NANYANG TECHNOLOGICAL UNIVERSITY

FROM AN IMAGE TO A TEXT DESCRIPTION OF THE IMAGE

Liu Yanli

School of Computer Science and Engineering

2012/2022

NANYANG TECHNOLOGICAL UNIVERSITY

SCSE21-0061

FROM AN IMAGE TO A TEXT DESCRIPTION

OF THE IMAGE

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Computer Science
of the Nanyang Technological University

by

Liu Yanli

School of Computer Science and Engineering

2021/2022

Abstract

Information technology is changing rapidly, multimedia video with its rich information content, diverse presentation, convenient transmission, storage form is rapidly replacing the traditional paper text. The amount of video data is growing in a spurt. In the face of the vast sea of news video, how to quickly and accurately retrieve and store video information has become a pressing problem. Video uses images and sound to convey information. To achieve this purpose, the visual summaries of broadcast news videos can first be recovered by extracting the video's important frames, resulting in a collection of images that is a good representation of the video's visual content. Image captioning is then used to assign relevant descriptions to the extracted keyframes. Meanwhile, the audio of the video is extracted to be processed. Not only the speech content itself but also the background sound indicate the news content. This project implements a fully automated video captioning system designed specifically for broadcast news video. To perform image captioning, the proposed system uses shot-based boundary detection to extract important frames, and a CLIP prefix + GTP2 model is used for image caption. The system's accuracy is measured using the MS COCO dataset, and it's compared to the current state-of-the-art in image captioning. Also presented is a method for evaluating the generated video captions against a set of annotated keyframes.

Acknowledgements

This Final Year Project was completed with the support and guidance of a select few people, who have provided me with the advice and direction that was necessary to me.

I would like to convey my heartfelt thanks to my supervisor Prof Chng Eng Siong, for allowing me to work on this project and for providing me with vital guidance. His dedication to his work and research inspired me to do better and to attempt to meet the high standards that he expects from his students.

I would also like to thank Mr. Kyaw Zin Tun, for his continuous assistance and advice throughout the duration of the project. He has taught me the proper methodologies to carry out the project, and always answered all of my queries patiently. For that, I am extremely grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Since the rise of short video in 2016, the whole short video industry has been in the ascendant. With the steady growth of traditional video platforms such as YouTube and the rapid growth in the number of users of short video platforms such as the first rising giant Tiktok, we have entered the era of video for all. Benefiting from the dividends of the 4G era, video is playing an increasingly important role in information dissemination. In the face of the vast sea of videos, the traditional fast-forward, fast-back, keyword search, and other methods can no longer meet the real needs, how to quickly and accurately retrieve and browse video has become the current urgent problem to be solved.

Video data is rich in content and can be broadly divided into two categories: Information Content, which refers to the semantic content in the video; and Audio Visual Content, which refers to the representation of sound and moving images contained in the video. Video data has a complex structure and relationships. Video data is three-dimensional data with both spatial and temporal attributes. Video data contains both spatial and temporal dimensions. The spatial dimension denotes that each image frame from the video represents a two-dimensional spatial structure. It is challenging to establish a clear structure because the spatial information included in the image is quite complicated. The video's temporal dimension denotes that it is made up of many image frames that are dispersed along the time axis, with each image frame serving as the video stream's unit(Xuemei, Yue, and Fang, 2017).

Video keyframes are an efficient and streamlined form of video presentation. With the help of image captioning, which links computer vision and natural language processing, its task is to give a machine a picture, which is then required to perceive the objects in the picture and even capture the relationships in the picture, and finally, gen-

erate descriptive words for the keyframes of the video. This can help us to quickly get the general information of the video through the presentation of pictures and text. Also, information such as language and background sounds in the video can enrich our extraction of video content.

We want to develop a system that integrates image captioning and sound event detection to extract the information from a video and present them in text. This paper focuses on news videos. In today's image captioning field, an encoder is used to extract image features and a text decoder is used to generate the description sentence. In principle, this demands such a need to connect visual representations and verbal representations. As a result, this type of model requires the collection and annotation of a large number of datasets needed for training, which leads to high costs. In addition, trained visual models are generally good at only one typical task type, and migration to other tasks is costly. Ron Mokadi (2021) proposed a ClipCap model, which uses CLIP encoding as a caption prefix. The image captions are produced after the language model fine-tuning with a basic mapping network. The core aim is to acquire a wide comprehension of visual and textual data by combining it with a pre-trained language model (GPT2). Thus, their approach requires only fairly fast training to produce a qualified caption model. It can efficiently generate meaningful captions for large and diverse datasets requiring no extra annotation or pre-training. Then, we use a Sound event detection system with ASR to extract speech and ambient sound information from the audio of the news video.

1.2 Objectives

The main purpose of this project is to build a video captioning system based on the ClipCap model, which integrates both image and sound information to generate more detailed and accurate captions of broadcast news videos.

The aims of this project are:
1. Image captioning: Examine how existing works of ClipCap for the purpose to generate a text description of an image.

2. Evaluation of generated captions: Evaluate the output of the ClipCap model. The output is evaluated against a standardized dataset to determine its accuracy.

3. Sound event detection with Automatic Speech Recognition: Integrate the existing

works of an SER system with ASR developed by Lee, 2021 for the purpose to determine not only the sound event class but also the onset and offset time of the sound event.

4. Video caption: This is accomplished by splitting a video into keyframes and then feeding these keyframes into the ClipCap model. Meanwhile, the audio of the video is extracted and fed into a sound event detection with an automatic speech recognition system. The output is text descriptions of each keyframe and audio clip.

1.3 Scope

The effective scope of this video captioning system is largely limited to broadcast news videos. This is because the keyframe extraction method used works well with videos that contain numerous frames with varying information.

The image caption model ClipCap is trained under zero-shot learning. The types of sound events detected by the SED system are limited and not trained typically on broadcast news datasets due to a lack of time and resources. The goal of the study is to explore and evaluate a new image captioning model ClipCap and develop a video captioning system integrate image captioning with sound event detection.

1.4 Report Organization

This report consists of six main chapters. An overview of each chapter is described below:

Chapter 1: An introduction of the project, its objectives, and scope.

Chapter 2: Review of past research and related works that appertain the concepts of the project—Video keyframe extraction, image captioning, sound event detection, and datasets.

Chapter 3: The proposed approach to the project is outlined. Introduces the component specifications, the dataset used, tools implemented, and caption evaluation metrics.

Chapter 4: Implementation of the complete video captioning program. Post-processing methods to refine the output, and custom annotation evaluation is discussed.

Chapter 5: Discussion of the results attained. The data is tabulated and compared, and limitations are outlined.

Chapter 6: Conclusion and summary of the project, along with recommendations for future improvement.

Chapter 2

Literature Review

In this chapter, the past works relating to the project are discussed, and the concepts involved in keyframe extraction, image captioning, and sound event detection are explained.

In section 2.1, the report explores different methods for story unit segmentation.

In section 2.2, the architecture of the ClipCap model for image captioning is discussed. Then, the State-of-the-Art Architecture of different image caption models is discussed as well.

In section 2.3, the prediction system of how Sound Event Detection works is discussed.

In section 2.4, different evaluation metrics are introduced for image captioning.

In section 2.5, common datasets used for training of image captioning model are discussed.

2.1 Video Key Frame Extraction

News videos have a certain regularity in their structural pattern. The news video is composed of a series of story units, and the news broadcast is usually preceded by an overview of the news by the presenter, followed by a detailed report from the scene, during which a theme subtitle that summarizes the content of the news appears, usually only one theme subtitle appears for each story unit. Between two different story units, there is a long silence between the host and the live report, and it is mostly accompanied by video cutscenes.

2.1.1 Audio feature-based story unit segmentation

The audio information-based news story unit segmentation method is used to determine the candidate cut points for news story units by extracting the audio in the video and selecting effective audio features like the silence point, transition words, different speakers for audio classification. This method can achieve high accuracy in detecting information such as advertisements, weather forecasts, and news opening and closing credits in news videos(Lu, Zhang, and S. Z. Li, 2003). However, it can usually only be used as an effective aid for news video story unit segmentation.

2.1.2 Visual feature-based story unit segmentation

Many of the news story units consist of a special structure of ”presenter footage + news footage”. A shot boundary detection can be performed to indicate the story boundary(Quénot, Moraru, and Besacier, 2003).
Screen text is another important visual feature that indicates high-level semantic content of the video and provides important clues for retrieval and browsing of multimedia data(). A screen text boxes detection method was proposed by Poignant et al., 2012. The idea is that several successive filters are passed over each frame of the video, filtering out connected parts that do not have a mandatory geometry and keeping only boxes that are stable enough to change over time.

2.1.3 Multimodal features based story unit segmentation

Multimodal feature fusion uses more than one feature from visual features, audio features, and text information to segment the story units of content-based news videos(Dumont and Quénot, 2012). The main idea is to first extract as many effective features as possible to build a framework. Then some a priori rules of news video are used to effectively

perform data fusion of multimodal features. Finally, the boundary point positions of story units are judged by some decision mechanisms.
Compared with using single modal information features for video content structure analysis, the method of using multimodal information features fusion extracts as many effective features as possible, which can effectively overcome the leakage and misdetection brought by using single modal information features, so more researches start to prefer the method of multiple features fusion for news video content structure analysis.

2.2 Image Captioning

Image captioning is the ability of a machine to generate a text description of an image. The recognition of individual objects on images is not difficult for a machine, which can be done by a standard classifier model. However, finding out the relation between these objects and understanding what is happening in a two-dimensional image is a lot more complex for a machine. The human brain can do this quite simply because we can extrapolate from previous experience and can even put ourselves in the shoes of the person in the photo to quickly understand what is happening. However, machines consider an image as a set of pixels only.

2.2.1 ClipCap

The core of the ClipCap model is that it can still produce meaningful subtitles when both the image encoder model of CLIP and the language model GPT-2 are frozen. A lightweight transformer-based mapping network is trained from the CLIP embedding space and the constants given by GPT-2 with the aim of retrieving fixed-length prefixes. Given a certain fixed prefix, the GPT-2 model can be applied to construct subtitles(Mokady, Hertz, and Bermano, 2021). Below is the transformer-based structure of the ClipCap model.

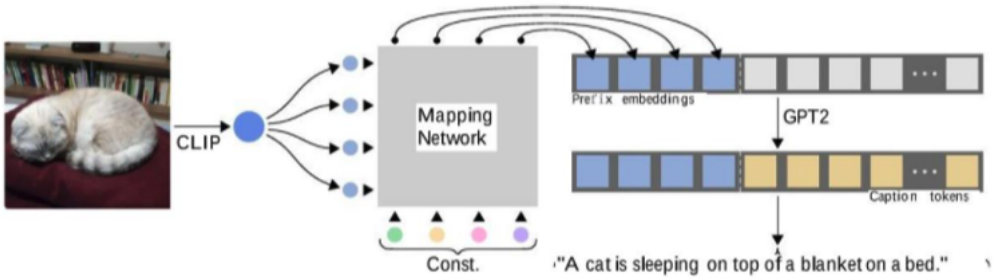

Figure 2.1: ClipCap Architecture

2.2.1.1 CLIP

CLIP is Contrastive Language-image pre-training, which compares the text-image pre-training model. You only need to provide the text description of the Image category to classify images. CLIP relies on 400 million text-to-image pairs collected by OpenAI from the Internet during pre-training. Then, with a zero-shot design similar to GPT-2/3, the CLIP shows superior performance without being optimized directly against the baseline: the robustness gap is reduced by 75%, and performance is comparable to that of the deep residual network ResNet50. That is, CLIP achieves the same precision on the ImageNet dataset compared to the original ResNet50. However, the training sample used is less than ResNet50(Radford, Kim, et al., 2021).

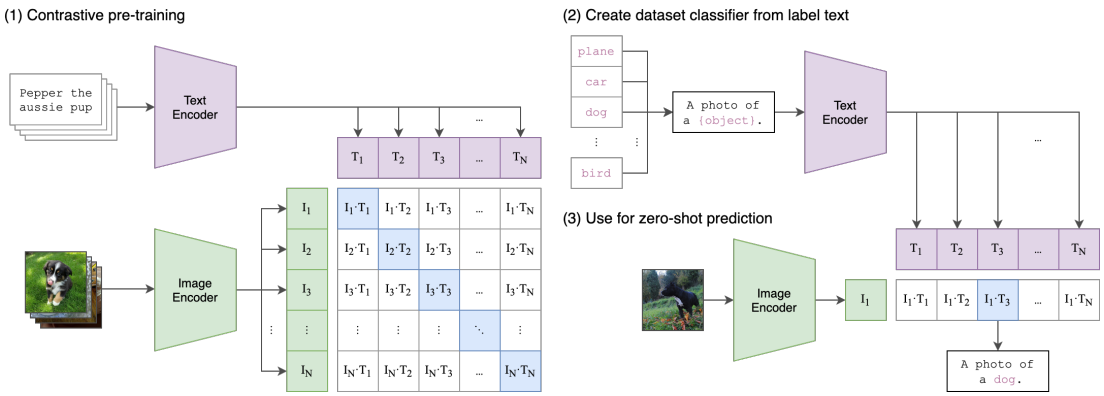

Figure 2.2: CLIP Architecture

2.2.1.1.1 Zero-shot learning

Purely supervised learning has achieved amazing results on many tasks, but its limitation is that it often requires enough samples to train a good enough model, and the classifier trained by cats and dogs can only be used for cats. Dogs are classified,

other species it cannot identify. Zero-shot learning aims to make the model be able to classify categories that it has never seen before to achieve true intelligence of machine(Lampert, Nickisch, and Harmeling, 2009).

The model is trained using the training set data to enable the model on classifying the items in the test set. However, there is no intersection of categories in the training set and categories in the test set. In this period, to make the model valid, a description of the categories is needed to build a bridge across the training set and the test set(Kodirov, Xiang, and Gong, 2017).

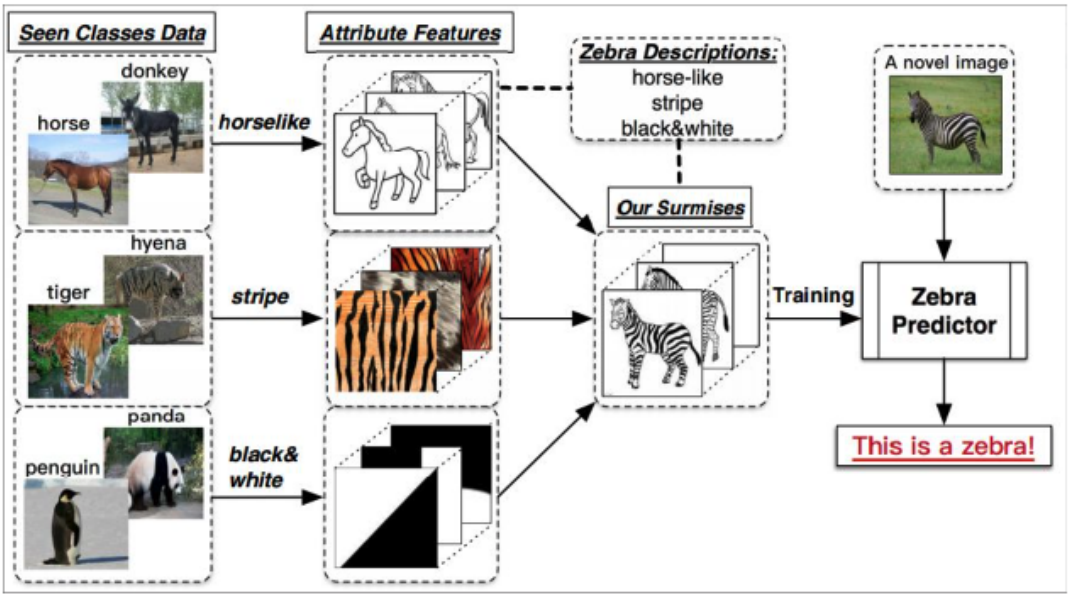

Figure 2.3: zero-shooting learning workflow

2.2.1.1.2 Multimodal model

The CLIP model has double flows, two encoder processing text and image data respectively. The text encoder used is Transformer. The image encoder can be one of the two models, ResNet or Vision Transformer(ViT). If we divide it more finely, there are 5 types of ResNet: Resnet-50, Resnet-101, RN50x4, RN50x16, RN50x64, and 3 types of ViT: VIT-B/32, VIT-B/16, VIT-L/14.
The implementation of CLIP is in the following steps. Firstly, the encoder representation directly linear projects to multi-modal embedding space. Secondly, calculate the cosine similarity between 2 modes to make pairs with N matches have the largest similarity, while the dispatched pairs have the smallest similarity. Then, the symmetrical cross-entropy loss is optimized from these similarities scores. Moreover, the data

enhancement method used is random square crop for resized images(Radford, Kim, et al., 2021).

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 2.4: Pseudo-code of how CLIP model works

2.2.1.1.3 ResNet

ResNet was first proposed by He et al., 2016 to solve the issue of gradient disappearance, explosion, and network degradation generated by increasing the number of network layers. Traditionally, this can be solved by normalized initialization and batch normalization of the data, but such methods are prone to degradation of the network performance, i.e., the error rate increases as the network depth deepens. ResNet can use residual networks to build deep networks and solve such degradation problems and gradient problems.

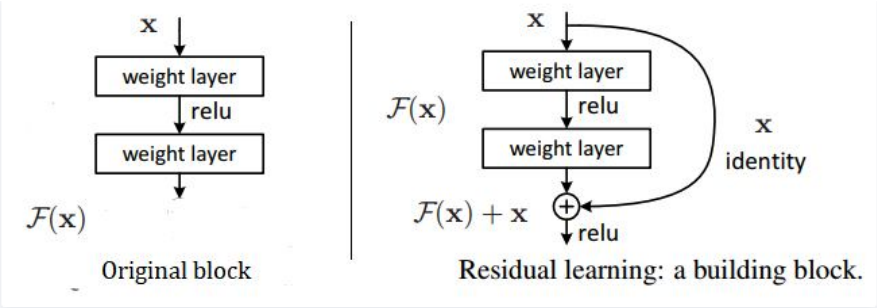

Figure 2.5: Comparison of residual and normal block

2.2.1.1.4 Vision Transformer(ViT)

The architecture of the ViT model is shown below. The image is reconstructed as a series of squashed two-dimensional pieces. Secondly, each of them is linearly embedded to the G dimension with a trainable linear projection. Thirdly, extra learnable position embeddings are added to retain positional information. Fourthly, a learnable embedding is pre-pended to the sequence of embedded patches. Then, the resulting sequence of vectors is fed to a standard Transformer encoder. Lastly, a classification head is attached(Dosovitskiy et al., 2020).

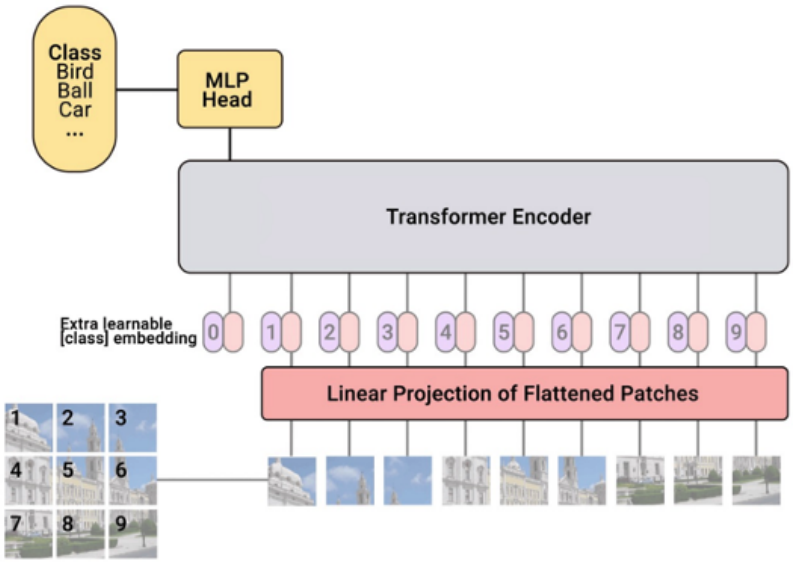

Figure 2.6: The architecture of a Vision Transformer

2.2.1.2 GPT2

GPT2 language model does an amazing job of generating text that exceeds the expectations of current language models(BERT, Transformer-XL, XLNet) in terms of both contextual coherence and emotional expression(Radford, Wu, et al., 2019). The GPT-2 is not an especially new structure regarding the structure of the model only. To be more specific, GPT-2 resembles a Transformer model with only a decoder.

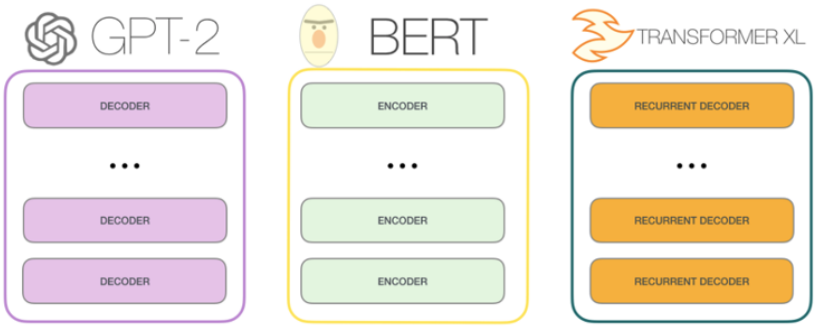

Figure 2.7: GPT2 compared to BERT and Transformer

2.2.1.2.1 Prefix-tuning

Language models can realize the effectiveness of quickly adapting to new tasks by concatenating learned prefixes(X. L. Li and Liang, 2021). Fine-tuning (top) involves maintaining a full model copy for each job and updating all Transformer settings (the red Transformer box). Prefix-tuning (bottom) is a technique that freezes the Transformer parameters while optimizing only the prefix (the red prefix blocks). As a result, we only need to save the prefix for each activity, which makes prefix tuning modular and space-saving. Each vertical block represents a one-time step of transformer activation.

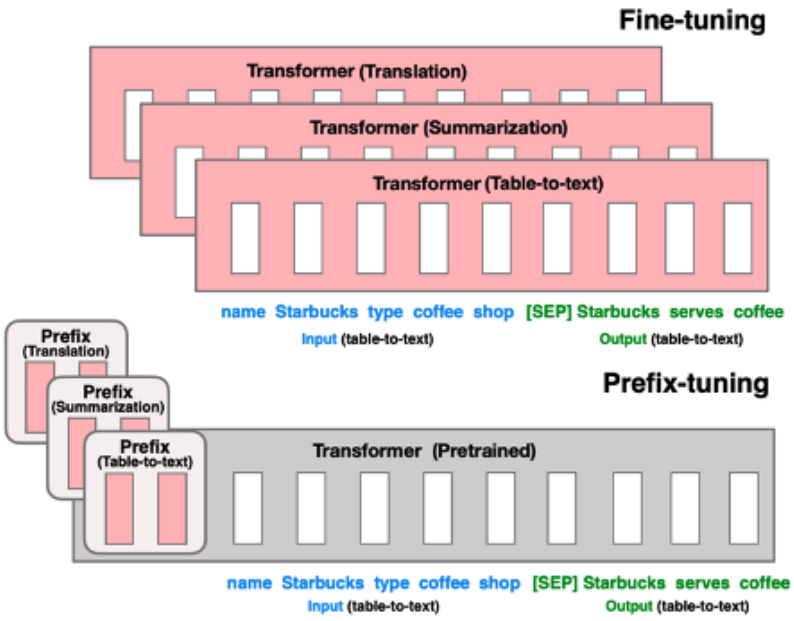

Figure 2.8: prefix-tuning workflow

2.2.1.3 Mapping Network Architecture

A key component of the ClipCap model is the mapping network. It transforms the CLIP embedding into GPT-2 space. The mapping becomes easier when both the language model and the network model are simultaneously fine-tuned. In this scenario, the actual architecture of the mapping network is a basic Multi-Layer Perception(MLP). Since the purpose of the pre-training of the CLIP model is vision-language, we can create realistic and understandable captions even with the simple use of a hidden layer. However, when the language model is frozen, a more expressive transformer design is recommended. For extended sequences, the transformer allows for global attention between input tokens while lowering the number of parameters(Mokady, Hertz, and Bermano, 2021).

2.2.2 State-of-the-Art Architectures

In this section, a summary of representative work in the image captioning area will be introduced according to the categorization of important work and ideas, with a focus on deep learning-based approaches.

2.2.2.1 Encoder-Decoder

The basic Encoder-Decoder framework, Neural Image Caption Generator, for image captioning is proposed for the first time in 2014 by Google(Vinyals et al., 2016). Compared to previous methods that obtained features based on object recognition and attribute detection, It replaces the RNN in the original machine translation model with CNN-based InceptionNet to extract image features. Meanwhile, these image features extracted are passed to the decoder of the RNN. For better long-term memory, RNN can also be replaced with LSTM, GRU, bi-RNN, ESN, etc.

2.2.2.2 Attention Mechanism

The bottleneck of Encoder-Decoder is that the vector to store extracted features is fixed and limited. The accuracy of machine translation decreases as the input sequence length increases(Cho et al., 2014).

People tend to pay attention to information that is useful in viewing images and ignore a lot of useless information. Two attention mechanisms, soft attention, and hard attention are proposed by Xu et al., 2015 to add to the Encoder-Decoder framework. The attention mechanism estimates the probability of each pixel of the feature map and then performs a weighted summation, after which it is sent to an RNN (LSTM) for decoding. The soft attention mechanism is more broadly used in real practice. The article also quantifies the performance of the network in Flickr8k, Flickr30k, MS COCO after adding the attention mechanism.

2.2.2.3 Reinforcement Learning Methods

Reinforcement learning can solve two problems caused by training with cross-entropy loss function: exposure bias and loss-evaluation mismatching. There are four basic elements in RL: state, action, reward, and policy. In image captioning, the image you see and the words that have been generated is ”state”. ”action” is what the next word is generated, and ”return” is a metric such as CIDER. The REINFORCE algorithm is used by Ranzato et al., 2015 for the first time for the image caption and seq2seq problems. Liu et al., 2017 proposed improvements on that by directly using evaluation metrics such as SPICE and CIDEr as rewards to optimize the parameters by strategy gradient.

2.2.2.4 Generative Adversarial Network

The basic architecture of Generative adversarial networks(GAN) consists of a generator and a discriminator. A generator's target is to maximize the distribution of probabilities of the real data in order to make sure that generates false samples that are "fake". A discriminator's training target is to realize the classification of the real and fake data from the generator to improve differentiation of the fake data in training(Pan et al., 2019).

A method for image captioning using conditional GAN was introduced at CVPR by Dai et al., 2017, with the motivation to generate more diverse image captions. The generator uses a traditional encoder-decoder structure to input images to obtain fake captions. The discriminator is LSTM that randomly receives real and fake captions at each step and accepts an image feature that is used to score the caption in order to correctly distinguish real answers from fake ones.

2.3 Sound Event Detection with Automatic Speech Recognition

Sound Event Detection (SED) is the procedure of recognizing sound events in an audio clip, and the timestamps of their onset and offset(Benetos, Stowell, and Plumbley, 2018). Automatic Speech Recognition (ASR) is a text-to-speech conversion method. Lee, 2021 developed a SED system with ASR that is able to analyze clips with long audio durations. Frame-wise prediction pre-processing and post-processing methods are proposed in order to address the issues with the existing SED system. The image below illustrates the structure of the prediction system of SED.

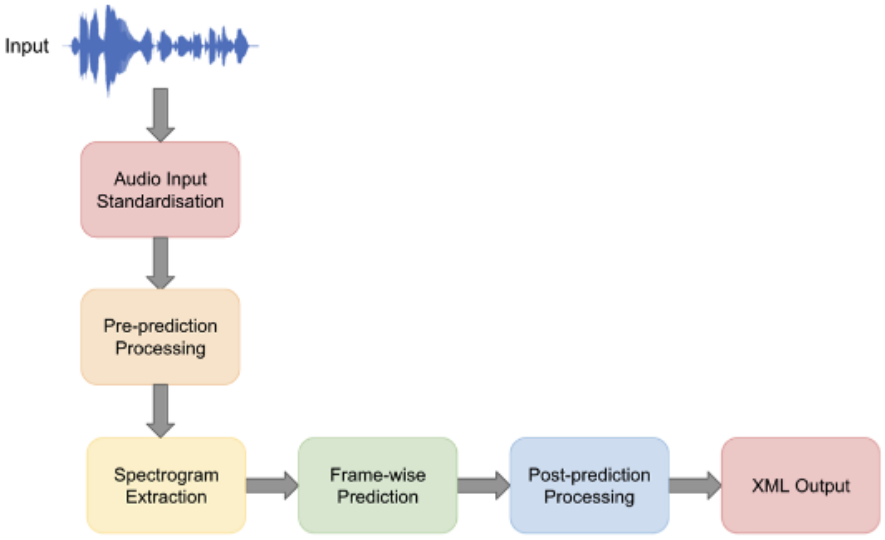

Figure 2.9: the prediction system architecture of a SED system

2.4 Evaluation Metrics

On the leaderboard of the COCO competition, you can see that the commonly used metrics to evaluate are BLEU, Meteor, ROUGE, CIDEr, and SPICE. The first two are for evaluating machine translation, the third is for evaluating automatic summary, and the last two should be customized for CAPTION.

2.4.1 BLEU

IBM suggested Bleu for the evaluation of machine translation projects in 2002, and it was published in ACL with over 10,000 citations(Papineni et al., 2002). The BLEU algorithm is determining how similar two sentences are. It compares the degree of overlap between the candidate translation and the n-gram (in practice, the 4-gram is taken from the unigram) in the reference translation, and the higher the degree of overlap, the higher the quality of the translation. This is an accuracy-only metric, meaning that it cares more about how many n-grams in the candidate translation are correct (i.e., appear in the reference translation) than about the recall (which n-grams in the reference translation do not appear in the candidate translation).
BLEU comes in a variety of forms. According to n-gram, the number of consecutive words for n may be separated into a range of assessment indicators, the most frequent of which are BLEU-1, BLEU-2, BLEU-3, and BLEU-4 four. The lower-order BLEU can assess sentence fluency, whereas the higher-order BLEU can measure word-level correctness.

2.4.2 METEOR

METEOR is based on BLEU with some improvements, aiming at solving some inherent flaws in the BLEU standard. Using WordNet to compute specific sequence matches, synonyms, roots, affixes, and paraphrases, BLEU's efficacy is improved, and it's more relevant to manual discrimination(Banerjee and Lavie, 2005).

2.4.3 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a collection of metrics for comparing the similarity between automatically generated summaries or translations and human-generated reference summaries or translations(C.-Y. Lin, 2004). The difference between ROUGE and BELU is that ROUGE only considers recall, i.e., it does not care whether the translation results are fluent, which is only concerned with whether the translation is accurate: how many n-grams from the reference translation are included in the candidate translation. Four ROUGE methods are proposed in his paper.

2.4.4 CIDEr

The BLEU and the vector space model are combined in CIDEr. It treats each phrase as a document and calculates the cosine angle of the TF-IDF vector (with the exception that the term is an n-gram rather than a word) to determine how similar the candidate and reference sentences are(Vedantam, Lawrence Zitnick, and Parikh, 2015).

2.4.5 SPICE

SPICE encodes objects, properties, and relationships in a caption using a graph-based semantic model. It first parses the caption to be evaluated and the reference captions into syntactic dependency trees using Probabilistic Context-Free Grammar (PCFG) and then maps the dependency trees into scene graphs using a rule-based approach. It first parses the caption to be evaluated and the reference captions into syntactic dependencies trees using the Probabilistic Context-Free Grammar (PCFG) dependency parser and then maps the dependency trees into scene graphs using a rule-based approach. Finally, the F-score values of objects, attributes, and relationships in the caption to be evaluated are calculated(Anderson, Fernando, et al., 2016).

2.5 Datasets

COCO-captions, Conceptual Captions, and NoCaps datasets are discussed and used for the training and evaluation of the CLIP model.

2.5.1 MS COCO Dataset

The full name of COCO is Common Objects in COntext. The contributor to this dataset is the Microsoft team. It is available for object detection, segmentation, and captioning tasks because the dataset has three relative types of annotation. The images in this dataset are divided into three sets: training, validation, and test. The images in COCO are gathered by scouring Flickr for 80 object categories and for a wide range of scene types using Amazon's Mechanical Turk(AMT)(Chen et al., 2015).

To date, this is the biggest dataset with semantic segmentation that offers 80 categories, over 330,000 images, out of which 200,000 are labeled and the whole dataset has over 1.5 million individuals(T.-Y. Lin et al., 2014).

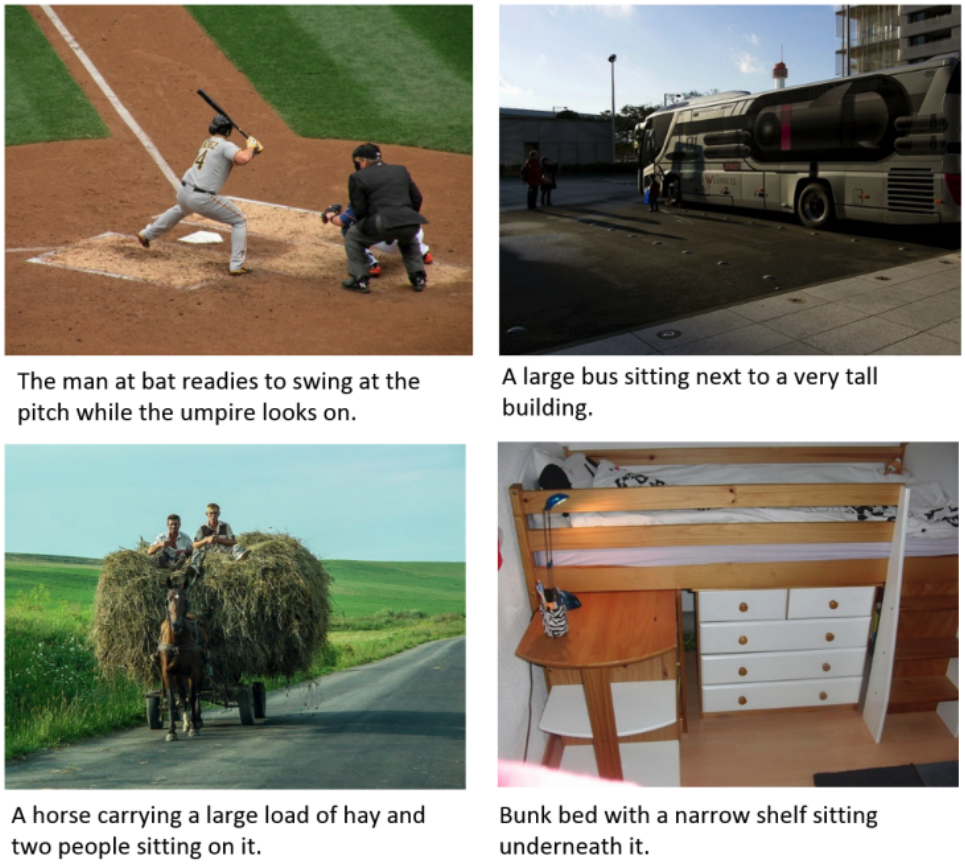

Figure 2.10: Example of images and it's description from COCO dataset

2.5.2 Conceptual Captions

Conceptual Captions (CC) dataset is a dataset containing image URLs and caption pairs, which is used for the training and evaluation of image caption systems in machine learning. The dataset with an order of magnitude more images than the MS-COCO dataset consists of two versions of approximately 3.3 million images (CC3M) and 12 million images (CC12M). Weak correlation descriptions are automatically collected from the Alt-text HTML attribute associated with web images through a simple filtering procedure called Flume pipeline(Chambers et al., 2010).
The images in the Conceptual Captions dataset and their original descriptions are derived from the Web and therefore represent a broader range of styles than many current image caption annotations such as the MS-COCO images. However, images from Conceptual Captions are not always available, as the dataset provides image URLs (P. Sharma et al., 2018).

Alt-text: A Pakistani worker helps to clear the debris from the Taj Mahal Hotel November 7, 2005 in Balakot, Pakistan.

Conceptual Captions: a worker helps to clear the debris.

Alt-text: Musician Justin Timberlake performs at the 2017 Pilgrimage Music & Cultural Festival on September 23, 2017 in Franklin, Tennessee.

Conceptual Captions: pop artist performs at the festival in a city.

Figure 2.11: Example of images and it's description from CC dataset

2.5.3 NoCaps

On datasets with restricted visual ideas and huge quantities of paired picture-caption training data, image captioning algorithms have shown outstanding results. In order for models to learn a wide variety of visual concepts, it is best to learn from less supervision. NoCaps aims to assess whether the model can accurately describe new categories of objects in the test image without corresponding training data(Agrawal et al., 2019).

NoCaps leverages alternative data sources, such as target detection, to enable the model to describe objects that do not exist in the header corpus of the training set. These objects with object detection annotations, but without a title corpus, are called novel objects, and the images describing those containing the new objects are called novel object captioning(Anderson, Gould, and Johnson, 2018).

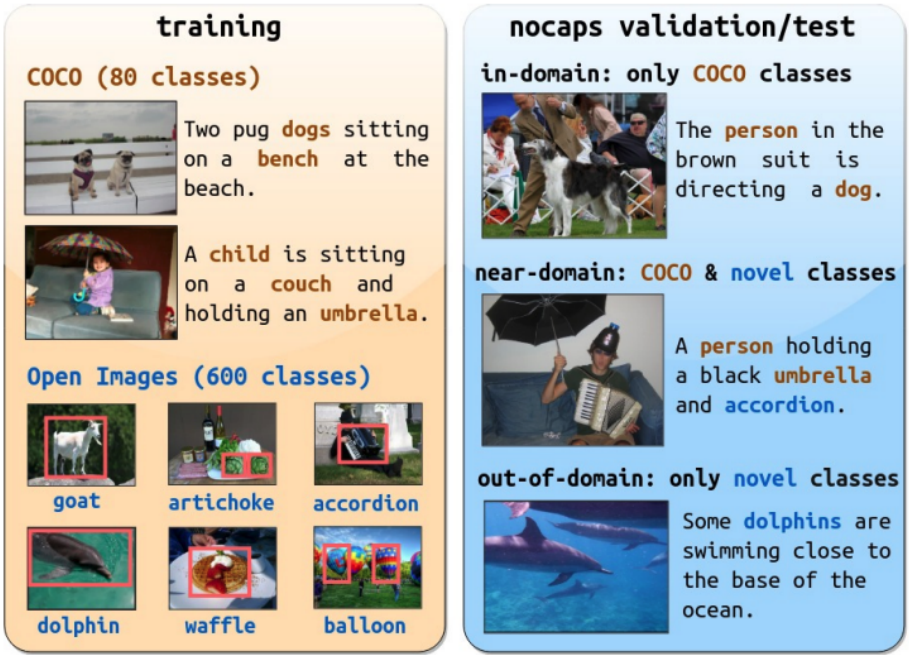

Figure 2.12: Example of images and it's description from NoCaps dataset

The main goal of NoCaps is to separate "how to recognize objects" from "how to talk about objects." Learn to generate syntactically correct titles using COCO image title data, and learn more visual concepts using a large open image detection dataset.

- The training set consists of COCO captions 2017(Chen et al., 2015) image-caption pairs (118,000 Images, 80 target categories) and Open Images V4(Krasin et al., 2016) target detection training set (1,700,000 Images with bounding boxes, 600 target categories).
- The validation set contains 4,500 Images, with 11 captions(10 reference citations and 1 human baseline) per image, from the Open Images V4 validation set.
- The test set contains 10,600 Images, with 11 captions(10 reference citations and 1 human baseline) per image, derived from the Open Images V4 test set.

The COCO dataset has fewer categories than the Open Images dataset. There are no training Caption equivalents for the almost 400 target classes found in the test Images. NoCaps assessment is separated into three subgroups to enable a more fine-grained analysis: intradomain, near-domain, and out-of-domain. In COCO, domains explain how items are related to categories.

Chapter 3

Project Methodology and Specification

This chapter outlines the project methodology and specifies the software that was used.

Section 3.1 discusses the proposed system architecture and proposed methodology of the project.

In section 3.2-3.5, the specification for each component and environment setup is discussed.

Section 3.4 briefly introduces the various tools and libraries that were employed in this project.

3.1 System Architecture and Proposed Methodologies

The video captioning system mainly consists of a scene detection system, an image captioning system, a sound event detection system to generate the captions of a video considering both visual and auditory information, and an evaluation system to evaluate the results. The flowchart below is the architecture of the video captioning system.

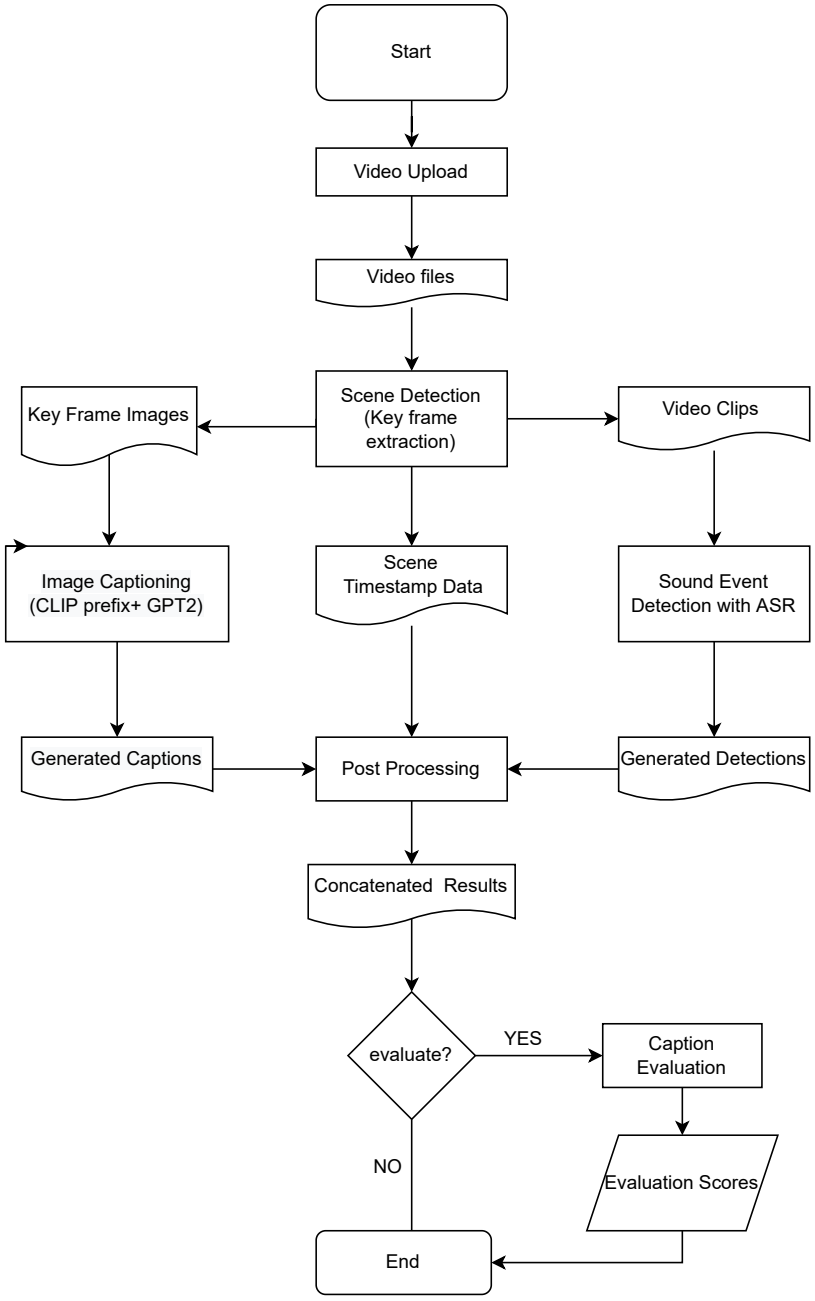

Figure 3.1: System Architecture of video captioning system

Firstly, the video is spitted into different story units based on shot changes. One

keyframe is extracted for every story unit and image captioning is implemented on it. Meanwhile, the audio of each video clip is extracted and passed to the sound event detection section.

In the image captioning area, ClipCap is a lightweight captioning model that uses CLIP as a pre-trained frozen model to extract visual characteristics and GPT-2 as a pre-trained frozen language model to create captions. The picture features will be extracted using the CLIP model's image encoder. The use of a mapping network on the CLIP embedding generates a prefix for each caption. It's a set of embeddings with a defined size that's related to caption embedding. The prefix will be given to the language model GPT2, which has been fine-tuned during the mapping network training procedure. For the sound event detection section, the Sound Event Detection with ASR system developed by Zhen, 2021 is used to detect human and emergency Sounds in each video clip.

For data manipulation and analysis, the pandas library is used for post-processing. Evaluation metrics like BLEU-1 to BLEU-4, CIDEr, SPICE, METEOR, and ROUGE-L will be used to evaluate the caption results.

3.2 Key Frame Extraction Specifications

pyscenedetect is a python command-line utility that analyzes video, detects scene changes, and cuts video using OpenCV. The resulting timecode can then be used to break the video into separate segments using another program (e.g. mkvmerge, ffmpeg), and also to generate a frame-by-frame analysis for the video. PySceneDetect uses two main detection algorithms: detect-threshold and detect-content. The former one is to compare each frame to a predetermined black level, which is beneficial for identifying fading from black to black. Although the processing speed is slower, the latter one looks for changes in content by comparing each frame, which is important for identifying quick changeover between video sequences.
For this project, the detect content algorithm of pyscenedetect is used to split the video into several scenes. The keyframe that represents the scene is the middle frame of the video clip.

3.3 Image Captioning Specifications

The image captioning is realized based on a ClipCap model largely from the paper, "ClipCap: CLIP Prefix for Image Captioning (Ron et al., 2021). The images are encoded using an image encoder of the pre-trained CLIP model(Radford, Kim, et al., 2021). Then, the encoded visual information generated is used as a prefix to guide the future process of generating text using another model. The language model used is GPT-2, which is a smaller and more intuitive version of the powerful OpenAI model.

The author proposed two different mapping networks to connect the CLIP prefix and GPT-2: Multi-Layer Perceptron(MLP) mapping network or Transformer mapping network. For the MLP mapping network, pre-trained models on COCO and Conceptual Captions are provided. The language model GPT-2 is fine-tuning during the training process. However, for the Transformers mapping network, the language model is fixed, which means the number of parameters will not increase compared to the training with fine-tuning. The COCO pre-training models of the Transformer are provided by the author.

3.4 Sound Event Detection Specifications

For this project, the sound event detection system with ASR are adapted from the repository, https://github.com/yazdayy/sound-event-detection (Zhen, 2021). The author provides eight pre-trained models that trained under different data augmentation methods, processing types as below.

Model	Data Augmentation	Processing Type	Threshold Type	Segment-based Error Rate	Segment-based F1-score
Cnn-9 + Gru + Attention	SpecAugment + Mixup	None	Non-optimised	0.555	0.624
Cnn-9 + Gru + Attention	SpecAugment + Mixup	None	Optimised	0.601	0.635
Cnn-9 + Gru + Attention	SpecAugment + Mixup	Frame-wise averaging	Non-optimised	0.557	0.618
Cnn-9 + Gru + Attention	SpecAugment + Mixup	Frame-wise averaging	Optimised	0.574	0.639
Cnn-9 + Transformer + Attention	SpecAugment + Mixup	None	Non-optimised	0.552	0.628
Cnn-9 + Transformer + Attention	SpecAugment + Mixup	None	Optimised	0.561	0.648
Cnn-9 + Transformer + Attention	SpecAugment + Mixup	Frame-wise averaging	Non-optimised	0.549	0.622
Cnn-9 + Transformer + Attention	SpecAugment + Mixup	Frame-wise averaging	Optimised	**0.550**	**0.647**

Figure 3.2: Pre-trained models and their performance

There are 19 types of human sounds and 5 types of emergency sounds may be detected:

Human sounds:

1. Applause, 2. Breathing, 3. Chatter, 4. Child_speech_kid_speaking, 5. Cheering, 6. Clapping, 7. Conversation, 8. Cough, 9. Crowd, 10. Crying_sobbing, 11. Female_speech_woman_speaking, 12. Laughter, 13. Male_speech_man_speaking, 14. Run, 15. Screaming, 16. Shout, 17. Sneeze, 18. Walk_footsteps, 19. Whispering

Emergency sounds:

1. Air_horn_truck_horn, 2. Car_alarm, 3. Emergency_vehicle, 4. Explosion, 5. Gunshot_gunfire, 6. Siren

Figure 3.3: Different sound event types

The SED system is integrated with DeepSpeech Automatic Speech Recognition(ASR). DeepSpeech is an open-source voice-to-text engine that employs a model which is based on Baidu's deep speech research papers and approaches(Hannun et al., 2014). ASR is activated when the event detected is either man, female, or child speaking.

3.5 Caption Evaluation Specifications

For the evaluation section, the evaluation model for various unsupervised automated metrics for Natural Language Generation proposed by S. Sharma et al., 2017 can evaluate the image captioning result. The model can help calculate the results of common evaluation metrics including BLEU-1 to BLEU-4, CIDEr, SPICE, METEOR, and ROUGE-L, SkipThoughtsCosineSimilarity, EmbeddingAverageCosineSimilarity, VectorExtremaCosineSimilarity, GreedyMatchingScore. The GitHub for nlg-eval is https://github.com/Maluuba/nlg-eval.

3.6 Tools and Technologies

This section summarises the tools and packages used in the development of the video captioning system.

3.6.1 Environment Setup

The tables below show an overview of the environment setup used for development.

Hardware	8GB RAM, 2.3 GHz Intel Core i5
Operating System	macOS Mojave 10.14.6
Development IDE	PyCharm

Table 3.1: Development Environment Setup

3.6.2 Language and Libraries

In this section, the various tools, languages, and libraries that were used in this project are explored. The table below summarizes the technologies and their component associations.

Component	**Technologies**
Key Frame Extraction	Python, OpenCV, FFmpeg,PySceneDetect
Image Captioning	Python, PyTorch, PIL, Scikit-image, Matplotlib, Transformers
Sound Event Detection	Librosa, SpeechRecognition, sed_eval

Table 3.2: Tools and Technologies Used

3.6.2.1 Python

Python was the main programming language used for this project, as the existing library was practical for developing a video captioning system.

3.6.2.2 OpenCv-Python

OpenCV(Bradski and Kaehler, 2008) is a cross-platform computer vision library that operates on Linux, Windows, Android, and Mac OS and is released under the BSD license (open source). It's small and fast, with only a few C functions and C++ classes, and it provides interfaces to Python, Ruby, MATLAB, and other languages, allowing it to implement a wide range of image processing and computer vision methods. The Python binding library OpenCV-Python is used to handle computer vision challenges. When installing OpenCV-Python, the new version is very limited because some classic algorithms have been copyrighted, so use version 3.4.9.31.

3.6.2.3 PySceneDetect

PySceneDetect is a Python library and command-line utility for analyzing video and detecting scene transitions or clips. When using the split-video command, PySceneDetect interfaces with other programs (e.g. mkvmerge, ffmpeg) to automatically divide the video into individual pieces. Frame-by-frame analysis, also known as statistics files, can be created for movies to aid in the determination of suitable thresholds or the detection of trends for specific films or other analysis methods.

3.6.2.4 FFmpeg

FFmpeg is a collection of open-source tools which could be used to record, transcode, and transmit digital music and video. It has very powerful features including video capture functions, video format conversion, video capture, adding watermarks to videos, and so on. Applications can utilize libavcodec, libavutil, libavformat, libavfilter, libavdevice, libswscale, and libswresample module from this library. End-users can also utilize ffmpeg, ffplay, and ffprobe to do transcoding and playback.

3.6.2.5 PyTorch

PyTorch(Paszke et al., 2019) is a Python machine learning package that is open-source. It's utilized in a variety of applications, including natural language processing. It was created by the Facebook Artificial Intelligence Research Group, and probabilistic programming is done with YouTuber's Pyro software. PyTorch is closely similar to Face-

book's Torch framework, which is built on Lua. PyTorch includes deployments for mobile and embedded frameworks.

3.6.2.6 PIL

The PIL (Python Image Library) library is a third-party library for the Python language, which needs to be installed via the pip tool. The PIL library is mainly used for image archiving and image processing.

3.6.2.7 scikit-image

Scikit-image, shortened as skimage, is a python scripting language-based digital image processing program. The skimage package consists of a number of sub-modules, each providing different functionality. The io module is used to read and display images.

3.6.2.8 Transformers

Huggingface Transformers(Wolf et al., 2020) is based on an open-source transformer-based model structure to provide a pre-training language library, which supports Pytorch, Tensorflow2.0, and supports the interconversion of the two frameworks. The framework supports a variety of the latest NLP pre-trained language models, users can quickly make model calls, and supports model further pretraining and downstream tasks fine-tuning. GPT2Tokenizer, GPT2LMHeadModel models are imported when training the mapping network of the ClipCap image captioning model.

3.6.2.8.1 Matplotlib

Matplotlib(Hunter, 2007) is a multi-platform data visualization package that uses NumPy arrays as its foundation. Using an object-oriented API includes charting, graphing, picture, and other visualization drawing features. Attention processes in image caption models are visualized using Matplotlib.

3.6.2.9 Librosa

Librosa is a Python library for analyzing and manipulating audio and music, which includes methods like time-frequency processing, feature extraction, and generating sound visuals, all of which may be utilized in the sound events detection system.

3.6.2.10 SpeechRecognition

SpeechRecognition is a voice recognition library that supports a variety of engines and APIs, both online and offline. The SpeechRecognition library is extremely flexible as it accommodates several major speech APIs. Among them is the Google Web Speech API, which supports default API keys hardcoded into the SpeechRecognition library without registration, making SpeechRecognition the best choice for writing Python programs because of its flexibility and ease of use.

3.6.2.10.1 sed_eval

sed_eval is an open-source python toolbox for evaluating sound event detection systems in a consistent and transparent manner.

3.6.2.11 h5py

In python programming, if you want to read and write large data that exceeds memory, you can introduce the h5py library to use. h5py(Group et al., 1997) is a module for the Python language to manipulate HDF5. h5py files are containers for two types of objects, datasets, and groups.

3.6.2.12 Pandas

Pandas is a NumPy-based open-source Python library that is extensively utilized for data analysis and data cleaning. Pandas works well with data from a variety of different sources, such as Excel tables, CSV files, SQL databases, and even with data stored on web pages, Pandas is based on Numpy and is often used together with Numpy and matplotlib. The captions concatenation of keyframe timestamps, image caption output, and SED output is based on this.

Chapter 4

Implementation

This section elaborates on the development and implementation process of the video captioning system one component by one component.

Section 4.1 discusses the implementation of keyframe extraction using the command line interface provided by pySceneDetect.

In section 4.2, the specification of how to use these two ClipCap models with different mapping techniques to do image captioning is discussed.

In section 4.3, the specification of how to implement the prediction system of the SED with ASR system on the video clips is discussed.

Section 4.4 briefly illustrates the post-processing on the generated image captioning and sound events detected.

Section 4.5 introduces how to run the video captioning system as a whole.

Section 4.6 focuses on the evaluation of the generated results of the video captioning system.

4.1 Key Frame Extraction

We can use the command line interface provided by pySceneDetect package to realize the extraction of key frame from the video. The example command is shown below.

```
$ scenedetect --min-scene-len 2s --input video.mp4 detect-content
   --threshold 29 list-scenes save-images -n 1 -o ./frames split-video
   -o ./clips
```

- –min-scene-len: Set the minimum duration of the scene when splitting.
- –input: The path of the input video file.
- detect-content: Choose the content detection algorithm to detect the scene boundary.
- –threshold: Set the threshold value of the delta_HSV frame metric. Only when the value is exceeded, a new scene is detected.
- list-scenes: Print the list of scenes and export to a CSV file.
- save-images: Save images from each scene of the video.

  -n: The number of images to be saved in each scene. If the value n is not specified, the default images saved will be the start frame, end frame, and middle frame.

  -o: The path to save the key images.
- split-video: Segment the video based on the detected scene boundary

  -o: The path to save the video clips after splitting.

After execution of this code line. All video clips generated after splitting, which are based on the content detection algorithm will be stored in the folder named clips. The middle frame of each scene is stored in another folder called frames. A result CSV file that contained the start, end, duration time information is a form similar to below.

Timecode List:	00:22.1	00:24.3	00:29.9	00:32.8	00:35.7	00:38.7	00:40.9	00:44.1	00:47.1
Scene Number	Start Frame	Start Timecode	Start Time (seconds)	End Frame	End Timecode	End Time (seconds)	Length (frames)	Length (timecode)	Length (seconds)
1	0	00:00.0	0	662	00:22.1	22.089	662	00:22.1	22.089
2	662	00:22.1	22.089	728	00:24.3	24.291	66	00:02.2	2.202
3	728	00:24.3	24.291	897	00:29.9	29.93	169	00:05.6	5.639
4	897	00:29.9	29.93	984	00:32.8	32.833	87	00:02.9	2.903
5	984	00:32.8	32.833	1070	00:35.7	35.702	86	00:02.9	2.87
6	1070	00:35.7	35.702	1161	00:38.7	38.739	91	00:03.0	3.036
7	1161	00:38.7	38.739	1225	00:40.9	40.874	64	00:02.1	2.135
8	1225	00:40.9	40.874	1322	00:44.1	44.111	97	00:03.2	3.237
9	1322	00:44.1	44.111	1412	00:47.1	47.114	90	00:03.0	3.003
10	1412	00:47.1	47.114	1485	00:49.6	49.55	73	00:02.4	2.436
11	1485	00:49.6	49.55	1595	00:53.2	53.22	110	00:03.7	3.67
12	1595	00:53.2	53.22	1727	00:57.6	57.624	132	00:04.4	4.404
13	1727	00:57.6	57.624	1852	01:01.8	61.795	125	00:04.2	4.171
14	1852	01:01.8	61.795	1973	01:05.8	65.832	121	00:04.0	4.037
15	1973	01:05.8	65.832	2054	01:08.5	68.535	81	00:02.7	2.703
16	2054	01:08.5	68.535	2454	01:21.9	81.882	400	00:13.3	13.347
17	2454	01:21.9	81.882	2697	01:30.0	89.99	243	00:08.1	8.108
18	2697	01:30.0	89.99	2793	01:33.2	93.193	96	00:03.2	3.203
19	2793	01:33.2	93.193	3029	01:41.1	101.068	236	00:07.9	7.875
20	3029	01:41.1	101.068	3347	01:51.7	111.678	318	00:10.6	10.611
21	3347	01:51.7	111.678	3485	01:56.3	116.283	138	00:04.6	4.605
22	3485	01:56.3	116.283	3707	02:03.7	123.69	222	00:07.4	7.407
23	3707	02:03.7	123.69	3769	02:05.8	125.759	62	00:02.1	2.069
24	3769	02:05.8	125.759	3922	02:10.9	130.864	153	00:05.1	5.105
25	3922	02:10.9	130.864	4010	02:13.8	133.8	88	00:02.9	2.936
26	4010	02:13.8	133.8	4255	02:22.0	141.975	245	00:08.2	8.175
27	4255	02:22.0	141.975	4359	02:25.4	145.445	104	00:03.5	3.47
28	4359	02:25.4	145.445	4474	02:29.3	149.282	115	00:03.8	3.837
29	4474	02:29.3	149.282	4652	02:35.2	155.222	178	00:05.9	5.939
30	4652	02:35.2	155.222	4832	02:41.2	161.228	180	00:06.0	6.006
31	4832	02:41.2	161.228	5368	02:59.1	179.112	536	00:17.9	17.885
32	5368	02:59.1	179.112	6019	03:20.8	200.834	651	00:21.7	21.722
33	6019	03:20.8	200.834	6523	03:37.7	217.651	504	00:16.8	16.817

Figure 4.1: A .CSV file that stores the timestamps for each story unit

4.2 Image Captioning

After we get the keyframes of the video, image captioning will be implemented on each keyframe. There are two kinds of ClipCap models. One is MLP mapping network, the other is transformer mapping network. For MLP mapping networks with a single hidden layer, the prefix length is set to be 10 after experiments. 8 multi-head self-attention layers with 8 heads each are used for the transformer mapping network, with the CLIP embedding set to 10 constant tokens(Mokady, Hertz, and Bermano, 2021).

To use the MLP mapping network model, the command line is shown below.

```
$ python caption.py --model coco --beam_size 5
```

- –model: There are two pre-trained models available for the MLP mapping network. One is trained on the MS COCO dataset and the other is trained on the Conceptual Caption dataset. The value of –model can either be ‘coco’ or ‘conceptual’.

- –beam_size: You can choose whether to use beam search or not and the size of the beam. The default value of the beam size used for beam search is set to be 5. Set value 0 to the beam_size to indicate that beam search is not used.

To use the transformer mapping network model, the command line is shown as below. The pre-trained is trained on the MS COCO dataset.

```
$ python caption_transformer.py --beam_size 5
```

- –beam_size: You can choose whether to use beam search or not and the size of the beam. The default value is set to be 5. Set value 0 to the beam_size to indicate that beam search is not used.

The generated captions for each keyframe of a video are stored in one .csv file as below.

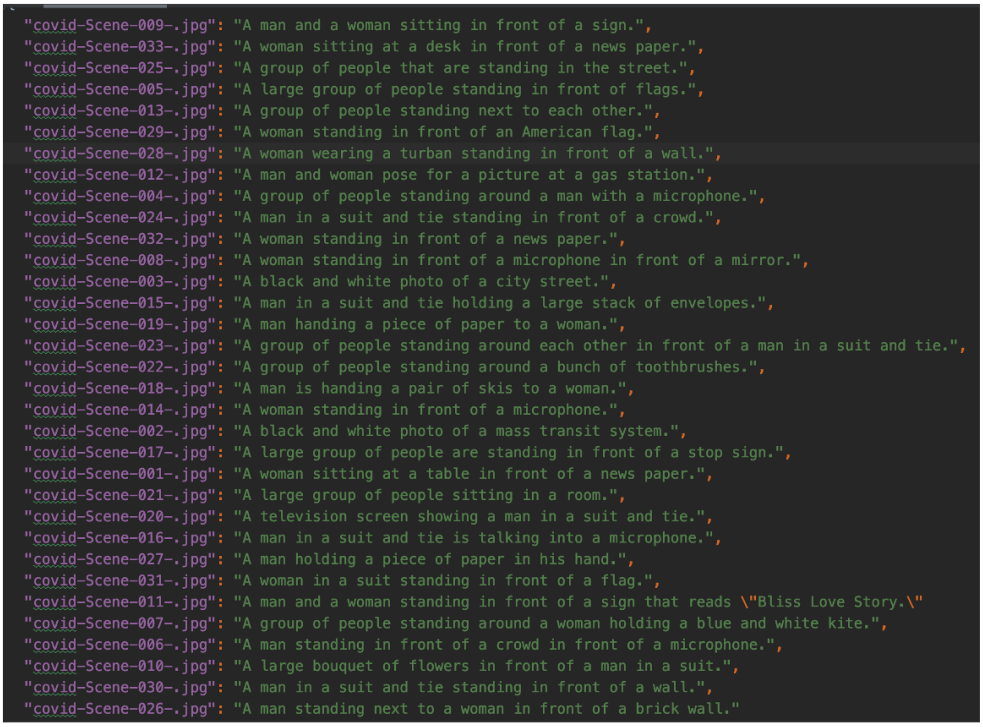

Figure 4.2: The .json file stores image captioning output

4.3 Sound Event Detection with Automated Speech Recognition

```
(base) LILIdeMacBook-Pro:~ lili$ python pytorch/predict.py predict_asr --input_d
ir='/Users/lili/PycharmProjects/SEDwithASR/upload' --workspace='/Users/lili/Pych
armProjects/SEDwithASR' --filename='main_strong' --holdout_fold=1 --model_type='
Cnn_9layers_Gru_FrameAtt' --loss_type='clip_bce' --augmentation='specaugment_mix
up' --batch_size=32 --feature_type='logmel' --cuda --sample_duration=5 --overlap
 --overlap_value=1 --sed_thresholds --language='eng'
```

Figure 4.3: Commadline example to run SED system

- –input_dir: The location of the folder where all these video clips are.

- –workspace: The location of the project folder of SEDwithASR system.
- –filename: Set the value to be ”main” for purely weakly-labelled system or ”mainstrong”combined weakly-labelled and strongly-labelled system.
- –model_type: The pretrained model type.
- –augmentation: The data augmentation methods used to train the model. The value can be one of the eight values: ’none’, ’spec_mixupaugment’, ’timeshift’, ’mixup’, ’timeshift_mixup’, ’specaugment_timeshift_mixup’, ’specaugment_mixup’, ’specaugment_timeshift’.

The Sound Event Detection is implemented on every video clip for each scene. The output .xml file includes the start time, duration, event detected and the human speech recognition data in the following structure.

Figure 4.4: The output .xml file that stores SED with ASR results

4.4 Captions Concatenation

The information of timestamps of each scene after slitting, the image captioning generated for each keyframe, and the sound events detected in each scene are supposed to be concatenated together.

Firstly, four columns ’Scene Number’, ’Start Time (seconds)’, ’Length (seconds)’,’End Time (seconds)’ from the ”vidoe-Scenes.csv” file are extracted and saved in a dataframe ”scenes_split_df” as below.

```
scenes_df = pd.read_csv(csvfilename,header=1)
scenes_split_df = scenes_df[['Scene Number', 'Start Time (seconds)',
    'Length (seconds)','End Time (seconds)']].copy()
```

The structure of the dataframe is as below:

	Scene Number	Start Time (seconds)	Length (seconds)	End Time (seconds)
0	1	0.00	21.08	21.08
1	2	21.08	2.60	23.68
2	3	23.68	3.00	26.68
3	4	26.68	3.00	29.68
4	5	29.68	3.00	32.68

Figure 4.5: The extracted timestamps from 'idoe-Scenes.csv'

Secondly, the image captions stored in "prediction.json" are extracted and concatenate dataframe according to the frame index. Meanwhile, the location of the video clip where each scene the key frame belongs to is also concatenated to the dataframe for further use.

```
# Read in image caption data
current_directory = "/Users/lili/PycharmProjects/image_caption"
path_to_file = os.path.join(current_directory, "prediction.json")
with open(path_to_file) as mydata:
    prediction_dict = json.load(mydata)
# Formatting image caption data
prediction_df = pd.DataFrame(list(prediction_dict.items()), columns =
    ['Frame', 'Caption'])
prediction_df.sort_values(by=['Frame'], inplace=True, ascending=True)
prediction_df.reset_index(inplace=True, drop = True)
# Concatenating timestamps with image captions
scenes_split_df['Caption'] = pd.Series(prediction_df['Caption'])
scenes_split_df.columns = ['frame#', 'stime', 'dur(s)', 'etime','image
    caption']
# The location of video clips
folder_loc = "/Users/lili/PycharmProjects/image_caption/clips"
clipfiles = [folder_loc+f for f in os.listdir(folder_loc)]
clipfiles.sort()
```

```
scenes_split_df['clip location'] = clipfiles
```

The structure of the dataframe is as below:

	frame#	stime	dur(s)	etime	image caption	clip location
0	1	0.00	21.08	21.08	A woman standing in front of an air conditioner.	/Users/lili/PycharmProjects/image_caption/clip...
1	2	21.08	2.60	23.68	A large jetliner sitting on top of an airport ...	/Users/lili/PycharmProjects/image_caption/clip...
2	3	23.68	3.00	26.68	A large white and blue airplane is on the runway.	/Users/lili/PycharmProjects/image_caption/clip...
3	4	26.68	3.00	29.68	A large white airplane sitting on top of an ai...	/Users/lili/PycharmProjects/image_caption/clip...
4	5	29.68	3.00	32.68	A picture of an airplane on the tarmac.	/Users/lili/PycharmProjects/image_caption/clip...

Figure 4.6: The dataframe that stores the caption results after image caption

Then, we access the folder of the video clips based on the clip location column. The audio of the video is extracted as a .wav file. After the sound event detection, the events detected are extracted from the generated .xml file. We iterate the row of the data frame and access the .xml file of the scene based on the frame index and save them as a new column ”sound event” to the data frame.

```
# Convert .mp4 to .wav
for filename in clipfiles:
    actual_filename = filename[:-4]
    if(filename.endswith(".mp4")):
        print("ffmpeg -i {0} {1}.wav".format(filename, actual_filename))
    else:
        continue
# Create new column to store sound events
scenes_split_df["event"] = np.nan
tree=et.parse('/Users/lili/PycharmProjects/image_caption/SEDwithASR/predict_results/cov
root=tree.getroot()
event_set = set([])
for SoundSegment in list(root[0]):
    event_set.add(SoundSegment.text)
prefix =
    "/Users/lili/PycharmProjects/image_caption/SEDwithASR/predict_results/covid-Scene-"
for i in scenes_split_df["frame#"]:
    loc = prefix + str(i).zfill(3) +".xml"
    tree=et.parse(loc)
    root=tree.getroot()
    event_set = set([])
    for SoundSegment in list(root[0]):
        event_set.add(SoundSegment.text)
```

```
scenes_split_df['event'] = scenes_split_df['event'].astype('object')
scenes_split_df.at[i-1, "event"]=event_set
```

The final structure of the dataframe is as below:

	frame#	stime	dur(s)	etime	image caption	clip location	event
0	1	0.00	21.08	21.08	A woman standing in front of an air conditioner.	/Users/lili/PycharmProjects/image_caption/clip...	{Female_speech_woman_speaking, Male_speech_man...
1	2	21.08	2.60	23.68	A large jetliner sitting on top of an airport ...	/Users/lili/PycharmProjects/image_caption/clip...	{Female_speech_woman_speaking, Breathing}
2	3	23.68	3.00	26.68	A large white and blue airplane is on the runway.	/Users/lili/PycharmProjects/image_caption/clip...	{Male_speech_man_speaking}
3	4	26.68	3.00	29.68	A large white airplane sitting on top of an ai...	/Users/lili/PycharmProjects/image_caption/clip...	{Conversation, Female_speech_woman_speaking}
4	5	29.68	3.00	32.68	A picture of an airplane on the tarmac.	/Users/lili/PycharmProjects/image_caption/clip...	{Conversation, Female_speech_woman_speaking}

Figure 4.7: The dataframe that stores the caption results

Lastly, the dataframe is output as a json file ”vidname-OUTPUT.json” as below.


```
[
  {
    "frame#": 1,
    "stime": 0.0,
    "dur(s)": 21.08,
    "etime": 21.08,
    "caption": "A couple of women standing next to a bunch of airplanes.",
    "clip location": "/Users/lili/PycharmProjects/image_caption/clips/singapore_airplane-Scene-001.mp4",
    "event": [
      "Breathing",
      "Female_speech_woman_speaking",
      "Male_speech_man_speaking",
      "Conversation"
    ]
  },
  {
    "frame#": 2,
    "stime": 21.08,
    "dur(s)": 2.6,
    "etime": 23.68,
    "caption": "A large jetliner sitting on top of an airport tarmac.",
    "clip location": "/Users/lili/PycharmProjects/image_caption/clips/singapore_airplane-Scene-002.mp4",
    "event": ["Breathing", "Female_speech_woman_speaking"]
  },
```


Figure 4.8: The output form of generated captioning

4.5 Complete Video Captioning

The previous sections can be run at once and generate the JSON file that stores both the image captioning and sound event detection of the video. Below is an example command to run the caption.py file.

```
$ python caption_video.py -i covid.mp4 -m mlp --keepframes
```

- –input, -i: The path of the input video.

- –model, -m: The type of CLIPCAP model used. The value can be either ”mlp” or ”transformer”.
- –keepframes, -k: Keep the keyframe images after image captioning or not.

4.6 Evaluation of Captions

The evaluation of the image captioning results is based on custom reference captions. Ground truth captions should be referenced for video annotations. The annotations for every keyframe of a video are stored in a text file named ’reference-n.txt’. The output of the image captions stored in the text file ’output.txt’ is then evaluated against these custom references. The number of custom references has no limits.
Firstly, make sure the Python dependencies of nlg-eval are installed. If not, run:

```
pip install git+https://github.com/Maluuba/nlg-eval.git@master
```

Then set up the nlg-eval package

```
nlg-eval --setup
```

Run the following command to get the evaluation metrics:

```
$ nlg-eval --hypothesis=examples/hyp.txt
    --references=examples/ref1.txt --references=examples/ref2.txt
```

- –hypothesis: The path to the files where stores the generated captions.
- –references: The path to the reference files.

The output is like this:

```
Bleu_1: 0.333333
Bleu_2: 0.202031
Bleu_3: 0.140205
Bleu_4: 0.091190
METEOR: 0.188980
ROUGE_L: 0.325681
CIDEr: 0.270803
SkipThoughtsCosineSimilarity: 0.285378
EmbeddingAverageCosineSimilarity: 0.855676
EmbeddingAverageCosineSimilairty: 0.855676
VectorExtremaCosineSimilarity: 0.506320
GreedyMatchingScore: 0.652763
```

Figure 4.9: The output form of evaluation metrics

Chapter 5

Discussion of Results

We took video covid.mp4 as an example. The video is splitted to 33 story units and 33 key frames are extracted as below.

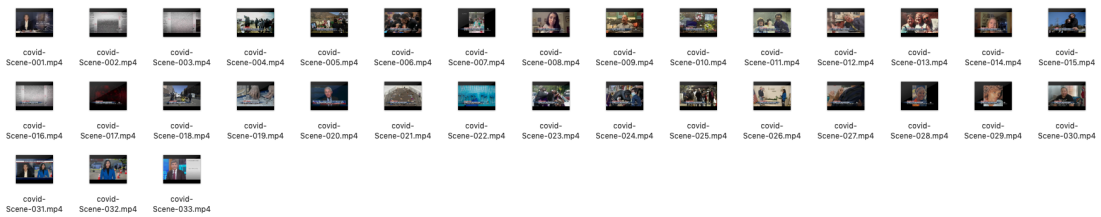

Figure 5.1: Story units splitted from covid.mp4

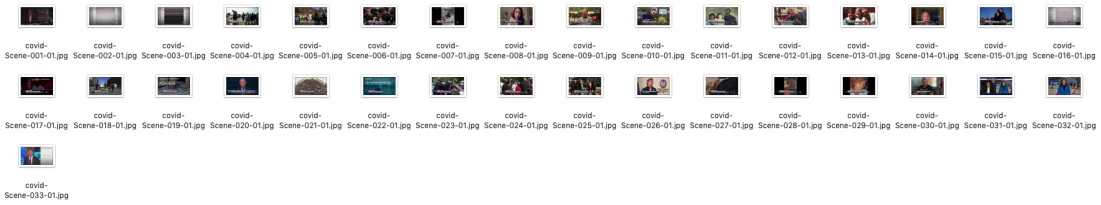

Figure 5.2: Key frames extracted from covid.mp4

The output file after image captioning and sound event detection is as below:

```
[
  {
    "frame#": 1,
    "stime": 0.0,
    "dur(s)": 21.08,
    "etime": 21.08,
    "caption": "A couple of women standing next to a bunch of airplanes.",
    "clip location": "/Users/lili/PycharmProjects/image_caption/clips/singapore_airplane-Scene-001.mp4",
    "event": [
      "Breathing",
      "Female_speech_woman_speaking",
      "Male_speech_man_speaking",
      "Conversation"
    ]
  },
  {
    "frame#": 2,
    "stime": 21.08,
    "dur(s)": 2.6,
    "etime": 23.68,
    "caption": "A large jetliner sitting on top of an airport tarmac.",
    "clip location": "/Users/lili/PycharmProjects/image_caption/clips/singapore_airplane-Scene-002.mp4",
    "event": ["Breathing", "Female_speech_woman_speaking"]
  },
```


Figure 5.3: Part of the output captioning file .mp4

The table below displays the evaluation scores of the sample covid.mp4 against the custom dataset for models with MLP+fine-tuning mapping networks or Transformer-based mapping networks using different beam width sizes. The best scores in each evaluation metric are highlighted in bold.

Beam Size	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
k = 1	0.333333	0.198645	0.127141	0.079675	0.181439	0.315039	0.275280
k = 2	0.351351	0.230590	0.156703	**0.099848**	**0.199613**	0.350947	**0.333740**
k = 3	**0.357143**	**0.240772**	**0.162279**	0.097911	0.198384	**0.353249**	0.299886
k = 4	0.335917	0.211185	0.145127	0.092841	0.186361	0.324710	0.270150
k = 5	0.330792	0.201288	0.138827	0.090037	0.187664	0.325831	0.265784

Table 5.1: Evaluation of COCO pretrained model(MLP+fine-tuning) with Different Beam Sizes

Beam Size	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
k = 1	0.110831	0.024677	0.000000	0.000000	0.078621	0.140782	0.013635
k = 2	0.142119	**0.063361**	**0.033478**	**0.018998**	0.081095	0.174045	**0.061769**
k = 3	0.153465	0.045478	0.023045	0.014153	**0.082249**	0.164971	0.054459
k = 4	0.142544	0.036714	0.015119	0.000002	0.076763	0.162922	0.027655
k = 5	**0.161731**	0.052806	0.019553	0.000002	0.081644	**0.176387**	0.031158

Table 5.2: Evaluation of Conceptual Caption pretrained model(MLP+fine-tuning) with Different Beam Sizes

Beam Size	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
k = 1	0.336182	0.183928	**0.115558**	0.065468	**0.172072**	**0.322668**	**0.222295**
k = 2	**0.341808**	**0.184592**	0.112393	0.063929	0.171827	0.320943	0.206352
k = 3	0.316804	0.180667	0.115445	**0.073472**	0.168861	0.311614	0.211641
k = 4	0.305085	0.163131	0.094039	0.050536	0.163957	0.308821	0.191658
k = 5	0.304469	0.167645	0.098734	0.052213	0.168423	0.312887	0.198670

Table 5.3: Evaluation of COCO pre-trained model(Transformer) with Different Beam Sizes

From the table above, we can find the for covid.mp4 video, COCO pre-trained model with the Transformer mapping network generated the most accurate captions. In general, the model pre-trained on the COCO dataset works better than the model pre-trained on the Conceptual Captions dataset.
The table below shows the quantitative evaluation of two types of ClipCap models on different datasets(Mokady, Hertz, and Bermano, 2021).

(*A*) **Conceptual Captions**

Model	ROUGE-L ↑	CIDEr ↑	SPICE ↑	#Params (M) ↓	Training Time ↓
VLP	24.35	77.57	16.59	115	1200h (V100)
Ours; MLP + GPT2 tuning	**26.71**	**87.26**	**18.5**	156	80h (GTX1080)
Ours; Transformer	25.12	71.82	16.07	**43**	**72h** (GTX1080)

(*B*) **nocaps**

Model	in-domain		near-domain		out-of-domain		Overall		Params↓	Time↓
	CIDEr↑	SPICE ↑	CIDEr	SPICE	CIDEr	SPICE	CIDEr	SPICE		
BUTD [4]	74.3	11.5	56.9	10.3	30.1	8.1	54.3	10.1	52	960h
Oscar [19]	79.6	**12.3**	66.1	**11.5**	45.3	**9.7**	63.8	**11.2**	135	74h
Ours; MLP + GPT2 tuning	79.73	12.2	**67.69**	11.26	**49.35**	**9.7**	65.7	11.1	156	7h
Ours; Transformer	**84.85**	12.14	66.82	10.92	49.14	9.57	**65.83**	10.86	**43**	**6h**

(*C*) **COCO**

Model	B@4 ↑	METEOR ↑	CIDEr ↑	SPICE ↑	#Params (M) ↓	Training Time ↓
BUTD [4]	36.2	27.0	113.5	20.3	52	960h (M40)
VLP [47]	36.5	28.4	117.7	21.3	115	48h (V100)
Oscar [19]	**36.58**	**30.4**	**124.12**	**23.17**	135	74h (V100)
Ours; Transformer	33.53	27.45	113.08	21.05	**43**	**6h** (GTX1080)
Ours; MLP + GPT2 tuning	32.15	27.1	108.35	20.12	156	7h (GTX1080)

Figure 5.4: Pre-trained models and their performance

From the table above, we can see that the training time decreased a lot compared to the state-of-art architectures. The MLP mapping network model with GPT-2 fine-tuning has an outstanding performance on the Conceptual Captions dataset. The Transformer mapping network model has the relatively same performance as Oscar on the NoCaps dataset but the training time is faster a lot.

5.1 Limitations

Some limitations of the project implementation are as follows.

- The threshold value varies for different videos. The accuracy of the division of the story unit cannot be confirmed.
- The image captioning model fails to produce accurate captions for some frames.
- The sound event detection system fails to produce accurate captions for some audio clips.
- The evaluation of captions from both image captioning and sound event detection is difficult to implement.

Chapter 6

Conclusion and Future Work

This chapter concludes the report with a summary and some recommendations for future improvements.

6.1 Conclusion

In this project, a video caption system that is able to generate image captions on the keyframe and detect sound events of each story unit is developed. For image caption, a simple model, ClipCap based on CLIP is used, which seems to be part of a new image captioning paradigm. The objective is to focus on utilizing current models while just training a small mapping network. Rather than learning new semantic entities, this strategy simply learns to adjust the pre-trained model's current semantic knowledge towards the style of the target dataset(Mokady, Hertz, and Bermano, 2021). The sound event detection with ASR is also implemented in this system to get more information about a video.

6.2 Future Work

The keyframe extraction method used in this project is only based on shot boundaries and the middle frame of each story unit is the keyframe. However, this middle frame may not always reflect the main content of the shot, and the extracted keyframes may not be representative. Nowadays, many scene segmentation systems based on multimodel features designed for news video have been proposed such as (Dumont and Quénot, 2012).

For image captioning, the ClipCap model is based on the CLIP prefix and GTP-2 language model. The pre-trained model should be typically trained on a video dataset. Moreover, the sound events detected can only be classified into 19 types of human sounds and 5 types of emergency sounds, which is limited. The SED system should be

trained on a dataset consisting of only news audios instead of a wide range of audios in the future. Meaningful sound types for news videos should be defined and trained on the news video dataset.

The video captioning system designed in this project is based on the image caption of one keyframe, which lacks context information of adjacent frames. The encoder techniques in the case of video caption are similar to those used in the image caption, with the exception that the characteristics retrieved by video caption vary with time. Furthermore, a 3D feature extraction method was proposed by Tran et al., 2015, which entails combining every frame of the video in the channel dimension and convolving it in 3D. The idea of which is to combine each frame of the video in the channel dimension and convolve it in 3D. Its key objective is to obtain the interrelationships among different frames of the video.

Bibliography

Xuemei, Wang, Jiang Yue, and Wang Fang (2017). "The study of subtitle translation based on multi-hierarchy semantic segmentation and extraction in digital video". In: *Humanities Social Sci.* 5.2, pp. 91–96.

Lee, Yan Zhen (2021). *Sound event detection with human and emergency sounds*.

Lu, Lie, Hong-Jiang Zhang, and Stan Z Li (2003). "Content-based audio classification and segmentation by using support vector machines". In: *Multimedia systems* 8.6, pp. 482–492.

Quénot, Georges, Daniel Moraru, and Laurent Besacier (2003). "CLIPS at TRECvid: Shot boundary detection and feature detection". In: *TRECVID 2003 Workshop Notebook Papers*. Citeseer.

Poignant, Johann et al. (2012). "From text detection in videos to person identification". In: *2012 IEEE International Conference on Multimedia and Expo*. IEEE, pp. 854–859.

Dumont, Emilie and Georges Quénot (2012). "Automatic story segmentation for tv news video using multiple modalities". In: *International journal of digital multimedia broadcasting* 2012.

Mokady, Ron, Amir Hertz, and Amit H Bermano (2021). "Clipcap: Clip prefix for image captioning". In: *arXiv preprint arXiv:2111.09734*.

Radford, Alec, Jong Wook Kim, et al. (2021). "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR, pp. 8748–8763.

Lampert, Christoph H, Hannes Nickisch, and Stefan Harmeling (2009). "Learning to detect unseen object classes by between-class attribute transfer". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE, pp. 951–958.

Kodirov, Elyor, Tao Xiang, and Shaogang Gong (2017). "Semantic autoencoder for zero-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3174–3183.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Dosovitskiy, Alexey et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.

Radford, Alec, Jeffrey Wu, et al. (2019). “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8, p. 9.

Li, Xiang Lisa and Percy Liang (2021). “Prefix-tuning: Optimizing continuous prompts for generation”. In: *arXiv preprint arXiv:2101.00190*.

Vinyals, Oriol et al. (2016). “Show and tell: Lessons learned from the 2015 mscoco image captioning challenge”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.4, pp. 652–663.

Cho, Kyunghyun et al. (2014). “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078*.

Xu, Kelvin et al. (2015). “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning*. PMLR, pp. 2048–2057.

Ranzato, Marc’Aurelio et al. (2015). “Sequence level training with recurrent neural networks”. In: *arXiv preprint arXiv:1511.06732*.

Liu, Siqi et al. (2017). “Improved image captioning via policy gradient optimization of spider”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 873–881.

Pan, Zhaoqing et al. (2019). “Recent progress on generative adversarial networks (GANs): A survey”. In: *IEEE Access* 7, pp. 36322–36333.

Dai, Bo et al. (2017). “Towards diverse and natural image descriptions via a conditional gan”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2970–2979.

Benetos, Emmanouil, Dan Stowell, and Mark D. Plumbley (2018). “Approaches to Complex Sound Scene Analysis”. In: *Computational Analysis of Sound Scenes and Events*. Ed. by Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis. Cham: Springer International Publishing, pp. 215–242. ISBN: 978-3-319-63450-0. DOI: 10.1007/978-3-319-63450-0_8. URL: https://doi.org/10.1007/978-3-319-63450-0_8.

Papineni, Kishore et al. (2002). “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318.

Banerjee, Satanjeev and Alon Lavie (2005). “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments”. In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Lin, Chin-Yew (2004). “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*, pp. 74–81.

Vedantam, Ramakrishna, C Lawrence Zitnick, and Devi Parikh (2015). “Cider: Consensus-based image description evaluation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575.

Anderson, Peter, Basura Fernando, et al. (2016). “Spice: Semantic propositional image caption evaluation”. In: *European conference on computer vision*. Springer, pp. 382–398.

Chen, Xinlei et al. (2015). “Microsoft coco captions: Data collection and evaluation server”. In: *arXiv preprint arXiv:1504.00325*.

Lin, Tsung-Yi et al. (2014). “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer, pp. 740–755.

Chambers, Craig et al. (2010). “FlumeJava: Easy, Efficient Data-Parallel Pipelines”. In: *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI ’10. Toronto, Ontario, Canada: Association for Computing Machinery, pp. 363–375. ISBN: 9781450300193. DOI: 10.1145/1806596.1806638. URL: https://doi.org/10.1145/1806596.1806638.

Sharma, Piyush et al. (July 2018). “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 2556–2565. DOI: 10.18653/v1/P18-1238. URL: https://aclanthology.org/P18-1238.

Agrawal, Harsh et al. (Oct. 2019). “nocaps: novel object captioning at scale”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. DOI: 10.1109/iccv.2019.00904. URL: http://dx.doi.org/10.1109/ICCV.2019.00904.

Anderson, Peter, Stephen Gould, and Mark Johnson (2018). *Partially-Supervised Image Captioning*. arXiv: 1806.06004 [cs.CV].

Krasin, Ivan et al. (Jan. 2016). *OpenImages: A public dataset for large-scale multi-label and multi-class image classification.*

Hannun, Awni et al. (2014). “Deep speech: Scaling up end-to-end speech recognition”. In: *arXiv preprint arXiv:1412.5567*.

Sharma, Shikhar et al. (2017). “Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation”. In: *arXiv preprint arXiv:1706.09799*.

Bradski, Gary and Adrian Kaehler (2008). *Learning OpenCV: Computer vision with the OpenCV library*. ” O’Reilly Media, Inc.”.

Paszke, Adam et al. (2019). “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32.

Wolf, Thomas et al. (2020). “Transformers: State-of-the-art natural language processing”. In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45.

Hunter, John D (2007). “Matplotlib: A 2D graphics environment”. In: *Computing in science & engineering* 9.03, pp. 90–95.

Group, HDF et al. (1997). “Hierarchical data format version 5”. In.

Tran, Du et al. (2015). “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497.