- Model 12, trained on the Flickr30k dataset using VGG16 as the image encoder and GRU as the decoder, achieved the highest BLEU-1 score of 0.558658, BLEU-2 score of 0.308375, and an average peer evaluation score of 3.62.

- Models trained on the Flickr30k dataset generally outperformed those trained on the Flickr8k dataset. For instance, Model 11 achieved a BLEU-1 score of 0.548011, BLEU-2 score of 0.302180, and an average peer evaluation score of 3.52.

- Models using GRU had loosely the same scores as models using LSTM. Although with best training parameters we see that GRU outperforms LSTM. For example, Model 6 achieved a BLEU-1 score of 0.545339, BLEU-2 score of 0.312137, and an average peer evaluation score of 3.39 while Model 1 achieved a BLEU-1 score of 0.530617, BLEU-2 score of 0.307579, and an average peer evaluation score of 3.32

For models using ViT as the image encoder:

- Model 6, which used GRU as the decoder, achieved the highest BLEU-1 score of 0.468518, BLEU-2 score of 0.217186, and an average peer evaluation score of 2.08.

- Models trained on the Flickr30k dataset generally outperformed those trained on the Flickr8k dataset. For instance, Model 5 achieved a BLEU-1 score of 0.460261, BLEU-2 score of 0.205046, and an average peer evaluation score of 2.05.

Throughout our experimentation, we observed a trend of overfitting as the number of training epochs increased. This overfitting is likely due to the limited amount of training data available. Consequently, the models that were trained for a higher number of epochs tended to memorize the training

data too closely, leading to poor generalization on unseen data. This phenomenon explains why models trained for 20 epochs generally yielded better results, as they were less prone to overfitting.

It is worth noting that the average scores for the models using ViT were significantly lower than those using VGG16. This could suggest that VGG16 might be better at capturing local features in the images, which could be more relevant for the task of image captioning. On the other hand, ViT, a transformer-based model, captures global, long-range dependencies in the image, which might not be as beneficial for this specific task.

In conclusion, the results of the experimentation and evaluation suggest that using VGG16 as the image encoder, GRU as the decoder, and training on a larger dataset like Flickr30k can lead to better performance in the task of image captioning. However, these findings are specific to the current task and datasets, and may vary for different tasks and datasets. Further research and experimentation are needed to generalize these findings.

5.2 Limitations

While the experimentation and evaluation of the image captioning models have yielded promising results generally, it is important to acknowledge the limitations of this study.

Firstly, the models were trained on two specific datasets, Flickr8k and Flickr30k. These datasets, although diverse, may not fully represent the vast array of images that the models may encounter in real-world applications. The performance of the models may vary when exposed to images from different areas or with different characteristics.

Secondly, the models were evaluated using the BLEU metric and peer evaluations. While these methods provide valuable insights into the model's performance, they may not fully capture the quality of the generated captions. For instance, the BLEU metric, which is based on n-gram precision, may not properly reflect the semantic relevance of the captions. Similarly, peer evaluations, although providing a human perspective, are subjective and may vary across different individuals.

Thirdly, the models used specific architectures, namely VGG16, ViT, LSTM, and GRU. While these architectures have proven effective for this task, there may be other architectures or variations that could potentially yield better results.

Fourthly, the computational demands of training complex models for image captioning were quite high. These models often require substantial computational resources, particularly a powerful GPU, for efficient training. Without such resources, training more complex models was not be feasible. This restricted the exploration of potentially more effective architectures and techniques in image captioning.

Lastly, the models were trained for a fixed number of epochs, which may not have been sufficient for the models to fully converge. Different training strategies might lead to improved performance.

5.3 Future Works

The results of this study provide a solid foundation for future research in the field of image captioning. However, there are several avenues that could be explored to further enhance the performance and applicability of these models.

Firstly, expanding the diversity of the training datasets could potentially improve the generalization of the models. The current study utilized the Flickr8k and Flickr30k datasets, which, while diverse, may not fully represent the wide array of images that the models may encounter in real-world applications. Incorporating more diverse datasets, including those with images from different domains or with different characteristics, could help the models learn more robust and generalizable features.

Secondly, exploring alternative model architectures could lead to improved performance. This study utilized VGG16 and Vision Transformer (ViT) as image encoders and LSTM and GRU as decoders. While these architectures have proven effective for this task, there may be other architectures or variations that could potentially yield better results. For instance, other types of convolutional neural networks like ResNet and DenseNet could be explored.

Thirdly, refining the training strategies could also enhance the model performance. The models in this study were trained for a fixed number of epochs, which may not have been sufficient for the models to fully converge. Different training strategies, such as early stopping or learning rate scheduling, might lead to improved performance.

Fourthly, due to the high computational demands of training complex models, more powerful computational resources would be beneficial. Access to more powerful GPUs would allow for the training of more complex models and the exploration of potentially more effective architectures and techniques in image captioning.

Lastly, developing more comprehensive evaluation metrics could provide a better understanding of the model's performance. While BLEU scores and peer evaluations provide valuable insights, they may not fully capture the quality of the generated captions. Future work could explore other evaluation metrics that consider additional aspects of the captions, such as their relevance, descriptiveness, and grammatical correctness.

In conclusion, while the results of this study are promising, there is ample opportunity for future research to further advance the field of image captioning. By addressing these potential areas of improvement, future work can continue to push the boundaries of what is possible in image captioning.

Bibliography

[1] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. Deep learning approaches on image captioning: A review. *ACM Computing Surveys*, 56(3):1–39, October 2023. ISSN 1557-7341. doi: 10.1145/3617592. URL http://dx.doi.org/10.1145/3617592.

[2] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. From show to tell: A survey on deep learning-based image captioning, Nov 2021. URL https://arxiv.org/abs/2107.06912.

[3] Priya Singh, Chandan Kumar, and Ayush Kumar. Next-lstm: a novel lstm-based image captioning technique. *International journal of system assurance engineering and management*, 14(4):1492–1503, Jun 2023. doi: https://doi.org/10.1007/s13198-023-01956-7.

[4] gaudenz boesch. Vision transformers (vit) in image recognition - 2022 guide, Sep 2021. URL https://viso.ai/deep-learning/vision-transformer-vit/.

[5] Madhuri Bhalekar, Shubham Sureka, Shaunak Joshi, and Mangesh Bedekar. Generation of image captions using vgg and resnet cnn models cascaded with rnn approach. *Advances in intelligent systems and computing*, page 27–42, Jan 2020. doi: https://doi.org/10.1007/978-981-15-1366-4_3.

[6] Jason Brownlee. A gentle introduction to calculating the bleu score for text in python, Nov 2017. URL https://machinelearningmastery.com/calculate-bleu-score-for-text-python/.

[7] Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Explaining transformer-based image captioning models: An empirical analysis. *AI Communications*, page 1–19, Oct 2021. doi: https://doi.org/10.3233/aic-210172.

[8] Mohit Tripathi. Image processing using cnn | beginner's guide to image processing, Jun 2021. URL https://www.analyticsvidhya.com/blog/2021/06/image-processing-using-cnn-a-beginners-guide/.

[9] geeksforgeeks. Vgg-16 | cnn model, Feb 2020. URL https://www.geeksforgeeks.org/vgg-16-cnn-model/.

[10] datagen. Understanding vgg16: Concepts, architecture, and performance. URL https://datagen.tech/guides/computer-vision/vgg16/.

[11] Great Learning Team. Introduction to vgg16 | what is vgg16?, Oct 2021. URL https://www.mygreatlearning.com/blog/introduction-to-vgg16/#VGG%2016%20Architecture.

[12] Rohit Thakur. Beginners guide to vgg16 implementation in keras | built in, Mar 2023. URL https://builtin.com/machine-learning/vgg16.

[13] Asrar Almogbil, Amjad Alghamdi, Arwa Alsahli, Jawaher Alotaibi, Razan Alajlan, and Fadiah Alghamdi. A comparison between vgg16 and xception models used as encoders for image captioning. *Computer Science and Information Technology*, Jul 2022. doi: https://doi.org/10.5121/csit.2022.121316.

[14] paperswithcode. Papers with code - vision transformer explained. URL https://paperswithcode.com/method/vision-transformer.

[15] Arjun Sarkar. Are transformers better than cnn's at image recognition?, May 2021. URL https://towardsdatascience.com/are-transformers-better-than-cnns-at-image-recognition-ced60ccc7c8.

[16] Manish Chablani. Vision transformer (vit) — an image is worth 16x16 words: Transformers for image recognition at. . . , Feb 2024. URL https://medium.com/@ManishChablani/vision-transformer-vit-an-image-is-worth-16x16-words-transformers-for-image-recogni

[17] Skylar Jean Callis. Vision transformers, explained, Feb 2024. URL https://towardsdatascience.com/vision-transformers-explained-a9d07147e4c8.

[18] Yingzi Huo, Kai Jin, Jiahong Cai, Huixuan Xiong, and Jiacheng Pang. Vision transformer (vit)-based applications in image classification. pages 135–140, 05 2023. doi: 10.1109/BigDataSecurity-HPSC-IDS58521.2023.00033.

[19] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers, 2022.

[20] GeeksforGeeks. Introduction to recurrent neural network - geeksforgeeks, Oct 2018. URL https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/.

[21] Data Science Courses. Sequential data analysis with recurrent neural networks, Feb 2024. URL https://datasciencectraining.medium.com/sequential-data-analysis-with-recurrent-neural-networks-e3d8f544d416.

[22] Jeremy Cohen. Recurrent neural networks (rnns) in computer vision: Image captioning, Aug 2023. URL https://www.comet.com/site/blog/recurrent-neural-networks-rnns-in-computer-vision-image-captioning/.

[23] geeksforgeeks123. Vanishing and exploding gradients problems in deep learning, Dec 2023. URL https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/.

[24] Yash Bohra. Vanishing and exploding gradients in deep neural networks, Jun 2021. URL https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/.

[25] gfg. Long short term memory networks explanation, Jul 2019. URL https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/.

[26] Shikha Gupta. Image caption generator using deep learning, Dec 2021. URL https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/.

[27] Anishnama. Understanding gated recurrent unit (gru) in deep learning, May 2023. URL https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2.

[28] d2l. 9.1. gated recurrent units (gru) — dive into deep learning 0.16.6 documentation. URL https://d2l.ai/chapter_recurrent-modern/gru.html.

[29] My Abdelouahed Sabri, Hamza El Madhoune, Chaimae Zouitni, and Abdellah Aarab. Image captioning: An understanding study. *Lecture notes in networks and systems*, page 482–491, Jan 2023. doi: https://doi.org/10.1007/978-3-031-29860-8_49.

[30] Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. URL https://www.ijcai.org/Proceedings/2017/0704.pdf.

[31] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. Deep learning approaches on image captioning: A review. *arXiv:2201.12944 [cs]*, Nov 2022. URL https://arxiv.org/abs/2201.12944.

[32] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator, 2014. URL https://arxiv.org/abs/1411.4555.

[33] tensorflow. Image captioning with visual attention | text. URL https://www.tensorflow.org/text/tutorials/image_captioning.

[34] Ketan Doshi. Image captions with attention in tensorflow, step-by-step, Apr 2021. URL https://towardsdatascience.com/image-captions-with-attention-in-tensorflow-step-by-step-927dad3569fa.

[35] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv:2201.12086 [cs]*, Feb 2022. URL https://arxiv.org/abs/2201.12086.

[36] Sophia Zell. Good caption hunting. URL https://tech.bertelsmann.com/en/blog/articles/good-caption-hunting.

[37] GFOFG. Regularization techniques in machine learning, Feb 2024. URL https://www.geeksforgeeks.org/regularization-techniques-in-machine-learning/.

[38] Ahmed Sabir. Paper summary: Blip: Bootstrapping language-image pre-training for unified vision-language..., Jul 2023. URL https://ahmed-sabir.medium.com/paper-summary-blip-bootstrapping-language-image-pre-training-for-unified-vision-lan

[39] URL https://paperswithcode.com/task/image-captioning.

[40] Sanqiang Zhao, Piyush Sharma, Tomer Levinboim, and Radu Soricut. Informative image captioning with external sources of information, Jun 2019. URL https://arxiv.org/abs/1906.08876.

[41] Yuepeng Li. Image caption using vgg model and lstm. *Applied and Computational Engineering*, 48:68–77, 03 2024. doi: 10.54254/2755-2721/48/20241175.

[42] Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. Image captioning with deep bidirectional lstms, 2016.

[43] . URL https://www.kaggle.com/datasets/dibyansudiptiman/flickr-8k.

[44] . URL https://datasets.activeloop.ai/docs/ml/datasets/flickr30k-dataset/#:~:text=dataset%20for%20Python%3F-.

[45] Kendall Fortney. Pre-processing in natural language machine learning, Nov 2017. URL https://towardsdatascience.com/pre-processing-in-natural-language-machine-learning-898a84b8bd47.

[46] Sanyamdhawan. Splitting data for machine learning models, Jun 2020. URL https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/.

[47] Mar 2024. URL https://algorithmicmind.org/gru-vs-lstm/.

[48] Joyita Ghosh and Subir Gupta. Adam optimizer and categorical crossentropy loss function-based cnn method for diagnosing colorectal cancer. In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pages 470–474, 2023. doi: 10.1109/CISES58720.2023.10183491.

[49] Oct 2020. URL https://vitalflux.com/keras-categorical-cross-entropy-loss-function/.

[50] Brian Mutea. Image captioning model with tensorflow, transformers, and kangas for image visualization, Dec 2023. URL https://www.comet.com/site/blog/image-captioning-model-with-tensorflow-transformers-and-kangas-for-image-visualizat

[51] Oct 2022. URL https://www.geeksforgeeks.org/nlp-bleu-score-for-evaluating-neural-machine-translation-python/.

[52] Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. *Learning to Evaluate Image Captioning*. URL https://openaccess.thecvf.com/content_cvpr_2018/papers/Cui_Learning_to_Evaluate_CVPR_2018_paper.pdf.

[53] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation, Jul 2019. URL https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.