Universiteit Leiden

Master Computer Science

Exploring the Application of the BLIP-2 Model on Soccer Video Data

Name:	Bohan Wei
Student ID:	s3600246
Date:	13/08/2024
Specialisation:	Computer Science
1st supervisor:	Hazel Doughty
2nd supervisor:	Lu Cao

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my first supervisor, Hazel, and my second supervisor, Lu Cao. Throughout the past few months of writing my thesis, I encountered numerous difficulties regarding ideas, implementation, and mental challenges. Hazel patiently helped me overcome these obstacles and provided invaluable guidance.

Secondly, I want to thank my friends. During the past six months, I went through many tough times, and their care and support gave me the hope and strength to face life positively.

Lastly, I would like to thank myself for my dedication and perseverance throughout this journey, overcoming challenges, and staying committed to achieving my goals.

Table of Contents

List of Figures

List of Tables

Chapter 1 Abstract

A new model called BLIP-2 has emerged in the multimodal field. This image-to-language model is designed for visual question answering, image captioning, and image-text retrieval. In this thesis, we explore an innovative approach to using the BLIP-2 model for video captioning tasks on the SoccerNet dataset. Several strategies are employed to adapt video data for use with an image-language model. Specifically, this experiment uses average pooling to fuse video features over time before integrating them into the BLIP-2 model.

Chapter 2 Introduction

As the development and commercialization of sports become increasingly successful, more and more people are enjoying the excitement that sports bring. Football, the world's most popular sport, is loved by people all over the globe. Meanwhile, the development of the gaming industry has given rise to many sports video games, such as basketball, FIFA for soccer, ML category for baseball, and many more.

There are numerous benefits to incorporating captions into sports broadcasts. For instance, captions enhance accessibility for diverse audiences: automated captions ensure that those who are deaf or hard of hearing can follow the game in real time. Additionally, captions can be automatically translated, allowing viewers from different linguistic backgrounds to enjoy the content.

Furthermore, captions enhance viewer engagement. Automated captions of live commentary can highlight key moments, helping viewers engage more deeply with the game and assist with video retrieval. Finally, in games like FIFA and NBA 2K, real-time captions of commentary enhance the realism of the experience, making gameplay more immersive.

To address this issue, we will employ video captioning technology. Video captioning refers to the automatic generation of descriptive text that aligns with the content of a video, typically in real time. This process involves creating sentences or captions that describe what is happening in the video, such as actions, events, or conversations. Video captioning for sports commentary uses advanced multimodal models to analyze both visual and textual elements of sports videos, generating descriptive captions in real time. By processing video feeds and leveraging natural language processing (NLP), the system can produce captions that explain vital events—such as goals, fouls, or player actions—as they happen. Pre-trained on large soccer video datasets, these models recognize patterns and sport-specific terminology, ensuring accurate and contextually relevant captions. Whether for soccer, basketball, or other sports, video captioning ensures that fast-paced sports content is accessible and understandable to a broad audience.

Although many existing video captioning models are pretty effective, no specific model is tailored to soccer video data. Ultimately, I chose to use BLIP-2 for two main reasons. First, traditional video captioning models typically require a video encoder to extract video features. Extracting features from soccer videos, usually 90 minutes long, demands significant computational resources beyond my capacity. BLIP-2, on the other hand, is an image captioning model. An image encoder requires significantly fewer computational resources than a video encoder, and a video can essentially be viewed as an ordered sequence of images. The second reason is that teams using BLIP (the predecessor of BLIP-2) on similar datasets have achieved excellent results, which led me to select BLIP-2 for this video captioning task.

Therefore, in this paper, I raise several key questions:

1. Can an image-to-caption model like BLIP-2 handle video captioning tasks?
2. How significant is the difference between video input (multiple frames with temporal information) and image input (a single frame)?

3. What are the most critical factors influencing the results?

The structure of this paper is as follows: I first introduce related work relevant to this task. Then, in the preliminary section, I explain the technical aspects of the various modules used in the model. Next, I describe the details of the model, followed by an introduction to the dataset and the metrics used to evaluate the experimental results. From an implementation perspective, I also outline the workflow of this experiment and its innovative points. The experiment process and results are detailed in the experiments section. Finally, I discuss the findings, propose future directions for research, and conclude the thesis.

Chapter 3 Related Work

3.1 Video Captioning

Image and video captioning is the task of generating textual descriptions of visual content. Early video captioning schemes employed template-based approaches, extracting semantic concepts like subject, object, and verb using visual classifiers and generating captions through a language model applied to predefined sentence templates. Rohrbach et al.Rohrbach et al. (2013) introduced a Conditional Random Field (CRF) to learn relationships between different video components, generating rich semantic representations. However, template-based methods are inflexible, struggle with large vocabularies, and do not generalize well to unseen dataKhurana and Deshpande (2021). Additionally, they cannot learn concepts end-to-end from low-level image features.

Deep learning-based VC often uses encoder-decoder frameworks Gao et al. (2020). In these models, the encoder extracts visual features via methods such as 2D-CNNs Venugopalan et al. (2014), which can be further processed using techniques like Mean Pooling (MP), Temporal Encoding (TE), or Semantic Attribute Learning (SEM) Aafaq et al. (2021). Typically, CNNs are used in encoders for video, while RNNs handle language generation in the decoding phase Olivastri et al. (2019). Modern methods often train encoder and decoder parts separately, improving performance.

Using Three Dimensional Convolutional Neural Networks (3D-CNN) in encoder-decoder frameworks captures spatio-temporal information more effectively than applying 2D-CNNs to individual frames. Recurrent Neural Networks (RNN) or Dual-Stream RNNs (DS-RNN) can be used in encoding. Language decoders utilize pre-trained word embeddings like Word2Vec, GloVe, and fastText to represent semantically similar words with similar vectors, and RNNs generate the output sequence due to the sequential nature of languageGoyal et al. (2018).

After the widespread application of Transformer, more new models have emerged. CLIP4CaptionTang et al. (2021) adopt a Transformer structured decoder network to effectively learn the long-range visual and language dependency, unlike models using LSTM or GRU as the sentence decoder.

Hierarchical Representation Network with Auxiliary Tasks (HRNAT)Gao et al. (2022) leverages hierarchical video representations and auxiliary tasks to improve performance in video captioning and question answering. The hierarchical representation of videos is guided by the three-level representation of languages and auxiliary tasks to generate syntax consistent with the ground-truth description.

The work addresses the limitations of large-scale video-text datasets by leveraging advancements in image captioning to pre-train video models without parallel video-text data. Using an OPT language model and a TimeSformer visual backbone, several video captioning models are pre-trained and fine-tuned on video captioning datasets. This approach Lialin et al. (2023) overcomes issues with existing mining techniques, such as low-quality captions and data scarcity, showing improved performance in video captioning.

To reduce the semantic gap between vision and language, Shi et al. (2023) proposes a video-text alignment module with a retrieval unit and an alignment unit for video captioning. The retrieval unit retrieves sentences as semantic anchors between visual scenes and language descriptions. In contrast, the alignment unit aligns the video with the retrieved sentences to learn video-text-aligned representations.

3.2 End-to-end Vision-Language Pre-training

Vision-language pre-training (VLP) is an area of research focused on developing models that can understand and generate visual and textual information. These models are pre-trained on large datasets containing both images and text, such as image-caption pairs, to learn joint representations of visual and linguistic data. Vision-language pre-training (VLP) research has experienced rapid advancements in recent years.

CLIPRadford et al. (2021) is dual-encoder architecture because it uses two separate encoders—one for images and one for text—to generate embeddings in a shared representation space. LXMERTTan and Bansal (2019) and ALBEFLi et al. (2021) employ a multi-stage process to fuse visual and textual information at the same time effectively. VL-T5Cho et al. (2021), PaLIChen et al. (2022) and SimVLMWang et al. (2021c) are called encoder-decoder architecture, which consists of two distinct components: an encoder and a decoder. The encoder processes the input data and converts it into a latent representation, while the decoder takes this latent representation and generates the output. While BLIPLi et al. (2022) and BeiTWang et al. (2022) as unified transformer architecture integrate vision and language processing within a single transformer framework, treating both modalities in a unified manner.

3.3 Modular Vision-Language Pre-training

Modular Vision-Language Pre-training represents a flexible, efficient, and powerful approach to building vision-language models. Composing the task into specialized modules and leveraging pre-trained components addresses the complexities of multimodal learning while optimizing computational resources and improving performance across various tasks. Some models like UNITERChen et al. (2019), OscarLi et al. (2020) and VinVLZhang et al. (2021) freeze the image encoder. Some methods freeze the language model to use the knowledge from LLMs for vision-to-language generation tasks, like MAPLMañas et al. (2023) and Img2LLMGuo et al. (2022).

Chapter 4 Preliminaries

4.1 Overview

In this thesis, the primary model I am using is BLIP-2. However, briefly introducing all the techniques and models is necessary in a multimodal architecture. This helps us understand the development of the groundbreaking Transformer family and the alignment of different modalities.

From the figure 4.1, I start from **attention** because attention is the primary and crucial mechanism in the Transformer. **Transformer** has revolutionized the NLP field by ingeniously applying non-sequential models to handle sequence data. It achieves parallel sequential data processing through self-attention mechanisms, significantly enhancing computational efficiency and performance.

Based on Transformer, two LLM models have experienced rapid development. **Pre-training of Deep Bidirectional Transformers(BERT)** is utilized for pretraining deep bidirectional representations, while **Open Pre-trained Transformer(OPT)** models are used to generate text. **Query-Transformer(Q-Former)** is initialized with the pre-trained weights of BERT to accelerate the training of downstream tasks and improve performance.

Vision Transformer(ViT) has successfully transferred the Transformer from the NLP field to the CV field, significantly influencing the workflow of computer vision tasks.

Align before Fuse(ALBEF) is a multimodal that integrates vision and language models by self-supervised and contrastive learning. And **Bootstrapping Language-Image Pre-training(BLIP)** was proposed to solve the problem that the model cannot handle the generation and retrieval tasks simultaneously by introducing a multimodal mixture of encoder and decoder. Due to the end-to-end training of large-scale models, the cost of training becomes prohibitive. **Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models(BLIP-2)** is improved from BLIP by freezing the training-costly ViT and LLM model and introducing the Q-Former.

4.2 Attention

In deep learning, RNN (Recurrent Neural Network) first handles sequential data. Unlike traditional isolated feedforward neural networks, it keeps a memory of previous input. This

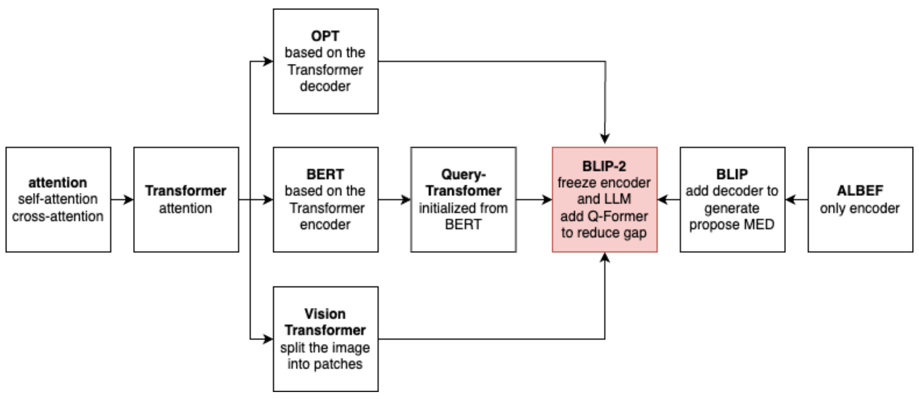


Figure 4.1 From attention to BLIP-2

mechanism allows RNNs to capture correlation and dependencies in the sequential data, making them well-suited for several tasks.

However, LSTM (Long Short-Term Memory) is proposed due to the limitation of capturing long-term patterns. It introduced the gate mechanisms, which include the input modulation gate, input gate, forget gate, and output gate. This enables LSTM to remember or forget the information by controlling the status of the gate. Therefore, LSTM performs better on tasks involving long sequences than RNN does.

Attention Vaswani et al. (2017) is invented to focus on the specific part of the sequence rather than the entire sequence, which improves the efficiency by dynamic selecting. Besides, attention enables models to process the input in parallel, while the parallel computation capability limits LSTM because its output highly depends on the input. From the perspective of interpretability, the attention mechanism outperforms LSTM significantly.

Attention is used to inform the model which specific part should be focused on, and it is a crucial concept of Transformer. Three trainable vectors will be introduced here; they query Query matrix(W^Q), Key matrix(W^K), and Value matrix(W^V).

From the word perspective, there will be:

- the word itself n_1
- the embedding x_1 (obtained by tokenization)
- the query q_1 (obtained by multiplying x_1 by the W^Q)
- the key k_1 (obtained by multiplying x_1 by the W^K)
- the value v_1 (obtained by multiplying x_1 by the W^V)

Similarly, the other words in the sentence have their own x, q, k, and v.

Introducing so many vectors here aims to calculate the scores of different inputs and find the semantic correlation among the sentences.

- the attention score $\alpha_{1,n}$ is calculated firstly by the dot product of the current q_1 and k_n
- the score will be passed to a softmax layer to get the attention score
- multiply each value by the attention score and sum up to obtain the output of the self-attention layer of the position

From the matrix perspective, we can compute the attention function on a set of queries simultaneously. Attention could be calculated by obtaining Q,K, and V.

$$Q = XW_Q \tag{4.1}$$

$$K = XW_K \tag{4.2}$$

$$V = XW_V \tag{4.3}$$

where X is the input matrix.

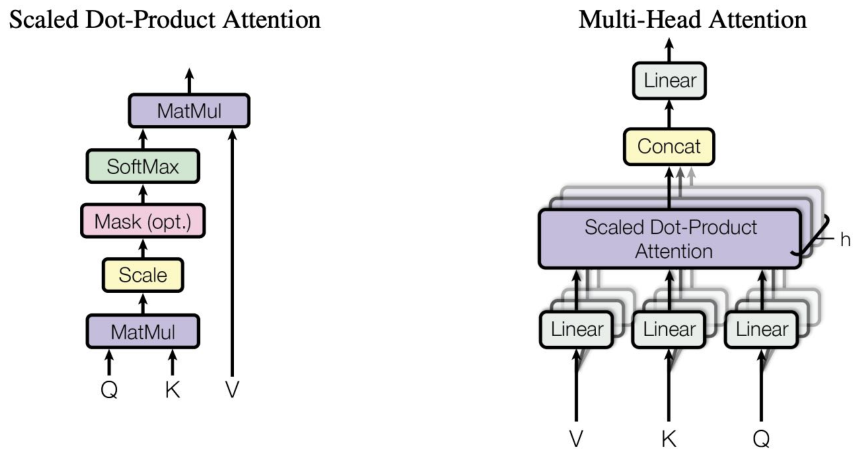


Figure 4.2 (Left)Scaled Dot-Product Attention(Right)Multi-Head Attention Vaswani et al. (2017)

And then, Attention A can be obtained by

$$A = \text{softmax}(QK^\top) \tag{4.4}$$

A scaling operation is added by adding the division of $\sqrt{d_k}$ to ensure numerical stability and uniformity in attention computations.

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \tag{4.5}$$

Finally, we have to multiply A with V to get the output O

$$O = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{4.6}$$

This is how self-attention works, as shown in figure 4.2 left, considering the importance or relevance of the input to the context and returning an output of the same size as the input, which could be passed to the next layer.

However, in the paper, multi-head attention is applied, as shown in 4.2 right. With it, there is one Q/K/V matrix and multiple sets. They aim to attend to information from different representation subspaces at other positions.

4.3 Transformer

The Transformer model was introduced by Vaswani in 2017Vaswani et al. (2017), characterized by its parallelization capabilities, encoder-decoder architecture, and variants of attention mechanisms, making it not only highly versatile for natural language processing tasks but also can be effectively adapted and transferred to other modalities.

The Transformer model consists of the encoding and decoding components and their connection.

The encoding component is composed of a stack of encoders; each encoder could be broken down into a self-attention layer and a feed-forward layer(an add&norm layer followed by them). The decoding component has a similar structure, except the decoder

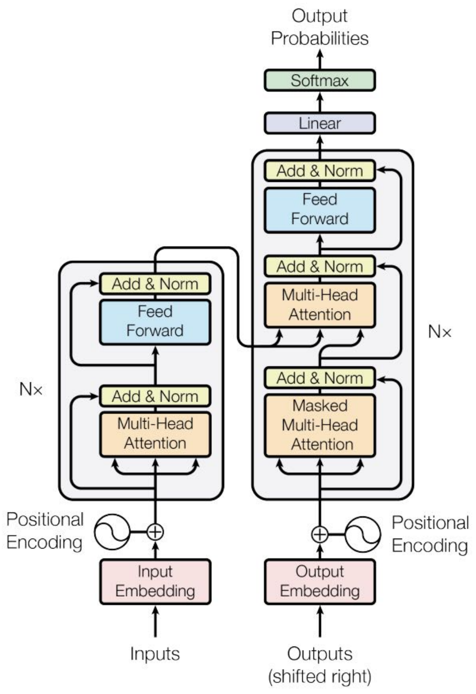


Figure 4.3 The Transformer - model architectureVaswani et al. (2017)

needs a bridge to pass the information to the encoder.

The Transformer groundbreaking architecture tackles NLP tasks by introducing self-attention mechanisms. This idea has been validated and proven effective in other modalities as well.

4.4 Bidirectional Encoder Representations Transformer(BERT)

BERT (Bidirectional Encoder Representations from Transformers) considers context from both directions, which makes it fundamentally different from traditional unidirectional language models. It is based on the Transformer architecture, explicitly utilizing the encoder part, which employs self-attention mechanisms to capture relationships between all words in a sentence, regardless of distance. This bidirectional nature allows BERT to generate deep, context-aware representations of words, significantly enhancing its ability to understand and process natural language.

BERT has gained significant popularity in pre-training for natural language processing. By pre-training on large corpora like Wikipedia and BookCorpus, BERT learns general language representations that can be fine-tuned for various downstream tasks such as question answering, sentiment analysis, and named entity recognition with minimal additional training. For instance, in the BLIP-2 model, the Q-Former component is initialized from BERT, leveraging its robust pre-trained language representations to handle specific tasks effectively. This approach exemplifies how BERT's pre-trained models are a strong foundation for specialized applications, demonstrating its versatility and impact in NLP.

4.5 Vision Transformer (ViT)

One of the most significant contributions of Transformers is their revolutionary impact on the computer vision (CV) field, which has long been dominated by Convolutional Neural

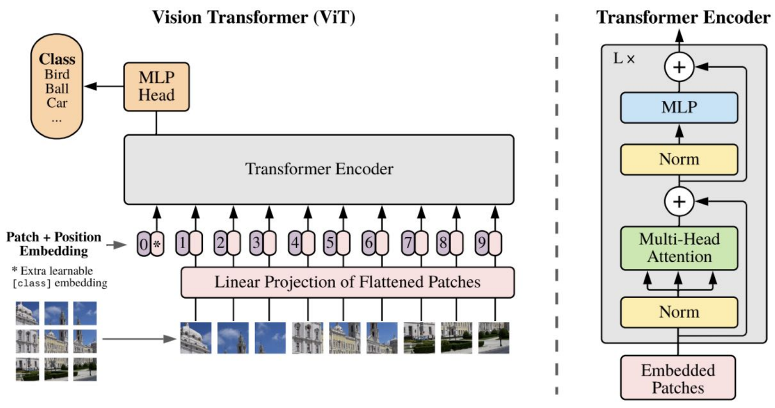


Figure 4.4 The ViT - model architectureDosovitskiy et al. (2020)

Networks (CNNs). Traditionally, CNNs were the go-to architecture for image-processing tasks due to their ability to capture spatial hierarchies through convolutional layers. However, Transformers, initially designed for natural language processing, have also performed remarkably in CV tasks. By leveraging self-attention mechanisms, Transformers can be applied directly to images without needing CNNs.

Due to the sheer number of pixels involved, it's impractical to use individual pixels as the last unit to adapt an image for input into a Tran that will be produced for the model. For instance, an RGB image of size 224x224 contains 50,176 pixels (224 * 224), not accounting for the three color channels. This number far exceeds the typical hidden layer size 768 in models like ViT-Base, making direct pixel input infeasible.

To address this, images are divided into smaller patches, and each patch is linearly embedded to form a sequence suitable for the Transformer. For example, if the patch size is 16x16, the image is divided into 196 patches (224 * 224 / 16 * 16). Each RGB patch of size 16x16x3 is then flattened into a 768-dimensional vector (16 * 16 * 3), resulting in a feature representation of the image as a sequence of 196 embeddings, each of size 768.

These patch embeddings are then fed into the Transformer model, preceded by a linear projection layer. Additionally, position embeddings are added to these patch embeddings, akin to the position embeddings used in natural language Transformers, to retain spatial information. Positional information is crucial for images, just as text, to maintain the relative positions of patches.

Similar to BERT, a learnable classification token [CLS] is introduced for classification tasks. This [CLS] token is appended to the beginning of the patch sequence and learns to aggregate information from all patches through attention mechanisms. The output corresponding to the [CLS] token is then used for final classification, capturing the essential information from the entire image.

4.6 Contrastive learning and CLIP

Contrastive learning is an unsupervised learning technique aimed at learning compelling features from data without needing labeled datasets. The core idea is to pull similar data points closer in the embedding space while pushing dissimilar ones apart. This is achieved through a loss function that penalizes the model when similar items are far apart, or dissimilar items are close together. Contrastive learning has proven useful in various fields, such as computer vision, natural language processing, and multimodal learning, as it captures the underlying structure of the data for tasks like clustering, retrieval, and classification.

A notable application of contrastive learning is the Contrastive Language-Image Pre-training (CLIP)Radford et al. (2021) model developed by OpenAI. CLIP leverages text and image data to learn a unified multimodal embedding space, effectively aligning visual and textual information. OpenAI collected 400 million image-text pairs from the internet to pre-train CLIP. The model uses a visual encoder to process images and a text encoder to process text, transforming each input into embeddings in a shared multimodal space. The training objective maximizes the similarity between matching image-text pairs and minimizes it for non-matching pairs using a contrastive loss function.

4.7 Align Before Fuse (ALBEF)

ALBEF (Align Before Fuse) is the foundational version of BLIP-2, and understanding its concepts is essential before delving into BLIP-2. In my learning field, the goal is to align and jointly model visual features and word tokens. Before ALBEF, models typically aligned visual and textual features only after fusion, which made it difficult for the multimodal encoder to learn image-text interactions due to their initial misalignment effectively. ALBEF addresses this issue by proposing an architecture that aligns features before fusion, significantly improving the subsequent fusion process.

The ALBEF model comprises three encoders: a 12-layer ViT-Base image encoder, a 6-layer BERT-Base text encoder, and a 6-layer BERT-Base multimodal encoder. When an image and text are input into their respective encoders, their embeddings are obtained. The [CLS] tokens from both embeddings are extracted to calculate Image-Text Contrastive (ITC) loss. The text embeddings are then fed into the multimodal encoder, fusing with the image embeddings through cross-attention mechanisms at each layer of the multimodal encoder. This setup allows for calculating Image-Text Matching (ITM) and Masked Language Modeling (MLM) losses, further enhancing the model's ability to learn from multimodal data.

ALBEF's innovative approach of aligning features before fusion enables more effective learning of image-text interactions, setting a solid foundation for subsequent models like BLIP-2. By pre-aligning visual and textual features, ALBEF enhances the capability of multimodal encoders to model and understand complex relationships between images and text jointly.

4.8 Bootstrap Language-Image Pre-training(BLIP)

Based on ALBEF, a new model BLIPLi et al. (2022) is proposed. BLIP improves on the concepts introduced by ALBEF (Align Before Fuse) by refining the alignment of visual and textual features before fusing them, ensuring more coherent and contextually relevant embeddings. Its scalable architecture and versatility make it suitable for various applications, from image captioning to visual question answering.

Unlike ALBEF, BLIP uses a transformer encoder and a decoder, which is called a multimodal mixture of encoder and decoder (MED). A MED can operate as an unimodal encoder or an image-grounded text decoder. BLIP's architecture includes more cross-attention mechanisms and other improvements that enhance the alignment and fusion

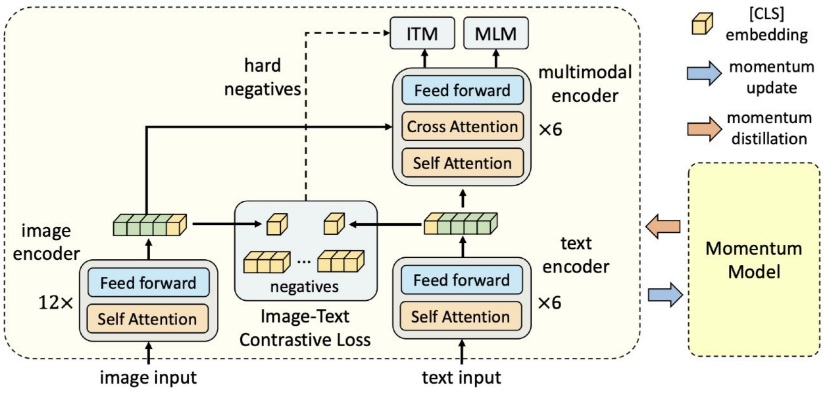


Figure 4.5 Illustration of ALBEFLi et al. (2021)

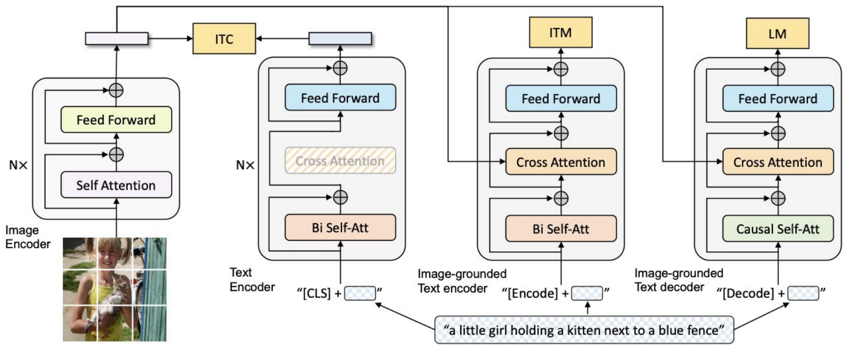


Figure 4.6 Illustration of BLIPLi et al. (2022)

process. The model is still trained with three objectives similar to ALBEF: image-text contrastive, image-text match, and image-text conditioned language modeling.

4.9 Pretraining & finetuning

Pre-training serves as a foundational step, particularly in the field of natural language processing (NLP). It involves training a model on a large scale of data, such as Wikipedia, Instagram, and websites, to learn general language patterns and structures. This initial phase allows the model to develop a rich and deep understanding of text features, semantic relationships, and contextual representation in the data.

Once pre-training is complete, the model can be fine-tuned on specific downstream tasks, such as text classification, named entity recognition, or machine translation. Fine-tuning involves adjusting the model's parameters and learning objectives to optimize performance for the target task while retaining the knowledge acquired during pre-training.

Pre-training and fine-tuning extend the model's availability by providing more possibilities for transfer learning. Models like BERT (Bidirectional Encoder Representations from Transformers) Devlin et al. (2019), Text-to-Text Transfer Transformer(T5) Raffel et al. (2019) and GPT(Generative Pre-trained Transformer) Brown et al. (2020) have demonstrated massive success across a wide range of NLP tasks. The effectiveness of pre-training in capturing language representation and the flexibility of fine-tuning for specific downstream tasks make them applicable to all kinds of multimodal tasks.

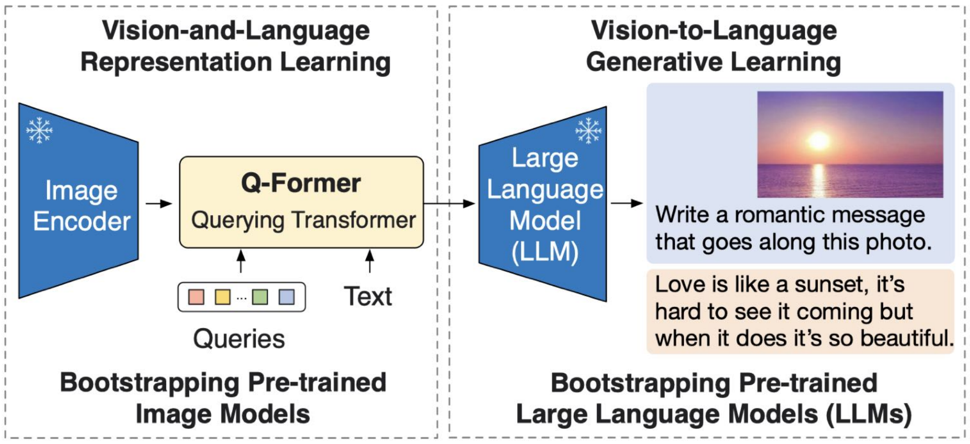

Vision-and-Language Representation Learning
Image Encoder
Q-Former
Querying Transformer
Queries
Text
Bootstrapping Pre-trained Image Models
Vision-to-Language Generative Learning
Large Language Model (LLM)
Write a romantic message that goes along this photo.
Love is like a sunset, it's hard to see it coming but when it does it's so beautiful.
Bootstrapping Pre-trained Large Language Models (LLMs)

Figure 4.7 Overview of BLIP-2's frameworkLi et al. (2023)

4.10 Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models(BLIP-2)

4.10.1 Overview

The model in this thesis is Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models (BLIP-2)Li et al. (2023), which is improved from ALBEF and BLIP.

BLIP-2 is a generic and efficient vision-language pre-training (VLP) strategy because of the frozen image encoders and large language models. The pre-training is divided into two stages. **The first stage is Vision-and-language representation learning**, by introducing a Qurey Transformer(Q-former) as a bridge to align the image embeddings and the corresponding text embeddings. A frozen image encoder obtains the image embeddings, while the text transformer obtains the text embeddings. The queries interact with each other through self-attention layers and interact with frozen image features through cross-attention layers. **The second stage is Vision-to-Language Generative Learning**, which bootstraps from frozen large language models.

As for the reasons for choosing this model, BLIP-2 achieves state-of-the-art performance on various vision-language tasks despite having significantly fewer trainable parameters than existing methods. Besides, the unimodal pre-trained models remain frozen during the pre-training, which reduces computation costs significantly and meets my hardware requirement.

4.10.2 Vision-Language Representation Learning

4.10.2.1 Image Encoder

The aim of an image encoder here is to get the visual embeddings of the input image.

The image encoder in the experiment is ViT-g/14 from EVA-CLIP. It is a variant of the Vision Transformer (ViT) used in the EVA-CLIP (Efficient Vision-Alignment)Fang et al. (2023) model, which is designed for visual representation learning. *g* represents 'giant,' which implies the model size is larger than the standard models. *14* refers to the patch size. *CLIP* is a model that learns to associate images and text by training on a large dataset of image-text pairs through contrastive learning. EVA-CLIP aims to make this

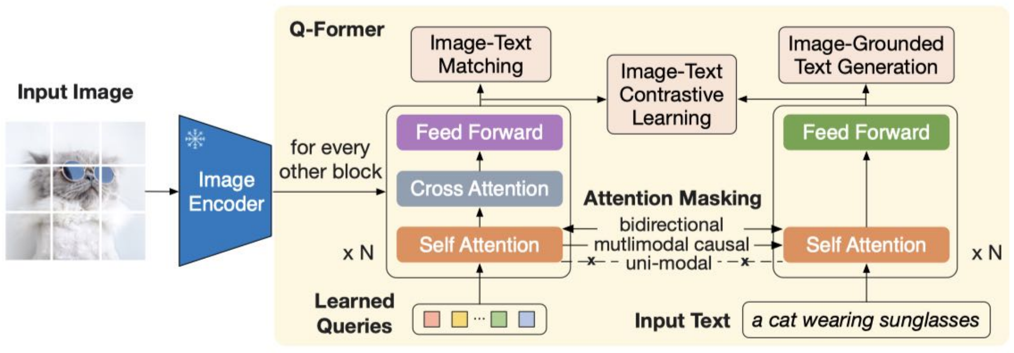


Figure 4.8 BLIP-2's first-stage vision-language representation learning Li et al. (2023)

process more efficient and effective, leveraging the Vision Transformer architecture.

In the model, the last layer of the ViT is removed to get slightly better performance.

4.10.2.2 Q-Former

Q-Former is essentially a trainable module consisting of vectors. It bridges the gap between the image encoder and the LLM. From figure 4.8, two sub-transformer blocks share the same self-attention layers. The left is an image transformer consisting of self-attention, cross-attention, and a feed-forward block, which interacts with the frozen image encoder. The right one is the text transformer, which consists of a self-attention and a feed-forward layer, which works as the text encoder and decoder.

We initialize learned queries as the input to the image transformer, and the queries interact with each other through self-attention layers and with frozen image features through cross-attention layers. At the same time, these queries could interact with the text content via the self-attention layers.

From the implementation perspective, since the query and text share a self-attention layer in the transformer, we first concatenate the query and text into a long sequence [query, text] and input it into the self-attention layer. In this layer, we need to control the interaction of attention based on different tasks, which I will detail below. After passing through the self-attention layer, the query goes through a cross-attention layer that interacts with the image embeddings. A feed-forward layer then processes the query. The text, on the other hand, passes through its feed-forward layer. After one transformer layer, we still obtain an updated [query, text], which can be fed into the next transformer block.

The Q-former is initialized with the pre-trained weights of BERTbase, one of the most popular transformer applications, but with the cross-attention layers randomly initialized because there are no precedents working on image embeddings and text together by BERT.

In the experiment, 32 queries with a dimension of 768 are used. The output size of queries is (32×768) after the transformer blocks, which is significantly smaller than the size of frozen image features (e.g. 257×1024 for ViT-L/14, 257×768 for ViT-g/14). Therefore, this design helps the model extract the most relevant relation between the visual information and the text content.

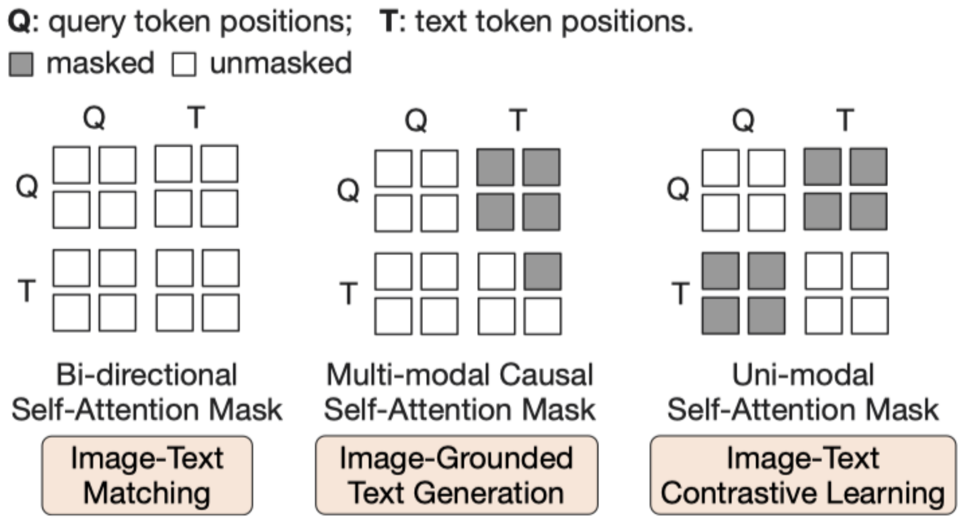

Q: query token positions; T: text token positions.
masked
unmasked
Q
T
Bi-directional Self-Attention Mask
Image-Text Matching
Multi-modal Causal Self-Attention Mask
Image-Grounded Text Generation
Uni-modal Self-Attention Mask
Image-Text Contrastive Learning

Figure 4.9 Self-attention masking strategyLi et al. (2023)

4.10.2.3 Attention Masking Strategy

As we mentioned before, BLIP-2 is improved from BLIP. Inspired by BLIP, three pre-training objectives are used to train the Q-Former for better representation. There are three objectives: Image-Text Contrastive Learning (ITC), Image-grounded Text Generation (ITG), and Image-Text Matching (ITM). Different objects must employ different attention masking strategies to control the interaction between query and text in the self-attention layer.

The final loss=loss_itc + loss_itm + loss_lm.

4.10.2.4 Image-Text Contrastive Learning (ITC)

ITCLi et al. (2022); Radford et al. (2021); Yao et al. (2021) learns to align the image and text representations as much as possible by contrasting the image-text similarity of the cheerful pair and negative pairs. We define the output query representation Z from the image transformer, and t is the output embedding of [CLS] token of the text embeddings. Since Q-Former is based on BERT, the [CLS] token represents the input text.

Unlike BLIP in BLIP, after the visual and text encoder, the ITC could be computed by two [CLS] tokens. However, in BLIP-2, Z has 32 queries; we need to calculate the pairwise similarity between each query and t and select the highest one as the image-text similarity.

Since we are pursuing contrastive learning, we aim to prevent interactions between images and text before computing the loss. Returning to the earlier issue, potential information leakage can occur at the self-attention layer. Therefore, we need to modify the self-attention mask. For instance, as shown in the 4.9 (Right), queries should only interact with queries, and text should only interact with text. The interaction between query and text is masked.

The ITC loss computes similarities between image and text features and calculates a contrastive loss to train the multimodal model. It gathers features across GPUs using concat_all_gather, computes **image-to-text** and **text-to-image** similarities via matrix multiplication and aggregation with *torch.max* to obtain the highest similarity value for each pair and prepare target labels based on GPU rank and batch size. Finally, it calculates the contrastive loss using cross-entropy for general training or a custom loss for COCO retrieval fine-tuning, encouraging the model to align matching image-text pairs closely and separate non-matching pairs.

4.10.2.5 Image-grounded Text Generation (ITG)

ITGLi et al. (2022); Wang et al. (2022); Yu et al. (2022) aims to train the Q-Former to generate texts. The queries can attend to each other exclusively, while the text tokens can attend to all questions and their preceding text tokens. From the 4.9 (Middle), we can see the current text and query can't interact with the future text, so there is no mask in queries.

The LM loss is calculated by first setting up the decoder input IDs and labels, with padding tokens replaced by -100 to exclude them from the loss computation. It then constructs the attention mask by combining query and text tokens' attention masks. The prepared inputs, including decoder_input_ids, the complete attention_mask, and labels, are passed to the Qformer model to generate the output. Finally, the language modeling loss is extracted from the model's output. This process enables the model to generate text and optimize based on the provided labels while integrating image features.

4.10.2.6 Image-Text Matching (ITM)

ITMLi et al. (2022); Wang et al. (2021b) is a binary classification task to predict whether an image-text pair is positive (matched) or negative (unmatched). In BLIP, the ITM could be computed by image [CLS] and text [encode] tokens. The 32 queries must go through a projection layer and get logits. Logits are then used to calculate ITM by cross-entropy. In this phase, the interaction between query and text is needed; we want the model to learn how to classify. No masked part is visible from the 4.9 (Left).

Calculating ITM loss begins by gathering input IDs and attention masks for text tokens and image embeddings across all GPUs. Then, it masks similarities for matching pairs by setting their similarity scores to a meager value, ensuring that self-matching within the same batch is avoided. Next, it computes weights for text-to-image and image-to-text similarities using softmax. The code selects negative samples for each text and image using weighted sampling, stacking them into tensors. It then constructs new input tensors, combining positive and negative text and image embedding samples. Finally, it uses *self. Qformer.bert* computes the output for image-text matching extra, ts the final hidden state, passes it through itm_head to get matching scores, and calculates the ITM loss using cross-entropy. This approach enhances the model's ability to distinguish between matching and non-matching image-text pairs.

4.10.3 Vision-Language Generative Learning

From figure 4.10, we can see the top figure and the bottom one have quite similar architecture. The learned queries go through the Q-Former, and a fully connected layer is used to get the projected queries. The only difference is that if we use a decoder-based LLM, the projected queries are directly prepended to the LLM decoder. Otherwise, they must first go through an LLM encoder if we use an encoder-decoder-based LLM. The experiment only focuses on a decoder-based LLM named the OPT model. Open Pretrained Transformers (OPT) is based on the powerful GPT-3 model but is designed to address the

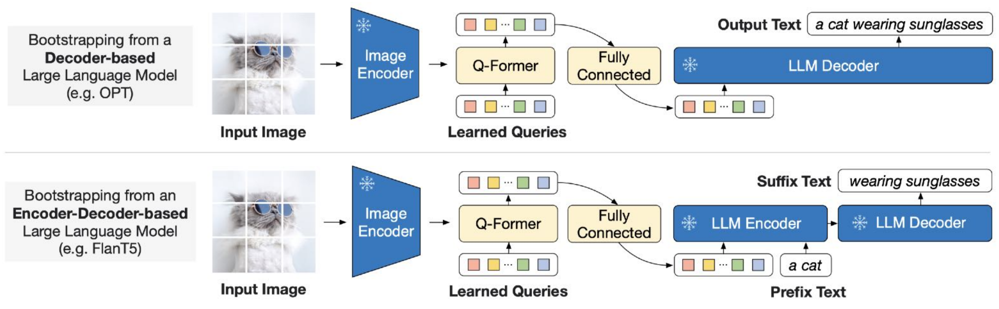


Figure 4.10 BLIP-2’s second-stage vision-to-language generative pre-training. Bootstrapping a decoder-based LLM (e.g. OPT)(Top). Bootstrapping an encoder-decoder-based LLM (e.g. FlanT5) (Bottom)

high computational cost associated with GPT-3.

In the second stage, the LLM is pertained by the language modeling loss, where it needs to generate the caption based on the queries from the Q-Former. Details will be discussed in the next chapter.

Chapter 5 Implementation

5.1 Workflow

Since this experiment is based on the BLIP-2 model framework, an image-to-caption framework, the model's input should be an image. After being encoded by the visual encoder, visual features are obtained. In the first stage, these visual features are aligned with text features, and the unique queries in BLIP-2 learn how to align these two modalities. Then, the learned queries serve as the input for the next stage. Passing through a Fully Connected layer, these queries are transformed into inputs suitable for the LLM model, which then generates the corresponding output.

However, BLIP-2 cannot perfectly adapt to video input for the video captioning task. Therefore, I first preprocessed the video files. Simply put, I selected keyframes from the video and averaged these keyframes to serve as the model's input.

The second adjustment is that I did not use the visual encoder in real time during the experiment due to limited computational resources. This means that in the experiment, image-to-caption was not end-to-end. I pre-extracted the features of the keyframes. Since the visual encoder is frozen, the features of each image are fixed. This approach saves computational resources because extracting video features often consumes much computational power. Additionally, it facilitates the average pooling operation.

Furthermore, LSTM was also attempted in subsequent experiments. LSTM replaced the LLM model. Although LSTM has a more straightforward structure, it requires specific design when integrating visual and text features. Specifically, LSTM excels at handling sequential data, effectively capturing long-term and short-term dependencies. By predicting the next token, LSTM can recursively generate coherent text. Specifically, the text input is fed into the LSTM, while the visual features are input as parameters into the LSTM, where these two representations are fused.

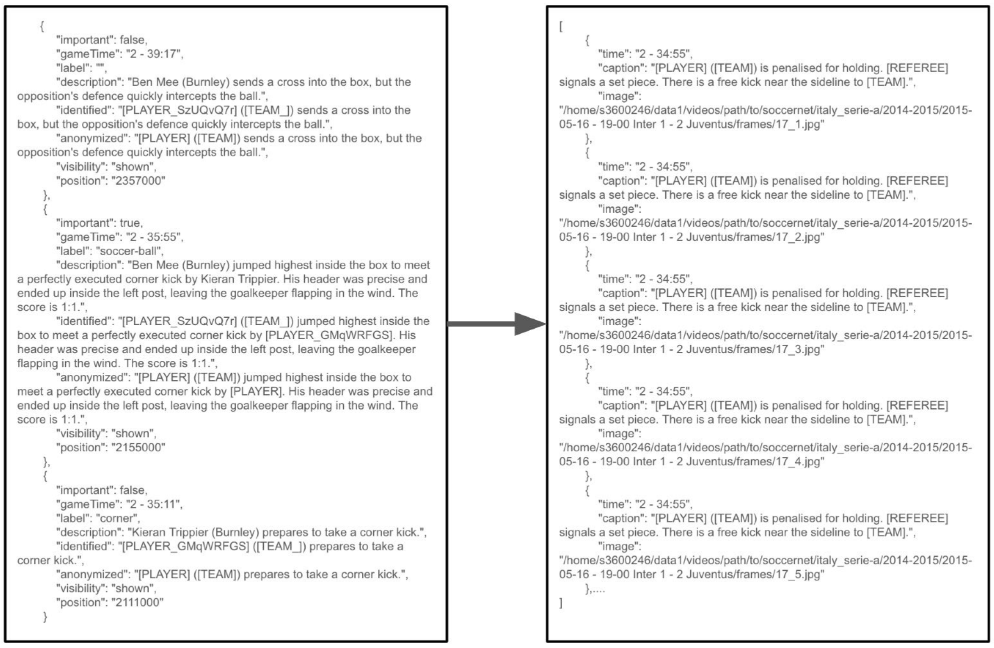
5.2 Data Preprocessing

First, I read the JSON file containing annotations that record the timestamp and the corresponding caption. I set the timestep, fps, and interval of frame parameters and use the methods of *cv2.VideoCapture* to get the corresponding timestamp point and the nearby frames, which represent a segment of the video input. After that, I align each of these frame inputs with the caption. And save all the pairs to the same JSON file. Then, I divided them into training, validation, and test sets.

In the blip-2 model, the authors provide the interface for fine-tuning the coco dataset. Here, I changed the dataset to the format that fits the model.

5.3 Dataset Building

When building the dataset, I made a few adjustments because of inconsistencies in the data. First, the training and eval sets (including the test sets) are initialized with different classes in the source code. For example, the image is read first for the training set, and the image and the corresponding caption go through the visual and text processors, respectively. The



```
    {
        "important": false,
        "gameTime": "2 - 39:17",
        "label": "",
        "description": "Ben Mee (Burnley) sends a cross into the box, but the
opposition's defence quickly intercepts the ball.",
        "identified": "[PLAYER_SzUQvQ7r] ([TEAM_]) sends a cross into the
box, but the opposition's defence quickly intercepts the ball.",
        "anonymized": "[PLAYER] ([TEAM]) sends a cross into the box, but the
opposition's defence quickly intercepts the ball.",
        "visibility": "shown",
        "position": "2357000"
    },
    {
        "important": true,
        "gameTime": "2 - 35:55",
        "label": "soccer-ball",
        "description": "Ben Mee (Burnley) jumped highest inside the box to meet
a perfectly executed corner kick by Kieran Trippier. His header was precise and
ended up inside the left post, leaving the goalkeeper flapping in the wind. The
score is 1:1.",
        "identified": "[PLAYER_SzUQvQ7r] ([TEAM_]) jumped highest inside the
box to meet a perfectly executed corner kick by [PLAYER_GMqWRFGS]. His
header was precise and ended up inside the left post, leaving the goalkeeper
flapping in the wind. The score is 1:1.",
        "anonymized": "[PLAYER] ([TEAM]) jumped highest inside the box to
meet a perfectly executed corner kick by [PLAYER]. His header was precise and
ended up inside the left post, leaving the goalkeeper flapping in the wind. The
score is 1:1.",
        "visibility": "shown",
        "position": "2155000"
    },
    {
        "important": false,
        "gameTime": "2 - 35:11",
        "label": "corner",
        "description": "Kieran Trippier (Burnley) prepares to take a corner kick.",
        "identified": "[PLAYER_GMqWRFGS] ([TEAM_]) prepares to take a
corner kick.",
        "anonymized": "[PLAYER] ([TEAM]) prepares to take a corner kick.",
        "visibility": "shown",
        "position": "2111000"
    }
```

```
[
    {
        "time": "2 - 34:55",
        "caption": "[PLAYER] ([TEAM]) is penalised for holding. [REFEREE]
signals a set piece. There is a free kick near the sideline to [TEAM].",
        "image":
"/home/s3600246/data1/videos/path/to/soccernet/italy_serie-a/2014-2015/2015-
05-16 - 19-00 Inter 1 - 2 Juventus/frames/17_1.jpg"
    },
    {
        "time": "2 - 34:55",
        "caption": "[PLAYER] ([TEAM]) is penalised for holding. [REFEREE]
signals a set piece. There is a free kick near the sideline to [TEAM].",
        "image":
"/home/s3600246/data1/videos/path/to/soccernet/italy_serie-a/2014-2015/2015-
05-16 - 19-00 Inter 1 - 2 Juventus/frames/17_2.jpg"
    },
    {
        "time": "2 - 34:55",
        "caption": "[PLAYER] ([TEAM]) is penalised for holding. [REFEREE]
signals a set piece. There is a free kick near the sideline to [TEAM].",
        "image":
"/home/s3600246/data1/videos/path/to/soccernet/italy_serie-a/2014-2015/2015-
05-16 - 19-00 Inter 1 - 2 Juventus/frames/17_3.jpg"
    },
    {
        "time": "2 - 34:55",
        "caption": "[PLAYER] ([TEAM]) is penalised for holding. [REFEREE]
signals a set piece. There is a free kick near the sideline to [TEAM].",
        "image":
"/home/s3600246/data1/videos/path/to/soccernet/italy_serie-a/2014-2015/2015-
05-16 - 19-00 Inter 1 - 2 Juventus/frames/17_4.jpg"
    },
    {
        "time": "2 - 34:55",
        "caption": "[PLAYER] ([TEAM]) is penalised for holding. [REFEREE]
signals a set piece. There is a free kick near the sideline to [TEAM].",
        "image":
"/home/s3600246/data1/videos/path/to/soccernet/italy_serie-a/2014-2015/2015-
05-16 - 19-00 Inter 1 - 2 Juventus/frames/17_5.jpg"
    },....
]
```

Figure 5.1 (Left)The original data in JSON file. (Right)The data after preprocessing.

visual processor randomly crops the image and randomly flips the image horizontally. The text processor is to add a prompt to the caption, such as **a photo of**. However, I am using a series of images as input instead of a single image in my dataset. So, when I read, I must process each image on the list. Again, the method should return a list rather than only an image.

Another unique point is that the source code creates a ground truth file in this experiment that generates the value set. Since coco is already precomputed in the dataset, it is necessary to modify this for any custom dataset. The primary role of this section is to generate a dataset suitable for evaluation. Here, it utilizes the ID of the image, pairs the ID of the image with the caption, and uses the ID as a unique identifier to identify the image. As for the original COCO dataset, in the JSON file, they include the attribute image_id, so I need to add the image_id while creating the dataset manually.

In this step, there is a slight difference between the training set and the evaluation set in that in the evaluation set, there is no need for the caption to be passed in.

Another modification I made is creating the value and test dataset ground truth files. When initializing the classes for train and val/test, the training class will return the caption to add the prompt, while the other class does not. Therefore, we need a function to create the ground truth for the val and test dataset. It is achieved by loading the caption and caption together and save as a JSON file.

5.4 Average Pooling

Since I will apply video to the BLIP-2 model, there is a problem with how to process the video. I attempted to use multiple video frames to represent a single image. There are two ideas for merging various frames into one frame. One is early fusion, which averages the images at the pixel level. However, this approach can cause the image to lose its inherent information and may even make it unrecognizable for ViT. Therefore, I adopted late fusion inspired by the paperLei et al. (2022), where the operation involves merging the features

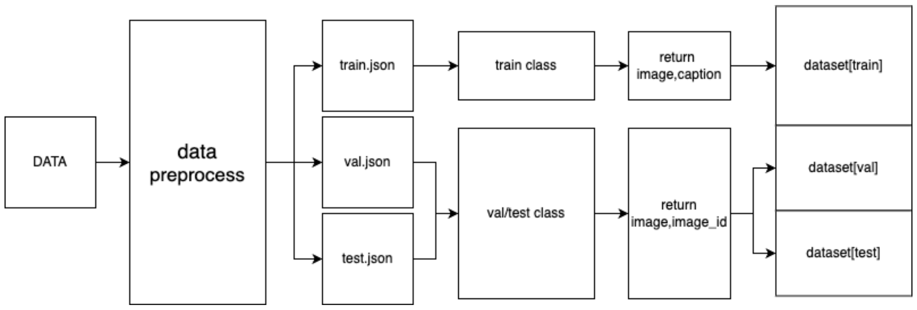


Figure 5.2 The workflow of building dataset

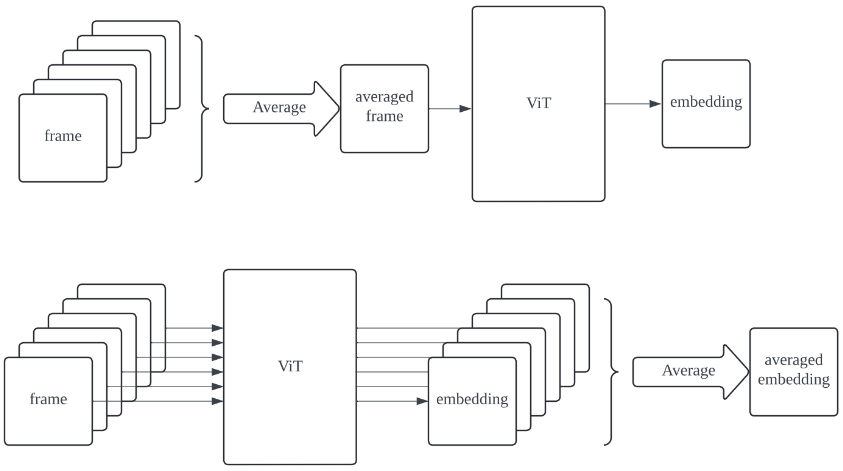


Figure 5.3 (Top)The early fusion (Bottom)The late fusion

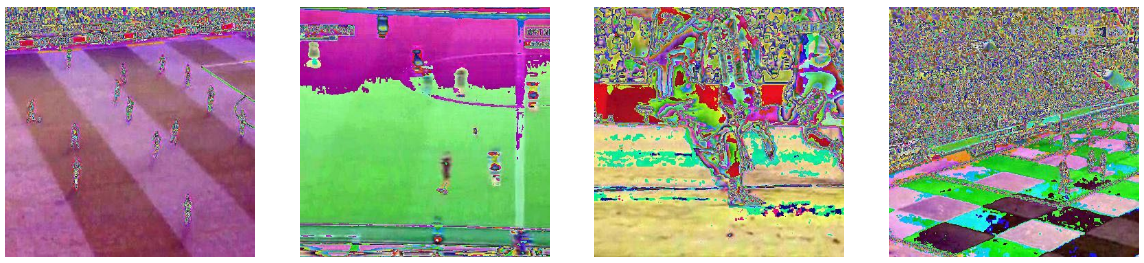


Figure 5.4 Four examples of early fusion

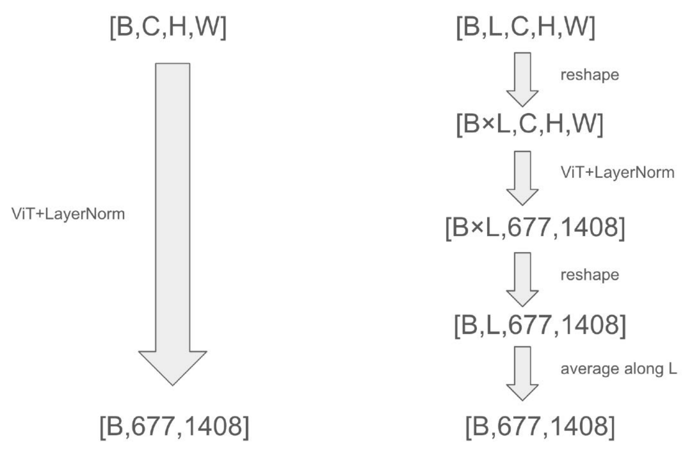


Figure 5.5 (Left)The original forward (Right)The modified forward

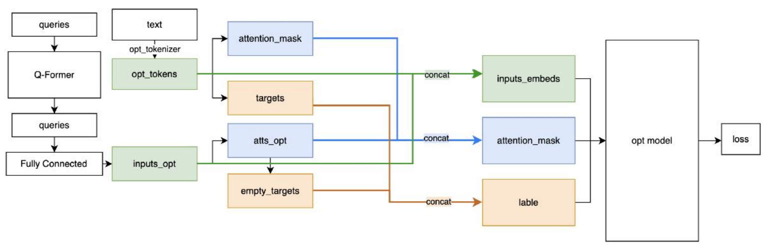


Figure 5.6 the workflow the the OPT model

extracted by ViT from the images. The averaged features are then used as the interaction object in BLIP-2's Q-Former.

This presents a significant dimensional challenge. Typically, the data inputted into the model is in the form of [B, C, H, W], where B denotes the batch size, C represents channels, and H and W stand for height and width, respectively. However, the current input shape is [B, L, C, H, W], where L signifies the number of video frames. Addressing this issue involves a series of dimensional transformations.

Firstly, [B, L, C, H, W] needs to be reshaped into [B $\times$ L, C, H, W]. Following this transformation, applying the visual encoder and layer normalization yields image embeddings of size [B $\times$ L, 677, 1408]. Subsequently, these embeddings must be reshaped back to [B, L, 677, 1408]. Finally, an average pooling operation along the L dimension computes the mean across frames, resulting in a final shape of [B, 677, 1408].

5.5 OPT model

The OPT (Open Pre-trained Transformer) model is a large-scale language model designed to perform various NLP tasks such as text generation, translation, summarization, and question-answering.

The OPT model is based on the Transformer architecture, similar to GPT-3 and BERT models. It consists of multiple layers of attention mechanisms and feedforward neural networks. It is a decoder-only architecture, which predicts the next token in a sequence based on the preceding tokens.

In the OPT model, the forward function has three inputs5.6: input embeddings, attention mask, and labels. In the BLIP-2 model, these three inputs are each composed of two parts concatenated together. The first part comes from the first stage of BLIP-2, which is the query second part from the OPT model.

Given that the OPT model is the key to our focus, understanding its inputs and outputs is crucial as it provides foundational knowledge for subsequent analysis. The OPT model loads a pre-trained OPT model, specifically OPTForCausalLM, for causal language modeling. OPTForCausalLM is a class from the HuggingFace Transformers library. OPT refers to a model developed by Meta, and OPTForCausalLM is designed for causal language modeling with the OPT model. Causal language modeling means the model generates each word based solely on the preceding words, not future words.

Inside the OPT model, the decoder produces outputs, including raw hidden states and possibly other information like attention weights or past vital values. These raw outputs from the decoder are then passed through the language modeling head to produce logits, representing the model's predictions for the next token. If labels are provided, the method calculates the loss using cross-entropy loss computed between the shifted logits and the shifted labels. Finally, the model returns a dictionary containing loss, logits, past_key_-values, hidden_states, and attentions. As shown in the diagram, we need the loss from this dictionary for model updates.

Next, we will closely examine the three critical inputs of the OPT model. They are input_embedding, attention_mask, and label.

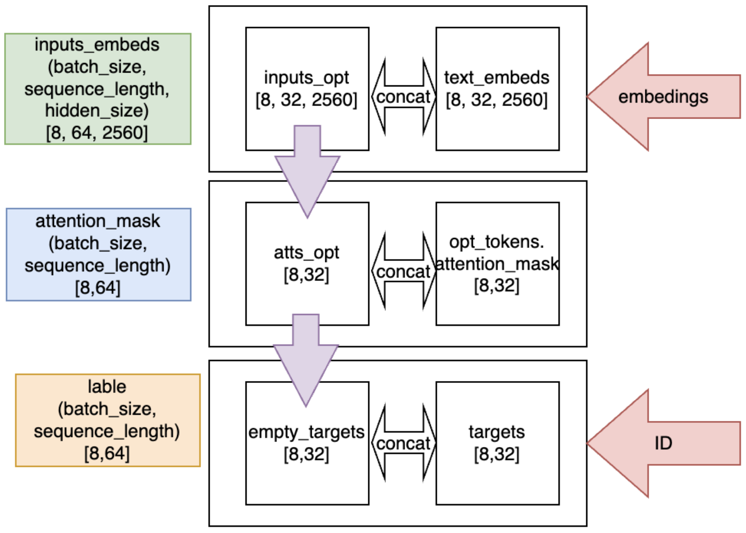


Figure 5.7 the shape of 3 inputs of the OPT model

5.5.1 input_embedding

Instead of passing input_ids, it is proper to pass an embedded representation directly. This is useful if you want more control over converting input_ids indices into associated vectors than the model's internal embedding lookup matrix.

The first part is 'inputs_op derived from the query output through a projection layer, which transforms the query into a suitable input for the OPT model. The shape of it is [8, 32, 2560].

The second part is the input embedding. It comes from 'opt_tokens, 'which results from tokenizing the text with the OPT tokenizer. The input embedding is obtained by embedding the ‘opt_tokens‘. Its shape is [8, 32, 2560].

5.5.2 attention_mask

A mask to avoid paying attention to padding token indices. Mask values are selected in [0, 1]: 1 for tokens that are not masked and 0 for tokens that are masked. An attention mask helps models selectively attend to relevant parts of the input sequence by marking positions to consider and positions to ignore. This is essential for handling padded sequences and controlling attention in various tasks.

The first part is the attention to the OPT model. It is a tensor of ones with a shape equal to all dimensions of ‘inputs_opt‘ except the last one. Its shape is [8, 32].

The second part is the attention mask from ‘opt_tokens‘. The output of a tokenizer in the Hugging Face ‘transformers‘ library typically includes several key components, depending on the options and the specific model/tokenizer being used. For example, ‘input_ids‘ and ‘attention_mask‘ are essential attributes. Therefore, the attention mask from ‘opt_tokens‘ can be directly obtained from the tokenizer function. Its shape is [8, 32].

5.5.3 label

Labels for computing the masked language modeling loss. Indice should either be in ‘[0, ..., vocab_size]‘ or -100. Tokens with indices set to ‘-100‘ are ignored (masked), and the loss is only computed for the tokens with labels in ‘[0, ..., vocab_size]‘.

The first part is ‘empty_targets‘. The pu pose of ‘empty_targets‘ creates a tensor of the same shape as the OPT attention, where all elements are set to ‘-100‘. The shape of it is [8, 32].

The second part is ‘targets.‘ ‘targets‘ is a new tensor where all padding token IDs in ‘opt_tokens.input_ids‘ are replaced with ‘-100‘. The shape of it is [8, 32].

Finally, input_embedding, attention_mask, and label all concatenate their two parts on the first dimension. Therefore, their shape are [8, 64, 2560], [8, 64], and [8, 64] respectively.

5.6 Potential Problems

The BLIP-2 is pre-trained as BLIP by 129M images, including COCOLin et al. (2014), Visual GenomeKrishna et al. (2016), CC3MSharma et al. (2018), CC12MChangpinyo et al. (2021), SBUOrdonez et al. (2011), and 115M images from the LAION400MSchuhmann et al. (2021) dataset additionally.

The pre-training process is conducted in two stages: 250,000 steps in the first stage and 80,000 in the second stage. Batch sizes are 2,320/1,680 for ViT-L/ViT-g in the first stage and 1,920/1,520 for OPT/FlanT5 in the second stage. Utilizing frozen models, this pre-training is more computationally efficient than existing large-scale VLP methods. For example, using a single 16-A100 (40GB) machine, the largest model (ViT-g and FlanT5-XXL) requires less than six days for the first stage and less than three days for the second stage. Although this model is already much more resource-efficient than others, it is still impossible for me to implement.

As mentioned before, one of the drawbacks of LLM on this task is that it is too omnipotent, which can lead to a problem. Because this experiment focuses on generating captions for football videos, the overpowered LLM model does not give a specific example.

For example, when a screenshot of a football video is fed into the BLIP-2 with the OPT model, the caption it generates is very general, such as a football match being played without generating captions describing the details. If the frame is a passing motion, the caption that BLIP-2 with the OPT model will generate will still be a football match, while the frame label should describe whether it is a long pass, a short pass, a free kick, etc. This is because even when a football match is being played, BLIP-2 with the OPT model can't generate a caption that describes the details of the game.

This is because even though fine-tuning is performed, there are only 20,000 entries in the fine-tuned dataset, which cannot be compared to the pre-trained dataset. The difference in scale makes it impossible to guarantee the effectiveness of fine-tuning. In other words, even after fine-tuning, the caption produced by the model is still very general and can not compared with the label caption.

5.7 LLM or LSTM?

Inspired by some models for video captioning tasksMkhallati et al. (2023)Wang et al. (2021a), LSTM is a frequently used module for the captioning process. LSTM has many advantages.

As a simple model, LSTM solves the problem of sequential data well through the gate mechanism. This makes it suitable for sequential problems like text generation. They perform well with smaller datasets and are less computationally intensive.

However, LSTMs are limited to understanding context over long distances in a sequence compared to LLMs, so they may not be suitable for some overly complex tasks.

As for LLMs, which have recently become very popular, are highly versatile and can be fine-tuned for a wide range of tasks such as translation, summarisation, and question-answering. They are embedded in large models as a critical part of processing textual content; for example, in the BLIP2 model, the OPT model is used for the default textual part.

Training and deploying LLMs require significant computational resources, memory, and storage for extensive data to train effectively, which can be a limited effort. Effectiveness can be minimal, which can be a limitation in data-scarce scenarios. LLMs' architecture and training process are more complex and require much longer than LSTM.

Based on this specific task, LSTM is worth a try. The reason is that the dataset is not large enough for training or fine-tuning. The second reason is that LLMs are too large for this task. In the BLIP-2 paper, the authors show various capabilities, including visual conversation, knowledge reasoning, commonsense reasoning, storytelling, personalized image-to-text generation, etc. This specific task is to generate captions. Third, the default opt model chosen is Facebook/opt-2.7b; 2.7 billion parameters are too big for this task. Therefore, an LSTM was designed to generate captions.

5.8 Introduction to LSTM

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that is well-suited for learning from data sequences. It is particularly effective at capturing long-term dependencies in sequences, which traditional RNNs struggle with due to the vanishing gradient problem. LSTMs are widely used in natural language processing, speech recognition, and time series prediction applications.

5.9 Tokenizer from Scratch

Creating a new language model is the most straightforward way to solve the problem. For example, for a football video, we would prefer to see nouns and terms related to football, such as offside, corner kick, free kick, etc., rather than general descriptions, such as player, football, football pitch, etc. In this way, it is necessary to generate your tokenizer from scratch.

Firstly, I collected all captions in the dataset, Loaded spaCy's English tokenizer via spaCy, and added special case rules for specific tokens like '[PLAYER],' '[COACH],' etc., ensuring these tokens are treated as '[PLAYER],' '[COACH],' etc., and '[COACH].' Next, we built vocabulary from the token counts, including only tokens that meet the minimum frequency requirement and adding special tokens like '[PLAYER],' '[COACH],' etc., ensuring these tokens are treated as single units. requirement and adding special tokens like '[PAD]', '[SOS]', '[EOS]', '[UNK]', '[MASK]', '[CLS]' and Set the default index to the '[UNK]' token, which is returned for out-of-vocabulary tokens. Finally, we also define the call Method to do the tokenization and the detokenize method to convert indices back to their corresponding tokens.

The difference between this task and the traditional generative task of LSTM is that the conventional LSTM only needs to go through the training and generate new content based on the textual information. The experiments give excellent results. However, in this experiment, the information from the visual part is equally important. Therefore, the experiment's focus is on how to combine visual information with textual information.

For the visual part, firstly, the features are dimensionally reduced by a convolutional layer and a pooling layer. We aim to split the text part and splice it with the visual features, aligning the visual and textual content by training the spliced multimodal features.

5.10 Caption Processing

The processing of text takes several steps. First, for a model like LSTM that considers sequences, the inputs and outputs differ for each step. This figure shows how LSTM is processed for text. The model should predict the following output for the input information. For example, for the sentence 'Problems for [COACH] as [PLAYER] cannot play anymore. His injury is too serious, and [PLAYER] ([TEAM]) replaces him.' the best result is when the input is ' Problems for [COACH] as [PLAYER] is not able to play,' the best result is that when the input is '[COACH] as [PLAYER] is not able to play,' the next word is predicted to be 'anymore'5.8. The cation must be made into its token ID in the vocabulary, which is then embedded by an embedding network before the prediction.

However, in this step, due to the different lengths of each sentence, the inputs in each batch are of various lengths. This will have a bad effect. So, I limit the max size of a sentence to the most extended token, truncate the part that exceeds the max length, and pad the part that is less than the max length, which ensures that the caption of each batch can have the same size as the input5.9.

5.11 Alternative Design

The second method directly feeds visual features into the LSTM model. The text part is first converted into tokens using the tokenizer, and unique tokens '[SOS]' and '[EOS]' are

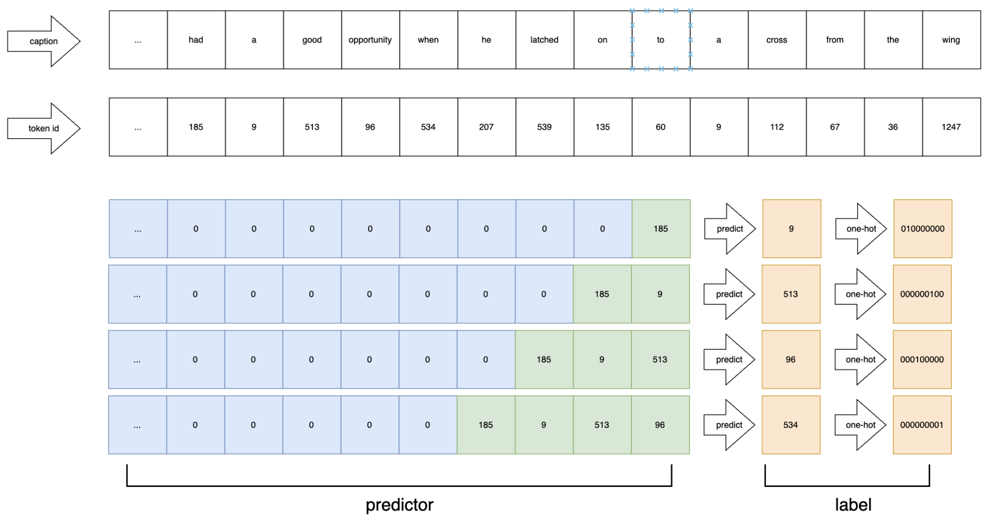


Figure 5.8 The first line is the caption, and the second line is the token ID corresponding to each token. On the left side is the predictor, and from top to bottom are the inputs for different time steps. The gr en part is the valid input part of the current time step. To ensure the sequence's length is certain, each sequence should be filled with padding. The blue part of the figure represents it. These are called predictors and will be used to predict the next token. It is represented in yellow in the figure. Immediately after that, the labels will be processed using the one-hot matrix.

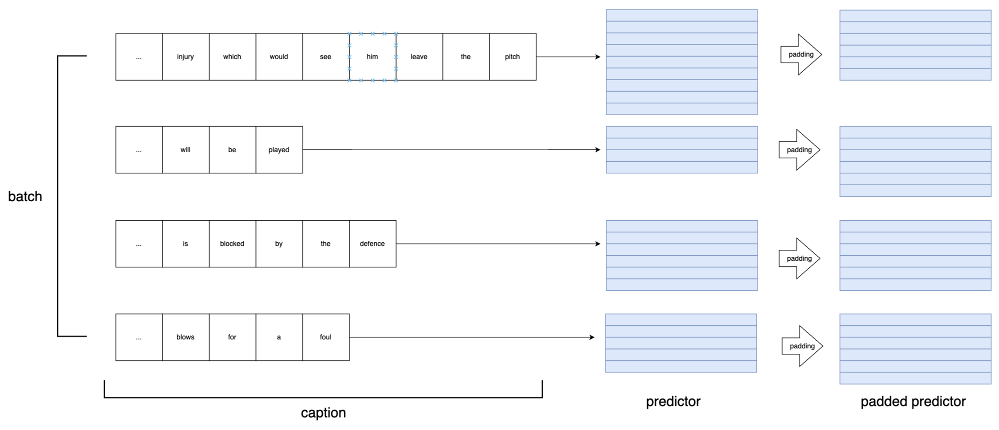


Figure 5.9 This is an example of padding. If the batch size is 4, we need to process four captions, and the length of each caption is different; according to the 5.8, we can know that different lengths of the sentence generated predictor are also different. The longer the sentence, the longer the predictor will be. The length of each batch can't be fixed, so it can't be fed into the network. Therefore, padding operation is essential to make the predictor of sentences of different lengths consistent.

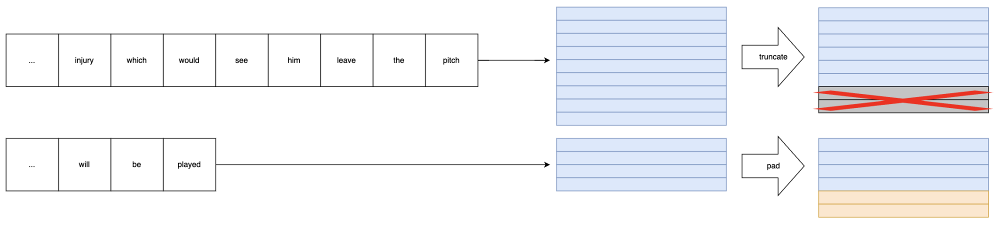


Figure 5.10 Here are the details of the padding section. For example, the length of the first caption is 8, and the length of the second caption is 4. If we set the max length to 6, the first sentence needs to be truncated, and after truncating two lines, it will be 6. The second sentence needs to be padded. This lowers different captions to have the same predictor matrix shape.

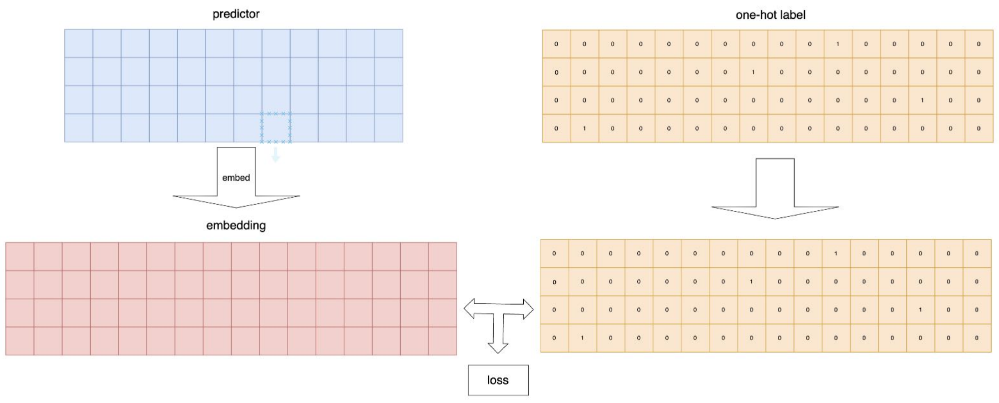


Figure 5.11 After the operation of ref and ref, the predictor can be embedded into embedding, and the one-hot label comes from ref1; at this time, embedding and one-hot label are the same shape. Therefore, we can calculate the loss of both by cross-entropy.

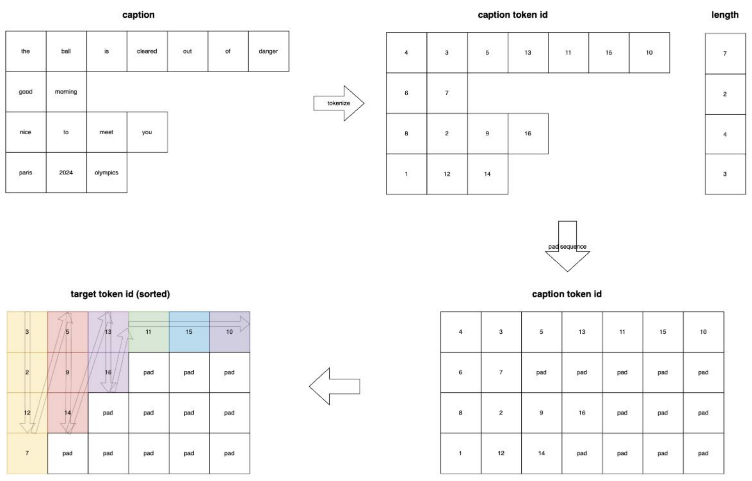


Figure 5.12 The workflow of the second design. (The data is fake and is used for demonstration purposes.)

added at the beginning and end of the captions. This approach represents the labels as a one-dimensional vector rather than a two-dimensional matrix. The *pack_padded_sequence* function is used here, which helps improve the model's convergence to some extent.

In this method, captions are first converted into token IDs, and the length of each caption in the batch is recorded. Next, addition is applied to make each caption sequence the same size. For the targets, only the portion following the first column is selected. The function reorders the sequences based on their lengths (most extended sequences first) and removes the padding. It then packs these sequences into a packed sequence object, a more compact representation for the RNN to process. *pack_padded_sequence* is designed to process sequences of varying lengths more efficiently within RNNs by packing them into a format that allows the RNN to skip computations on the padding tokens, which speeds up training and reduces memory usage.

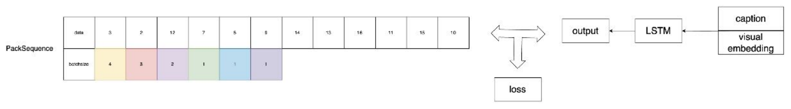


Figure 5.13 PackSequence data and the output of LSTM calculate the loss.

The output of the function is a packed sequence object. It can be used as input to RNNs (such as LSTMs or GRUs), which can handle the packed data more efficiently. The da in the pack sequence object calculates the loss with the model's output.

Chapter 6 Dataset and Metrics

6.1 Dataset

6.1.1 Overview

In this thesis, I use the dataset called SoccerNet-CaptionMkhallati et al. (2023). The SoccerNet-Caption dataset is specifically designed for video captioning in soccer. It includes 471 untrimmed broadcast soccer games from the top five European leagues (EPL, La Liga, Ligue 1, Bundesliga, and Serie A) and the Champions League, spanning 2014 to 2017. In addition to the videos themselves, the dataset provides novel textual comments embedded in time describing the game. The datasets scrape comments from the flashscore website for 471 games.

This comprehensive dataset is divided into two primary components: the video section and the caption section. It encompasses a staggering 715.9 hours of soccer broadcasts for the video segment and boasts 36,894 text commentaries for the caption segment.

6.1.2 Data Anonymization

The dataset undergoes anonymization to mitigate bias, which involves replacing player, team, referee, and coach names with generic tokens. For example, the original caption is: textitMark Clattenburg blows the whistle, Javier Pastore (Paris SG) is penalized for a foul. Barcelona are awarded a free kick from a dangerous position. But after the anonymization, the caption could changed to be like this: *[REFEREE] blows the whistle, [PLAYER] ([TEAM]) is penalized for a foul. [TEAM] are awarded a free kick from a dangerous position.*

6.1.3 Data format

The textual annotations are stored in JSON files and organized by individual games. Each file comprises a dictionary containing all metadata related to the game and a list of annotated comments. Each annotation is accompanied by a timestamp, three versions of the comment (original description, identified, and anonymized), a boolean value indicating whether the comment pertains to a critical moment in the game, and a contextual label, such as corner, substitution, yellow card, whistle, soccer ball, time, injury, fun fact, attendance, penalty, red card, own goal, or missed penalty. However, in this experiment, only comments are needed.

6.1.4 Data statistics

The SoccerNet-Caption dataset features an average of 78.33 temporally localized comments per game, totaling 36,894 captions across all matches. This averages to nearly one comment recorded every minute throughout the dataset. As for the distribution of the comments, most are scattered uniformly in the middle of the game, except there is always a peak at the start, followed by fewer captions for 10 minutes.

Analyzing the distribution of word counts per comment plot reveals that captions typically contain between 4 and 93 words. The distribution shows a pronounced long tail,

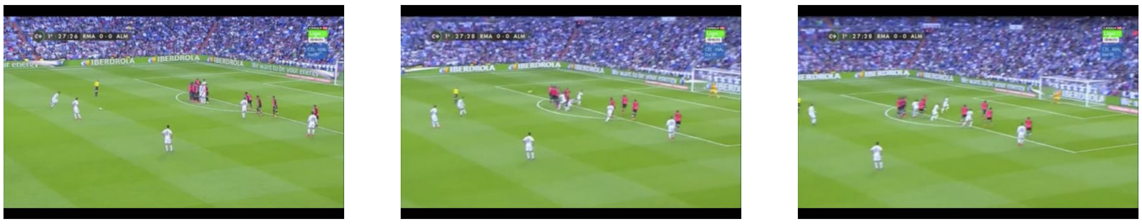


Figure 6.1 **Caption:** Close! [PLAYER] ([TEAM]) stands over the resulting free kick from around 27 meters out, and his shot goes just over the crossbar. The ball goes out of play, and [TEAM] will have a goal kick.

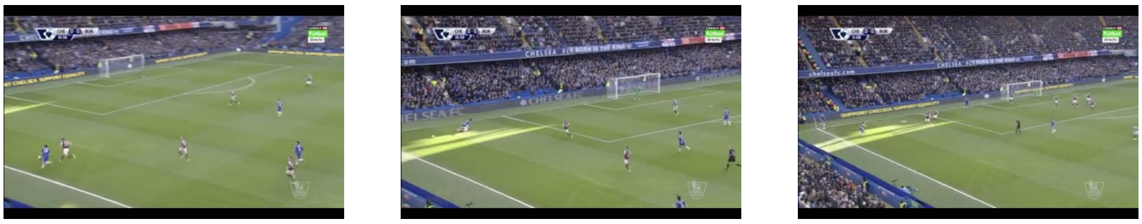


Figure 6.2 **Caption:** [PLAYER] ([TEAM]) slaloms his way past challenges, but an opposing player does well to get the ball away. The ball goes behind for a corner. [TEAM] will have an opportunity to threaten the opposition's goal.

averaging 21.38 words per comment. Aside from generic tokens, the most frequently used words are verbs associated with soccer actions (such as "kick," "pass," "cross") and nouns related to soccer (like "corner," "box," "goal").

6.1.5 Examples

From the 6.1, it can be seen that the athlete is boosting his run and getting ready to go for this free kick. In the second picture, you can see that the ball is flying in the air at the moment, while in the third picture, it is difficult to observe the ball's position.

From the 6.2, in the first picture, you can see two players chasing the ball; in the second, you can see that the blue player appears to be trying to get into the penalty area only to be put down by an opposing player. In the third, you can see the blue team taking a corner.

6.2 Evaluation Metrics

6.2.1 Overview

Some evaluation metrics are widely used in captioning, such as METEOR, BLEU, ROUGE-L, CIDEr, and SPICE. A practical evaluation metric should yield satisfactory results even when words are replaced with their synonyms, redundant words are added, the word order is altered, or sentences are abbreviated without changing the meaning. This is the overview of these metrics from figure 6.3.

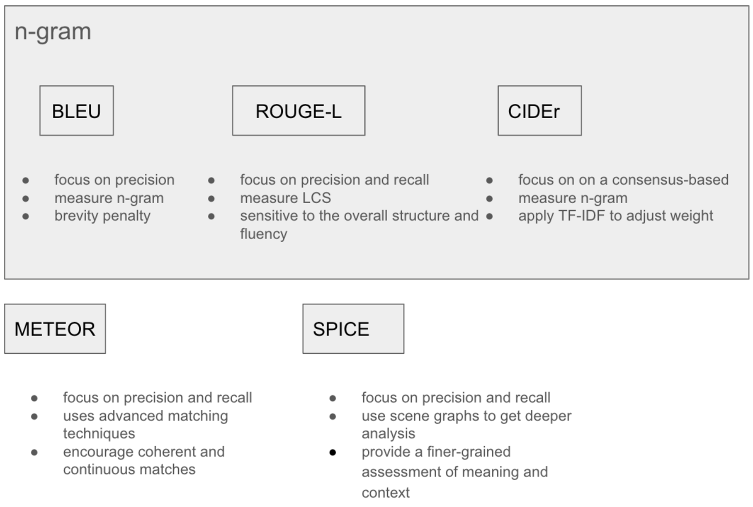

n-gram
BLEU
ROUGE-L
CIDEr
focus on precision
measure n-gram
brevity penalty
focus on precision and recall
measure LCS
sensitive to the overall structure and fluency
focus on on a consensus-based
measure n-gram
apply TF-IDF to adjust weight
METEOR
SPICE
focus on precision and recall
uses advanced matching techniques
encourage coherent and continuous matches
focus on precision and recall
use scene graphs to get deeper analysis
provide a finer-grained assessment of meaning and context

Figure 6.3 Overview of the metrics

6.2.2 BLEU

The Bilingual Evaluation Understudy (BLEU)Papineni et al. (2002) is designed for evaluating machine translation at first. Still, it can assess other NLP tasks, such as captioning, text summarization, and paraphrasing. It is easy and fast to calculate, although there are some drawbacks. There is a very famous example:

reference: The cat is on the mat.,

And there are two human versions of English translations:

candidate1: The cat is on the carpet.

candidate2: There is a cat on the mat.

To express the quality of the candidate, the score is calculated by counting how many words in the candidate translation are present in the reference translation and dividing the result by the number of words in the candidate to get a percentage. For example, in this case, BLEU(c1) = 5/6 =83.3%, BLEU(c2) = 5/7 =71.4%.

However, if there is a

candidate3:The the the the the the.

the BLEU(c3) would be 6/6 = 1, which does not make sense. We count the certain word "the" only for the times it appears at most on each reference translation, which is 2 in this case. BLEU(c3) then becomes 2/6 = 33.3%.

If there is a

candidate4:Cat on is at the the.

according to the formula, BLEU(c4) = 6/6 =1, but it is wrong obviously. Instead of considering only 1-grams, n-grams are introduced to address this problem. If we divide the reference and candidate4 into 2-grams, we can get

The cat, cat is, is on, on the, the mat and

Cat on, on is, is mat, mat the, the the.

The BLEU(c4) becomes 0/5 = 0.

Finally, if

candidate5: The cat

No matter 1-grams or 2-grams, BLEU(c5) = 2/2 =1. Missing information helps candidate 5 get a higher score. The Brevity Penalty (BP) is used to avoid this, penalizing some short sentences that lose much relevant information.

$$p_n = \frac{\sum_{C\in\text{Candidates}} \sum_{\text{ngram}\in C} \min(\text{count}_{\text{clip}}(\text{ngram}), \text{count}_{\text{ref}}(\text{ngram}))}{\sum_{C\in\text{Candidates}} \sum_{\text{ngram}\in C} \text{count}_{\text{clip}}(\text{ngram})} \tag{6.1}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \tag{6.2}$$

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{6.3}$$

where BF is the Brevity Penalty factor, p_n is the geometric mean of the modified n-gram precision up to length N, and w n is the weight of n-gram precision, where the sum of w n terms is equal to 1. Assume c is the length of the candidate sentence, and r is the reference length.

The BLEU evaluation metric is precision-based and does not incorporate recall. It treats synonymous words as distinct, penalizing even minor word-choice variations. This limitation is mitigated by METEOR, which generally aligns more closely with human judgments.

6.2.3 METEOR

The Metric for Evaluation of Translation with Explicit ORdering (METEOR)Banerjee and Lavie (2005) is a metric designed to evaluate machine translation quality. It aims to improve upon the shortcomings of BLEU by incorporating more nuanced approaches to measuring translation quality, such as synonymy, stemming, and word order. The metric is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision, combines them into an F1 score, and adjusts for word order using a chunk penalty. This approach makes METEOR more closely aligned with human judgment in evaluating translation quality.

$$\text{METEOR} = H_m \cdot (1 - \text{Penalty}) \tag{6.4}$$

$$P = \frac{\text{Number of matched words}}{\text{Total words in candidate}} \tag{6.5}$$

$$R = \frac{\text{Number of matched words}}{\text{Total words in reference}} \tag{6.6}$$

$$H_m = \frac{10 \cdot P \cdot R}{R + 9P} \tag{6.7}$$

$$\text{Penalty} = \gamma \cdot \left(\frac{\text{chunk\_count}}{\text{matched\_word\_count}}\right)^{\beta} \tag{6.8}$$

Here is an example,
reference: The cat is on the mat.
candidate: On the mat is the cat.

All 1-gram elements match due to the candidate being the jumbled version of the reference. So the precision is P=6/6=1 and R=6/6=1. We can match ’on the mat’ and ’the cat’; there are two chunks.

The H_m given by 6.7 is $H_m = \frac{10\times1\times1}{1+9\times1} = 1$.
Normally, we set γ to 0.5, β to 3, the penalty in 6.8 is $0.5 \times (\frac{2}{6})^3 = 0.000013$.
The METEOR score in 6.4 is $1 \times (1 - 0.000013) = 0.9997$

6.2.4 ROUGE-L

Longest Common Subsequence (ROUGE-L)Lin (2004) is a sequence-based similarity assessment method for evaluating the similarity between generated and reference summaries. It calculates the similarity by comparing the longest common subsequence of two sequences; the longer the subsequence length, the more similar the two sequences are.

$$ROUGE - L = \frac{\sum_{s\in\text{summary}} \sum_{l\in\text{reference}} \text{LongestCommonSubsequence}(s, l)}{\sum_{l\in\text{reference}} \text{length}(l)} \tag{6.9}$$

ROUGE-L comes from ROUGE-n, n means n-gram, and the scoring method is similar to BLEU, which I will ignore here. But L means LCS(Longest Common Subsequence). For example,

reference: Today is a good day!
candidate1: Today is a lovely day!

The common subsequence between reference and candidate1 is **Today is a day**, although these words are not right next to each other. The advantage of LCS is that it does not require the candidate to match strictly to the reference. Compared with ROUGE-n, n-gram needs to be defined in advance, which makes it less flexible. In this way, ROUGE-L can calculate the relationship between sentences.

Here is an example which explains the benefits of ROUGE-L:
reference: police killed the gunman
candidate1: police kill the gunman
candidate2: the gunman kill police
By ROUGE-2,
ROUGE-2(c1)=1/4=0.25 (the gunman)
ROUGE-2(c2)=1/4=0.25(the gunman)
The meanings of the two sentences are entirely different, but they score the same, which is unreasonable.
By ROUGE-L,
ROUGE-L(c1)=3/4=0.75 (police the gunman)
ROUGE-L(c2)=2/4=0.5 (the gunman)
C1 has a higher score, which summarizes the reference better than c2 by calculating the ROUGE-L.

6.2.5 CIDEr

Consensus-based Image Description Evaluation (CIDEr)Ghosal et al. (2021) is a metric to evaluate the machine-generated captions developed based on BLEU. CIDEr aligns more closely with how humans determine the similarity between two sentences. It employs TF-IDF to assign varying weights to different n-grams. Intuitively, this means that commonly occurring phrases are given lower weights, while rare phrases are considered more specific and are assigned higher weights. This reflects the tendency to pay more attention to unique or distinctive words. For example, **I will go to the bar tonight**, the 'bar' needs more attention than 'go to.'

Term frequency-inverse document frequency(TF-IDF) counts the number of occurrences of each word in the document. While a simple method, there is a lot of weight put on words that occur frequently that might not be significant and adequate for the classification of a document. TF is the counts divided by the total words of a document; this makes relative word frequencies more equal than all the documents as document size is no longer considered. TF-IDF is the TF multiplied by the IDF, where IDF is defined as the logarithm of the total documents divided by the document that contains the word of interest to which a one is added. If the IDF value is high, this indicates that the word of interest has significant value in document classification.

In CIDEr, TF-IDF is a statistical measure used to evaluate how relevant n-gram is to a video in a collection of videos in the database.

$$\text{CIDEr}(c_i, s_i) = \frac{1}{n}\sum_{n=1}^{N}\sum_{g\in G_n}\text{IDF}(g)\cdot\frac{\text{count}_g(c_i)\cdot\text{count}_g(s_i)}{\text{count}_g(c_i)+\text{count}_g(s_i)} \tag{6.10}$$

where c_i is the generated caption, s_i is the reference caption, n is the number of reference captions, G_n is the set of all n-grams in the reference sentences, IDF(g) is the inverse document frequency of the n-gram g, count$_g(c_i)$ is the count of n-gram g in the generated caption c_i, count$_g(s_i)$ is the count of n-gram g in the generated caption s_i.

6.2.6 SPICE

Semantic Propositional Image Caption Evaluation (SPICE)Anderson et al. (2016) is a metric designed to evaluate the quality of image captions generated by models. Unlike traditional metrics that primarily focus on lexical similarities (n-gram), SPICE focuses on the semantic content of captions.

Let us see some limitations of the n-gram,

A young girl standing on top of a tennis court. and

A giraffe standing on top of a green field.

These two sentences are similar because they both contain 'standing on top of if we use 5-grams. However, these two sentences do not correlate with the semantic level. However, if we see

A shiny metal pot filled with some diced veggies. and

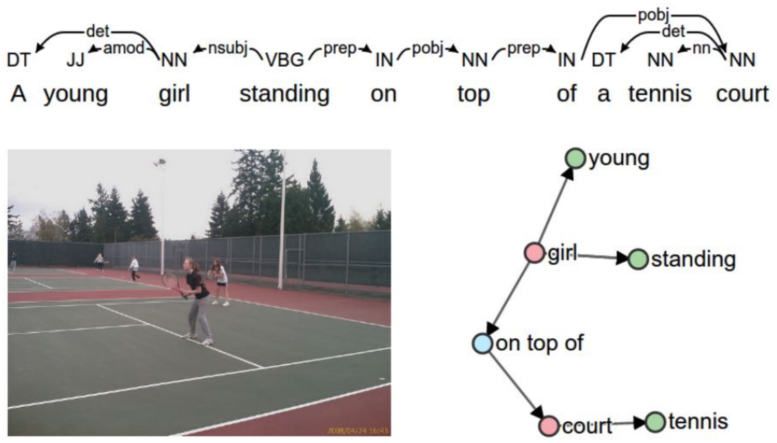
The pan on the stove has chopped vegetables in it.

We could know they are highly correlated on a semantic level, but the n-gram score is low.

Although n-gram is easy to use and calculate when sentences with similar meanings contain different words, the performance of n-gram is poor; SPICE is designed to capture higher semantic similarity by comparing the scene graph representations of both the generated and reference captions. These scene graphs are structured representations that encapsulate the objects, attributes, and relationships described in a caption.

We define C as the set of classes,R as the set of relation types,A as the set of attribute types, and c as a caption. So, the sense graph of c is

$$G(c) = \langle O(c), E(c), K(c)\rangle \tag{6.11}$$



DT JJ NN VBG IN NN IN DT NN NN
det
amod
nsubj
prep
pobj
prep
pobj
det
nn
A young girl standing on top of a tennis court
young
girl
standing
on top of
court
tennis

Figure 6.4 The caption(top) and corresponding semantic scene graphs(right)

where $O(c) \subseteq C$ is the set of object mentions in c, , $E(c) \subseteq O(c) \times R \times O(c)$ represents relations between objects, $K(c) \subseteq O(c) \times A$ is the set of attributes associated with objects.

Here is an example: the caption is

A young girl standing on top of a tennis court

girl and *court* are objects; *young*, *standing*, *tennis* are attributes, *on top of* is the relation. Then we can get the set {(girl), (court), (girl, young), (girl, standing), (court, tennis), (girl, on-top-of, court)}

$$\text{Precision} = \frac{|G_c \cap G_r|}{|G_c|} \tag{6.12}$$

$$\text{Recall} = \frac{|G_c \cap G_r|}{|G_r|} \tag{6.13}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.14}$$

where G_c represents the set of semantic units in the generated caption, including objects, attributes, and relationships, G_r represents the set of semantic units in the reference caption, $|G_c \cap G_r|$ represents the size of the intersection of semantic units between G_c and G_r.

By 6.12, 6.13, 6.14, we could get the SPICE score of the generated caption.

6.3 Cross-entropy

Cross-entropyMao et al. (2023) loss is a commonly used loss function for classification tasks in machine learning. It measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

For multi-class classification, where the target label y is a one-hot encoded vector, the cross-entropy loss is defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c})$$

Where:

- N is the number of samples.
- C is the number of classes.
- $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for sample i.
- $p_{i,c}$ is the predicted probability that sample i is of class c.

This loss function calculates the loss of multimodal features with labels after going through LSTM and OPT models. For example, in LSTM, the output and target of LSTM are 2D tensors of the same shape, and cross-entropy is used to calculate the loss of these two tensors.

Chapter 7 Experiments

7.1 Overview

First, the baseline model is the BLIP-2 with **fine-tuning OPT model**. In the baseline model, **late fusion** visual features are extracted using **average pooling**, and the text encoder utilizes **BERT**. I will attempt to use LSTM as another text generation model in the experiment. Similarly, I will experiment with single-frame and random features for visual features. The Soccernet Tokenizer, generated from this dataset, will be used for the tokenizer. Finally, we will explore the impact of early fusion and inference without fine-tuning the experimental results.

Table 7.1 The overview of all experiments.

	Caption Model	Visual Features	Tokenizer	Note
1	**OPT**	**Average Pooling**	**BERT**	**baseline**
2	LSTM	Average Pooling	BERT	model
3	OPT	Average Pooling	Soccernet	tokenizer
4	OPT	Single Feature	BERT	10/20 frames
5	OPT	Average Pooling	BERT	early fusion
6	OPT	Random Feature	BERT	none
7	OPT	Average Pooling	BERT	no FT
8	OPT	Average Pooling	BERT	1/3/5/10 epochs

7.2 Baseline Model

The model architecture is blip2_opt, the BLIP-2 model embedded with the OPT model. The OPT model used is the 2.7b version. Regarding image preprocessing, we use images of size 364×364, augmented with random resized cropping and horizontal flipping. The pre-training hyper-parameters are as follows: we use the AdamW optimizer with β_1 = 0.9, β_2 = 0.98, and a weight decay 0.05. We use a cosine learning rate decay with a peak learning rate of 1e-4 and a linear warmup of 2000 steps. The minimum learning rate in the second stage is 5e-5. For the fine-tuning part, the initial learning rate is 1e-4 with a weight decay 0.05. The batch size for training and evaluation is 8.

7.3 Fine-tuning

In this section, I attempted to use the BLIP-2 model for inference without fine-tuning directly. Based on the original pre-training of BLIP-2, we fine-tuned the model using SoccerNet data.

Table 7.2 The metrics of the fine-tuning and without it.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
FT	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
no FT	0.148	0.112	0.083	0.023	0.143	0.274	0.005	0.079

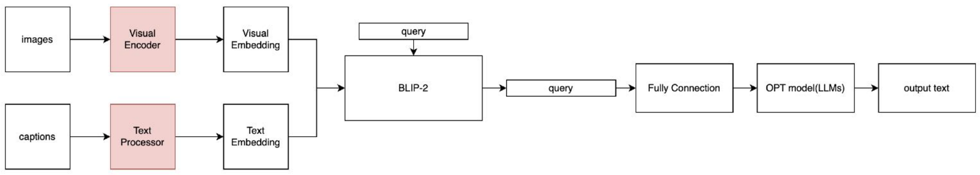


Figure 7.1 The BLIP-2 model in paperLi et al. (2023).

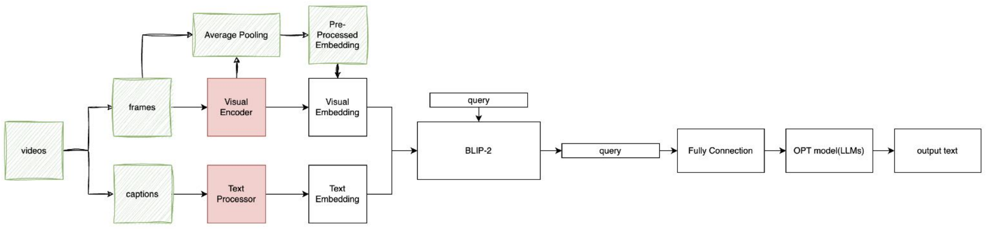


Figure 7.2 The baseline BLIP-2 OPT model.

From the results shown in 7.2, it is clear that fine-tuning significantly improves most of the evaluation metrics. Specifically, with fine-tuning (FT), there are higher scores in BLEU, METEOR, ROUGE_L, and CIDEr compared to the model without fine-tuning (no FT). For example, BLEU_1 increases from 0.148 to 0.185, and CIDEr improves from 0.005 to 0.049, indicating better alignment between the generated text and the reference.

However, it's worth noting that the SPICE score is higher in the model without fine-tuning, with a value of 0.079 compared to 0.028 with fine-tuning. This suggests that while fine-tuning helps improve surface-level metrics focusing on word and phrase matching, it might not capture the deeper semantic aspects as effectively as the non-fine-tuned model.

The improvements seen with fine-tuning likely occur because fine-tuning adjusts the model to understand better and generate text that aligns with the specific tokens of the fine-tuning training data. This process refines the model's parameters, enhancing its ability to generate more accurate and contextually soccer-related outputs. The decrease in SPICE might be due to the model overfitting to the specific data during fine-tuning, focusing more on word and phrase matching than maintaining semantic diversity. Therefore, fine-tuning is necessary.

7.4 Qualitive Results

In this section, we will analyze the captions generated by the model. These results were produced using the baseline model hyperparameters. The images shown represent a sequential six-frame sample from a video clip. Figure 7.3 shows that the model successfully predicted a free kick, even though the label caption provided a more comprehensive and detailed description of the video clip. This outcome is reasonable since the action likely extended beyond the range of the video clip I used, meaning the cross-pass action may not have been captured within the selected frames. As a result, the generated caption only captured the free kick.

In figures 7.4 and 7.5, we can observe that the generated captions and the label captions convey essentially the same semantic information, despite some differences in word order, particularly in figure 7.4. By examining the captions, we can see that actions such as "blow the whistle," "end the first half," and "takes a short corner kick" are mentioned in both generate caption and label caption, although the verb forms differ. These can be considered sound generation examples.

However, in figure 7.6, I present a case of significant failure where the model generates completely incorrect captions. Nonetheless, this example still demonstrates that the model has undergone pre-training. Even though the word "screen" was not present in the training set, it appears in the generated text, indicating that this word was part of the pre-training dataset. Moreover, the model also shows evidence of effective fine-tuning, as it can accurately extract information from the input video. This is proven by generating captions containing symbols like '[' and ']', which were not present in the pre-training dataset. Therefore, the fine-tuning process is indeed effective.

These results indicate that our baseline model can generate accurate captions, but there is still room for improvement, particularly in capturing temporal and complex contexts.

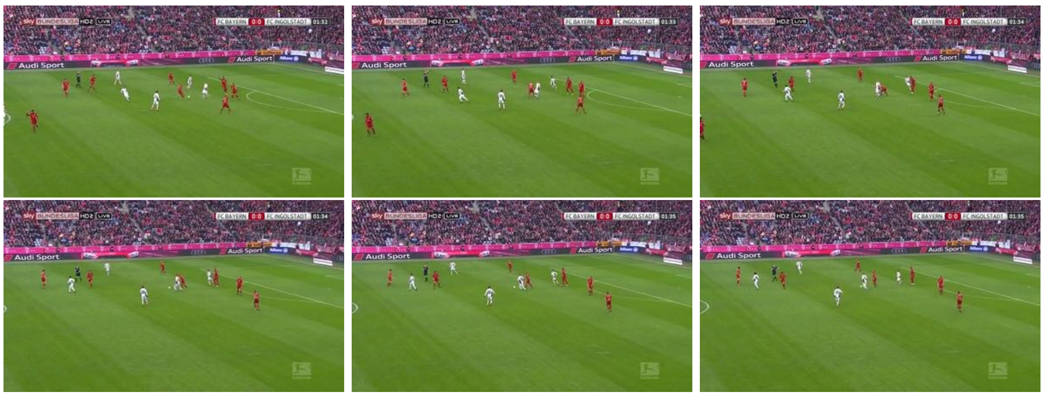


Figure 7.3 2015-12-12, Germany Bundesliga 2015-2016, Bayern Munich 2-0 Ingolstadt
Generated Caption: [player] [team] is taken by [referee] [official] and [team] is awarded a free kick.
Label Caption: [PLAYER] ([TEAM]) takes the resultant free kick from the edge of the box and tries to find one of his teammates with a cross.

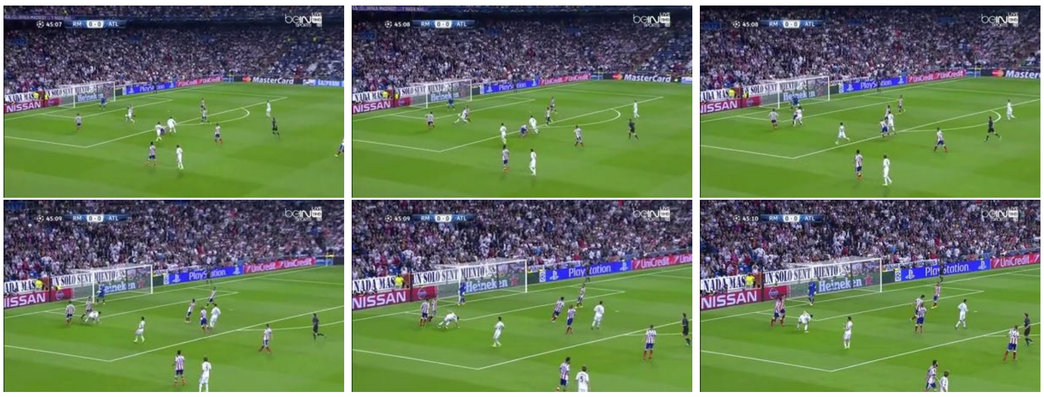


Figure 7.4 2015-04-22, Europe UEFA Champions League 2014-2015, Real Madrid 1 - 0 Atl. Madrid
Generated Caption: [player] [team] [referee] [official] blows the whistle to end the first half.
Label Caption: [REFEREE] has ended the first half by blowing the whistle.

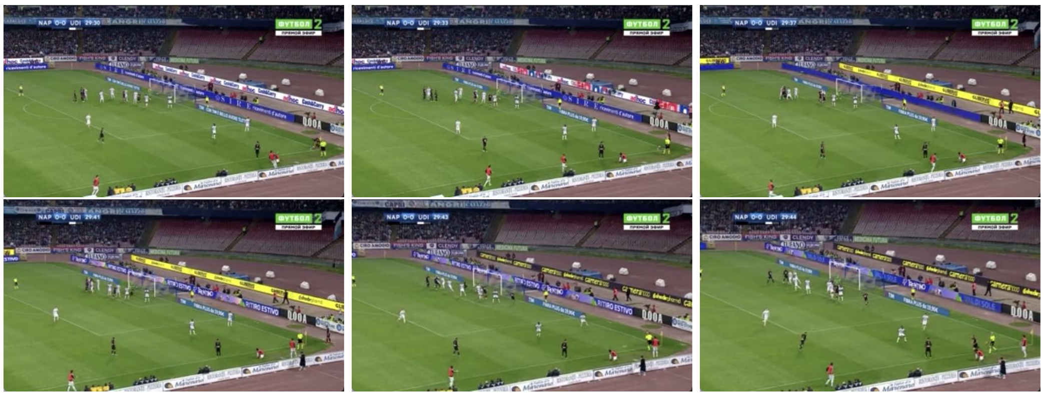


Figure 7.5 2017-04-15, Italy Serie A 2016-2017, Napoli 3 - 0 Udinese
Generated Caption: [player] today's match [player] [team] takes a short corner kick.
Label Caption: [PLAYER] ([TEAM]) takes a short corner kick.

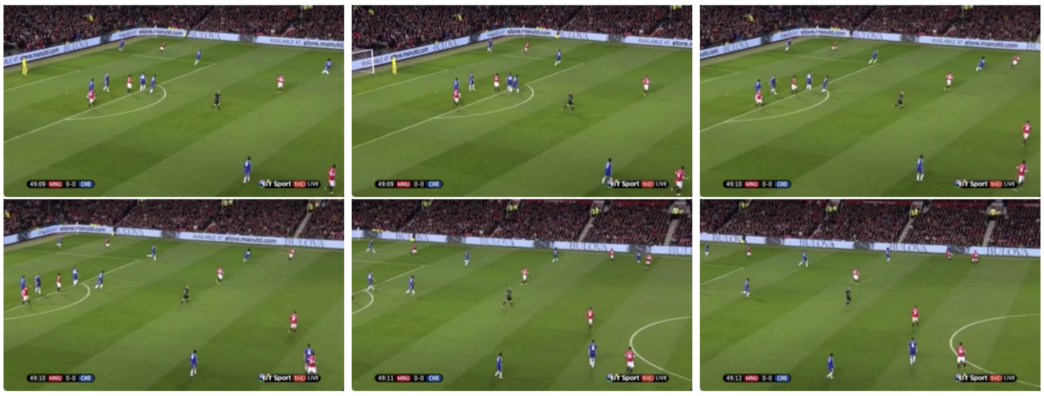


Figure 7.6 2015-12-28, England Premier League 2015-2016, Manchester United 0 - 0 Chelsea
Generated Caption: [player] [team] is shown on the big screen.
Label Caption: [PLAYER] ([TEAM]) collects a pass and finds himself in a good position to strike from the edge of the box, but his attempt is blocked.

7.5 Epochs

Setting the number of epochs is crucial to training machine learning models. It requires careful tuning to balance underfitting and overfitting, ensure model convergence, optimize validation performance, manage computational costs, and consider the learning rate and dataset characteristics. Properly configuring the number of epochs can lead to more effective and efficient training processes, resulting in better-performing models. The experiment was based on the baseline hyperparameter setting(except the epoch varies).

Table 7.3 1, 3, 5, and 10 epochs metrics.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
1	0.142	0.110	0.083	0.019	0.157	0.320	0.007	0.004
3	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
5	0.197	0.154	0.109	0.030	0.174	0.313	0.047	0.104
10	0.227	0.167	0.122	0.043	0.167	0.301	0.050	0.112

From the table7.3, we can see that the metrics generally improve as the number of epochs in the experiment increases. The BLEU score clearly illustrates this trend. As the

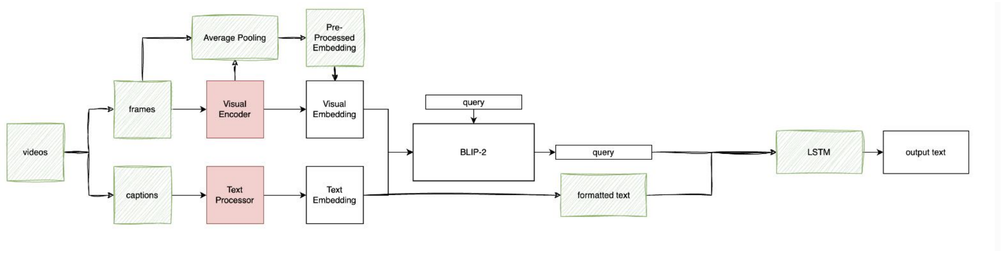


Figure 7.7 The LSTM replaces the OPT model.

number of epochs increases and the experimental results improve, the model learns and better captures the underlying patterns in the data. However, considering the total training time and the metrics, I ultimately set the number of epochs to **5**.

7.6 Language Generative Model

Based on the previous analysis, I attempted two different types of LSTM models—the difference lies in how each batch's predictor and label are constructed. In the first method, the visual and text embedding are concatenated to achieve alignment between the two modalities. However, this method did not perform well, and the loss did not decrease.

In the second method, the visual embedding and text embedding are used as inputs to the LSTM module. After training, this approach proved to be effective, and the model showed convergence.

Table 7.4 The metrics of OPT and LSTM model.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
OPT	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
LSTM	0.114	0.004	0.000	0.000	0.095	0.206	0.001	0.009

From the table7.4, it is clear that OPT significantly outperforms LSTM. A possible explanation for this difference lies in the alignment between the two modalities. While LSTM has proven effective in other video captioning tasks, simply replacing the OPT module with LSTM in the figure7.7 without additional adjustments may not be sufficient to maintain performance.

7.7 Visual Features

7.7.1 Number of Frames

In the experiment, we use single-frame features, features from 10 frames, and features from 20 frames with average pooling as inputs. The goal is to examine the effectiveness of average pooling and determine whether performance improves with an increasing number of frames7.8.

Based on the experimental table7.5, each of the three settings for the number of frames has its advantages and disadvantages. For instance, the single frame setting scores the highest on BLEU and METEOR, ten frames achieve the highest scores on ROUGE and CIDEr, while 20 frames score the highest on SPICE.

This variation can be explained by the fact that different metrics emphasize different aspects of the generated captions. BLEU and METEOR, which focus on n-gram overlap,

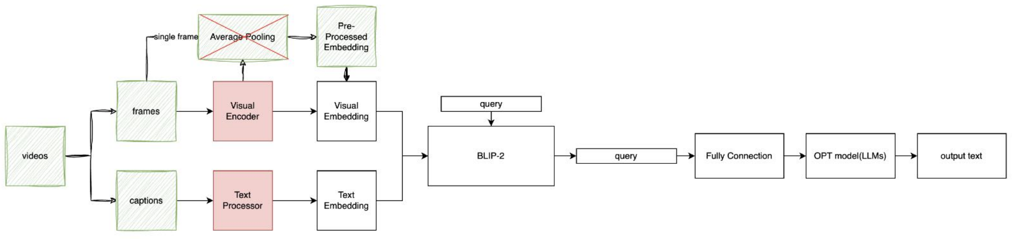


Figure 7.8 The single frame replaces average pooling.

Table 7.5 The metrics of different numbers of frames.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
single frame	0.199	0.151	0.111	0.022	0.174	0.304	0.031	0.065
10 frames	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
20 frames	0.157	0.121	0.090	0.025	0.158	0.321	0.011	0.099

may benefit from the detailed information captured in a single frame. ROUGE and CIDEr, which consider longer sequences and overall content, might gain from the context of ten frames. SPICE, which evaluates the semantic content, may be more sensitive to the more prosperous, diverse information available in 20 frames.

Finally, I used **10 frames** in subsequent experiments because its metrics were more balanced than a single frame and 20 frames. Additionally, since the video encoder requires time and computational resources to extract features, using ten frames is more resource-efficient than 20 frames while avoiding the randomness of single frames.

7.7.2 Early and Late Fusion

Because in the visual feature extraction part, we used ViT, which was pre-trained on regular images. The early fusion images involve fusion at the image level, while the late fusion we previously adopted was at the feature level. Therefore, ViT cannot theoretically recognize the images resulting from early fusion5.4. As it shows, some soccer game elements could be identified for some processed pictures. However, for others, they are just random noisy patterns in the pictures. Hence, I want to try using these early fusion images as the model's input to evaluate the performance on image generation in the figure7.9.

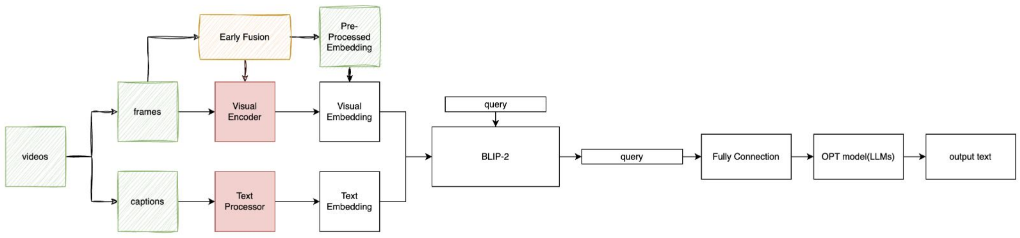


Figure 7.9 The early fusion replaces late fusion.

Regarding BLEU scores in the table7.6, which measure n-gram overlap between generated and reference captions, late fusion significantly outperforms early fusion at every level. This suggests that late fusion generates captions by average features more successfully. Similarly, the METEOR score, which accounts for synonymy and recall, is much higher for early fusion, indicating it generates more linguistically diverse and

Table 7.6 The metrics of early fusion and late fusion.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
late fusion	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
early fusion	0.190	0.080	0.049	0.010	0.113	0.219	0.003	0.003

accurate captions.

ROUGE_L, which focuses on the longest common subsequence between generated and reference captions, and CIDEr, which measures consensus based on TF-IDF weighting, show that late fusion produces captions that align more closely with reference captions. The SPICE score, which evaluates semantic content, is also better for early fusion, although both methods score relatively low on this metric.

Overall, these results suggest that late fusion is more effective than early fusion in generating captions that are both syntactically and semantically aligned with reference captions. Early fusion, on the other hand, seems to dilute important information by merging visual and textual data too early, leading to poorer performance across all metrics, particularly in content and semantic accuracy, as indicated by the CIDEr and SPICE scores.

7.7.3 Image Encoder and Noisy Input

Additionally, to verify the effectiveness of the visual encoder, I attempted to input random noise features directly to check the generated captions as shown in figure 7.10. This can help determine if the lack of diversity in caption generation is due to the visual encoder's outputs being too similar for different inputs, causing the model to have a significant bias.

The results7.7 show that the performance with random feature input is abysmal, as all metrics are significantly lower than the baseline. This indicates that the visual encoder extracts meaningful information, generating coherent text with pre-training. In contrast, random feature input fails to achieve this, demonstrating the effectiveness and necessity of the visual encoder.

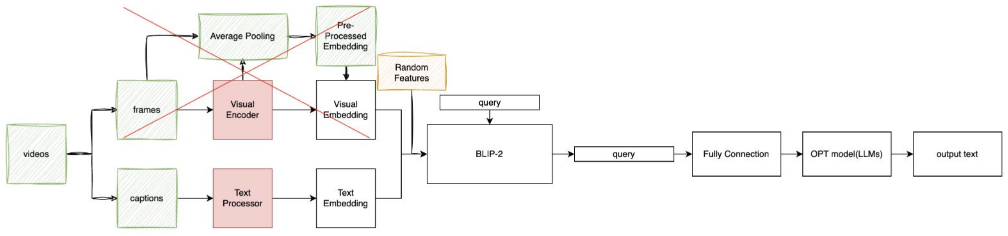


Figure 7.10 The random input replaces the visual encoder output.

Table 7.7 The metrics of the image features and noisy feature input.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
image	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
random	0.013	0.002	0.001	0.000	0.017	0.065	0.002	0.002

7.8 Tokenizer

The OPT model used the tokenizer from the BERT class, while I created a new Soccernet tokenizer based on the tokens from the dataset. Both tokenizers use unique tokens such as [SOS], [EOS], and [PAD]. However, the main difference lies in the vocabulary size. The vocabulary size for BERT is typically around 30,000 tokens, whereas the Soccernet vocabulary size is less than 3,000 tokens.

Therefore, I replaced the tokenizer in the OPT part of the BLIP-2 model with the Soccernet tokenizer and conducted experiments in figure7.11.

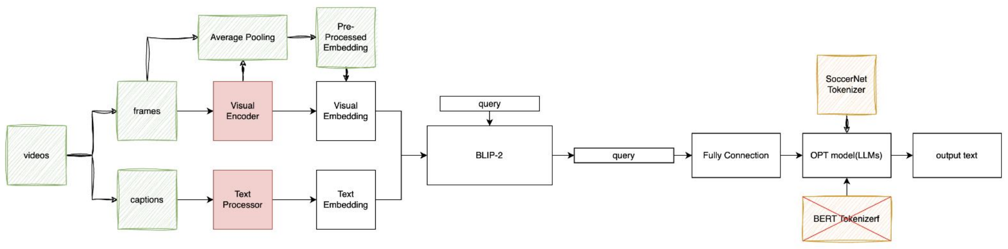


Figure 7.11 The SoccerNet tokenizer replaces the original tokenizer.

Table 7.8 The metrics of BERT and SoccerNet tokenizer.

	B_1	B_2	B_3	B_4	METEOR	ROUGE_L	CIDEr	SPICE
BERT	0.185	0.141	0.104	0.024	0.170	0.310	0.049	0.028
SN	0.189	0.144	0.106	0.026	0.171	0.310	0.051	0.029

The two models ' outcomes are very similar based on the experimental results7.8. This indicates that merely changing the tokenizer does not have a significant impact. The reason lies in the fact that when using BERT's tokenizer, not only is the text of the label captions tokenized with BERT's tokenizer, but we also need to process information such as the queries and attention masks from the Q-Former during the model's forward pass5.7. BERT generates these pieces of information. The tokenizer change ultimately only alters the target that enters the OPT model. Therefore, the results of using the two tokenizers will be similar from this perspective.

On the other hand, it is reasonable that the results using the Soccernet tokenizer are slightly better than those using BERT. BERT's tokenizer has a vocabulary size of over 30,000, while Soccernet's is only 2,759. During the generation phase, the generated vocabulary is restricted to 2,759 words, meaning the vocabulary space is much smaller than BERT's. Consequently, the generated words are more likely to fit the distribution of the dataset, resulting in better performance than BERT.

Chapter 8 Discussion

8.1 Analysis

Based on the experimental results, we can see a significant difference between the results of the OPT and LSTM models, with OPT outperforming the LSTM I created. The single-frame approach ensures the accuracy of the generated text, while multi-frame approaches better capture temporal information. Early and Late Fusion have a considerable impact on the experimental results, with Early Fusion severely affecting the performance of the visual encoder. In the experiments, increasing the number of epochs improved the results. The impact of the tokenizer on the results was insignificant, as I did not remove all parts of BERT from the model but only modified the text processing part, so the new tokenizer did not lead to a significant improvement. Lastly, fine-tuning also had an effect, but a limitation of this experiment is that the fine-tuning dataset was too small.

After concluding this experiment, future work has many directions to explore. Overall, this thesis is based on the BLIP-2 model, using frames from specific moments in the video as input to the model, which then generates corresponding captions.

Below, I analyze potential future work from the perspectives of the encoder, data, and applications.

8.2 Future Work

First, the video encoder is the most promising way to explore. The reason for not using a video encoder in this experiment is that it requires extensive computational resources, which I could not access. Therefore, I used consecutive video frames to represent the video input. For a dataset like soccer matches, the relationships between frames are crucial. Thus, trying a video encoder is worthwhile.

Second, my method can be applied to other video datasets. As mentioned previously, the information in soccer match videos heavily depends on the relationships between frames, making video features indispensable. However, if the information gap between frames is not significant for other types of videos, image frames might suffice to represent a video segment. A video captioning model based on image frames might perform better.

Third, data cleaning of video frames is essential. Many factors could affect the results of the method proposed in this paper. For instance, in the dataset's videos, there are frames of the match on the field and other shots like player close-ups, coach expressions, and audience reactions. These features differ significantly from the match features needed for caption generation. Through average pooling, valuable features might get diluted. Thus, one possible approach is to filter the quality of video frames and exclude those unrelated to soccer activities.

Fourth, the VIT model in BLIP-2 is another aspect worth enhancing and improving. BLIP-2 can be divided into two stages: the first stage aligns text and image representations, and the second stage generates text. The results were unsatisfactory during the second stage, despite attempts to fine-tune large language models like OPT or designing my language model from scratch. The possible reason might be issues occurring in the first stage. Due to the videos' low resolution and low fps, it is sometimes challenging for humans to discern the ball’s position. When extracting specific frames from the video, the image quality is poor, and VIT might fail to recognize the scene or deeper information, such as actions and the ball’s state. Therefore, fine-tuning VIT in the first stage to make it suitable for specific downstream tasks might be worth exploring.

Finally, it is also worth trying this method on videos of other sports. Although soccer has specialized terminology, other sports share common terms such as score, pass, and run. Applying a soccer-specific model to other sports could be an interesting experiment.

Chapter 9 Conclusion

This thesis aims to explore the use of VLP models like BLIP-2 for video captioning.

I pre-processed video data and text files in my experiments, then modified the existing model to suit this specific task. I conducted various comparative experiments, including testing different methods of video feature fusion, models for text generation, choices of language models, and hyperparameter adjustments, among others. Most of these experimental results align with the expectations and analyses.

Different language models require varying resources to achieve convergence, and the experimental outcomes differ. The OPT model performs significantly better than LSTM but requires more epochs. Regarding the number of frames, the results for a single frame are pretty good, but in some metrics, average frames yield better results due to the inclusion of more temporal information. The visual encoder is also a crucial component, and a fine-tuned visual encoder may achieve better results. As for epochs, more epochs generally lead to better outcomes. Changing the language model's tokenizer has little impact on the experimental results, but fine-tuning does have a significant effect, significantly improving the results.

Last, I will address the questions raised in the introduction.

1. Through experiments, it has been demonstrated that BLIP-2 can handle video understanding tasks.

2. The main difference between using multiple frames and a single frame is the presence or absence of temporal information. While single frames perform well in many video captioning tasks, football videos often lack temporal context. Temporal information is crucial in football match videos. Since this experiment utilizes average pooling to merge frame data instead of directly using video features, temporal information is inherently missing. As a result, multiple frames and single frames show similar performance in this experiment.

3. From the experiments, it is evident that the most significant factor affecting the results compared to the baseline model is whether the model has been fine-tuned. The most important factors are the correct use of the image encoder, the choice of language generation model, and the number of epochs. The number of frames, feature fusion methods, and tokenizer choices show relatively minor differences in impact.

Bibliography

Aafaq, N., Akhtar, N., Liu, W., & Mian, A. (2021). Empirical autopsy of deep video captioning encoder-decoder architecture. Array, 9, 100052.

Anderson, P., Fernando, B., Johnson, M., & Gould, S. (2016). Spice: Semantic propositional image caption evaluation. ArXiv, abs/1607.08822.

Banerjee, S. & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., & Voss, C., editors, Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., teusz Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. ArXiv, abs/2005.14165.

Changpinyo, S., Sharma, P. K., Ding, N., & Soricut, R. (2021). Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3557–3567.

Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A. J., Padlewski, P., Salz, D. M., Goodman, S., Grycner, A., Mustafa, B., Beyer, L., Kolesnikov, A., Puigcerver, J., Ding, N., Rong, K., Akbari, H., Mishra, G., Xue, L., Thapliyal, A. V., Bradbury, J., Kuo, W., Seyedhosseini, M., Jia, C., Ayan, B. K., Riquelme, C., Steiner, A., Angelova, A., Zhai, X., Houlsby, N., & Soricut, R. (2022). Pali: A jointly-scaled multilingual language-image model. ArXiv, abs/2209.06794.

Chen, Y.-C., Li, L., Yu, L., Kholy, A. E., Ahmed, F., Gan, Z., Cheng, Y., & Liu, J. (2019). Uniter: Learning universal image-text representations. ArXiv, abs/1909.11740.

Cho, J., Lei, J., Tan, H., & Bansal, M. (2021). Unifying vision-and-language tasks via text generation. In International Conference on Machine Learning.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In North American Chapter of the Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., & Cao, Y. (2023). Eva: Exploring the limits of masked visual representation learning at scale. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19358–19369.

Gao, L., Lei, Y., Zeng, P., Song, J., Wang, M., & Shen, H. T. (2022). Hierarchical representation network with auxiliary tasks for video captioning and video question answering. IEEE Transactions on Image Processing, 31, 202–215.

Gao, L., Wang, X., Song, J., & Liu, Y. (2020). Fused gru with semantic-temporal attention for video captioning. Neurocomputing, 395, 222–228.

Ghosal, D., Hong, P., Shen, S., Majumder, N., Mihalcea, R., & Poria, S. (2021). CIDER: Commonsense inference for dialogue explanation and reasoning. In Li, H., Levow, G.-A., Yu, Z., Gupta, C., Sisman, B., Cai, S., Vandyke, D., Dethlefs, N., Wu, Y., & Li, J. J., editors, Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 301–313, Singapore and Online. Association for Computational Linguistics.

Goyal, P., Pandey, S., & Jain, K. (2018). Deep learning for natural language processing. New York: Apress.

Guo, J., Li, J., Li, D., Tiong, A. M. H., Li, B. A., Tao, D., & Hoi, S. C. H. (2022). From images to textual prompts: Zero-shot visual question answering with frozen large language models. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10867–10877.

Khurana, K. & Deshpande, U. (2021). Video question-answering techniques, benchmark datasets and evaluation metrics leveraging video captioning: A comprehensive survey. IEEE Access, PP, 1–1.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., & Fei-Fei, L. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision, 123, 32 – 73.

Lei, J., Berg, T. L., & Bansal, M. (2022). Revealing single frame bias for video-and-language learning. arXiv preprint arXiv:2206.03428.

Li, J., Li, D., Savarese, S., & Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In International conference on machine learning, pages 19730–19742. PMLR.

Li, J., Li, D., Xiong, C., & Hoi, S. C. H. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine Learning.

Li, J., Selvaraju, R. R., Gotmare, A. D., Joty, S. R., Xiong, C., & Hoi, S. C. H. (2021). Align before fuse: Vision and language representation learning with momentum distillation. In Neural Information Processing Systems.

Li, X., Yin, X., Li, C., Hu, X., Zhang, P., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., Choi, Y., & Gao, J. (2020). Oscar: Object-semantics aligned pre-training for vision-language tasks. ArXiv, abs/2004.06165.

Lialin, V., Rawls, S., Chan, D., Ghosh, S., Rumshisky, A., & Hamza, W. (2023). Scalable and accurate self-supervised multimodal representation learning without aligned video and text data. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, pages 390–400.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In European Conference on Computer Vision.

Mañas, O., Rodriguez Lopez, P., Ahmadi, S., Nematzadeh, A., Goyal, Y., & Agrawal, A. (2023). MAPL: Parameter-efficient adaptation of unimodal pre-trained models for vision-language few-shot prompting. In Vlachos, A. & Augenstein, I., editors, Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2523–2548, Dubrovnik, Croatia. Association for Computational Linguistics.

Mao, A., Mohri, M., & Zhong, Y. (2023). Cross-entropy loss functions: Theoretical analysis and applications. In International conference on Machine learning, pages 23803–23828. PMLR.

Mkhallati, H., Cioppa, A., Giancola, S., Ghanem, B., & Van Droogenbroeck, M. (2023). Soccernet-caption: Dense video captioning for soccer broadcasts commentaries. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5074–5085.

Olivastri, S., Singh, G., & Cuzzolin, F. (2019). End-to-end video captioning. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pages 0–0.

Ordonez, V., Kulkarni, G., & Berg, T. L. (2011). Im2text: describing images using 1 million captioned photographs. In Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11, page 1143–1151, Red Hook, NY, USA. Curran Associates Inc.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In Isabelle, P., Charniak, E., & Lin, D., editors, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In International Conference on Machine Learning.

Raffel, C., Shazeer, N. M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21, 140:1–140:67.

Rohrbach, M., Qiu, W., Titov, I., Thater, S., Pinkal, M., & Schiele, B. (2013). Translating video content to natural language descriptions. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., & Komatsuzaki, A. (2021). Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114.

Sharma, P., Ding, N., Goodman, S., & Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In Gurevych, I. & Miyao, Y., editors, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2556–2565, Melbourne, Australia. Association for Computational Linguistics.

Shi, Y., Xu, H., Yuan, C., Li, B., Hu, W., & Zha, Z.-J. (2023). Learning video-text aligned representations for video captioning. ACM Trans. Multimedia Comput. Commun. Appl., 19(2).

Tan, H. H. & Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. In Conference on Empirical Methods in Natural Language Processing.

Tang, M., Wang, Z., LIU, Z., Rao, F., Li, D., & Li, X. (2021). Clip4caption: Clip for video caption. In Proceedings of the 29th ACM International Conference on Multimedia, MM '21, page 4858–4862, New York, NY, USA. Association for Computing Machinery.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Neural Information Processing Systems.

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., & Saenko, K. (2014). Translating videos to natural language using deep recurrent neural networks. arXiv preprint arXiv:1412.4729.

Wang, T., Zhang, R., Lu, Z., Zheng, F., Cheng, R., & Luo, P. (2021a). End-to-end dense video captioning with parallel decoding. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 6827–6837.

Wang, W., Bao, H., Dong, L., Bjorck, J., Peng, Z., Liu, Q., Aggarwal, K., Mohammed, O. K., Singhal, S., Som, S., & Wei, F. (2022). Image as a foreign language: Beit pretraining for all vision and vision-language tasks. ArXiv, abs/2208.10442.

Wang, W., Bao, H., Dong, L., & Wei, F. (2021b). Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. ArXiv, abs/2111.02358.

Wang, Z., Yu, J., Yu, A. W., Dai, Z., Tsvetkov, Y., & Cao, Y. (2021c). Simvlm: Simple visual language model pretraining with weak supervision. ArXiv, abs/2108.10904.

Yao, L., Huang, R., Hou, L., Lu, G., Niu, M., Xu, H., Liang, X., Li, Z., Jiang, X., & Xu, C. (2021). Filip: Fine-grained interactive language-image pre-training. ArXiv, abs/2111.07783.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., & Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. Trans. Mach. Learn. Res., 2022.

Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., Choi, Y., & Gao, J. (2021). Vinvl: Making visual representations matter in vision-language models. ArXiv, abs/2101.00529.