2024 Doctoral Thesis

Study of Interpretable and Efficient Video Understanding

Chubu University

Graduate School of Engineering

Department of Computer Science

Yuzhi Shi

Supervisor Takayoshi Yamashita

Video understanding is a critical field within computer vision that aims to comprehend the content and context of video data. It involves various tasks such as action recognition, action detection, and video captioning. These tasks are vital for various applications, including video editing, anomaly detection, and autonomous driving. For example, in anomaly detection, video understanding enables the identification of suspicious activities or objects, enhancing security measures. This technology is also helpful for recognizing pedestrians, vehicles, and traffic signs, ensuring safe navigation in autonomous driving systems. Since the traditional methods are limited by processing time, deep learning models have emerged as powerful tools for exploring video content. These models can analyze and process complex video data, leading to improved performance of video understanding tasks.

To deploy various video understanding models into real-world applications, two main challenges have been identified in this thesis: interpretability and efficiency. The interpretability of predictions poses a significant issue since people cannot trust a model that cannot be explained. This lack of clear reason for predictions hinders the practical application of video understanding models, as people might feel confused about the output from models. Additionally, the sheer volume of video data presents an efficiency challenge. Although deep-learning models can explore potential relationships buried in complex video data, they need some time to predict a correct result. Generally, a model with better performance contains more parameters and needs more time to handle inputs. It is an important issue for applications requiring performance and speed. To address this issue, improving the efficiency of video understanding algorithms is crucial, particularly for real-time applications where quick decision-making is essential, such as video anomaly detection. Via enhancing interpretability and efficiency, video understanding models can provide more accessible and practical solutions for a wide range of applications and make our lives more convenient.

In this research, several novel methods are proposed to address the two challenges in video understanding. (i) Firstly, the use of an attention mechanism is introduced to enhance interpretability in Chapter 3. Via leveraging attention mechanisms, the temporal information in videos can be efficiently explored. The important frames in videos are highlighted according to the attention scores to help us to analyze the reason of predicted class labels. (ii) Secondly, the issue of inefficiency caused by the presence of similar frames in videos is addressed in Chapter 4. The frame saliency weighting module that identifies the salient frames and enhances efficiency by focusing on keyframes is proposed. Additionally, the frame saliency module can utilize saliency scores to present the importance of frames and interpret the predictions of detection models. (iii) Thirdly, the limitation of score-based interpretation methods is observed. It is the inability to compare importance

scores across different videos directly. To overcome this challenge, video captions are introduced to represent video content semantically and it is employed for video anomaly detection. (iv) Fourth, since normal video captioning models do not contain much common sense and cannot answer questions from people, it limits its application. Therefore, pre-trained Large Language Models (LLMs) are employed to provide better interpretability via communication and reduce incorrect video captions. However, since the cost of fine-tuning large models is difficult to cover, therefore the knowledge selection is proposed to apply related knowledge in the specific domain.

In future work, decreasing the cost of applying large models and evaluating the proposed method on various video understanding tasks will be continually researched for a large-scale interpretable video understanding model.

Contents

Figure Contents

Table Contents

Chapter 1

Introduction

This chapter offers insight into the background and motivations behind this research and outlines the thesis structure.

1.1 Background

In contemporary society, the consumption of video content has become ubiquitous due to the proliferation of digital media platforms and advances in web technology. The ability to understand and interpret visual narratives has become an essential skill as people view endless streams of video ranging from entertainment to educational content. We obtain this ability by viewing large amounts of video content, which takes a lot of time. Therefore, we need a technology that can extract important information from all the video content around the globe when needed. Video understanding is the tool that can help us. It encompasses various tasks, including action understanding, natural language processing, sound understanding, visual understanding, etc. The core of video understanding is to comprehensively analyze and interpret the visual and semantic information delivered through the medium of video. This multifaceted process includes extracting semantics from each frame, recognizing objects, detecting actions, understanding spatial and temporal relationships, and discerning contextual clues embedded in the visual narrative.

Recent advances in computer vision technology have played a key role in realizing automatic video understanding models. These models employ various algorithms to understand and analyze video content. Object recognition and detection algorithms enable the models to recognize objects and predict their category labels in each video frame. Action detection algorithms analyze temporal frame sequences to detect specific interactions between humans and objects. Moreover, video understanding explores semantic information in video data to generate video descriptions or answer questions related to video content. By training on vast amounts of annotated video datasets, video understanding models continuously improve their performance, leading to the development of more helpful applications.

Video understanding holds significant implications for various domains, including entertainment, education, healthcare, and security. In the entertainment domain, the personalized video recommendation function leverages video understanding techniques to analyze user preferences and viewing patterns, thereby enhancing the user experience by providing relevant content recommendations. Additionally, video understanding models are useful to create interactive and adaptive learning environments in the education field. In such environments, educational videos are tailored to the specific needs and learning styles of individual students. In addition, video analysis techniques facilitate the interpretation of medical imaging data, helping clinicians in diagnosis, treatment planning, and patient monitoring. In the domain of security, video understanding approaches play a crucial role in detecting and analyzing suspicious activities, identifying objects and individuals of interest to ensure public safety.

While the technological advancements in video understanding have been remarkable, there are still a number of challenges that hinder the application of video understanding models. A significant challenge is the inherent complexity and variability of video data, which encompasses diverse visual content and dynamic scenes. This makes deep learning models the preferred solution for video understanding tasks. However, for real-time applications such as video anomaly detection, deep learning models may not be able to output timely predictions. Furthermore, the large amount of video data generated every day poses scalability and computational challenges for video understanding algorithms, necessitating the development of efficient processing and analysis techniques. In addition, many important domains require interpretation of prediction results, especially in security and healthcare. We need a reasonable interpretation of each prediction to help us understand and analyze the predictions of the model. In order to speed up the prediction of the model and to interpret the prediction results rationally, many researchers have proposed many effective algorithms through extensive research.

As the consumption of video content continues to grow rapidly, the need for advanced video understanding models becomes more pressing. In order to promote the implementation of video understanding technology, I focus on two key issues that affect its deployment: efficiency and interpretability, and use them as the motivation for our research.

1.2 Motivation

There are two motivations behind this research on video understanding.

- The interpretability of predictions.

Interpretability aims to help people fully understand the model's prediction results. It enables models to provide convincing explanations of predictions in a way that is intuitively understandable to humans.

- The efficient comprehension of video content.

  Efficiency aims for real-time processing. By focusing on the content of important frames, the model's understanding of the overall content of the video is promoted, thereby reducing the size and amount of computation of the model and improving computational efficiency.

Interpretability and efficiency are critical requirements not only in the field of video understanding, but also for other AI models. I believe that the methods proposed in this thesis will contribute to the development and application of AI models.

1.2.1 Interpretability

Interpretability is rooted in the need to bridge the gap between the complex inner workings of artificial intelligence (AI) models and the need for transparency and accountability in video analysis tasks. Transparency and accountability are essential for a variety of applications. For example, automatic video editing relies on the interaction between AI models and humans. It is necessary for AI models to edit video content based on human guidance. Consequently, the lack of explainability of prediction outcomes represents a significant barrier to the deployment of AI models. One solution to this problem is to visualize the judgment basis of the model. Grad-CAM [84] uses pixel-based scores to show the classification basis of the AI model. Similarly, using frame-based scores can also show the influence of each frame on the prediction results, thereby helping humans to understand the judgment basis of the model. However, this method can only calculate the importance of frames within the same video, and it is difficult to use it in long videos. Another more intuitive solution is to use text to generate content similar to explanatory text. This method is more aligned with human comprehension than the score-based approach and enables direct comparison of diverse video content. More comprehensive interpretability requires continuous dialogue between humans and models. Through dialogue, it is possible to gain insight into the underlying rationale of the model and comprehend the logical structure of its decision-making process. This thesis proposes score-based, text-based, and communication-based methods to enhance the interpretability and transparency of the video understanding model.

1.2.2 Efficiency

With the growth of digital video content, the demand for efficient video understanding solutions has never been greater. Due to the presence of similar frames in the video, this adds redundant information and confuses the AI models. This makes video representation learning difficult. The frame saliency-based method is proposed to learn the importance of each frame and improve video representation by paying more attention to important frames. The method makes video understanding tasks easy and reduces model size and computational complexity. Improved efficiency leads to faster inference times, resulting in real-time or near real-time video analysis, which is critical for application implementation. For example, anomaly detection applications need to detect anomalous actions as soon as possible to stop the anomalies from occurring. Therefore, the pursuit of efficiency in video understanding models represents a significant step towards empowering AI models widespread adoption across various domains.

1.3 Structure of the thesis

As shown in Figure 1.1, this thesis is structured as follows:

Chapter 1 introduces the background of the research and the two motivations, interpretability and efficiency.

Chapter 2 provides an overview of the various approaches to video understanding. Additionally, it discusses prominent video understanding tasks and their evaluation metrics employed to assess model performance. Finally, several publicly available datasets suitable for training video understanding models are presented.

Chapter 3 dives into the interpretability of predictions and the efficient handling of temporal information. To improve efficiency, the attention mechanisms is employed to process temporal frames and explore temporal relationships. The visualized attention scores present the importance of each frame for the final prediction result. In this way, the predictions of models can be analyzed using attention scores. This approach is evaluated on two public datasets, achieving state-of-the-art performance on the action spotting task. Furthermore, extensive experiments are conducted to analyze the effect of chunk size and the number of encoders for action spotting.

Chapter 4 addresses the heavy computation within the attention mechanism, especially for long-term videos. To mitigate this, a frame saliency module is proposed to pay more attention to keyframes and enhance video representation. Similar to Chapter 3, the visualization of saliency scores aids in interpreting predictions. Moreover, extensive experiments are conducted on several public datasets, video understanding tasks, and base

models to evaluate the robustness of the proposed method.

Chapter 5 proposes an interpretability approach semantically and evaluates it on the video anomaly detection task. The caption-guiding module is introduced to direct models in detecting specific anomaly actions via utilizing stored video captions of abnormal situations. This module generates an anomaly memory space to store abnormal prototypes and leverages them to interpret detection results semantically. The evaluation on two public datasets demonstrates considerable performance in video anomaly detection.

Chapter 6 extends the application of large language models (LLMs) to image understanding and explores the potential of LLMs for the communication-based interpretability method. The chapter focuses on few-shot image classification performance and proposes knowledge selection methods aimed at reducing the cost of fine-tuning.

Finally, Chapter 7 summarizes the key findings of this thesis and outlines the directions of future research.

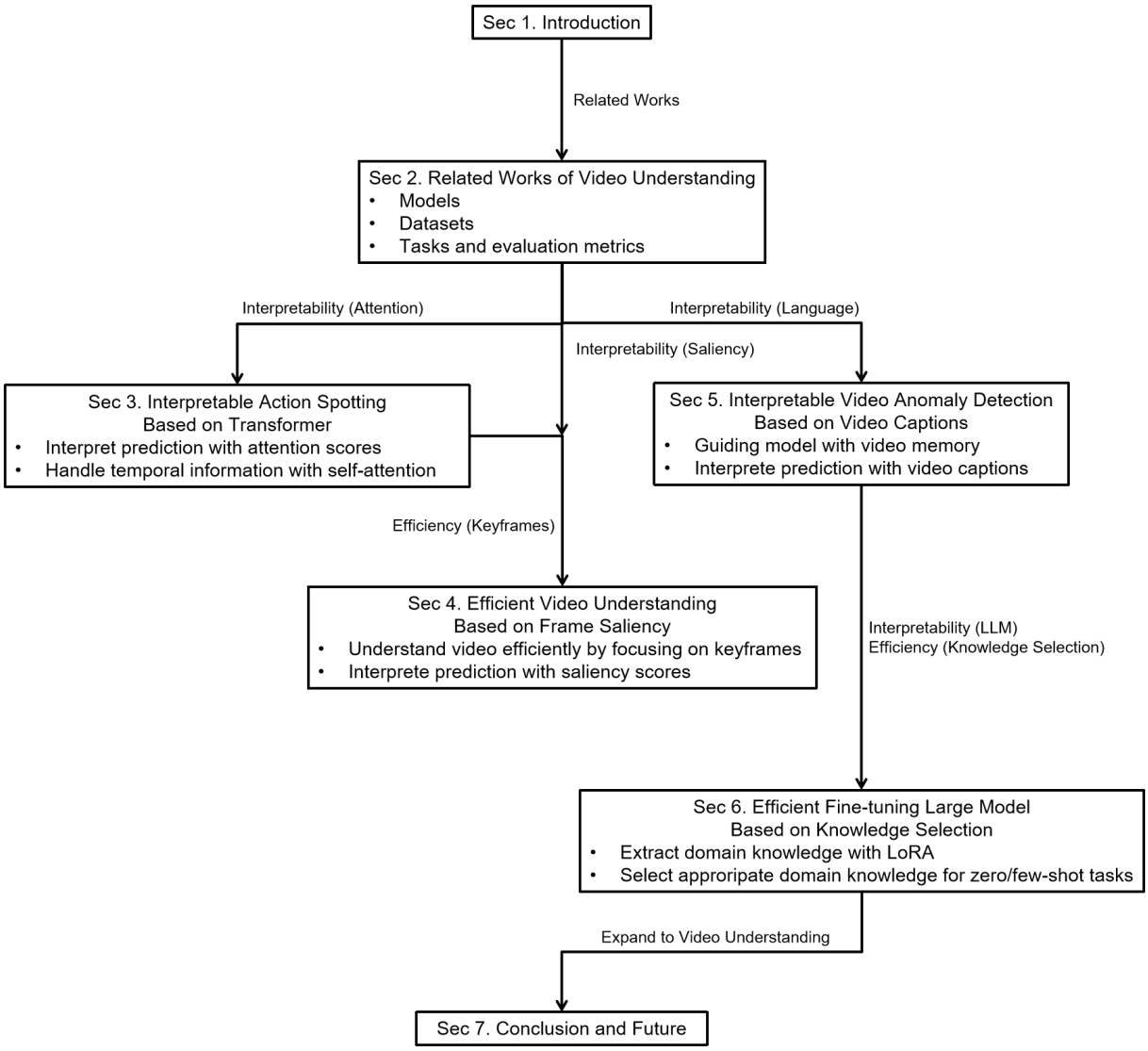


Figure 1.1: Structure of the thesis.

Chapter 2

Related Works of Video Understanding

Before introducing the proposed methods for video understanding tasks, fundamental knowledge of video understanding are provided in this chapter to aid in comprehending this research.

Various video understanding models are presented in Section 2.1. Section 2.2 introduces video understanding tasks and associated metrics used for evaluation. Section 2.3 introduces several public datasets suitable for training video understanding models. Finally, Section 2.4 provides a concluding summary of the related works of video understanding.

2.1 Video Understanding Models

Over the past decade, action understanding research has experienced a paradigm shift from primarily shallow, hand-crafted approaches to deep learning techniques. In this new paradigm, multi-layer artificial neural networks are capable of learning complex non-linear relationships in structured data and achieve considerable performance. Several video understanding models based on deep learning are presented and categorized according to their building blocks in this section.

2.1.1 CNN-based

Convolutional Neural Networks (CNNs) [41] have revolutionized the field of computer vision and have been widely adopted for video understanding tasks. A CNN is primarily composed of convolutional, pooling, normalization, and fully-connected layers. CNNs are useful in video understanding because the sharing of weights dramatically decreases the number of trainable parameters and therefore reduces computational cost compared to fully connected networks.

1-Dimensional CNNs (C1D), 2-Dimensional CNNs (C2D), and 3-Dimensional CNNs

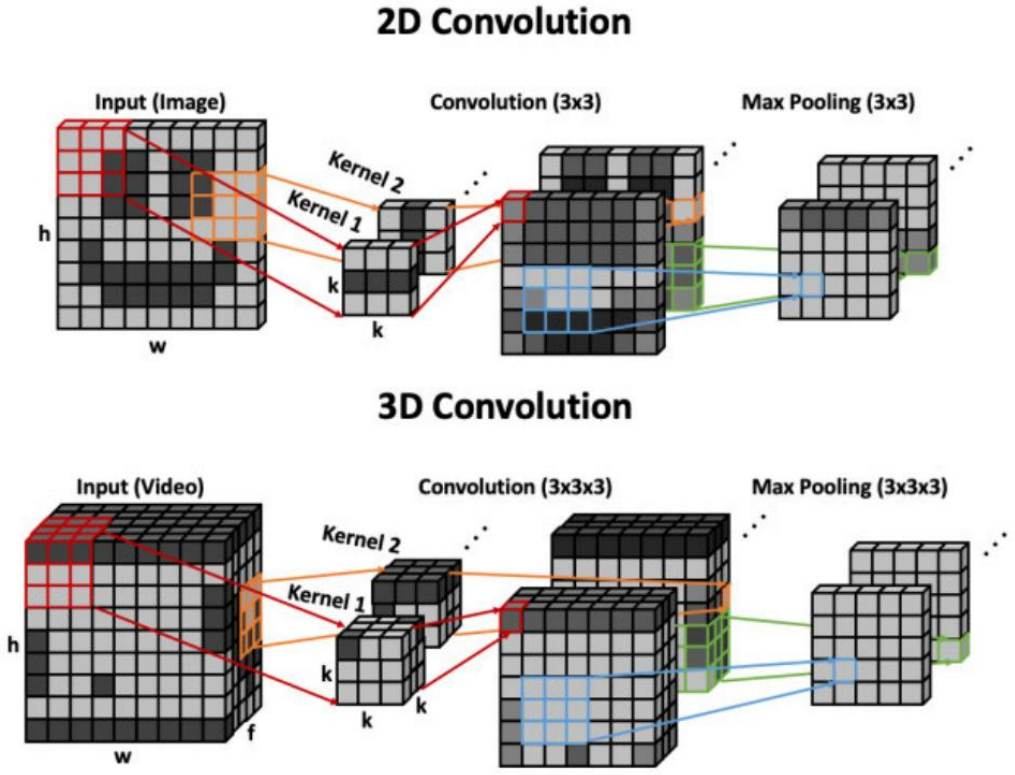


Figure 2.1: Convolution layer. *An example of 2D and 3D convolutional layers are shown. Figure is cited from [46].*

(C3D) are key components of many approaches [61, 50, 108, 125, 94, 90, 14, 39]. They use 1D, 2D, and 3D kernels, respectively. C1D is used for convolutions along the time dimension of embedded features, while C2D and C3D are used for extracting feature vectors from individual frames or frame sequences. 3D-convolution layer allows for a temporal receptive field in addition to the standard spatial one. Single-channel examples of 2D and 3D convolutions are shown in Fig. 2.1. When using multi-channel inputs, the convolutional kernels must be expanded to include a depth dimension with the same number of channels as the input tensor and the output is summed across channels. CNNs are particularly well-suited for analyzing spatial features in video data, thanks to their ability to automatically learn hierarchical representations from raw pixel values.

Numerous CNN architectures have been proposed and tailored specifically for video understanding, achieving remarkable success across a variety of tasks. For instance, a two-stream CNN architecture was introduced, consisting of separate spatial and temporal streams for processing video frames and optical flow information, respectively [87]. This model achieved considerable performance in action recognition tasks by effectively cap-

turing both appearance and motion cues in videos. Similarly, 3D CNNs were proposed to directly operate on spatiotemporal volumes of video data, achieving competitive performance in action recognition tasks without the need for optical flow computation [28]. Moreover, the availability of pre-trained 2D-CNN models on large-scale image datasets (e.g., ImageNet[82]) has facilitated the transfer learning and fine-tuning of CNN architectures for video understanding tasks. They are often used as the frame feature extractors, enabling efficient training with limited annotated video data.

2.1.2 RNN-based

3D convolution kernels handle temporal information like spatial information, however it is counter-intuitive. Because video content can change in the next frame, therefore two adjacent images can contain completely different content. It limits the performance of the convolution kernel. Recurrent Neural Networks (RNNs) emerged as a powerful tool for modeling sequential frames and capturing temporal dependencies in video understanding tasks. Different from traditional feed-forward neural networks, RNNs are equipped with recurrent connections that enable them to maintain internal state representations and process sequential data over time. In video understanding, RNNs leverage their recurrent connections to capture temporal dependencies and motion dynamics across video frames. By processing video frames sequentially, RNNs can effectively model the evolution of features and capture complex interactions within video data. Moreover, variants of RNNs such as long short-term memory (LSTM) [42] and gated recurrent unit (GRU) [16] have been specifically designed to address the challenges of vanishing gradients and long-range dependencies, making them well-suited for modeling sequential data with varying lengths and durations.

RNN architectures have been extensively explored and adapted for video understanding, demonstrating their effectiveness in capturing motion dynamics and temporal context in video data. For example, a LSTM-based RNN architecture was proposed for video understanding tasks [24]. This model combined convolutional layers for feature extraction with LSTM layers for temporal modeling, achieving state-of-the-art performance in tasks such as action recognition and video description generation. Similarly, an RNN architecture was employed to model long-range temporal dependencies in video sequences, demonstrating significant improvements in video classification accuracy [116].

2.1.3 Hybrid Models

Hybrid video understanding models combine multiple modalities or architectures to leverage their complementary strengths in analyzing and comprehending video data. These models integrate different types of neural networks, such as CNN, RNN, and attention mechanisms, to capture both spatial and temporal information within video sequences. Some focus on reducing the large computational costs of C3D: P3D [80], R(2+1)D [33, 98], ARTNet [104], MFNet [15], GST [69], and CSN [96]. Others focus on recognizing long-range temporal dependencies: LTC-CNN [100], NL [105], Timeception [45], and STDA [56]. Some unique modules include TSM [61] which shifts individual channels along the temporal dimension for improved C2D performance, TrajectoryNet [123] which uses introduces a TDD-like [106] trajectory convolution to replace temporal convolutions.

One common approach in hybrid models is to combine CNNs with RNNs or attention mechanisms. CNNs are well-suited for extracting spatial features from individual video frames, while RNNs or attention mechanisms are effective at capturing temporal dependencies and long-range interactions across frames. For example, a two-stream architecture was proposed [87], consisting of spatial and temporal streams. The spatial stream processes RGB frames using CNNs to capture appearance-based features, while the temporal stream computes optical flow between consecutive frames and feeds it through a separate CNN to capture motion-based features. These streams are then fused at later stages to make predictions, resulting in improved action recognition performance.

Another approach is to integrate CNNs with transformer-based architectures, such as the Video Transformer Network (VTN) [76]. In this approach, CNNs are used to extract spatial features from video frames, which are then fed into the transformer encoder to capture temporal dependencies and global interactions across frames. This hybrid architecture combines the strengths of CNNs in spatial feature extraction with the self-attention mechanism of transformers to achieve state-of-the-art performance in various video understanding tasks.

By combining multiple modalities or architectures, these models can effectively capture both spatial and temporal information within video data, leading to more robust and accurate video understanding capabilities.

2.1.4 Graph-based

Graph neural networks (GNNs) are often used for modeling complex relationships and structures for video understanding tasks. Different from CNNs which generally operate

on input data with fixed size, GNNs can effectively capture the spatial and temporal dependencies inherent in video data by representing them as graphs. In the context of video understanding, GNNs leverage the graph representation of video data to model interactions between video frames, objects, and actions. By encoding the spatial and temporal relationships as edges in the graph, GNNs can effectively propagate information across the video sequence.

In recent years, several research studies handled video understanding tasks with have explored the application of GNNs. For example, [117] apply the graph convolutional networks (GCNs) over the graph to model the relations among different proposals and learn powerful representations for the action classification and localization. [120] use a graph convolutional network and a tracking network to derive person-object detections.

2.1.5 Generative Models

Generative models learn to generate representations of future timesteps for prediction. It is generally used for action prediction, video generation tasks,

For the action prediction task, generative models produce *future* features and predict class labels for them. For example, Conv3D [38] utilizes a C3D to generate unseen features for prediction. RGN [122] utilizes a recursive generation and prediction scheme with a Kalman filter during training. RU-LSTM [31] utilizes a multi-modal rolling-unrolling encoder-decoder with modality attention.

2.1.6 Attention-based

The attention mechanism [101] has been widely used in various domains of deep learning in recent years, because of its powerful versatility. It allows models to dynamically allocate resources to the most informative parts of the input data, thereby enhancing their ability to capture fine-grained details and long-range dependencies in videos. In video understanding tasks, attention mechanisms enable models to selectively focus on relevant regions in frames and important temporal segments within video sequences. The attention mechanism can be applied at different levels, including spatial, temporal, and spatio-temporal attention. Spatial attention focuses on relevant regions within individual video frames, while temporal attention highlights representative segments across the temporal dimension. Spatio-temporal attention integrates both spatial and temporal cues to selectively attend to relevant spatial regions and temporal segments simultaneously. By leveraging attention mechanisms, video understanding models can effectively extract

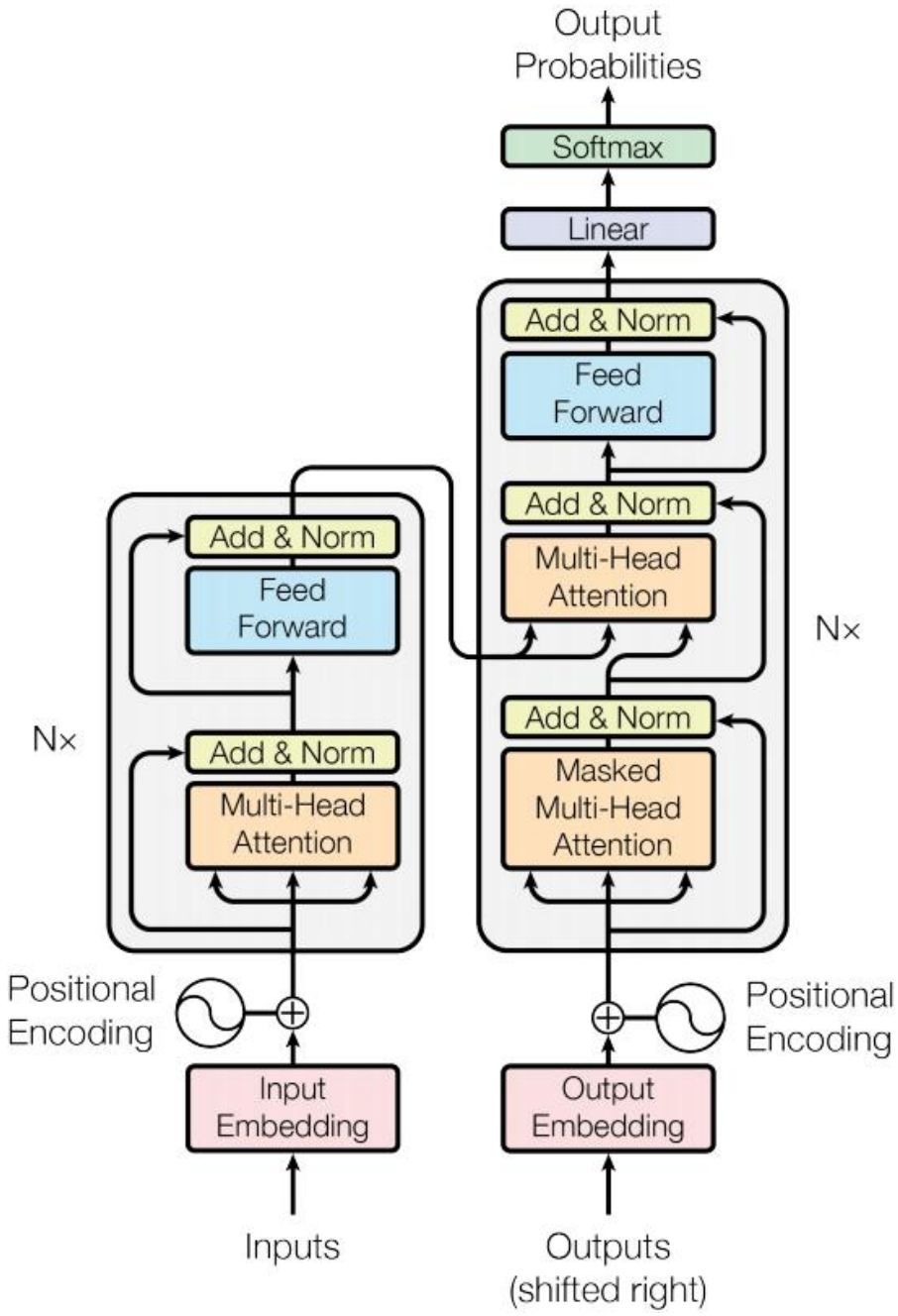


Figure 2.2: Structure of Transformer. *Figure is cited from [101].*

salient features and capture complex interactions within video data.

Transformers, which made their debut in the natural language processing (NLP) field in 2017 [101], are an encoder decoder sequence-to-sequence modeling schema that uses self-attention rather than recurrent neural networks or convolution. Transformers excel at capturing long-range dependencies and modeling complex interactions across sequences,

making them well-suited for analyzing temporal data such as videos. A family of action recognition models that employs vision transformers coalesced in 2020 and 2021. For instance, TimeSformer [6] utilizes embeddings of frame patches augmented with positional information as a sequence of tokens for the transformer. VTN [76] is based off a transformer model that processes long sequences of tokens. ViViT [1] employs another pure-transformer architecture. MViT [27] which introduce resolution and channel scaling in combination with the vision transformer. These models operate directly on video frames or feature sequences, allowing them to capture global context and temporal relationships efficiently.

Moreover, transformer-based models enable parallel processing of video frames, leading to faster inference and training times compared to recurrent architectures. With their ability to handle long-range dependencies and capture temporal context effectively, transformer-based models show promise for various video understanding tasks, including action recognition, video captioning, and video generation. As research in this area continues to advance, more attention-based approaches will be proposed to improve the performance of attention-based models in video understanding domain.

2.1.7 Large-scale Models

With the burgeoning growth of online video platforms and the escalating volume of video content, the demand for proficient video understanding tools has intensified markedly. Given the remarkable capabilities of Large Language Models (LLMs) in language and multi-modal tasks, many video understanding large language models (Vid-LLMs) [118, 70] have been proposed. The emergent capabilities of Vid-LLMs are surprisingly advanced, particularly their ability for open-ended spatial-temporal reasoning combined with commonsense knowledge, suggesting a promising path for future video understanding.

2.2 Tasks and Metrics

In this section, video understanding tasks are divided into two categories, action understanding task and text-based understanding task. In action understanding tasks, models need to utilize videos as input and explore required information from them, such as the class label of video content, and the timestamp that indicates when actions happened. Action recognition and action detection tasks belong to action understanding tasks. Text-based understanding tasks need models to comprehend multi-modality data (text and

videos), and describe and explore video content using text. Examples of them include video question and answer tasks.

2.2.1 Action Understanding Tasks

In this subsection, action understanding tasks are split into three groups according to the complexity of output data. They are simple understanding tasks, temporal understanding tasks, and spatio-temporal understanding tasks. The simple understanding task requires models to recognize specific activities in a video and predict class labels based on video content. If the timestamps where specific actions happen need to be predicted, the models should be trained by the temporal understanding task aiming to localize actions based on timestamps in a video. Spatio-temporal understanding is more difficult than previous tasks, which train models to identify both the when and where of events within video frames.

■ Simple Understanding Tasks

Action Recognition and Video Classification. The goal of the task is to assign a category to a video, known as action recognition or video classification. Action recognition is a fundamental task in video understanding. The objective is to automatically recognize and classify the actions performed by individuals or objects within the video frames.

Action recognition algorithms analyze the spatial and temporal features of video frames to detect and classify different actions. The challenge lies in accurately identifying and distinguishing between various action classes, considering factors such as appearance variations, motion dynamics, and contextual information. Action recognition models typically learn to capture discriminative features that represent different actions and assign action labels to videos.

Different benchmarks for activity classification are: Kinetic-400 [51], Kinetic-600 [10], Kinetic-700 [11], SomethingSomething-V2 (SSv2) [71], ActivityNet [9], HACS [121], HMDB51 [54], UCF-101 [89], and Diving-48 [59].

This task is evaluated using Top-K accuracy, confusion matrix, F1 score, etc. Top-K accuracy measures the proportion of correctly classified videos when the ground truth label can be found in the top-K predicted classes. Top-1 accuracy is referred to as accuracy. It provides a straightforward measure of the ability to correctly identify actions. Additionally, confusion matrix analysis is commonly employed to assess the performance across different action classes. The confusion matrix provides insights into the model's classification errors, enabling researchers to identify challenging action classes for improvement.

Considering both the ability to correctly identify positive instances (precision) and the ability to capture all positive instances (recall), the F1-score provides a balanced measure of precision and recall, offering a single metric to evaluate the overall performance.

Video Retrieval. The video retrieval task aims at finding videos containing specific actions, objects, or scenes. It is often used to train video recommendation models. This task exists in the literature with different names including: Multi-Instance Retrieval (MIR) [2], Paragraph-to-video (P2V) [92] retrieval. MIR focuses on both text-to-video (T2V) and video-to-text (V2T) retrieval. The common benchmark for T2V task is EPIC-Kitchen-100 [19], and the metrics are mAP for V2T retrieval and normalized Discounted Cumulative Gain (nDCG) for T2V retrieval. P2V retrieval bridges the gap between language and video, finding videos relevant to a paragraph (several sentences). Different datasets to solve this task includes ActivityNet Captions [9] and CondensedMovies [3].

■ Temporal Understanding Tasks

Temporal Action Localization/Detection. Different from action recognition, which only identifies the presence of actions in entire videos, temporal action localization/detection (TAL/D) aims to pinpoint the exact moments within videos when specific actions occur by providing both the action label and the temporal boundaries of each action instance. This task is essential for various applications, including video editing and action analysis. Temporal action localization/detection models output temporal segments that correspond to the occurrences of the actions in the video.

The challenge lies in accurately identifying the start and end timestamps of the actions within the video. Since there are many consecutive and similar frames exist in a video, locating the temporal boundaries between actions of interest and backgrounds is difficult for AI models.

Common benchmarks for this task are ActivityNet-v1.3 [9], HACS Segment [121].

The evaluation metrics are the intersection over union (IoU), the average precision (AP) for each action category, and the mean average precision (mAP). Intersection over union (IoU), which measures the overlap between the predicted temporal intervals and the ground truth annotations. IoU quantifies the temporal localization accuracy by assessing the extent of overlap between the predicted and ground truth intervals. Another important metric is average precision (AP), which measures the precision of temporal localization predictions across different confidence thresholds. AP provides insights into the model's performance at various levels of confidence, helping assess its robustness and reliability in localizing actions. Mean Average Precision (mAP) measures the average precision across multiple action categories. Recall at a fixed IoU threshold evaluates the

proportion of ground truth actions that are successfully localized above a certain IoU threshold. This metric highlights the model's ability to accurately localize actions with high overlap with the ground truth annotations, providing a measure of its sensitivity to different levels of localization accuracy.

Video Anomaly Detection. Video anomaly detection is a special task under action detection. It aims to automatically detect and localize abnormal activities, behaviors, or events that deviate from normal patterns or behaviors in a given video stream. Since anomaly rarely happens, this task is a few-shot action detection task. The difference between action detection and anomaly detection is that action detection generally prepares the action classes that need to be detected. While anomaly detection does not have a specific action category since all abnormal situations cannot be listed. The definition of anomaly will be difficult in difficult application scenarios. Video anomaly detection is applied to the applications of security monitoring, industrial inspection, and healthcare.

In video anomaly detection, the challenge is to distinguish between normal activities and anomalous events in video data, often with limited or no labeled anomaly data for training. Anomaly detection models typically learn to capture the underlying distribution of normal behavior in video sequences and identify deviations or outliers that indicate anomalies. These anomalies may manifest as sudden changes, irregular patterns, or unexpected occurrences in the video data.

Common benchmarks for this task are UCF-Crime [91], ShanghaiTech [64] datasets.

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is commonly used to assess the performance of video anomaly detection models, which is a conventional threshold-independent metric [91, 111]. It measures the ability to distinguish between true positive and false positive rates across varying threshold values, thereby quantifying its discrimination performance.

Action Spotting

Action spotting [34] task requires models to output a multi-label classification prediction for each frame. Different from the TAL/D task which detects a temporal boundary for each action instance, action spotting needs the model to output the exact frame for each action instance. The challenge of action spotting is the influence of consecutive similar frames in videos, which makes classifying each frame with an action label become difficult. Action spotting is necessary for applications that need accurate detection results, such as autonomous driving, and detection of rule violations in sports competitions.

The benchmarks are SoccerNet-v1, SoccerNet-v2 dataset [34, 20].

Average-mAP [34, 20] is the main evaluation metric for action spotting task. If the distance between the ground truth timestamp and the predicted timestamp is less than Δ seconds, the prediction is considered positive. Δ is a threshold ranging from 5-60

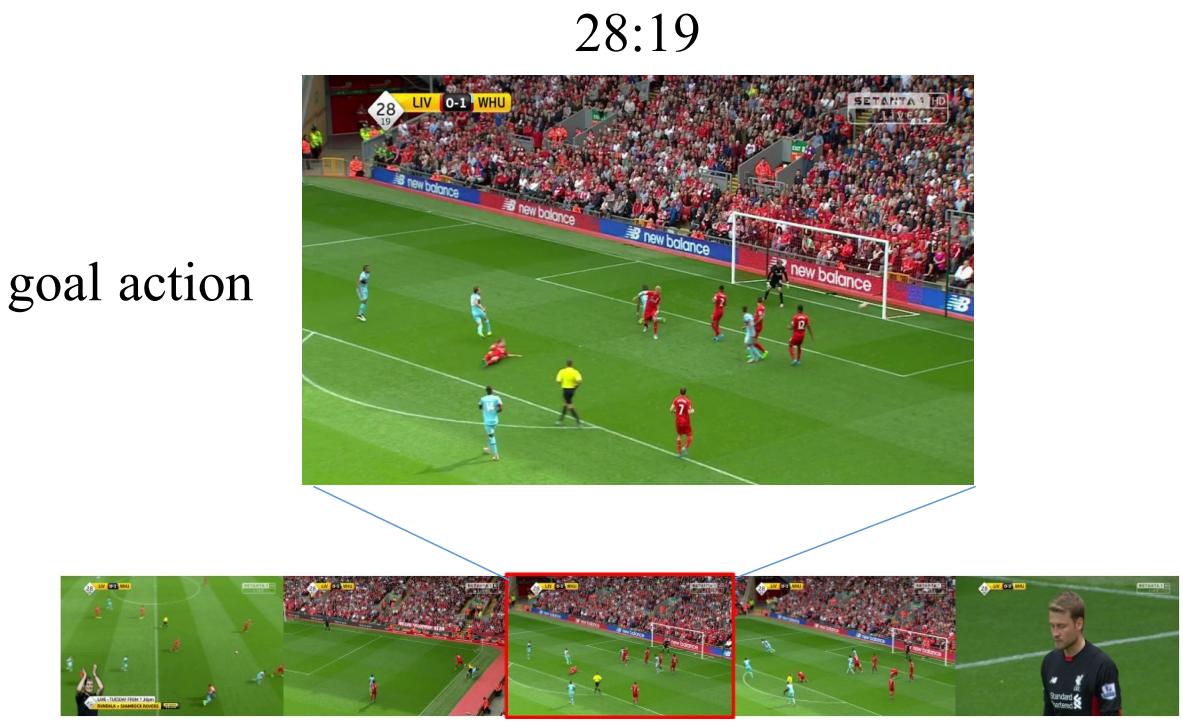


Figure 2.3: Action spotting. *The goal action is localized from a soccer video. The images are cited from [34]*

seconds using a 5-second step size. The average precision (AP) is calculated for each action class and each Δ. The mean average precision (mAP) score is calculated by taking the mean AP over all classes with Δ. The Average-AP is the average of 12 AP values calculated over 12 tolerances Δ for each class. The Average-mAP metric is the average of 12 mAP values calculated over 12 tolerances Δ.

■ Spatio-temporal Understanding Tasks

Spatio-temporal Action Localization/Detection. Spatio-temporal action localization/detection (SAL/D) task aims to locate both when and where specific actions within a video. Notable datasets for this particular category are UCF101-24 [89], JHMDB-22 [48], and UCF-MAMA [74]. These datasets contain annotations for each video frame. Datasets like AVA [37], AVA-Kinetics [55] contain box-annotations at 1Hz sampling frequency over a clip of 15 mins. The evaluation metric for this task is the f-mAP and video-mAP measuring frame-level and video-level localization performance respectively.

Video Object Detection and Tracking. The objective of video object detection (VOD) and tracking is detecting and tracking objects of interest within a video. VOD task

typically aims to detect objects across frame sequences. Since videos typically contain a lot of redundant temporal information, it helps detectors detect an object in the current frame and anticipate the position in the subsequent [67, 127]. The tracking task aims at identifying and following the movement of specific objects throughout a video. A more fine-grained tracking approach known as point tracking has emerged, which tracks specific points on the surface of an object regardless of pixel location. They are essential for various applications, including video surveillance, autonomous driving, and augmented reality.

Video object detection and tracking models need to analyze the spatial and temporal features of videos to detect and localize important objects and track their movements across consecutive frames. The challenge is handling factors such as occlusions, scale variations, motion blur, and object appearance changes while maintaining accurate object localization and tracking performance.

Benchmark datasets for VOD are ImageNet-VID [21]. Popular benchmarking datasets for tracking are KITTI [32], UADETRAC [110], LaSOT [26].

Reported Metrics of VOD are generally mAP with results reported for different speeds of motion of objects. HOTA [68] and Clear-MOT [5] are the evaluation metrics for tracking tasks.

Referring Video Segmentation (RVS). RVS [49] segments the objects referred to by either textual descriptions or the first frame' s segmentation. The benchmarks for RVS using textual description are RefCOCOg [75], Refer-Youtube-VOS [85]. The evaluation metrics for this task are mAP and mIoU.

2.2.2 Text-based Understanding Tasks

In this subsection, text-based understanding tasks are composed of video captioning task and video question answering task. The video caption task focuses on generating an appropriate textual description according to the input video content. Video question answering task enables models to predict the answers to related questions based on video content.

■ Video Captioning

Video captioning is a multi-modality task that generates textual descriptions of video content. The goal of the tasks is to automatically generate human-like captions that accurately describe the content and context of video sequences. Video captioning models typically take video frames as input and produce corresponding textual descriptions that

capture the temporal dynamics, objects, actions, and scenes depicted in the video. A video captioning model should generate informative, coherent, and semantically meaningful captions, providing a concise and accurate summary of the video content. The challenge lies in understanding the visual content of the video and translating it into natural language descriptions that convey relevant information to humans.

Popular benchmarks for this task are MSRVTT [112], Youcook2 [126], and MSVD datasets [13].

The evaluation metric for this task are BLEU@4 [78], METEOR [4], ROUGE [60], and CIDEr [103]. BLEU (Bilingual Evaluation Understudy) measures the n-gram overlap between generated and reference captions, providing insights into linguistic similarity. METEOR (Metric for Evaluation of Translation with Explicit Ordering) offers a more holistic assessment by considering synonyms and paraphrases, incorporating a weighted harmonic mean of precision and recall. CIDEr (Consensus-based Image Description Evaluation) evaluates consensus between generated and human-judged captions, capturing diversity and informativeness. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) computes recall-based scores for n-grams, assessing overlap and similarity between generated and reference captions. These metrics collectively enable a comprehensive evaluation of caption quality and alignment with human judgments.

■ Video Question Answering (VQA).

VQA answers questions about the video content based on visual information and potentially textual queries. According to the literature, this task is subdivided into three subcategories: Multiple-Choice (MC), Open-Ended (OE), and Long-Form (LF). MC-VQA addresses multiple-choice question answering. OE-VQA answers subjective, creative, and logical questions. LF-VQA goes beyond single answers, generating comprehensive explanations that understand video content, reason temporally, and adapt to diverse question types.

Common benchmarks for this subtask are TGIF-Action and TGIF-Transition [47], MSRVTT-MC [114]. Common benchmarks for this subtask are TGIF-Frame [47] and ActivityNet-QA [115]. Common benchmarks for LF-VQA subtask are ActivityNet-QA [115] and VIOLIN [63].

VQA is evaluated using Top-1, Top-K accuracy as a metric.

2.3 Video Understanding Datasets

2.3.1 Something-Something

Something-Something [36, 71] is a human-object interaction benchmark, released in 2017. Video creation was crowd-sourced through Amazon Mechanical Turk (AMT). The dataset consists of 108,499 videos and 174 classes. 108,499 videos are divided with an 80/10/10 training/validation/test split. Each single-instance lasts for 2–6 seconds. Examples of the 174 classes contain *holding something*, *turning something upside down*, and *folding something*. A second and larger version was released in 2018. It also added object annotations, reducing label noise, and improving video resolution. These are important benchmarks for human-object interaction due to their scale and quality.

2.3.2 UCF101

The UCF101 [89] dataset is a benchmark dataset for action recognition, containing 13,320 video clips across 101 action categories. It offers diversity in action types, actors, environments, and camera viewpoints. The dataset is Widely used for evaluating the performance of the action recognition task.

UCF101-24, the spatiotemporally labeled data subset of THUMOS' 13, was produced as part of the THUMOS' 13 challenge. Examples of the 24 human action classes include *Basketball dunk*, *Ice dancing*, *Surfing*, and *Walking with dog*. The majority of the classes are sports. It consists of 3,207 videos from the original UCF101 dataset [89]. Each video contains one or more spatio-temporally annotated action instances. While multiple instances within a video have separate spatial and temporal boundaries, they have the same action class label. The average video length is approximately 7 seconds. The dataset is organized into three train/test splits.

2.3.3 Kinetics

The Kinetics [12] dataset is a large-scale dataset with over 650,000 video clips spanning 600 action categories. Sourced from YouTube, it offers diversity in video quality, camera viewpoints, and actor demographics. It is valuable for training video understanding models to recognize a broad spectrum of human actions accurately. The Kinetics dataset family was produced as a large-scale, high-quality dataset of URL links to human action video clips focusing on human-object and human-human interactions. Class examples from Kinetics-400 [12] include *hugging*, *mowing lawn*, and *washing dishes*. Clips were

Figure 2.4: UCF101. *Images are cited from [89].*

collected from YouTube and annotated by AMT crowdworkers. The dataset consists of 306,245 videos, and within each class, 50 and 100 are reserved for validation and testing, respectively. Each single-instance video lasts for 10 seconds. Additional videos and classes were added in 2018 and 2019. These are among the most cited human action datasets in the field and continue to serve as a standard benchmark and pretraining source.

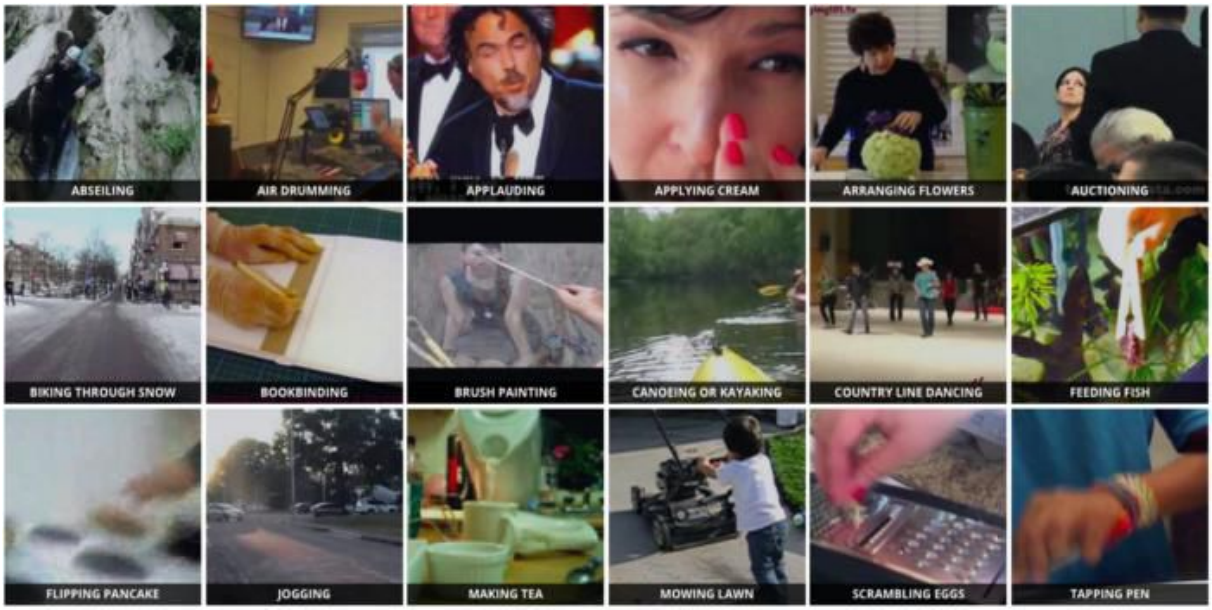


Figure 2.5: The Kinetics dataset. *Images are cited from [51].*

2.3.4 HMDB51

The HMDB51 [54] dataset focuses on action recognition, comprising 6,766 video clips across 51 action categories. Each clip is manually annotated with action labels. It is suitable for benchmarking and comparing the effectiveness of different action recognition algorithms, particularly in fine-grained action recognition tasks.

2.3.5 ActivityNet

ActivityNet [9] is a large-scale dataset designed for activity recognition and temporal localization. The ActivityNet dataset [9] family was produced for both action recognition and detection. Example human action classes include *Drinking coffee*, *Getting a tattoo*, and *Ironing clothes*. ActivityNet 100 (v1.2) is a 100-class dataset divided into a 4,819 videos (7,151 instances) training set, a 2,383 videos (3,582 instances) validation set, and a 2,480 videos test set. It was expanded to ActivityNet 200 (v1.3) with 200-classes divided into a 10,024 videos (15,410 instances) training set, a 4,926 videos (7,654 instances) validation set, and a 5,044 videos test set. On average, action instances are 51.4 seconds long. Web videos were temporal annotated by AMT crowd-workers. ActivityNet remains as a foundational benchmark for TAP and TAL/D because of the dataset scope and size. It enables the evaluation of models for action recognizing within untrimmed videos, reflecting real-world scenarios.

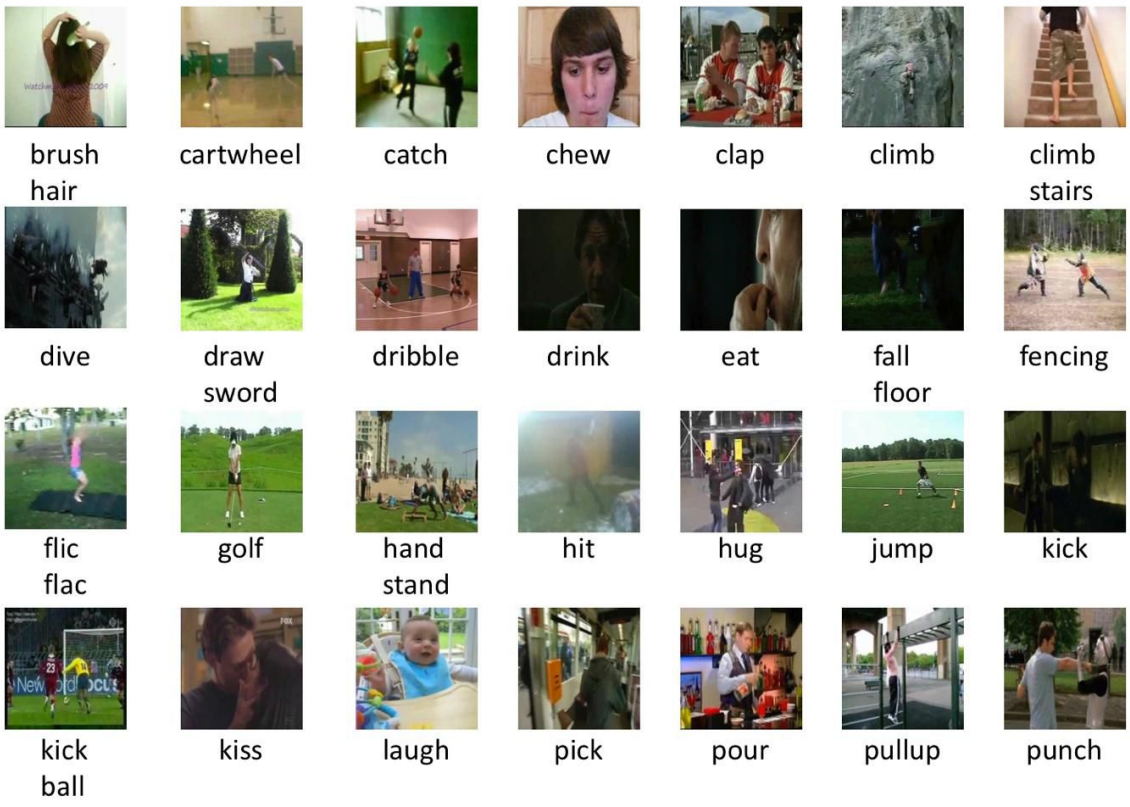


Figure 2.6: HMDB51. *Images are cited from [54].*

2.3.6 Charades

The Charades [86] dataset focuses on daily activity recognition in home environments. Charades was produced as a crowd-sourced dataset of daily human activities (e.g., *pouring into cup*, *running*, and *folding towel*). The dataset consists of 9,848 videos (66,500 temporal action annotations) with a roughly 80/20 training/validation split. Videos were filmed in 267 homes with an average length of 30.1 seconds and an average of 6.8 actions per video. Action instances average 12.8 seconds long. Charades-Ego used similar methodologies and the same 157 classes. However, in this dataset, an egocentric (first-person) view and a third-person view are available for each video. The dataset consists of 7,860 videos (68.8 hours) capturing 68,536 temporally annotated action instances. Charades serves as a TAL/D benchmark along with ActivityNet [9], but it is also useful as a multi-label AR benchmark because of the high average number of actions per video. Charades-Ego presents a multiview quality unique among large-scale daily human action datasets. It is useful for evaluating models on complex and realistic activity recognition

Figure 2.7: ActivityNet. *Images are cited from [9]*

tasks in naturalistic settings.

2.3.7 SoccerNet

SoccerNet-v2 contains 765 hours of videos of 500 soccer games, with 300,000 annotated timestamps and 17 action classes, such as *goal*, *ball out of play*, and *yellow card*. SoccerNet-v2 is divided into training, validation, and test sets as 300, 100, and 100 games, respectively [20]. The frame rate of videos is 2 frames for each second. The ground truth for each frame is a label vector. The label vector contains 17 different action labels as well as a label for the background. The action label in the label vector is set as 1 if the corresponding action occurs in the frame and other labels are set as 0. If none of the 17 actions appears in the chunk, then the background label is set to one.

SoccerNet-v1 [34] contains the same soccer videos as SoccerNet-v2, however it only includes three action classes and 6,637 annotations.

2.3.8 ShanghaiTech Campus

ShanghaiTech Campus dataset [64] contains 437 videos from 13 campus surveillance scenes with complex light conditions and camera angles. It contains 130 abnormal events and over 270,000 training frames. In this dataset, 238 videos are used for training and 199 videos for testing in the weakly-supervised setting. Moreover, the pixel-level ground truth of abnormal events is also annotated in the dataset. Different from the previous dataset which only contained videos captured with one fixed-angle camera, it includes multiple scenes with multiple view angles to increase scene diversity Further, it introduces the anomalies caused by sudden motion in this dataset, such as chasing and brawling in our

Figure 2.8: SoccerNet-v1. *Images are cited from [34]*

dataset, which are not included in previous datasets.

2.3.9 UCF-Crime

The UCF-Crime [91] dataset contains 1900 surveillance videos covering 13 real-world anomalous classes such as robbery, explosion, and road accident. It contains 1610 training videos and 290 test videos. Compared to ShanghaiTech [64], which mainly includes pedestrian activities in a university setting, the scenes in the UCF-Crime dataset are more diverse and complex.

2.4 Conclusion

Video understanding is a critical research in computer vision, which encompasses various tasks aimed at comprehending video content and context. Because it is required in various applications, such as surveillance, security monitoring, industrial inspection, and healthcare. Since the complexity and large volume of video data, deep-learning mod-

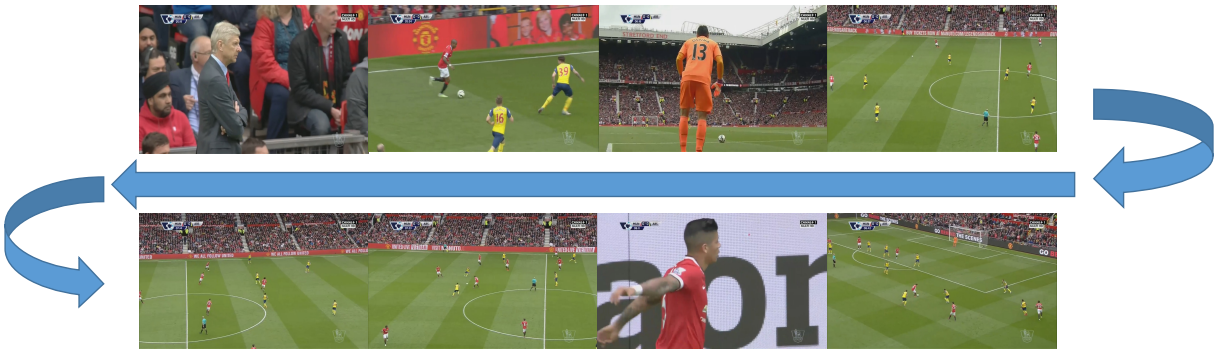


Figure 2.9: SoccerNet-v2. *Images are sampled from [20]*

Figure 2.10: ShanghaiTech Campus dataset. *Images are cited from [64]*

els have become the mainstream of video understanding. Typical video understanding approaches are introduced and categorized by the model structures.

To achieve automatic video understanding, video-related tasks are proposed, such as action recognition, action spotting, and object tracking. These tasks involve analyzing video content and identifying actions, objects, events, or anomalies within video sequences. To evaluate the performance of these tasks, several metrics are used. These include Intersection over Union (IoU), Average Precision (AP), accuracy, recall, F1-score, Mean Average Precision (mAP), coverage, and redundancy. These metrics provide quantitative measures of model performance, enabling researchers and practitioners to assess accuracy,

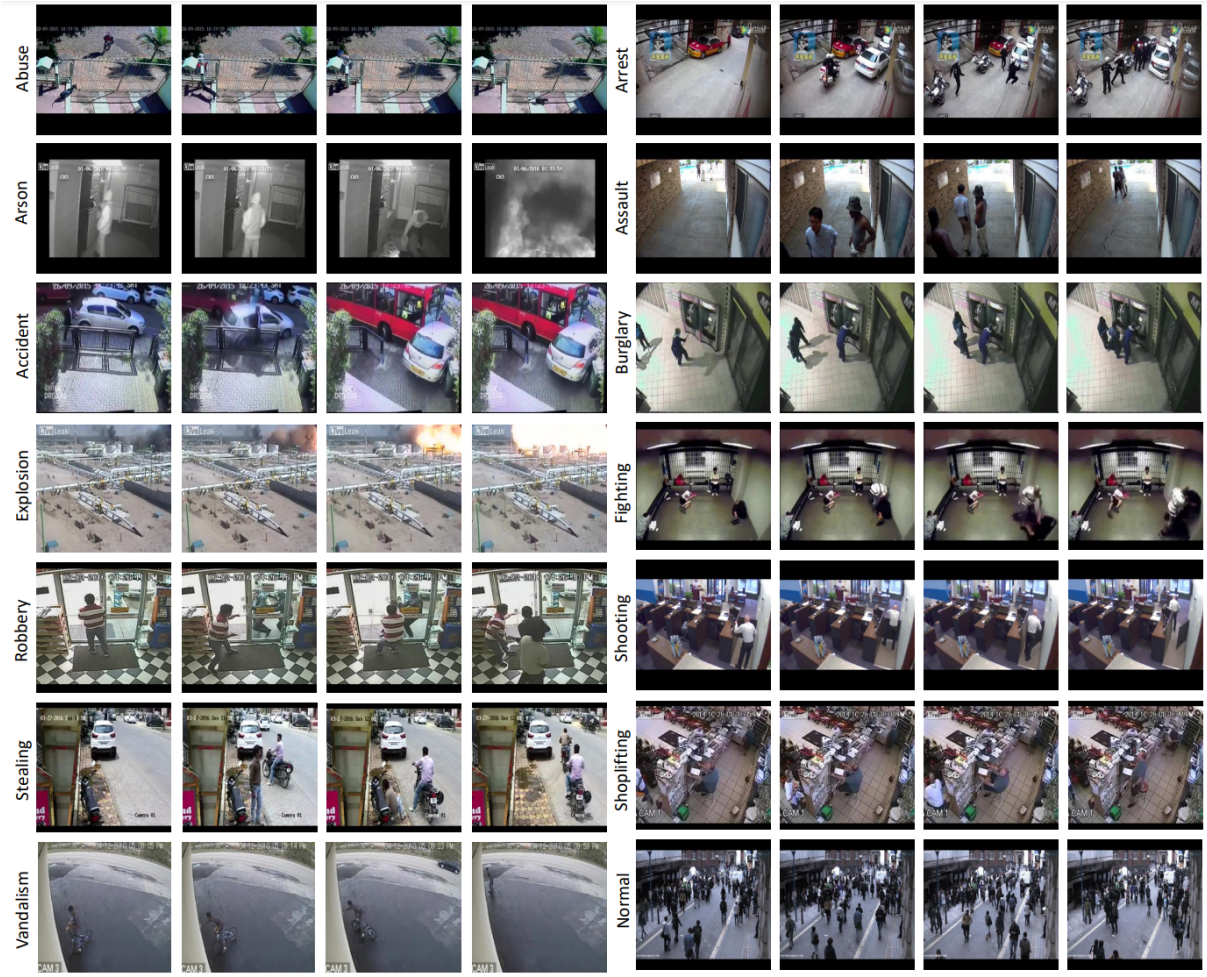


Figure 2.11: UCF-Crime dataset. *Images are cited from [91]*

localization, summarization quality, and anomaly detection capabilities across different video understanding tasks. Video understanding tasks and their evaluation metric are introduced in Section 2.2. Some of the tasks are used to evaluate my proposed methods in the following chapters.

To support these tasks, Many datasets are released to train and evaluate video understanding models, as such: UCF101 [89], ActivityNet [9], and SoccerNet-v2 [20]. The datasets are demonstrated in the Section 2.3.

Chapter 3

Interpretable Action Spotting Based on Transformer

As shown in Chapter 2, various action-related tasks are proposed to train video understanding models, such as action detection, action recognition, and action spotting. Since the actions that happen in a video can provide significant information, comprehending each action in a video is necessary for video understanding. Action spotting, which temporally localizes specific actions in a video, is an important task for understanding high-level semantic information. Different from action detection, it is required to find the most relevant frame where the action occurred. Fewer frames need to be focused on the action spotting task. It is helpful to evaluate the interpretability of models. Therefore, we propose an interpretable model for the action spotting task in this chapter.

The scene encoder, which is similar to the transformer encoder, is proposed to recognize actions/scenes from input videos. It utilizes self-attention to model temporal relationships in a video segment. Furthermore, the attention scores are visualized to present the importance of frames in an input video. The attention score is used to analyze the predictions. The experimental results on the public dataset, SoccerNet-v1 and SoccerNet-v2, demonstrate state-of-the-art accuracy. By using embedding features, our method obtains an Average-mAP of 75.3%, which is close to human-level accuracy.

This chapter shows the background and motivation of the proposed action spotting model in Section 3.1 and demonstrates model architecture in Section 3.2. Two public datasets are used to evaluate the proposed method and extensive experiments are conducted to analyze the effectiveness of the proposed method in Section 3.3. At last, a conclusion of the chapter is provided in Section 3.4.

3.1 Introduction

Automatic or semi-automatic video editing algorithms are required by many applications. For example, the generation of summary videos will reduce the time from event

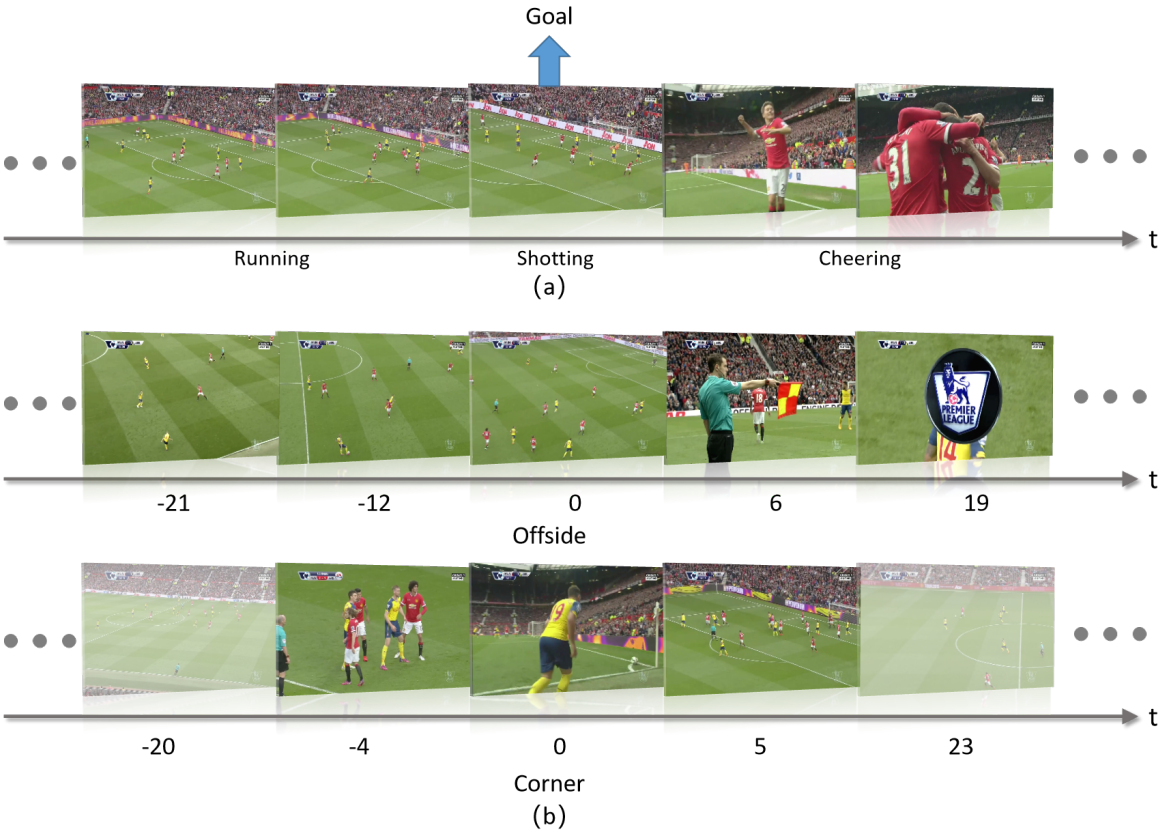


Figure 3.1: Action spotting. *A Goal action is shown in sub-figure (a), which consists of running, shooting, and cheering scenes. The top and bottom of sub-figure (b) show an Offside and a Corner action, respectively. The temporal duration of these actions is different. Note that the action happens in zero second as the center image of each example. Frame images are cited from [20].*

to broadcast in sports. To approach this task, understanding video content is significant. Action spotting is a video understanding task of localizing an event anchored to a single timestamp.

There are two main challenges in action spotting. The first challenge is how to adequately explore temporal information in videos. A video generally includes sequential images that are visually similar, but have a different context. For instance, the football field is typically the image background in a football scene, as shown in Figure 5.1. Therefore, it is important to consider not only visual information but also underlying temporal information for action spotting in such videos [35, 73].

The second challenge is interpreting predictions of action spotting. A deep learning model [84] could provide visual clues for image classification tasks and interpret classification results. However, the importance of each frame in a video still cannot be represented. Without interpretability, it is difficult to apply an AI model in a real-world application.

To address these issues, self-attention is employed to model the temporal patterns and utilize attention scores to interpret prediction results. Furthermore, the structure of multiple encoders is introduced and action-aware chunk size is applied to improve the performance of action spotting. Chunk size is the temporal length of the input video.

One may consider an action to be composed of different subactions, e.g., a *Goal* action is often composed of *Running*, *Shooting*, and *Cheering* subactions. A *Yellow card* action typically includes a scene of a player falling and a scene of the referee raising the card. The novel model based on multiple encoders is proposed for action spotting to tackle these challenges. Specifically, as shown in Figure 3.2, first image features are extracted from sequential frames of a video as a chunk. The chunk is split into multiple segments, and fed into multiple transformer encoders, respectively, to recognize scene content and capture the changes of scene in an action. Finally, the proposed model classifies actions by recognizing the scene sequence. Each type of action generally has a different duration, since they include different subactions. Therefore, different actions should utilize a different chunk size to learn the pattern of action and employ the action-aware chunk size with the aim of increasing action-related frames and reducing unrelated frames in a chunk.

In summary, the interpretable action spotting model is proposed based on scene encoders, which utilizes the self-attention mechanism to model temporal relationships and employ the attention score to interpret predictions. Additionally, the performance is further improved by the multiple encoder structure and the action-aware chunk size. Moreover, the proposed model achieves the state-of-the-art Average-mAP of 75.3% for action spotting on the SoccerNet-v2 dataset and evaluates the model design choices in ablation studies.

3.2 Transformer-based Model for Action Spotting

In this section, a novel model with multiple encoders is proposed, as shown in Figure 3.2. The extracted features are split into multiple subsets with the same duration. They are fed into a scene encoder, respectively. The structure is designed to learn semantic information in each subset to recognize scene sequences.

3.2.1 Scene Encoder

To recognize a scene in each video segment, the scene encoder is introduced in the proposed model. The scene encoder employs the structure of a transformer encoder. To accommodate different features, which may have different dimensions, an MLP layer is

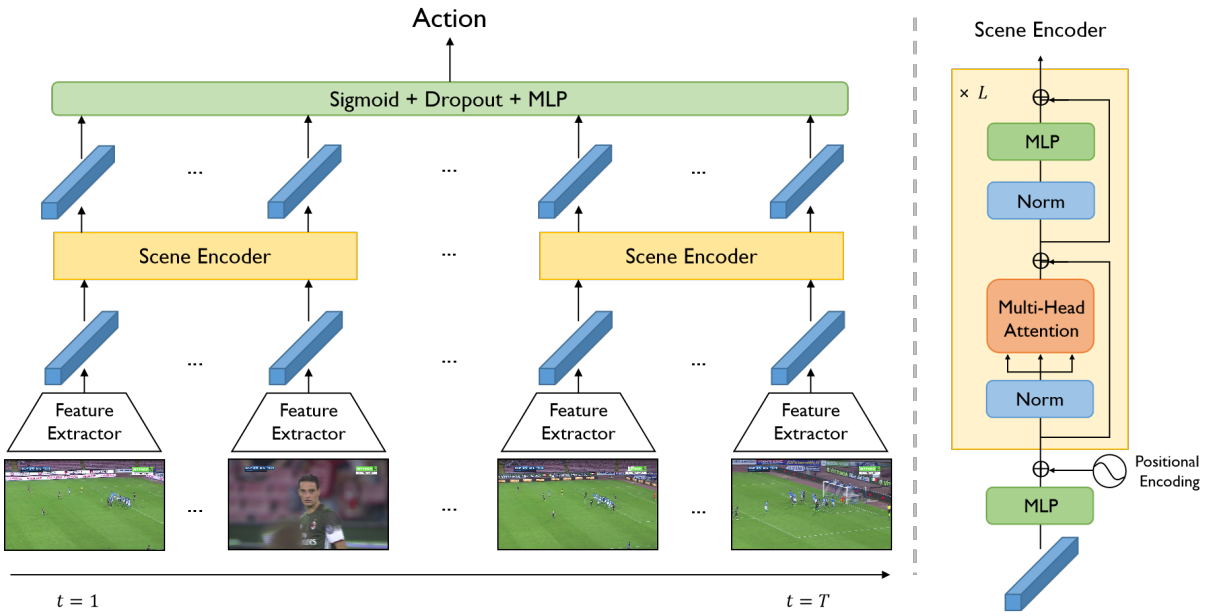


Figure 3.2: Proposed Model Architecture. *Images of a football game are cited from [20].*

utilized as the first layer of the scene encoder to map features to a fixed length E. The temporal information is injected to model the temporal relationship between frames by the positional encoding layer. The positional encoding is represented as a sinusoidal function, as proposed in [102].

After the positional encoding layer, the multi-head self-attention mechanism [102] is used to explore latent patterns from the temporal relationship among frames. The number of scene encoders, the number of stacked encoder layers, and the number of heads in the multi-head attention are denoted as N, L and H, respectively. The attention calculated by the query matrix $Q^n = \phi_q^n(\{\mathbf{f}_t\}_{t=1}^{\frac{T}{N}})$, the key matrix $K^n = \phi_k^n(\{\mathbf{f}_t\}_{t=1}^{\frac{T}{N}})$ and the value matrix $V^n = \phi_v^n(\{\mathbf{f}_t\}_{t=1}^{\frac{T}{N}})$, where ϕ_q, ϕ_k and ϕ_v are MLP layers and n is the index of the encoder.

The attention of the h^{th} attention head in the n^{th} encoder is calculated by

$$\text{head}_h^n = \text{Attention}_n^p(Q^n, K^n, V^n), \tag{3.1}$$

$$O_n = \phi_o^n([\text{head}_h^n]_{h=1}^H), \tag{3.2}$$

where ϕ_o^n is an MLP layer, and the *Attention* function is the scaled dot-product attention in [102]. The output of n^{th} encoder is denoted as O_n. The query, key, and value matrices are three different representations of the same frames. In Attention, the dot-product of Q and K represents the correlation of every two frames. Two similar frames have a strong correlation because their features are similar. The result is used as the weight on V to

help the model focus on similar frames. Since the frames in a scene are generally similar, self-attention can find scenes in a segment by finding similar images.

The scene encoder is used to extract scene features in each subset. The outputs of all encoders, which present the scene features in the subsets, are merged via averaging them temporally. A sigmoid layer and a dropout layer are used to avoid overfitting. Finally, an MLP ϕ_r layer and a sigmoid layer are employed to classify an action as:

$$\mathbf{C} = [O_1; O_2; ...; O_n], \tag{3.3}$$

$$\mathbf{m} = \frac{1}{T}\sum_{t}^{T}\mathbf{c}_t, \tag{3.4}$$

$$\mathbf{y} = \sigma(\phi_r(dropout(\sigma(\mathbf{m})))), \tag{3.5}$$

where [;] is a concatenation operator, and $\mathbf{c}_t$ is the presentation vector of the t^{th} frame output by encoders.

3.2.2 Model Architecture

Here action spotting is formulated to the task of recognizing scene sequences in each chunk. To recognize scene sequence in a video, the multiple encoder structure is proposed, as presented in Figure 3.2. The set of input feature vectors are partitioned into multiple subsets and each subset are handled by a respective encoder. If not specified otherwise, the number of scene encoders is set to two. Every scene encoder contains 8 heads and 2 encoder layers.

3.2.3 Implementation Details

For training models, the Adam optimizer [52] and the binary cross entropy loss function are used, and a starting learning rate is set to 0.002. The dropout rate is set to 0.1, the batch size to 128 and the chunk size to 15 seconds on training and validation datasets. The training process is stopped once the mAP metric on the validation dataset stops increasing for 6 continuous epochs.

■ Video Encoding

The frames are sampled at 2 fps from videos with a resolution of 224×224. The frame of an input video at the time index $t = 1, 2, \ldots, T$ is denoted by $\mathbf{x}_t \in \mathbb{R}^{H\times W\times C}$, where T is the number of frames in a chunk, H and W are height and width, and C is the number of

image channels. A feature vector of d-dimensional per frame is denoted by $\mathbf{f}_t \in \mathbb{R}^{1\times d}$. A ResNet-152 [41], which is pre-trained on ImageNet [21], is utilized as a feature extractor. As a result, the feature extractor outputs a 2,048-dimensional vector for every frame. The features are remapped to a 256-dimension vector per frame.

■ Training

During training, a long video is divided into multiple non-overlapping chunks. The proposed model is trained to predict the label of the actions that happened in each chunk. Since multiple actions can occur within a chunk, action spotting is formulated as a multi-label action classification task.

■ Inference

There are two types of inference processes: using a fixed chunk size for all actions and using an optimal chunk size for each class, called action-aware chunk size. When using a fixed chunk size, the prediction of each chunk is used as the classification result of its center frame. The prediction result for the entire video is obtained by sliding chunks frame by frame, to predict the classification results of all frames. The inference process with the action-aware chunk size is shown in Figure 3.3. The features with different chunk sizes are fed into the trained model to obtain the classification result. The result using an action-aware chunk size is taken as the classification result of the corresponding action. For example, the 20-second chunk result is used as the classification result of *penalty* action. In this manner, chunks are slid frame by frame to obtain the results of every frame. The appropriate chunk sizes are selected based on the performance of the validation dataset.

■ Non-Maximum Suppression (NMS)

For reducing positive action spotting results with low confidence, non-maximum suppression is used on confidence of a whole video in each class, following previous works [34, 18, 20, 35]. The NMS process is shown in Figure 3.4. For each class, the first peak of confidence from all frames is kept, and the confidence of the rest frames in a NMS window are set to 0. Then, the next peak is located and the same process is performed until all peaks above the NMS threshold are found. Finally, the confidence scores below the threshold are set to 0. There are two hyper-parameters in the process, the NMS threshold is 0 and the NMS window size is 60 frames (30 seconds).

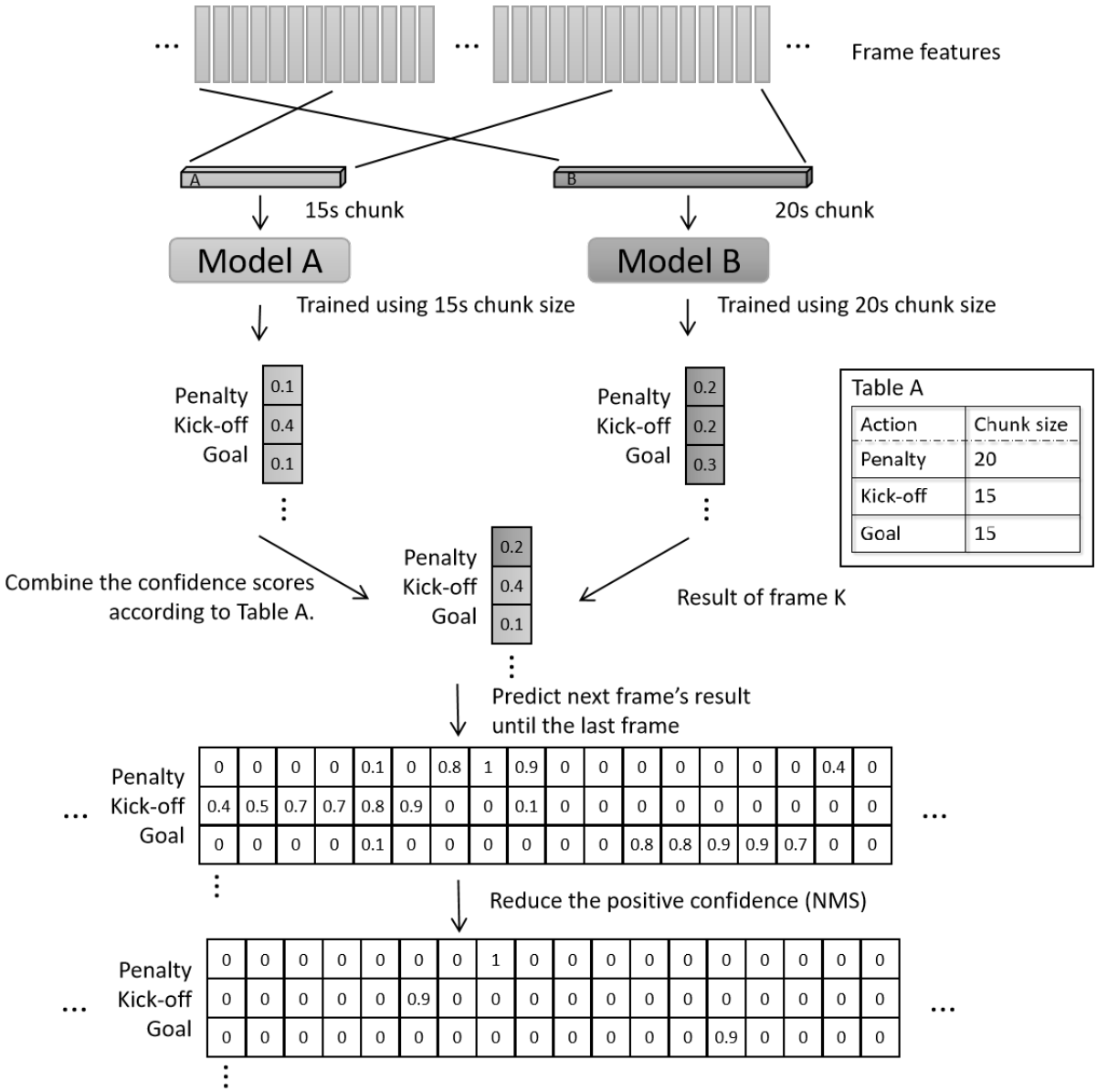


Figure 3.3: Inference. *Models A and B are trained with the chunk size of 15 and 20 seconds, respectively.*

3.3 Experiments

In this section, the proposed model is compared with several existing methods on the SoccerNet-v2 dataset [20]. The influence of chunk size for action spotting is analyzed to evaluate the effectiveness of chunk size optimization. Moreover, some ablation studies are conducted to confirm the design choices of the method. Finally, the model is evaluated on the SoccerNet-v1 [34] dataset to show the generalizability of the method.

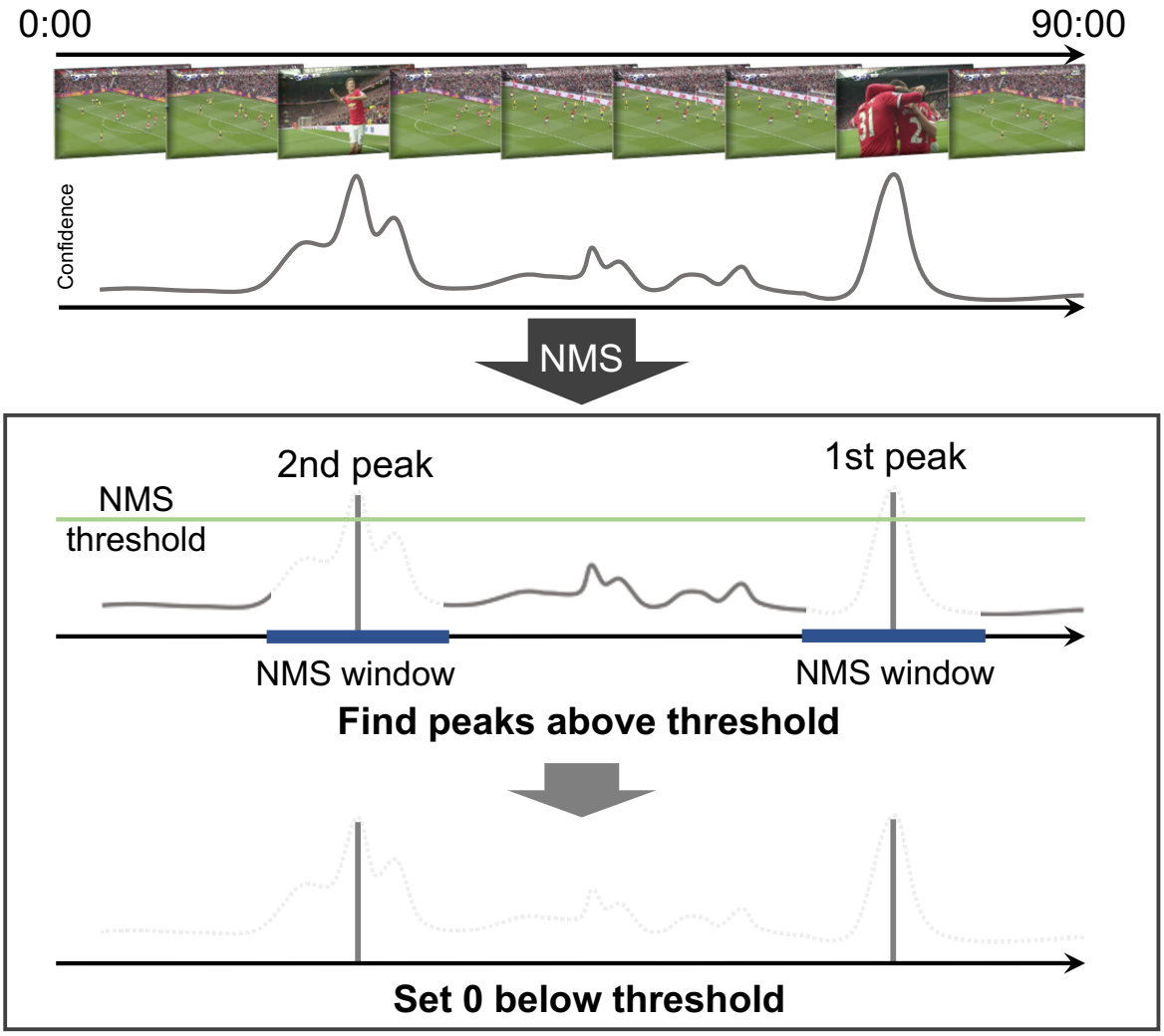


Figure 3.4: Non-Maximum Suppression (NMS).

3.3.1 Dataset and Evaluation Metric

The SoccerNet-v2 [20] is used to train and evaluate the method. SoccerNet-v2 contains 765 hours of videos of 500 soccer games, with 300,000 annotated timestamps and 17 action classes, such as *goal*, *ball out of play*, and *yellow card*. It is divided into training, validation and test sets as 300, 100 and 100 games, respectively [20].

SoccerNet-v1 is the predecessor of SoccerNet-v2, released by the same authors. It has 6,637 temporal annotations, including three classes of actions (goal, card, and substitution). SoccerNet-v1 contains fewer annotations and thus longer intervals between actions than SoccerNet-v2. There are no 120-second frame intervals containing more than five actions in the SoccerNet-v1 dataset. On the contrary, SoccerNet-v2 has up to 14 actions in 120-second intervals.

Table 3.1: Action spotting on SoccerNet-v2. *The ResNet is a pre-trained ResNet-152 and PCA is principal component analysis. Ours(1) uses 15 seconds as the chunk size. Ours(2) uses an optimal chunk size for each action. Number of data is the number of samples in every class.*

Method	Feature Extractor	Average-mAP	Penalty	Kick-off	Goal	Substitution	Offside	Shots on target	Shots off target	Clearance	Ball out of play	Throw-in	Foul	Indirect free-kick	Direct free-kick	Corner	Yellow card	Red card	Yel.→Red card
NetVLAD	ResNet+PCA	31.4	47.4	42.4	32.0	16.7	32.7	21.3	19.7	55.1	51.7	45.7	33.2	14.6	33.6	54.9	32.3	0.0	0.0
AudioVid	ResNet+PCA	39.9	54.3	50.0	55.5	22.7	46.7	26.5	21.4	66.0	54.0	52.9	35.2	24.3	46.7	69.7	52.1	0.0	0.0
CALF	ResNet+PCA	40.7	63.9	56.4	53.0	41.5	**51.6**	26.6	27.3	**71.8**	47.3	37.2	41.7	25.7	43.5	72.2	30.6	0.7	0.7
NetVLAD++	ResNet+PCA	50.7	67.7	59.6	70.2	70.3	35.3	37.1	38.3	56.0	68.2	65.3	62.4	43.4	55.2	78.9	50.0	1.5	1.7
NetVLAD++	ResNet	53.4	**79.3**	**62.1**	71.6	68.7	39.3	39.3	41.0	57.0	70.3	69.0	**64.2**	44.4	57.8	79.7	**56.7**	4.0	3.7
Ours(1)	ResNet	54.7	75.8	60.8	72.0	70.6	38.6	41.8	40.2	60.6	71.3	70.3	63.5	49.2	59.9	81.6	53.4	8.0	**11.5**
Ours(2)	ResNet	**55.2**	68.7	60.8	**72.0**	**70.6**	42.2	**41.8**	**42.2**	60.6	**73.1**	**72.3**	63.4	**49.2**	**59.9**	**83.6**	53.9	**15.9**	7.2
Number of data			173	2566	1703	2839	2098	5820	5256	7896	31810	18918	11674	10521	2200	4836	2047	55	46

The evaluation metric of action spotting is Average-mAP value. If the temporal offset between prediction and its closest ground truth is less than a given tolerance Δ, the prediction is regarded as positive. The average precision (AP) for the prediction per class within the threshold Δ, averaged over action classes to calculate the mAP. The Average-AP is the average of AP values calculated over 12 error tolerance values Δ (from 5 to 60 seconds, the step size is 5 seconds), respectively, for each class. The Average-mAP is the average of 17 Average-AP.

3.3.2 Performance of Action Spotting

■ Evaluation on SoccerNet-v2

Table 3.1 shows the results of the methods and several state-of-the-art methods (NetVLAD [81], AudioVid [99], CALF [18] and NetVLAD++[35]). As seen in Table 3.1, the method achieves an Average-mAP of 55.2% on the test dataset. The Average-mAP is an absolute 1.8% higher compared to NetVLAD++. This improvement is consistently seen for 12 of the 17 action classes, especially for classes with few examples (*Red Card* and *Yellow→Red*). This indicates the advantage of modeling the temporal relationship in every subset, especially for actions with few samples. The proposed method adequately utilizes the temporal relationship due to the multiple encoder structure.

Performance has a low correlation with the number of samples as seen in Table 3.1. In several actions where there are distinctive patterns of frame changes such as *corner* and *goal*, a better performance is achieved even with few samples. On the other hand, on actions with less distinctive patterns, such as *indirect free-kick*, the performance is lower. For better recognition of temporal patterns, it is important to consider optimal chunk

Table 3.2: Evaluation results for different chunk sizes. *ResNet-152 is used as the feature extractor. The Average-AP on the validation dataset changes with chunk size on three actions (direct free-kick, corner and yellow card).*

Chunk Size	10	15	20	25	30
Direct free-kick	52.7	**57.4**	52.8	49.7	46.4
Corner	**83.0**	81.1	77.8	74.6	69.6
Yellow card	54.9	54.7	**55.7**	52.3	52.6

Table 3.3: Action spotting on SoccerNet-v1. *The proposed method compares with prior works on SoccerNet and achieves the best performance.*

Model	**Feature Extractor**	**Average-mAP**
NetVLAD [81]	ResNet+PCA	49.7
AudioVid [99]	ResNet+PCA	56.0
CALF [18]	ResNet+PCA	62.5
NetVLAD++ [81]	ResNet+PCA	61.1
Ours	ResNet+PCA	64.5
Ours	ResNet	**66.8**

size because as many as action-related frames are included and the number of unrelated frames are decreased in chunks (see next subsection).

■ Evaluation on SoccerNet-v1

SoccerNet-v1 [34] contains the same soccer videos as SoccerNet-v2, but only includes three action classes and 6,637 annotations. The proposed model is evaluated on the SoccerNet-v1 dataset [34] comparing with related works. The model obtains an Average-mAP of 66. 8%, exceeding CALF [18] by 4.3%, as shown in Table 3.3. When using ResNet and PCA as feature extractor, the proposed method exceeds CALF [18] by 2%.

3.3.3 Interpretability Based on Attention Score

■ Visualization of Confidence Score

For further analysis of the results, the confidence scores of several classes are visualized in Figure 3.5.

Figures 3.5 (a) and (d) show examples of the confidence score of *corner* action on a temporal axis. As seen in these figures, the confidence score is high for frames where

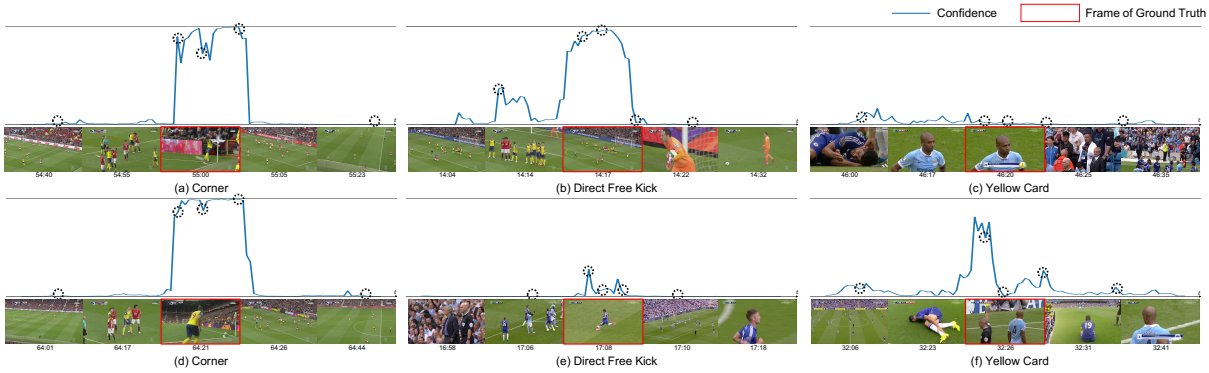


Figure 3.5: Confidence score examples. *Ground truth labeled frames of different actions are shown in frames within red boxes. The blue line represents the change in confidence scores in the time series axis. The confidence scores for adjacent frames are shown as circles in the graphs above.*

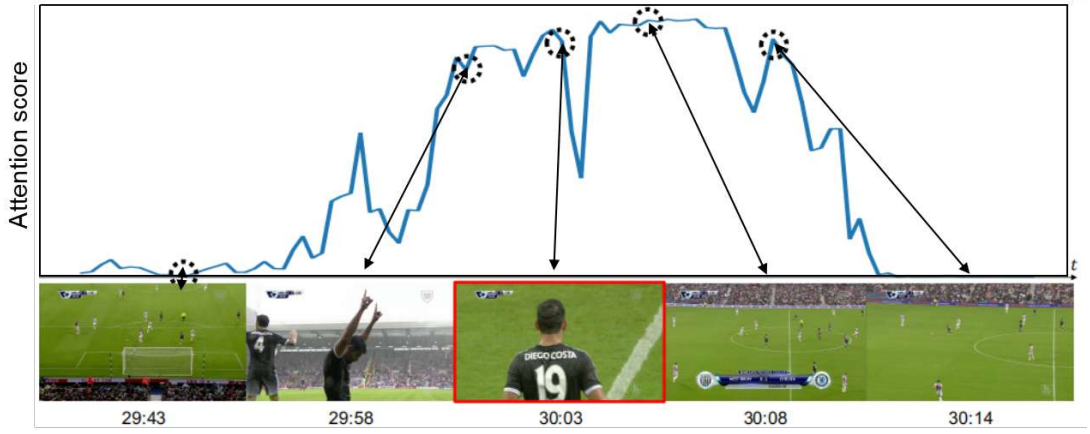
several people are shown in a close-up view. Such frames ordinarily last for around 10 seconds; therefore, an appropriate chunk size of *corner* action would be 10 seconds, as shown in Table 3.2.

The confidence of the *direct free-kick* action is high for frames where players gather in front of a goal. As shown in Figure 3.5 (b), the confidence score increases in frames where players gather in front of the goal. On the other hand, in Figure 3.5 (e), such frames appear for only a few seconds. Consequently, the confidence becomes low.

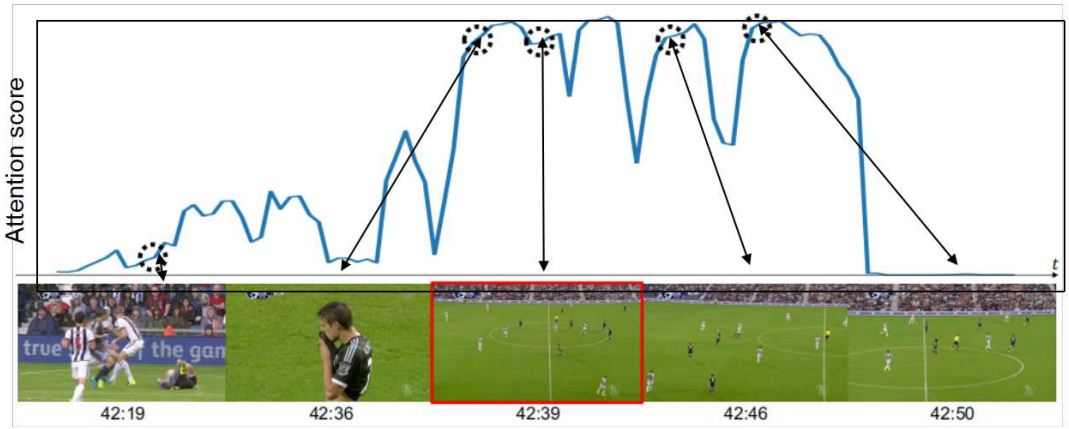
In the *yellow card* action, we often observe scenes of a player falling and a referee appearing. In contrast, no referee appears in Figure 3.5 (c), however, the yellow card information is displayed on the screen. The confidence score is low in this case. In another case, we observe a high confidence score in Figure 3.5 (f) as the scene of a falling player followed by a referee scene. $10-20$ seconds is an appropriate range of chunk size for yellow card actions. Because the scenes where a player falls often happen $3-10$ seconds before a *Yellow card* action, there is often a replay scene where a player falls, increasing the appropriate chunk size for yellow cards to over 10 seconds.

■ Visualization of Attention Score

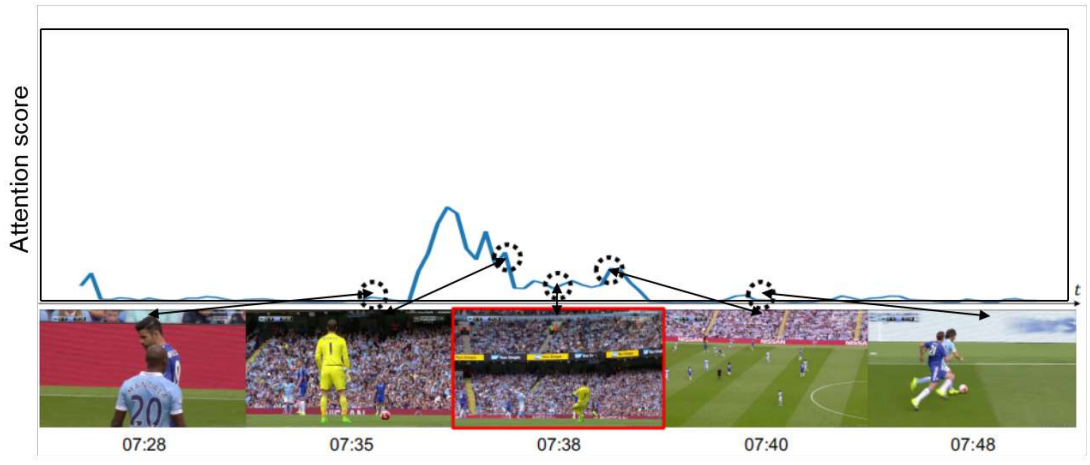
To evaluate the interpretability based on attention score for action spotting, three attention score graphs are plotted in Fig. 4.5 The attention score is calculated by the last frame attention estimation module in the model. The stride length is set to 15 seconds in inference, and it is equal to the chunk size used in training. The attention scores in every chunk are concatenated to obtain the attention of all frames. As shown in Figures 4.5



(a) Kick-off



(b) Kick-off



(c) Indirect free kick

Figure 3.6: Attention Score Examples. *Labeled actions from the SoccerNet-v2 dataset [20] are shown in frames marked by red boxes. The attention scores for the adjacent frames are marked by circles in the graph. The translucent red rectangle indicates the range within 5 seconds from the ground truth timestamp.*

Table 3.4: Evaluation of chunk size optimization. *In Fixed size, every class uses a 15 second chunk size. Results with and without chunk size optimization are shown in Fixed and Optimized. The optimized chunk size of action classes is shown in the row of Chunk Size. For most action classes, performance improvement is observed by chunk size optimization.*

	Penalty	Kick-off	Goal	Substitution	Offside	Shots on target	Shots off target	Clearance	Ball out of play	Throw-in	Foul	Indirect free-kick	Direct free-kick	Corner	Yellow card	Red card	Yel.→Red card	Avg-mAP
Fixed	**75.8**	**60.8**	**72.0**	**70.6**	38.6	**41.8**	40.2	**60.6**	71.3	70.3	**63.5**	**49.2**	**59.9**	81.6	53.4	8.0	**11.5**	54.7
Optimized	68.7	**60.8**	**72.0**	**70.6**	**42.2**	**41.8**	**42.2**	**60.6**	**73.1**	**72.3**	63.4	**49.2**	**59.9**	**83.6**	**53.9**	**15.9**	7.2	**55.2**
Chunk Size (sec)	20	15	15	15	20	15	10	15	10	10	10	15	15	10	20	30	20	

(a) and (b), the high attention scores appear in the frames where the center of a football ground appears. The center of the football field is related to the *kick-off* scene. Another example of an *indirect free-kick* is shown in Figure 4.5 (c). Attention scores in frames in which a player is standing next to a ball are high. The attention mechanism takes such frames as a clue to locating indirect free-kick actions. Unlike the confidence scores providing the chunk-level importance scores, the attention scores can predict the frame-level importance scores based on the temporal relationship of frames. Therefore, attention scores are more helpful for interpreting predictions.

3.3.4 Analyses

■ Action-aware Chunk Size

To find an appropriate chunk size for each action, the model is trained with different chunk sizes, ranging from 10 to 40 seconds (5 seconds as step size). Then, the chunk size that achieves the best performance in the validation dataset is selected. The optimal chunk sizes are shown in Table 3.4. As presented in this table, each action requires its own chunk size. Thanks to this optimization, several improvements of performance are reached. For example, the Average-AP of *offside* is improved by 3. 6% using the 20-second chunk size. The Average-AP of *shot-off target* is also improved by 2% with the chunk size of 10 seconds. The Average-AP of few-shot actions, such as *red card* and *yellow→red card*, varies considerably with chunk size, and the lack of training samples makes it challenging to select an optimal value. On the other hand, actions containing many samples have similar optimal chunk sizes in the test and validation datasets. The Average-mAP is improved by 0.5%.

Table 3.5: Comparison of different number of encoders. *Models are evaluated by the Average-mAP with the number of encoders from 1 to 5. The ResNet-152 [41] is used as the feature extractor. The highest value is obtained by a model using two encoders. Chunk Size shows the best fixed chunk size for each model.*

Num. of Encoder	1	2	3	4	5
Average-mAP(%)	52.1	**54.7**	53.5	52.9	52.2
Chunk Size(s)	15	15	15	15	15

Table 3.6: Comparison of different feature extractors. *The performance of the proposed method is evaluated using three feature extractors. Length shows the feature vector length of each feature extractor.*

Feature Extractor	ResNet+PCA	ResNet	Embedding
Average-mAP(%)	52.9	54.7	**75.3**
Length	512	2048	8576

■ Influence of Scene Encoders

To analyze the influence of encoders, models with different numbers of encoders are developed and evaluated, respectively. For each model, the hyper-parameters including chunk size are tuned to obtain a better performance. Note that a fixed chunk size is used for all models in this experiment, and the best performance is obtained with a 15-second chunk size. Models with multiple encoders achieve better results than the single encoder model, as shown in Table 3.5. Interestingly, the performance of models with more than two encoders is decreasing as the number of frames in each chunk is reduced when covering the same time window. The model with two encoders achieves the best performance on SoccerNet-v2, modeling pre-action and post-action windows with a duration of 7.5 seconds each.

■ Influence of Video Features

The proposed method is evaluated using three feature extractors (ResNet, ResNet + PCA, and Embedding feature extractor [128]). The results are shown in Table 3.6. The embedding feature extractor [128] consists of TPN [113], GTA [40], VTN [76], irCSN [97], I3D-Slow [28]. The Average-mAP increases to 75.3% using embedding features. To extract scene context in videos, it is required to correctly acquire important information in frames. Therefore, the selection of feature extractor has a significant effect on action spotting accuracy.

3.4 Conclusion

In this chapter, a novel action spotting model with the attention mechanism is proposed. The multiple scene encoders are employed to learn from the temporal relationship in subsets. Each scene encoder utilizes the attention mechanism to handle temporal information and represents the importance of frames with attention scores to interpret predictions. Furthermore, the multiple encoder structure and the action-aware chunk size demonstrate their effectiveness for analyzing action information from videos. According to the results of the experiments, the proposed model increases the state of the art by 1. 8% utilizing the attention mechanism to learn the temporal patterns of each action. The proposed model obtains an Average-mAP of 75.3% with embedding features.

Chapter 4

Interpretable and Efficient Video Understanding Based on Frame Saliency

In the last chapter, the effectiveness of the attention mechanism has been evaluated on the action spotting task. In addition, the visualized attention scores can interpret which frames are focused by the model. They can help us analyze the limitations and advantages of models. However, the heavy calculation of self-attention limits the real-time applications.

To address this problem, a novel model is proposed for video understanding tasks. Every video contains some similar frames, which cannot provide important information and require an equal amount of features to represent.

The large number of similar frames poses a challenge for recognizing actions in videos. Therefore, the frame saliency weighting module is proposed to focus on keyframes and use frame saliency scores to represent the importance of frames and interpret predictions. The proposed model contains two encoders, for pre-action and post-action time windows, to encode video context. We validate our design choices and the generality of the proposed method in extensive experiments. In the public SoccerNet-v2 dataset, the method achieves an average mAP of 57.3%, improving over the state of the art. Using embedding features obtained from multiple feature extractors, the average mAP further increases to 75%. Reducing the model size by over 90% does not significantly affect performance. Additionally, we use ablation studies to prove the effectiveness of the saliency weighting module. Moreover, considering high performance on action spotting could not show the generic of the proposed method, evaluation tasks are extended from action spotting to other video understanding tasks. The frame saliency weighting strategy is shown to be applicable to existing methods on more general action datasets, such as SoccerNet-v1, ActivityNet v1.3, and UCF101.

The structure of this chapter is as follows. Section 3.1 introduces the concept of the proposed method, and Section 3.2 demonstrates the proposed model. Section 3.3 shows

the experiment results of video understanding tasks. Section 3.4 concludes this chapter.

4.1 Introduction

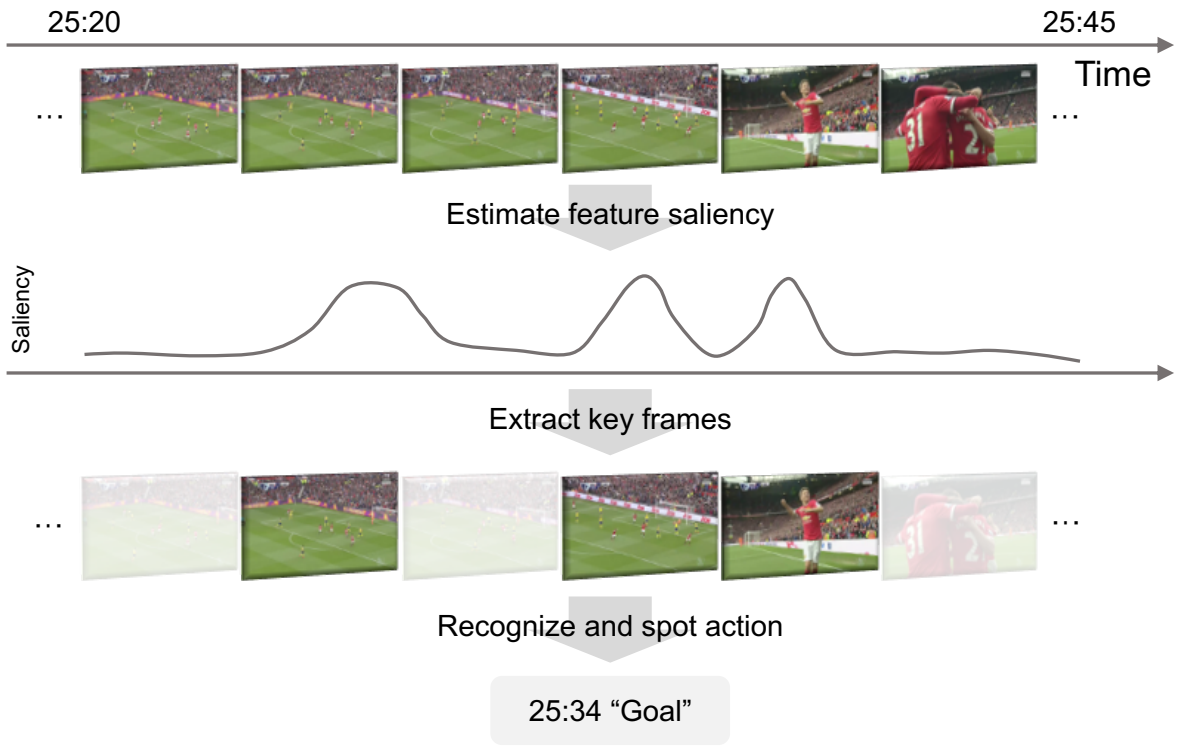


Figure 4.1: Action Spotting. *The action can be recognized from a single frame showing a cheering player. The proposed method estimates frame saliency in order to focus on discriminative keyframes for efficient action spotting.*

Video understanding has been researched for decades, and many models have been proposed, as mentioned in Chapter 2. Most of them focus on the expression of the temporal relationship.

A less explored research direction is understanding video content efficiently by focusing on keyframes. Videos contain many similar frames, which provide little information related to the task at hand. Much like a storyboard or cartoon strips, actions, and their context can be expressed in far fewer pictures if they contain sufficient information. To locate keyframes in videos, the saliency score is proposed to represent the importance of frames based on the similarity of frame features. In this work, the frame saliency weighting module is proposed to calculate a saliency score for each frame and use these as weights of frame features. Different from traditional saliency calculation methods, the proposed method improves video representation and saliency scores based on the cosine

similarity of frame features. Experiment results show that this module has significant benefits in terms of guiding the model to extract meaningful representations from videos. They demonstrate that frame saliency is efficient and highly effective in improving video representation while being significantly less complex than models that use self-attention such as transformers [101]. As an additional benefit, focusing on the salient features allows the model to understand videos with fewer parameters and reduce the model size by over 90% while maintaining high action spotting precision. On SoccerNet-v2 [20], the proposed model reaches a 57.3% Average-mAP for action spotting, an absolute improvement of +3.9% with respect to the current state of the art. The confidence scores and saliency scores are leveraged to analyze the performance of the proposed model in detail. Using multiple feature extractors and computing the embedded features of them, the Average-mAP metric further increases to 75.0%. In extensive experiments, the model structure, feature extractors, and hyper parameter choices are evaluated for video understanding tasks. To highlight its generality and versatility, the proposed module is evaluated on various public datasets and improves the performance of existing models.

In this chapter, the frame saliency weighting module is proposed to focus on salient information in videos by using temporal saliency weighting of feature vectors within temporal windows. Extensive experiments are conducted to analyze the proposed method on video understanding tasks, As a result, the proposed model achieves state-of-the-art performance in SoccerNet-v2 [20] on the task of action spotting and improves existing methods on several tasks of video understanding.

4.2 Frame Saliency Weighting Module for Video Understanding

The videos contain many similar frames in temporal segments (chunks) that are redundant for action spotting. Additionally, when an action occurs, the content of the video changes, and the changed frame is different from others. Such frames appear when actions occur, consequently, they should be focused on. The proposed method explicitly addresses the two facts. Considering that simply removing similar frames would reduce the duration of actions and change the content of video, an efficient frame saliency weighting module is proposed, which reduces the weight of redundant information and highlights keyframes that are important for the action spotting task. Action spotting models benefit from such distinct frames, which have low inter-frame similarity and high saliency.

4.2.1 Frame Saliency Weighting

To focus on keyframes efficiently, frame features are weighted by saliency, which is used to represent the importance of frames. The formula for computing a weighted frame feature is as follows.

$$\boldsymbol{k}_i = \sum_{j=1}^{N_f} \frac{e^{-(s_{i,j}-\theta)}}{\sum_{m=1}^{N_f} e^{-s_{m,j}}} \boldsymbol{f}_i, \tag{4.1}$$

where $\boldsymbol{f}_i$ means the i^{th} feature in a chunk, $\boldsymbol{k}_i$ is the weighted feature vector of the i-th frame, N_f is the number of frames in a chunk, and θ is a hyper parameter to adjust feature weights. When not explicitly stated, the value of θ is 0. By weighting features with saliency, model could locate keyframes and improve video representation efficiently. When an action happens in a video, the frame and frame features change. Consequently, we can consider that locating distinct frames according to the similarity of frame features is an efficient way to detect actions. $s_{i,j}$ represents the similarity of i-th and j-th frame features. The calculation of $s_{i,j}$ is as follows:

$$s_{i,j} = \boldsymbol{f}_i^{\mathrm{T}} \boldsymbol{f}_j. \tag{4.2}$$

If $s_{i,j}$ is high, the i-th and j-th frames are consider similar and the content of them redundant.

To further improve the representation, two fully connected layers and a ReLU layer are used to remap features.

$$\boldsymbol{k}_{\text{remap}} = \max\left(0, \boldsymbol{k}\boldsymbol{W}_1 + \boldsymbol{b}_1\right)\boldsymbol{W}_2 + \boldsymbol{b}_2 + \boldsymbol{f}, \tag{4.3}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the weights and $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the biases of the fully connected layers, respectively. $\boldsymbol{f}$ means the features of all frames in a chunk and $\boldsymbol{k}$ represents the weighted features. This step improves the representation ability of the model via mapping features into a higher-dimensional space and projecting them back into the original space. Finally, a residual structure is leveraged to solve the gradient degradation problem and two layer-normalization are used to make training more stable.

4.2.2 Model Architecture

The structure of the proposed model is shown in Figure 4.2. When an action occurs, the scenes before and after the action are generally different, especially in soccer videos. In order to locate actions by detecting scene changes before and after an action, video

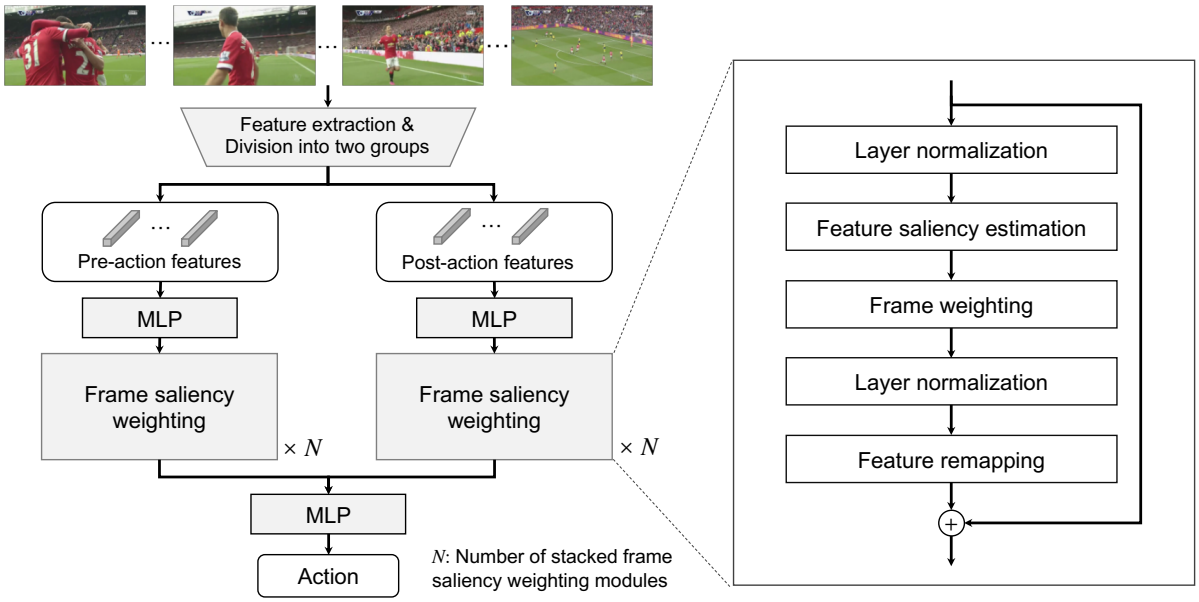


Figure 4.2: Proposed Model. *The proposed model consists of two encoders. An encoder is stacked by frame saliency weighting modules. A frame saliency weighting module consists of a frame saliency estimator, feature weighting, and remapping.*

clip features are split into two parts of the same length, pre-action and post-action. Two encoders are used to process each part, respectively. Given an image feature dimension L and the number of frames in a chunk N_f, the shape of the output of the feature extractor is $N_f \times L$. A fully connected layer reduces the dimensions of the features for efficiency and for a fair comparison among different feature extractors, which output features with different lengths. The size of the matrix output by the fully connected layer is $N_f \times E$, where E is the output dimension of the fully connected layer. The input of the remapping module is an $N_f \times E$ matrix, which is expanded to an $N_f \times 2E$ matrix by a fully connected layer. After an activation layer of ReLU, it uses another fully connected layer to remap the features to a $N_f \times E$ matrix. In this manner, the shape of the input data of the frame saliency module is the same as the output data. Therefore, an encoder can contain the N frame saliency weighting module. In the last multilayer perceptron (MLP) module, the output of two encoders is concatenated and averaged in the temporal direction. Finally, the action classification result is output through a softmax activation layer, a fully connected layer, and a sigmoid layer.

4.2.3 Implementation Details

A binary cross-entropy loss is used to train proposed model. An Adam optimizer with an initial learning rate of 0.003 is leveraged to optimize gradients of the model. To

Table 4.1: Action Spotting on SoccerNet-v2. *Evaluation results in terms of Average-mAP, where available, on the SoccerNet-v2 dataset with two different stride values during testing. N is the number of frame saliency weighting modules and E is the dimension of the features that output by the first fully connected layer.*

Model	Feature Extractor	Stride 1	Stride 20	Size/Param
NetVLAD	ResNet+PCA	31.4	-	7.50MB/0.66M
CALF	ResNet+PCA	40.7	-	6.64MB/0.58M
MaxPool	ResNet+PCA	18.6	-	0.11MB/0.01M
NetVLAD++	ResNet+PCA	50.7	46.7	7.50MB/0.66M
NetVLAD++	ResNet	53.4	48.1	19.50MB/1.70M
Ours $[N = 1, E = 512]$	ResNet+PCA	**56.0**	**51.0**	27.24MB/2.38M
Ours $[N = 1, E = 64]$	ResNet	54.0	49.5	**1.93MB/0.17M**
Ours $[N = 2, E = 256]$	ResNet	56.9	53.0	18.19MB/1.59M
Ours $[N = 1, E = 512]$	ResNet	**57.3**	**53.1**	66.29MB/3.16M

reduce the false detection rate, low-confidence predictions are filtered using non-maximum suppression (NMS), which is the same as in previous work [34, 18, 20, 35]. The batch size is set to 64 and the chunk size is set to 15 seconds. The non-maximum suppression (NMS) [34] is carried out over a 30 second window with a threshold 0. The model obtaining the highest mAP value on the validation dataset is used as the final model. It is used to evaluate on the test dataset.

4.3 Experiments

In this section, the performance of the proposed model is evaluated. The confidence graph as well as the saliency graph are visualized to interpret the results and conduct extensive experiments to analyze the influence of chunk size and the number of encoders, etc. To evaluate the effectiveness of the frame saliency weighting module, the fixed saliency is compared with the proposed learnable saliency. Furthermore, to evaluate the generality of the proposed method, the proposed module is tested on action spotting, temporal action proposal generation, and video classification tasks using SoccerNet-v1, Activity-v1.3, and UCF101, respectively.

4.3.1 Datasets

To demonstrate the applicability of the proposed model to different public datasets and different tasks, the evaluation results are presented on four datasets, SoccerNet-v1 [34], ActivityNet v1.3 [9], UCF101 [89], and SoccerNet-v2 [20]. Experiments were conducted for three video understanding tasks, action spotting, temporal action proposal generation, and action recognition.

■ SoccerNet-v2

The public SoccerNet-v2 dataset [20], which contains video footage and annotations of 500 soccer games, is used to train and evaluate the method. It is split the dataset into training, validation, and testing sets following the same procedure as the original paper (300, 100, and 100 games, respectively). The frame rate of videos is 2 frames for each second. The ground truth for each frame is a label vector. The label vector contains 17 different action labels, as well as a label for the background. The action label in the label vector is set as 1 if the corresponding action occurs in the chunk, and other labels are set as 0. If none of the 17 actions appears in the chunk, then the background label is set to one. If there is a *goal* action and two *play out of ball* actions happen in the chunk, the labels of *goal* and *play out of ball* are 1 and other labels are 0 in the label vector.

■ SoccerNet-v1

SoccerNet-v1 is the predecessor of SoccerNet-v2, released by the same authors. It has 6,637 temporal annotations, including three classes of actions (goal, card, and substitution). SoccerNet-v1 contains fewer annotations and thus longer intervals between actions than SoccerNet-v2. There are no 120-second frame intervals containing more than five actions in the SoccerNet-v1 dataset. On the contrary, SoccerNet-v2 has up to 14 actions in 120-second intervals.

■ ActivityNet v1.3

ActivityNet v1.3 [9] is a large-scale dataset consisting of 19,994 videos with 200 activity classes for action recognition, temporal action proposal generation and detection.

■ **UCF101**

The UCF101 dataset consists of 13,320 video clips, which are classified into 101 categories. All the videos are collected from YouTube and have a fixed frame rate of 25 FPS with a resolution of 320 × 240.

4.3.2 Evaluation Metric

The performance of action spotting is evaluated by the Average-mAP. If the distance between the ground truth timestamp and the predicted timestamp is less than Δ seconds, the prediction is considered positive. Δ is a threshold ranging from 5-60 seconds using a 5 second step size. The average precision (AP) is calculated for each action class and each Δ. The mean average precision (mAP) score is calculated by taking the mean AP over all classes with Δ. The Average-AP is the average of 12 AP values calculated over 12 tolerances Δ for each class. The Average-mAP metric is the average of 12 mAP values calculated over 12 tolerances Δ.

4.3.3 Performance on Video Understanding Tasks

■ **Action Spotting on SoccerNet-v2**

Table 4.1 shows the results of the proposed model as well as the models from the literature. The proposed model achieves 57.3% Average-mAP, representing an absolute increase of 3.9% over the previous state-of-the-art. NetVLAD [81] and MaxPool pooled features regardless of their temporal relationship between frames. The NetVLAD++ model uses two NetVLAD pooling modules to handle the first and second half of every chunk, respectively. However, it does not consider the importance of frames within each half chunk, treating them equally. In contrast, the proposed model uses the saliency of frames as feature weights to focus on keyframes and learn the importance of each frame.

Performance decreases when increasing the stride value from 1 to 20 frames, however, with an Average-mAP of 53.1% it is an absolute 5% higher than NetVLAD++. This means that the model can run the inference at a significantly higher speed for a similar average mAP value. Additionally, the model still outperforms NetVLAD++ in terms of Average-mAP at one-tenth of the model size. Consequently, the frame saliency weighting module is an elegant and efficient structure for action spotting.

Table 4.2: Results on SoccerNet-v1. *The Average-mAP is used a the evaluate metric.*

Method	Feature Extractor	Average-mAP
NetVLAD	ResNet+PCA	49.7
CALF	ResNet+PCA	62.5
NetVLAD++	ResNet+PCA	61.1
Ours	ResNet+PCA	**65.0**
Ours	ResNet	**68.1**

■ Action Spotting on SoccerNet-v1

In this experiment, the Average-mAP is reported for evaluation. The results of SoccerNet-v1 are shown in Table 4.2. Consistent with the results of SoccerNet-v2, the proposed method achieves the highest Average-mAP with 68.1% using ResNet features, an increase of +7.0% over NetVLAD++. An interesting observation is that in contrast to the SoccerNet-v2 dataset, CALF [18] outperforms NetVLAD++ on the SoccerNet-v1 dataset. The loss function in CALF defines six temporal segments around each ground-truth action to include temporal relationships. When actions occur frequently in a dataset such as SoccerNet-v2, their temporal segments may overlap, resulting in decreasing performance. Therefore, CALF is better suited for datasets with fewer annotations and longer intervals between actions, such as SoccerNet-v1.

■ Temporal Action Proposal Generation on ActivityNet

Using ActivityNet v1.3, the methods are evaluated by the performance of temporal action proposal generation. To adapt the approach to the task, the saliency weighting module is added as the first module in SSTAP [109], named by the SSTAP (+saliency) model. SSTAP is a temporal actions detection method based on self-supervised learning. The average recall (AR) is measured for different average numbers of proposals (AN) as AR@AN, and calculate the area under the AR *vs.* AN curve (AUC) as the metric on ActivityNet v1.3, where AN varies from 0 to 100. The results are shown in Table 4.3. SSTAP with the added saliency module improves the AUC score by 0.1%, confirming the effectiveness of the saliency module for the task of generating temporal action proposals and improving video features. In addition, using the proposed module, the importance of frames could be provided to interpret the prediction of existing models.

Table 4.3: Temporal Action Proposal Generation on ActivityNet v1.3 [9]. *The results are obtained on fully-supervised training. SSTAP (+saliency) use frame saliency weighting module to improve video presentation before the SSTAP model.*

Method	AUC (%)
SSTAP	67.5
SSTAP (+saliency)	**67.6**

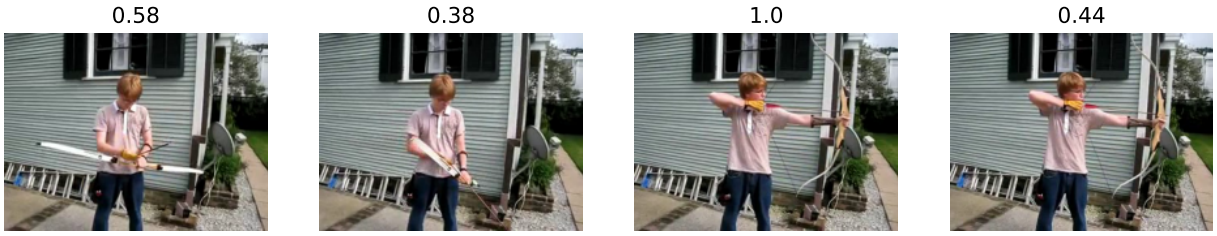


Figure 4.3: Saliency Score of the Archery Action.

■ Action Recognition on UCF101

To confirm the effectiveness of the proposed model for action recognition, another model is developed by combining the proposed module with SlowFast [29], denoted by SlowFast(*), and compare it with SlowFast using the UCF101 [89] dataset. The SlowFast is pre-trained on Kinetics 400 dataset [12]. As shown in Table 4.4, the performance is improved by 3.2% by using the proposed model as another branch. The proposed model could capture important information that the SlowFast model might miss.

To evaluate the interpretability of the proposed method, the proposed method generates some saliency scores of the archery action using the video from the UCF101 dataset, as shown in Figure 4.3. The durations of the videos from the UCF101 dataset are short and each video only contains one type of action. Therefore most of the frames in the videos are related to the action label. The saliency scores of these videos are higher than other datasets. The saliency score is higher in the third frame, where a man is holding a bow and an arrow than in other frames. The frame is related to the action of archery. Therefore, the proposed model could recognize the action by locating keyframes.

Table 4.4: Action Recognition on UCF101. *SlowFast(*) is the combination model of SlowFast and the proposed model.*

Model	Accuracy
SlowFast	85.4
SlowFast(∗)	**88.6**

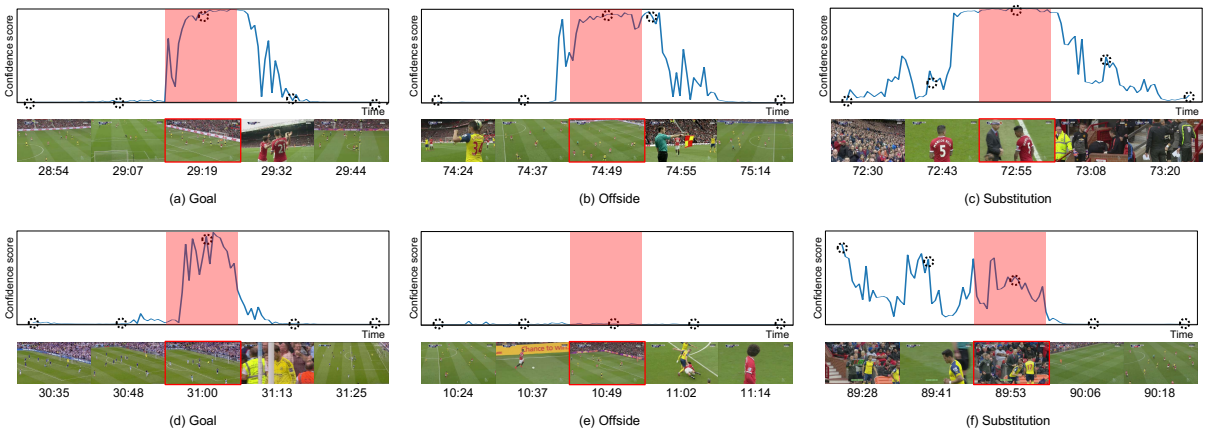


Figure 4.4: Confidence Score Examples. *Labeled actions from the SoccerNet-v2 dataset [20] are shown in frames marked by red boxes. The confidence scores for the adjacent frames are marked by circles in the graph. The translucent red rectangle indicates the range within 5 seconds from the ground truth timestamp.*

4.3.4 Interpretability Based on Frame Saliency Scores

■ Confidence Score

To analyze the experiment results, the results is visualized using a confidence score graph as shown in Figure 4.4. The confidence score is the prediction of the model before being processed by NMS. From (a) and (d) in Figure 4.4, for the *goal* action label, the confidence is high after the action itself, corresponding to the celebration scenes of players and fans. *Offside* actions typically include scenes where a referee raises a flag. In Figure 4.4 (b), the scene where a referee raises the flag occurs for 6 seconds after the offside action itself. In Figure 4.4 (e), no referee scene appears, and the confidence of an offside action is low. Figure 4.4 (c) shows a *substitute* action, and confidence is high for the frames 10 seconds before and after the action, as the camera typically focuses for several seconds on the player running to the sideline and the substitute player entering the field. The confidence in the substitution action is close to zero after a substitution, as shown in Figure 4.4 (f). The confidence score could indicate the scenes related to specific actions.

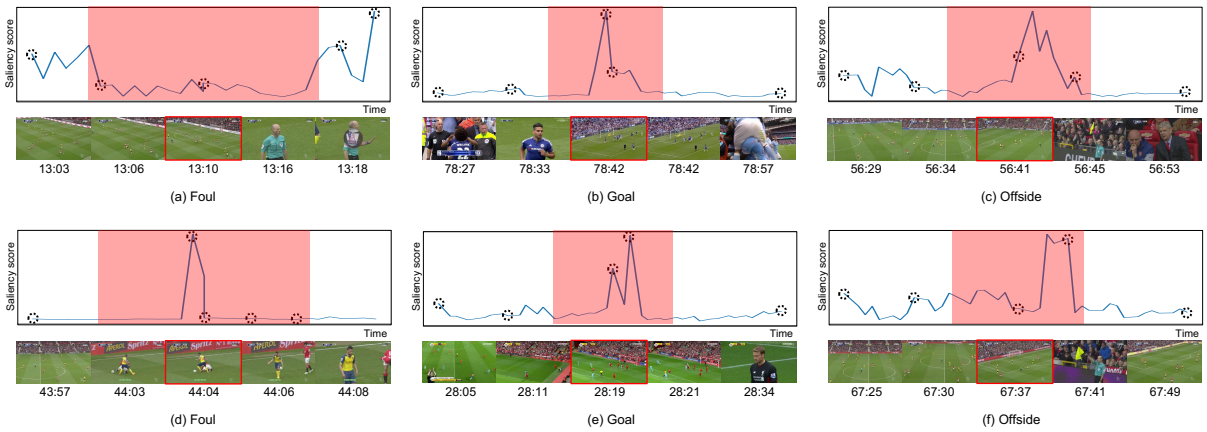


Figure 4.5: Saliency Score Examples. *Labeled actions from the SoccerNet-v2 dataset [20] are shown in frames marked by red boxes. The saliency scores for the adjacent frames are marked by circles in the graph. The translucent red rectangle indicates the range within 5 seconds from the ground truth timestamp.*

■ Saliency Score

To explore whether the saliency score is critical for action spotting, the saliency score graphs of six examples are plotted. The saliency score is calculated by the last frame saliency estimation module in the model. The stride length is set to 15 seconds in inference, equal to the chunk size used in training and concatenate the saliency scores in every chunk to obtain the saliency of all frames. From Figures 4.5 (a) and (d), it is observed that high-saliency scores appear at the time when a referee appears or players fall to the ground, which are salient features to spot a *foul* action. For *goal* actions, the model focuses on the frames in which the ball enters the goal. Because the saliency score is high on such frames, shown in Figures 4.5 (b) and (e). In Figures 4.5 (c) and (f), the frames in which a referee raises a yellow flag have high saliency scores, which are focused when locating off-side action. Based on these results, the saliency score of the frames that are related to actions is high. Therefore the frame saliency weighting module can focus on action-related frames and improve video representation by using the saliency scores as feature weights. Furthermore, the saliency score can be helpful in interpreting the prediction results and helps the proposed model be applied in the areas where interpretability is required. The frame-level saliency scores are helpful to improve model performance.

4.3.5 Analyses

■ Influence of Chunk Size and Number of Encoders

The effectiveness of the chunk size and the number of encoders are analyzed using the Average-mAP metric. The results are presented in Table 4.5.

Chunk Size. Because the model is trained to recognize action in every chunk, the chunk size is a necessary hyper parameter. If the chunk size is too large, a chunk would contain multiple actions that can affect each other and make the training challenging. If there is at most one action within a chunk, the adjacent chunks do not affect each other. The model needs to memorize $C + 1$ types of patterns, where C is the number of action classes in the dataset. If at most two actions occur in a chunk, the model needs to memorize $C^2 + 1$ types of patterns. On the other hand, if the chunk size is too small, video chunks where no action happens would increase and result in a more unbalanced dataset. Therefore, selecting an appropriate chunk size is important for action spotting. As shown in Table 4.5, 15 seconds is an appropriate chunk size, which is also reported in NetVLAD++[35]. The appropriate chunk size is determined by duration of frames which are related to specific actions. Detailed analysis has been done in the Chapter 3.

Number of Encoders. When an action occurs, the scene of the video would change. For example, before a goal action, players are running to the goal, while after a goal is scored, players celebrate and assemble. A model with two encoders is able to detect such scene changes easily, by learning from pre-action and post-action scenes, respectively. The model with two encoders performs best, as shown in Table 4.5. Models with three or four encoders increase the complexity of model structure while not further improving accuracy. When increasing the number of encoders with a fixed chunk size, the temporal segments fed into each encoder will be smaller and they will contain less temporal information. It increases the difficulty in capturing temporal correlations among frames in a chunk.

■ Influence of Model Architecture and Feature Extractor

The effect of the model and feature extractor are analyzed. The proposed module is compared with self-attention mechanism using the Average-mAP metric. The results are presented in Table 4.6 and Table 4.7.

Model Architecture. The proposed model is compared with NetVLAD++ [35] and a transformer-based model [101] using different feature extractors. As shown in Table 4.6, the proposed method achieves the best performance. NetVLAD++ leverages two NetVLAD encoders to learn temporal information, however it could not capture the importance of each frame. The transformer-based model only has one encoder, which leads

Table 4.5: Comparison of Chunk Sizes and Numbers of Encoders. *The Average-mAP is calculated for different chunk sizes from 10 to 30 seconds, while changing the number of encoders from 1 to 4, using ResNet-152 as the feature extractor. The highest value was obtained for a model using two encoders and a chunk size of 15 seconds.*

Chunk size (s)	Number of encoders			
	1	2	3	4
10	50.98	54.92	54.96	54.30
15	52.90	**57.32**	55.51	55.47
20	50.81	54.97	53.69	54.01
25	48.12	53.30	53.03	51.41
30	45.69	50.99	49.55	49.52

Table 4.6: Comparison of Model Architectures and Feature Extractors *in terms of Average-MAP. The proposed method achieves the best performance in all choices of the feature extractor.*

Feature Extractor	Transformer	NetVLAD++	Ours
ResNet+PCA	47.8	50.1	**56.0**
ResNet	48.3	53.4	**57.3**
Embedding	73.8	74.1	**75.0**

to the difficulty of locating the frame changes before and after actions.

Feature Extractor. The embedding features extracted by five models [128] greatly improve the performance of all methods, highlighting the importance of efficient video representations for the action spotting task. For any feature extractor, the proposed method achieves a better result than other models. Consequently, the proposed method would improve the presentation of video features by increasing the weights of keyframes.

■ Comparison with Self-attention Model.

In order to compare with self-attention, two self-attention-based models with a transformer encoder and two transformer encoders are developed, respectively. The method reaches a higher Average-mAP, as presented in Table 4.7. I consider that it is because the frame saliency weighting module does not have three fully connected layers to calculate the vectors K, Q, V in a transformer encoder [101]. The calculation of K, Q, V changes the mapping results of the feature extractor and reduces the ability of capturing the similarity among frames. In addition, the three fully connected layers increase the model parameters. Different from them, the model learns to calculate the saliency of each frame

Table 4.7: Comparison with Self-attention. *ResNet-152 is used as a feature extractor. The two self-attention based models with an encoder and two encoders are developed respectively.*

Model	Average-mAP	Size (MB)
Self-attention (1 encoder)	48.3	97.2
Self-attention (2 encoders)	53.0	249.0
Ours (2 encoders)	**57.3**	**66.3**

Table 4.8: Comparison with Keyframe Extraction Methods. *The Average-mAP on SoccerNet-v2 is used as the metric and ResNet152 with PCA is used as the feature extractor. The chunk size is 15 seconds, and frame rate is 2.*

Method	Average-mAP
Sampling-based	49.0
Shot-based	38.0
Cluster-based	46.9
Ours	**56.0**

based on the similarities of the learnable frame features.

■ Influence of Frame Extraction Methods

To further demonstrate the effectiveness of the method, three frame extraction methods are leveraged to select frames before feeding features into the proposed model. The proposed model is compared with different methods. Keyframe extraction is an efficient method used to clearly express the important contents of a video file by extracting a set of representative frames and removing the duplicated ones. The techniques of keyframe extraction can be classified into three main classes: sampling-based, shot-based and clustering-based techniques [83]. The implementation details are as follows. The sampling-based method randomly selects 10 frames from every chunk that contains 30 frames. The shot-based method splits every video chunk into several shots and uses the center frame in each shot as a keyframe. The cluster-based method clusters all 30 frames in every chunk into 10 frames. The model without a keyframe extraction module achieves the best performance, as shown in Table 4.8. Selecting frames would change the original temporal information and influence the performance of action spotting. Compared to them, the method can focus on the keyframes and learn to locate keyframes based on the inter-frame similarity and annotation labels automatically without changing the original temporal relationship. Therefore, the proposed method achieves the best performance.

Table 4.9: Comparison of Fixed Saliency and Variable Saliency. *In the model with fixed saliency, all saliency scores are set to 1 to ignore the influence of saliency.*

Model	Average-mAP
Fixed saliency	49.5
Variable saliency	**56.0**

Table 4.10: Comparison NetVLAD++ with and without proposed method. *NetVLAD++(*) denotes NetVLAD++ with the saliency weight module. Training time is the time used to obtain the highest mAP on the validation dataset.*

Model	Average-mAP	Training time(Second)
NetVLAD++	53.4	368.9
NetVLAD++(∗)	**54.5**	**292.7**

■ Importance of Learnable Saliency

The performance of the method is compared with fixed and learnable saliency on SoccerNet-v2 dataset to analyze the influence of the proposed saliency weighting module. The features are extracted using ResNet with PCA. To remove the influence of the saliency score, all saliency scores in the pre-trained model are set as 1. As a result, the weight of every frame is equal, and the influence of saliency score can be ignored. Without saliency weighting, the average mAP of the model drops to 49.5%, a decrease by absolute 6.5% as shown in Table 4.9. The saliency weighting module improves the Average-mAP for action spotting by weighting frames using learnable saliency scores.

■ Efficiency of Saliency

To prove that the frame saliency weighting module could find the keyframes and improve feature representation by weighting features through saliency scores, a model is introduced by adding a frame saliency weighting module before the NetVLAD++ model, denoted by NetVLAD++(∗). The learnable frame saliency weighting module could be seen as a preprocessing module. As shown in Table 4.10, by using the frame saliency weighting module to improve frames, the performance of NetVLAD++ is improved by 1.1%. Additionally, by using frame saliency, the training time is reduced. As presented in Figure 4.6, NetVLAD++ obtains the highest mAP at epoch 20 using 368.9 seconds, whereas NetVLAD++(∗) obtains the highest mAP at epoch 11 with 292.7 seconds. The frame saliency module gives larger weights to keyframes, which encourages the model to

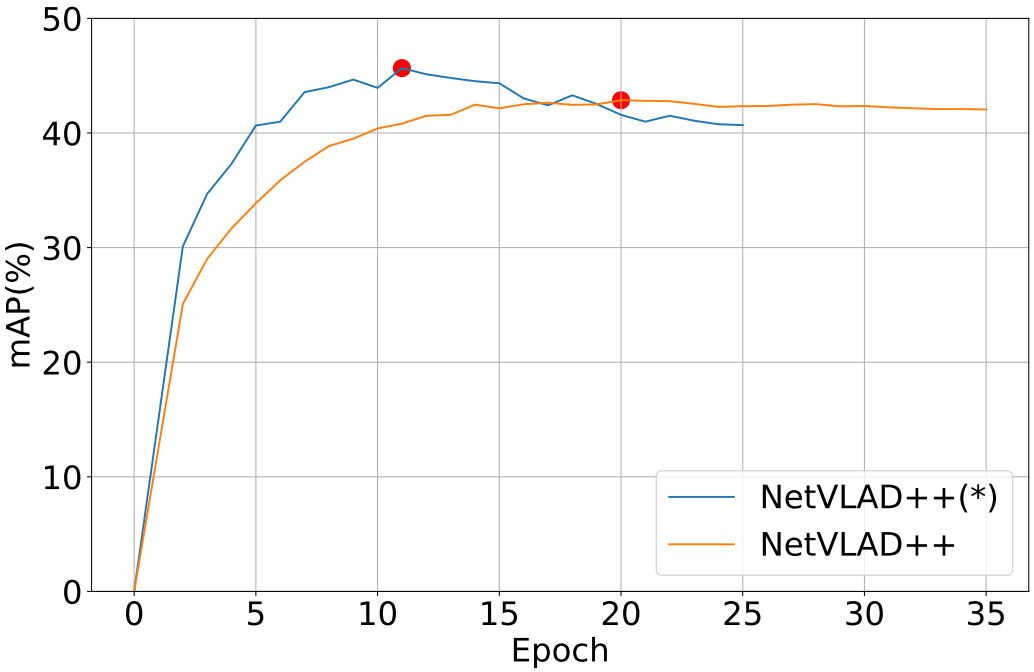


Figure 4.6: Change of mAP with respect to Epochs on the Validation Dataset. *The changes of NetVLAD++ and NetVLAD++(∗) during training are shown respectively.*

focus on important frames and understand video content from keyframes effectively.

4.4 Conclusion

The problem of redundant information in videos is addressed by paying more attention to important frames. The frame saliency weighting module, which weights frames according to feature similarity, is proposed to reduce the influence of redundant frames and improve frame saliency and video representation. The performance of the proposed model is analyzed and the proposed model achieved state-of-the-art accuracy on SoccerNet-v2, obtaining 57.3% Average-mAP. Additionally, the proposed model obtains the 75. 0% average mAP using embedding features. Furthermore, the saliency weighting module can be effectively applied to existing video understanding methods.

Chapter 5

Interpretable Video Anomaly Detection Based on Video Captions

The frame saliency module is proposed to improve video representation and interpreted predictions based on frame saliency in the last chapter. However, there is a limitation of the score-based interpretability. It is the importance scores of frames in a video cannot be compared with the importance scores from the other videos directly. The scores are calculated based on the frame features in a video. Therefore, when the input is a long video, the calculation would be heavy. If a similar action happens in the video at different timestamps, the importance scores of different actions may not be correct.

Therefore, a novel interpretation method is required to resolve these problems. Different from scores, language can be understood directly by humans. With the development of natural language processing (NLP), AI models can more correctly understand the semantic meaning of each sentence. It is a fundamental technology for text-based interpretability. To comprehensively evaluate the idea, the video anomaly detection task is used as the objective goal of this work. The video anomaly detection task is a specific action detection task. Most video anomaly detection approaches are based on non-semantic features, which are not interpretable, and prevent the identification of anomaly causes. To address the issue, a caption-guided interpretable video anomaly detection framework that explains the prediction results based on video captions (semantic) is proposed. It utilizes non-semantic features to fit the dataset and semantic features to provide common sense and interpretability to the model. It automatically stores representative anomaly prototypes and uses them to guide the model based on similarity with these prototypes. Specifically, we use video memory to represent the content of videos, which includes video features (non-semantic) and caption information (semantic). The proposed method generates and updates a memory space during training, and predicts anomaly scores based on the memory similarities between the input video and the stored memories. The stored captions can be used as descriptions of representative anomaly actions. The proposed module can be easily integrated with existing methods. The interpretability and reliable

detection performance of the proposed method are evaluated through extensive experiments on public benchmark datasets.

The structure of this chapter is as follows. Section 5.1 introduces the problems in existing anomaly detection tasks. Section 5.2 shows the proposed method. Section 5.3 shows the results of the experiment and analyzes them. Section 5.4 provides a conclusion to this chapter.

5.1 Introduction

The large number of monitoring videos has made video anomaly detection an increasingly daunting task for human operators. Consequently, video anomaly detection has become more crucial than ever before. Furthermore, depending on how anomalies are defined, anomaly detection techniques can be applied to various video understanding tasks, such as action detection, action recognition, and video classification. Given its significance, video anomaly detection has been extensively researched for decades. However, developing a video anomaly detection model is challenging, as the definition of an anomaly is subjective and depends on the specific application scenario. For instance, *fighting* is considered an abnormal behavior in daily life, yet it is a normal action in boxing matches. Additionally, the predictions of models lack interpretability. Most previous approaches [124, 30, 93, 111] extract visual features from videos based solely on pixel changes across frames, without understanding the video content. This results in unexplainable prediction results and limits their practical application.

To address these problems, it is necessary to understand the semantic content of the videos. Semantic features are similar to human understanding. People understand video content based on the information of objects in the image and the interactions between them. Such information is typically included in video captions. Furthermore, captions are more easily understood than other explanation methods [84]. For example, visual information-based approaches, such as Gradient-weighted Class Activation Mapping (Grad-CAM [84]). Grad-CAM offers limited visual interpretability since it cannot provide clear boundaries between objects of interest and the background.

As shown in Figure 5.1, captions include important information needed for anomaly detection. Therefore, detection models can easily identify the positions of related objects from the image and video caption. Additionally, semantic features tend to be more stable than video features because they are less affected by object appearance or capture conditions. The caption embeddings generated by a pre-trained language model also contain common sense knowledge. For example, the semantic similarity between *fighting* and *vi-*

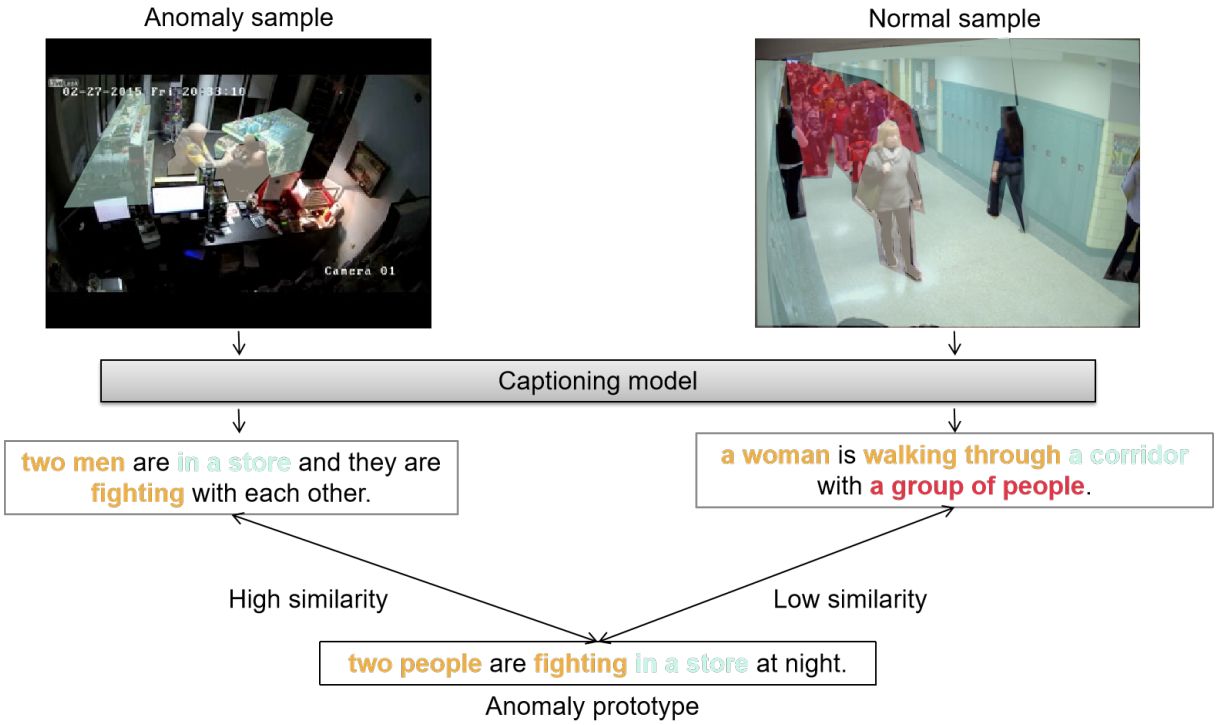


Figure 5.1: *Semantic similarity. The similarity based on semantic information provided in captions is closer to the understanding of humans because it contains high-level information, such as objects and their interactions. Images from [91].*

olence is greater than that between *walking* and *violence*. Therefore, video captions and caption embeddings can be utilized as semantic features to identify abnormal situations based on the similarity of *video memories*, where each memory contains a video feature, a video caption, and a caption embedding. Representative anomaly video memories are stored as the definitions of abnormal situations, guiding the anomaly detection model and explaining its predictions. This video memory is used to represent the content of a video. And the similarity among the memories is used to guide the model and predict anomaly scores.

In summary,

- To address the lack of interpretability in anomaly detection models that rely on non-semantic features, the video memory is introduced to represent video content and a novel caption-guided interpretable framework is proposed for video anomaly detection, which utilizes text as semantic features to guide the model and explain predictions.

- The anomaly actions stored in the memory space are visualized to understand what constitutes an anomaly for the models. To analyze the utility of the proposed

method and demonstrate the necessity of video captions, extensive experiments are conducted.

- The method achieves state-of-the-art performance on the ShanghaiTech [64] dataset and shows the interpretability and efficiency of the proposed approach using the UCF-Crime [91] dataset.

5.2 Caption-guided Model for Video Anomaly Detection

In this section, the caption-guiding module, the architecture of the proposed model, and implementation details are introduced.

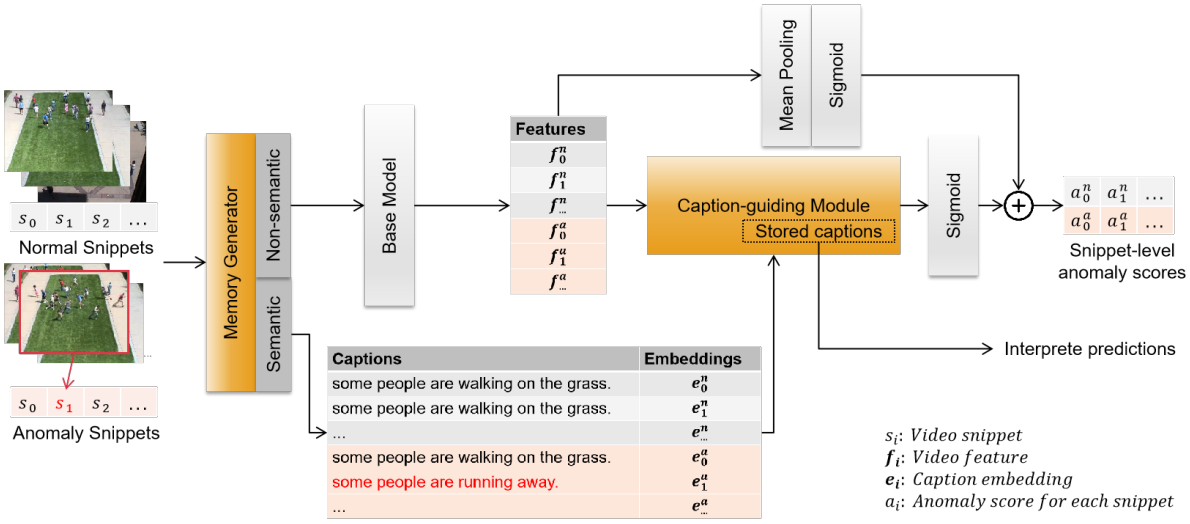


Figure 5.2: *The overview of the proposed method. The proposed method contains two main modules, a memory generator and a caption-guiding module. The memory generator extracts semantic and non-semantic features as the video memory to represent video content. The caption-guiding module stores anomaly video memories to guide the model and interpret predictions using video captions. The snippet images on the left are cited from [44].*

5.2.1 Caption-guiding Module

To enable the interpretability of text-guided models, a novel module is introduced. It can utilize text as part of the video representation and guide the model based on this representation, allowing the text to generate interpretable predictions. The anomaly scores are calculated based on similarities with the stored memories. Since similar memories

provide redundant information, it is important to store only representative memories in the memory space to ensure each memory represents a distinct anomaly situation. On the other hand, since the parameters of the base model change during training, the video memory will also evolve. Therefore, old memories should be removed to optimize the memory space. The caption-guiding module has three key functions: prediction of anomaly scores, generation of the memory space, and optimization of the memory space.

Prediction of anomaly scores. The anomaly scores AS are calculated based on the memory similarities between the input memory $\mathbf{m}^{input}$ and the stored memories $\{\mathbf{m}_i \mid \mathbf{m}_i \subset M\}$ in memory space M. $\mathbf{m}^{input}$ consists of a video feature $\mathbf{f}^{input}$, a video caption $\mathbf{c}^{input}$, and a caption embedding $\mathbf{e}^{input}$. To accurately characterize the relationship of video content, the memory similarity contains the non-semantic similarity based on the video features and the semantic similarity based on the caption embeddings. The calculation of non-semantic similarity s^f is presented as follows:

$$s_i^f = \mathbf{f}^{input} \cdot \mathbf{f}_i \,, \tag{5.1}$$

$$s^f = \text{mean}\left(\underset{\text{max}}{topK}(s_0^f, s_1^f, s_i^f, ..., s_I^f)\right), \quad i \subset [0, I] \,, \tag{5.2}$$

where I is the number of memories stored in the memory space, K is a hyperparameter, and s_i^f is the non-semantic similarity between $\mathbf{f}^{input}$ and $\mathbf{f}_i$. The number of memories I stored in the memory space changes during training.

To reduce the influence of outliers, we use the mean of the $topK$ similarities instead of the maximum or mean of all memories to represent the non-semantic similarity s^f. The semantic similarity s^e based on caption embeddings is calculated as follows:

$$s_i^e = \frac{\mathbf{e}^{input} \cdot \mathbf{e}_i}{\|\mathbf{e}^{input}\| \|\mathbf{e}_i\|} \,, \tag{5.3}$$

$$s^e = \text{mean}\left(\underset{\text{max}}{topK}(s_0^e, s_1^e, s_i^e, ..., s_I^e)\right), \quad i \subset [0, I], \tag{5.4}$$

where s_i^e represents the semantic similarity between $\mathbf{e}^{input}$ and $\mathbf{e}^i$. The semantic features are utilized to calculate anomaly scores because the similarity of captions is more akin to human understanding than video features, and video captions can be directly interpreted. Additionally, the common sense included in the pre-trained language model can guide the model using the caption embeddings.

The anomaly score AS is calculated based on the non-semantic and semantic similarities:

$$AS = \frac{s^f + \theta s^e}{1 + \theta} \,, \tag{5.5}$$

where θ is a temperature parameter that adjusts the weight of the semantic similarity. Note that if the memory space is empty, the anomaly score is set to 0.5, as shown in Figure 5.3. The anomaly score AS also represents the memory similarity between the input memory $\mathbf{m}^{input}$ and the stored memories $\mathbf{m}_i$. When the input memory is similar to the stored memories, the content of the input video is similar to the representative anomaly situation, and consequently, the anomaly score AS would be high. The anomaly score AS is calculated based on both the semantic and non-semantic features. The non-semantic features can fit the training samples, while the semantic features can provide video understanding that is closer to common sense.

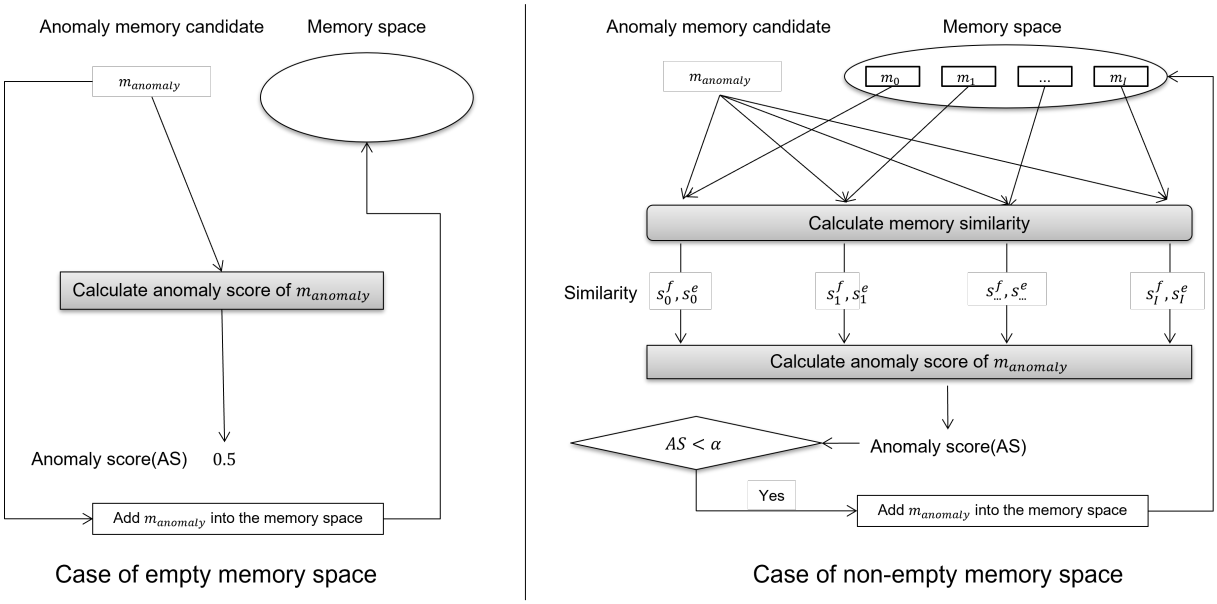


Figure 5.3: *Prediction of anomaly scores and generation of memory space. The left subfigure shows that when the memory space is empty, the anomaly score (AS) of the anomaly video candidate is 0.5, and the memory would be added to the memory space. The right subfigure shows that when the memory space is not empty, the anomaly score is calculated based on the memory similarity with all stored memories. If the anomaly score is greater than a threshold α, the anomaly memory candidate will be added to the memory space.*

Generation of memory space. The memory space is the core of this module. It stores anomaly memories and outputs anomaly scores based on similarities to the stored memories. It is generated for two purposes: to explain the prediction results and to guide the model in detecting anomalies. To make full use of memory space, each memory needs to represent a different anomaly situation, and the memory space needs to store representative and distinct memories.

There are two steps in adding a new memory to the memory space. The first step is to locate the anomaly memory candidates $\mathbf{m}_{\mathbf{anomaly}}$ from the anomaly videos. Anomaly

videos contain both anomaly snippets and normal snippets. If the memories of normal snippets are put into the memory space, the model would mistakenly classify normal actions as anomalies. Therefore, normal snippets from the anomaly videos should be filtered out.

The anomaly memory candidates are located by selecting the snippets from anomaly videos that have low similarities to the snippets from normal videos. Specifically, we calculate the memory similarities between the snippets of an anomaly video and the snippets from a normal video and take the mean of the memory similarities with the snippets from a normal video as the normal score for each snippet from an anomaly video. The snippet with the lowest normal score is selected as an anomaly snippet candidate from each anomaly video.

The second step is to decide whether each anomaly memory candidate needs to be added to the memory space. To store representative and distinct memories, only the memory candidates with low similarities to the stored memories should be added to the memory space. To find such memories, the memory similarities between an anomaly memory candidate and all stored memories $\{\mathbf{m}_i \mid \mathbf{m}_i \subset M\}$ in the memory space are calculated. This process is the same as the calculation of the anomaly score AS, therefore both calculations are completed concurrently, as depicted in Figure 5.3.

If the AS of an anomaly memory candidate is smaller than a threshold value α, then the input memory is considered to represent a new type of anomaly situation and add it to the memory space. α is initially set to 1 and is updated through the optimization of the memory space. In this manner, the memory space is generated automatically and filled with representative anomaly memories. Therefore, the memory space can guide the model in finding anomaly actions based on the memory similarities between the stored memories.

Optimization of memory space. Due to changes in the parameters of the base model during training, the video features $\mathbf{f}$ and the video memories $\mathbf{m}$ would also change. Consequently, the meaning of old memories may become outdated, and some stored redundant memories could misguide the model. To detect anomalies efficiently, distinct and representative anomaly memories should be kept, while redundant memories should be removed. The optimization of the memory space is shown in Figure 5.4. To find redundant memories, the redundancy Re_i of the i^{th} memory is calculated as follows:

$$Re_i = \text{mean}\left(\{{s^m}_{ij} \mid {s^m}_{i0}, {s^m}_{i1}, ..., {s^m}_{iI}\}\right); \quad j \neq i\,, \tag{5.6}$$

$${s^m}_{ij} = \mathbf{f}_i \cdot \mathbf{f}_j + \theta \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\|\|\mathbf{e}_j\|}\,, \tag{5.7}$$

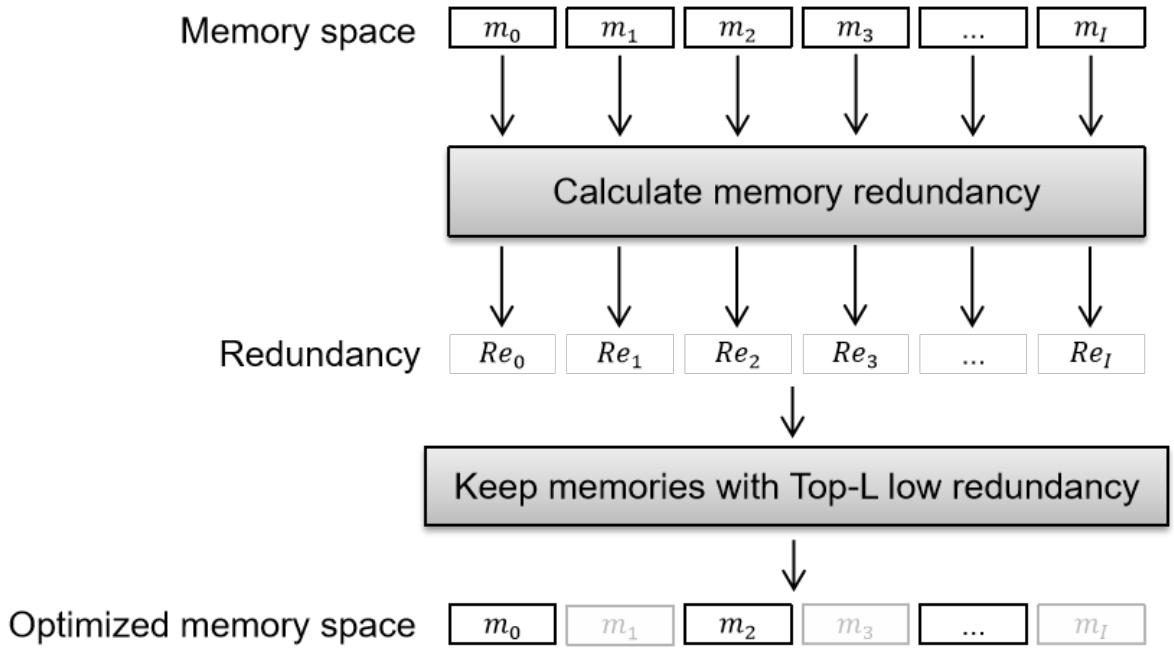


Figure 5.4: *Optimization of memory space. To optimize the memory space, the redundancy of each memory are calculated and only the L memories with the lowest redundancy are kept in the memory space.*

where ${s^m}_{ij}$ denotes the similarity between $\mathbf{m}_i$ and $\mathbf{m}_j$. i and j both represent memory numbers, with the same range of $[0 \sim I]$.

The memory m_i with a higher Re_i contains less useful information. The L memories with the $topL$ minimum redundancy are kept and the others are removed to optimize the memory space. Because the base model would update its parameters to fit the training dataset, the similarity of the video features would change. Therefore, the threshold α needs to be updated to adapt to these changes and suppress the addition of similar memories to the memory space. The threshold α is updated as follows:

$$\alpha = \max\left(\underset{\min}{topL}(Re_0, Re_1, Re_2, ..., Re_I)\right) , \tag{5.8}$$

where L is a hyperparameter to limit the number of memories retained. If the number of memories in the original memory space is greater than L, only L memories are stored.

5.2.2 Model Architecture

As shown in Figure 6.2, multiple instance learning (MIL) is employed to tackle the weakly-supervised video anomaly detection task, as the public datasets only contain video-level annotations. At first, an anomaly video and a normal video are split into 32 video

snippets, respectively. Subsequently, the frozen memory generator extract non-semantic and semantic features from the video snippets. Specifically, the pre-trained I3D [28] model is used to extract raw video features of length 2048. Additionally, the pre-trained Swin-BERT [62] model is employed to generate video captions and the pre-trained MPNet [88] employed to extract caption embeddings from these video captions. The MPNet is chosen for extracting caption embeddings because it is a widely-used model for calculating sentence similarity. A caption embedding is a vector of length 786. The base model is used to project raw video features into vectors of length 32 as the optimized video features. This base model is a simple composition of three fully-connected layers. Note that the base model could be replaced by any existing model capable of representing video content through feature vectors.

A video memory $\mathbf{m}$ is defined as the representation of a video snippet, which consists of an optimized video feature vector $\mathbf{f}$, a video caption $\mathbf{c}$, and a caption embedding $\mathbf{e}$. Feature $\mathbf{f}$ is a non-semantic feature, while $\mathbf{c}$ and $\mathbf{e}$ are semantic features. Video memories are fed into the caption-guiding module to generate the memory space, which stores important video memories related to anomaly actions. The stored memories are used to predict anomaly scores based on memory similarities, guide the model, and explain the predictions. A sigmoid layer is added after the caption-guiding module, a mean pooling layer, and another sigmoid layer is added after the base model. Finally, a residual structure is employed to reuse the optimized video features. The model outputs the snippet-level anomaly scores by combining the outputs from these two sigmoid layers. With the help of the stored video memories, the proposed method can locate anomalies based on semantic and non-semantic features, and provide interpretable predictions.

5.2.3 Implementation Details

Since most anomaly detection datasets only have video-level annotations, multiple instance learning (MIL) is used to train anomaly detection models, following previous work [91, 111, 93]. For a fair comparison, the I3D model [28] pre-trained on Kinetics-400 [12] is adopted for video feature extraction. On the ShanghaiTech dataset, the model is trained using the Adam [52] optimizer with a learning rate of 10^{-3}, following the training procedure of S3R [111]. On the UCF-Crime dataset [91], the model is trained using the AdaGrad optimizer with an initial learning rate of 0.1, reducing it by a factor of 10 after epochs 25 and 50, respectively. During inference, the memory space is not be changed. The anomaly score is calculated based on the similarity with the stored memories.

Regarding the hyperparameters of the proposed model, the top-K parameter is set to 5, the number of stored memories L is set to 7, and the temperature parameter θ is set

to 1. Additionally, memory space optimization is conducted every 3 iterations. THE optimization would be skipped if the number of stored memories falls below 10.

5.3 Experiments

In this section, extensive experiments are conducted to evaluate the performance and interpretability of the proposed method on two datasets: ShanghaiTech [64] and UCF-Crime [91]. Both datasets are used for weakly-supervised video anomaly detection.

5.3.1 Dataset and Evaluation Metric

■ Dataset

The ShanghaiTech dataset [64] contains 437 videos from 13 campus surveillance scenes. In this dataset 238 videos are used for training and 199 videos for testing in the weakly-supervised setting. The UCF-Crime [91] dataset contains 1900 surveillance videos covering 13 real-world anomalous classes such as robbery, explosion, and road accident. It contains 1610 training videos and 290 test videos. Compared to ShanghaiTech, which mainly includes pedestrian activities in a university setting, the scenes in the UCF-Crime dataset are more diverse and complex.

■ Evaluation Metric

For evaluating the model performance on video anomaly detection, the Area Under Curve (AUC) is calculated. It is a conventional threshold-independent metric, has been used in previous work [91, 111].

5.3.2 Performance of Video Anomaly Detection

To evaluate the performance of the proposed method for anomaly detection, MLP and S3R are used to evaluate the proposed method on ShanghaiTech and UCF-Crime datasets, respectively.

As shown in Table 5.1, when using S3R [111] as the base model, the proposed method achieves an AUC score increase of 0.21%, reaching state-of-the-art performance on the ShanghaiTech dataset. Because the proposed method can detect anomalies based on caption embeddings, allowing it to leverage semantic information to improve the existing method.

Table 5.1: *Comparison of frame-level AUC performance for video anomaly detection on the ShanghaiTech dataset. The proposed method uses S3R as the base model.*

Method	Feature	AUC (%)
GCN-Anomaly [124]	C3D [95]	76.44
GCN-Anomaly	TSN [107]	84.44
MIST [30]	C3D	93.13
MIST	I3D	94.83
RTFM [93]	C3D	91.51
RTFM	I3D	97.21
MSL [57]	C3D	94.81
MSL	I3D	96.08
S3R [111]	I3D	97.48
Ours (S3R)	I3D + Caption	**97.69**

Table 5.2: *Comparison of frame-level AUC performance for video anomaly detection on the UCF-Crime dataset. MLP is an MLP-based model that contains 4 fully connected layers. The proposed method uses an MLP model excluding the last fully connected layer as the base model.*

Method	Feature	Interpretable	AUC (%)
GCN-Anomaly [124]	TSN		82.12
MIST [30]	I3D		82.30
MLP	I3D		82.81
RTFM [93]	I3D		84.30
S3R [111]	I3D		**85.99**
Ours (MLP)	I3D+Caption		**84.64**

Moreover, the interpretability and performance of the proposed method are evaluated on the public UCF-Crime dataset for video anomaly detection, as presented in Table 5.2. Using a multi-layer perceptron (MLP) with the fully connected layers as the base model, the proposed approach achieves an improvement in the AUC score of 1. 79% over the standalone MLP model. The S3R model is not used as the base model on the UCF-Crime dataset due to the limitation of GPU resources. However, the method still reaches a comparable performance even with the MLP model as the base model, as video captions provide the necessary clues for anomaly detection. The UCF-Crime dataset contains untrimmed videos, which makes it difficult to generate accurate video captions. For example, many videos contain a logo scene and some scenes repeat several times within a video. This limitation reduces the effect of the video captions. Despite these challenges,

the proposed model can still interpret the predictions using the available video captions. More analyses are provided in Section 5.3.3.

5.3.3 Interpretability Based on Language

Interpretability is a critical function for an anomaly detection model, as it requires the model not only to detect anomalous actions but also to understand the video context. The proposed model represents the video content using semantic and non-semantic features, detects anomaly actions based on memory similarities with the stored anomaly memories, and explains the predictions via video captions. Various experiments are conducted to analyze the interpretability of the proposed method.

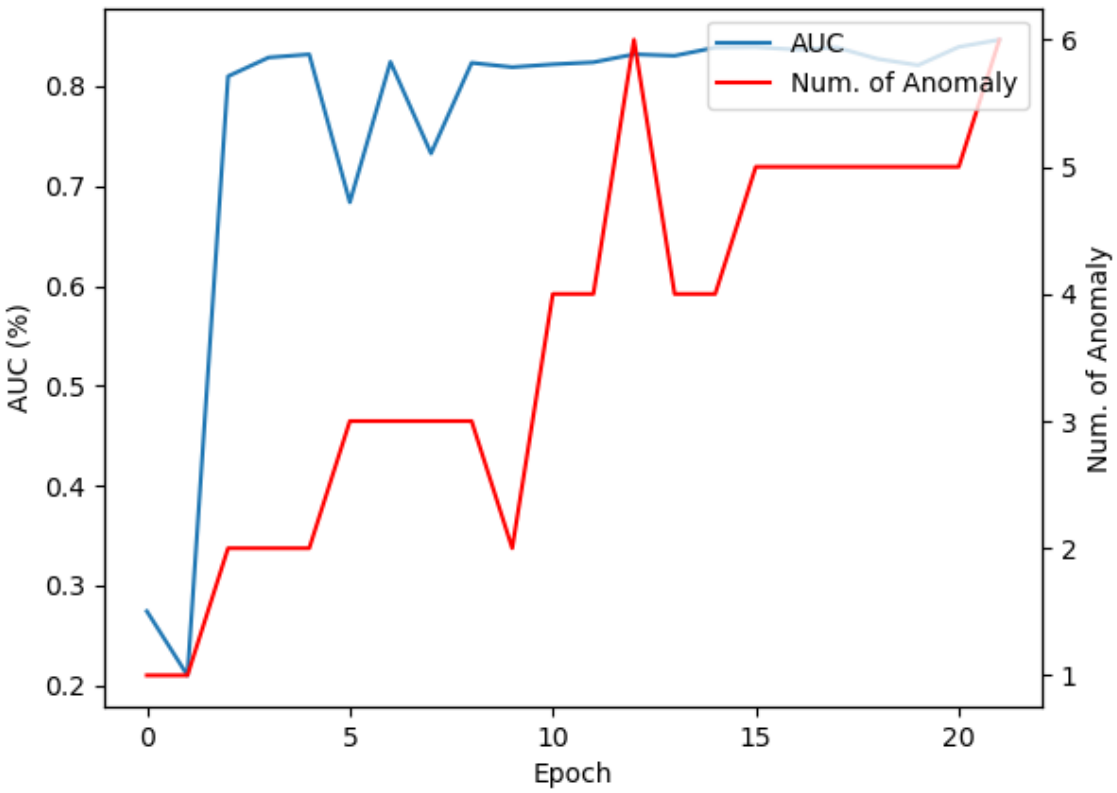


Figure 5.5: *Change of memory space during training. The influence of anomaly memories is evaluated by analyzing the relationship between the number of anomaly memories in the memory space and the AUC score.*

Meaningful predictions. As shown in Figure 5.6, only the moment when a man sets fire is annotated as an anomaly. However, the fire grew larger and the situation became

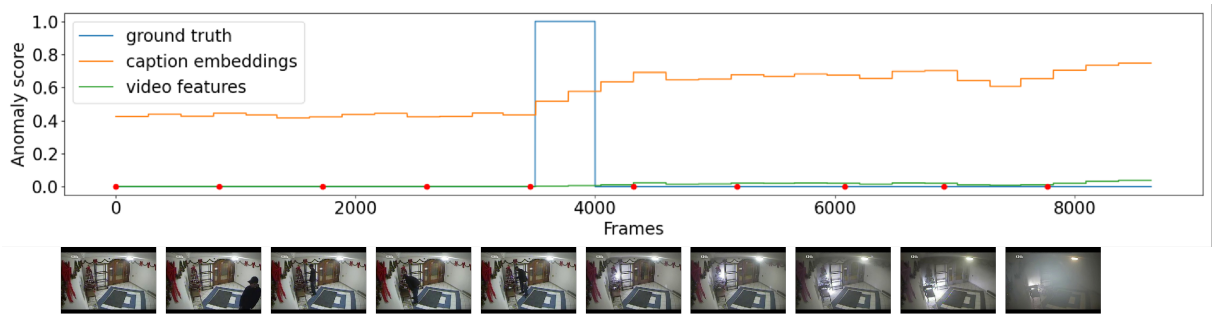


Figure 5.6: *Comparison of predictions from the models with caption embeddings and with video features. The figure shows video frames at the bottom, with the timestamps of selected frames plotted as red points. The graph displays the predicted anomaly scores of two models and the annotated ground truth. The blue line represents the ground truth of the anomaly. The green line represents the anomaly scores of the model that use video features as video memory. The orange line represents the anomaly scores of the proposed method that utilizes both video features and caption embeddings. The frames are taken from the UCF-Crime dataset [91].*

more dangerous after that initial moment. The model using only video features outputs small anomaly scores for the scenes where the fire grows larger. In contrast, the model using caption embeddings predicts increased anomaly scores in scenes where the room is full of smoke. This is because the semantic similarity between *smoke* and the descriptions of representative anomaly situations, such as *explosion* and *fire*, is large. The meaningful predictions from the model using caption embeddings are more suitable for real-world applications. The definition of anomaly and the annotations in datasets are subjective. If the model is trained solely to fit the annotation data, it would lack common sense and ignore some dangerous situations. However, using video captions to guide the model allows it to output more meaningful anomaly scores.

Change of memory space during training. The stored memories guide the model to detect anomaly actions based on the memory similarities. If there is a video memory related to *fighting* in the memory space, it helps the model to detect *fighting* actions. Therefore, if the memory space stores more anomaly memories, the performance will improve. The change in the number of anomaly memories in the memory space is analyzed and the performance is evaluated by the AUC score during training. After each training epoch, the memory space would be optimized if the number of memories is larger than ten. As a result of the optimization, only seven representative memories are kept, and each stored memory is determined whether it is an anomaly memory based on the corresponding caption. As shown in Figure 5.5, the number of stored anomaly memories increases, and the AUC score improves during training. The stored memories guide the model via semantic similarity to reach better anomaly detection performance.

Stored memories. For further analysis, several stored captions and frames are shown

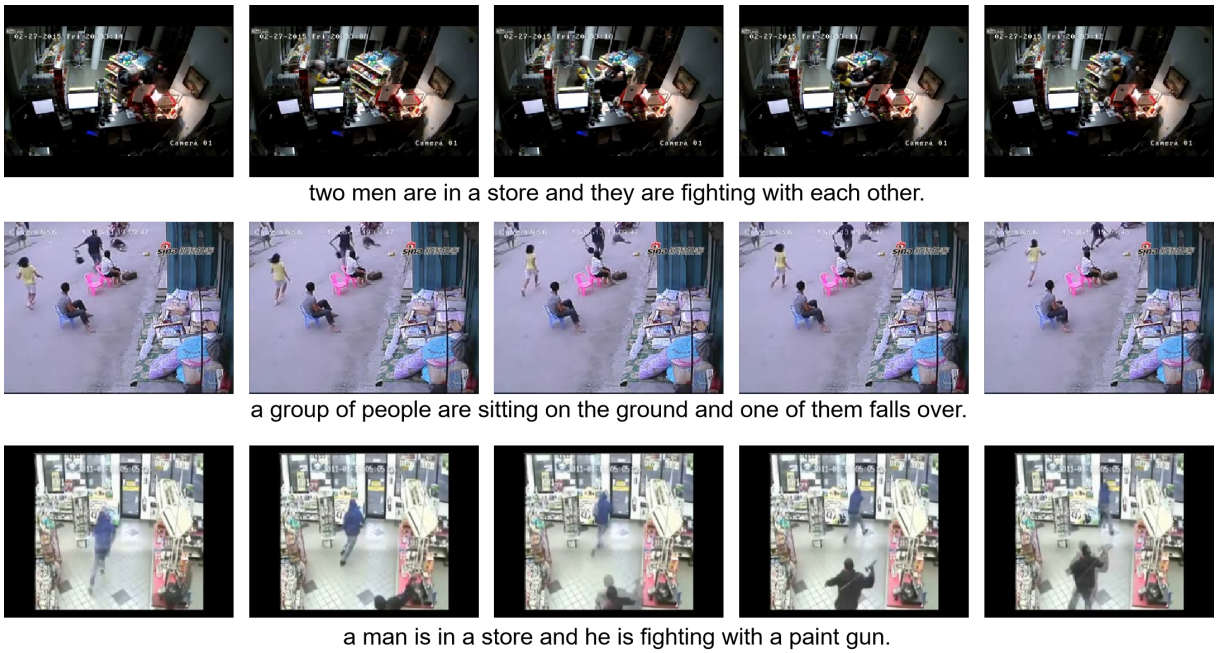


Figure 5.7: *Examples of stored video memories. Five frames are sampled from each video to show the video content. The video caption is generated by the memory generator and stored in the memory space. They are the definition of anomalies for the model. The video is from the UCF-Crime dataset [91].*

in Figure 5.7. The stored captions describe the anomaly actions, allowing the proposed method to recognize related anomaly actions through the memory similarities with the input video snippet. For example, the caption *two men are in a store and they are fighting with each other* contains the word *fighting*, which is associated with violence and has a high semantic similarity with video captions including violence-related words. By storing such memories in the memory space, the model is guided to detect anomaly actions based on these semantic similarities. Furthermore, an interesting phenomenon is observed. The memory space stores some captions that describe the scenes preceding the anomaly actions, such as *a person is throwing a package onto a door of a house*. These captions can help the model detect anomalies earlier. As training progresses, the memory space becomes more stable, as the updates to the base model become less frequent and slower. Eventually, the memory space holds the appropriate memories for effective anomaly detection.

Table 5.3: *Comparison with different memory types. To evaluate the performance of semantic features, three different features are leveraged as video memory.*

Memory type	AUC (%)
Caption embedding (CE)	62.40
Video feature (VF)	82.60
Ours (CE+VF)	**84.64**

5.3.4 Analyses

To analyze the importance of memory space optimization and the influence of the proposed method on training and inference time, additional experiments are conducted.

Strength of caption embeddings.

The method is the first to use visual captions (semantic features) for anomaly detection. Previous work extracted video features to detect anomalies. However, these features cannot be directly interpreted. In contrast, the proposed method introduce a semantic video representation by using video captions as a part of video memory to represent video content. To show the usefulness of semantic features, three different memory types are compared. They are video features (VF), caption embeddings (CE), and the proposed memory type (CE+VF), which contains both video features and caption embeddings.

The results are shown in Table 5.3. Using caption embeddings (CE) as memory, the AUC score is 62.4%. We attribute this to the fact that the untrimmed videos in the dataset lead to some incorrect video captions, limiting the effectiveness of the caption embeddings. Although CEs contain important information and common sense to guide the model, they have less information than video features (VF) and are influenced by inaccurate captions. In contrast, VFs include detailed information from the videos, allowing the model to fit the training samples and achieve an AUC of 82.6%. In contrast to VF, the proposed memory type (CE+VF) contains common sense information that guides the model without extensive training, enabling it to understand video content and achieve the best performance. Furthermore, the proposed memory type can leverage video captions to explain the prediction results, as shown in Figure 5.7. Moreover, using caption embeddings as part of the video memory allows the model to output meaningful predictions.

■ **Memory Space Optimization**

The optimization of the memory space removes redundant memories from the memory space and updates the threshold α to suppress the addition of similar memories. The three

Table 5.4: *Comparison of different optimization methods. The three calculation methods which based on three different feature types are compared. The fixed threshold α is compared with the variable threshold. Note that only the feature types used for optimization are changed and all models use CE+VF as the video memory.*

Feature type for optimization	Threshold α	AUC (%)
Video feature (VF)	variable	83.09
Caption embedding (CE)	variable	82.55
CE+VF	0.3	83.33
CE+VF	0.4	83.56
CE+VF	0.5	83.94
CE+VF	0.6	83.54
CE+VF	variable	**84.64**

Table 5.5: *Comparison of training and inference times. The number of video snippets processed per second is reported.*

Method	Training time (h)	Inference time (snippets/s)	AUC (%)
S3R	38.33	41.67 ± 2.26	97.48
Ours (S3R)	42.67	36.29 ± 1.92	97.69

methods of calculating memory redundancy are compared: based on the similarity of video features, based on the similarity of caption embeddings, and based on the similarity of video memories, respectively. Selecting redundant memories based on the video memories reaches better performance, as shown in Table 5.4. This suggests that leveraging video memory can correctly identify representative anomaly memories. To demonstrate the need for updating the threshold α, a variable threshold is compared with a fixed threshold. The experiments show that the variable threshold results in the best performance. This is because the base model projects the same video feature into different feature vectors to fit the training dataset during training, changing the distances among video memories. Therefore, the threshold is updated based on the stored memories to prevent adding similar video memories to the memory space.

■ Training and Inference Time

To assess the impact of the proposed method on both training and inference times, several experiments are conducted for comparing with the S3R model on the ShanghaiTech dataset. The method builds upon the S3R model as the base model. Both models are

trained for 15,000 epochs on an NVIDIA® A100 GPU and recorded the time taken to achieve optimal performance as the training time. As shown in Table 5.5, due to the additional computations involved in the caption-guided memory module, the method requires more time for both training and inference compared to the S3R model. However, the approach offers interpretability through video captions and demonstrates improved performance over the base model.

■ Failure Case

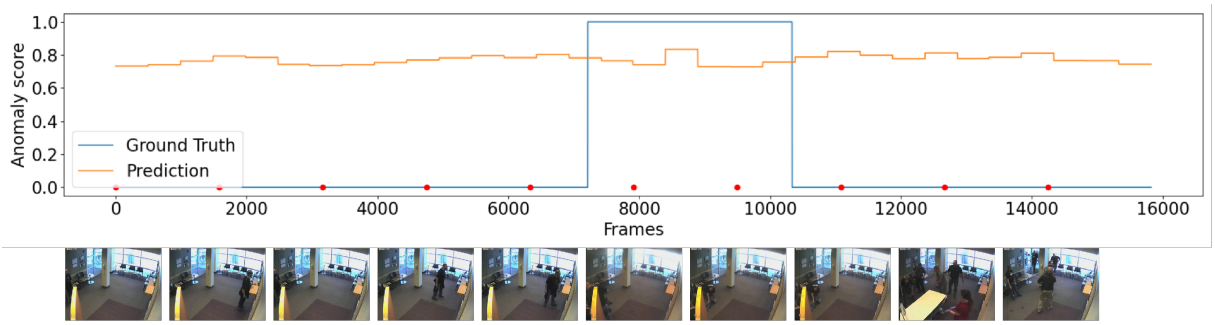


Figure 5.8: ***Failure Case.*** *The figure shows ten frame images of the video at the bottom, the timestamps of selected frames are presented in the graph using the red points. The blue line presents the ground truth of the anomaly. The orange line presents the anomaly scores of the proposed method. The video frame images are cited from [91].*

A failure case is depicted in Figure 5.8, where an arrest scene is obscured by a yellow door in the sixth and seventh frames. The model assigns high anomaly scores to all scenes, including normal ones, due to this obscured critical event. The absence of *arrest* related terms in video captions further complicates anomaly detection. To improve performance, exploring anomaly detection based on changes in video captions is important.

5.4 Conclusion

To provide interpretability based on text, the text-guided interpretable framework is proposed for the task of video anomaly detection. By incorporating video captions, the proposed module offers interpretability to the predictions and guides the model in detecting specific anomalies. Through extensive experiments, the method has demonstrated performance gains on two public anomaly detection datasets, while also shedding light on the interpretability. Furthermore, the introduced video representation, termed video memory, enables the model to produce meaningful predictions grounded in common sense, drawing from pre-trained video captioning models.

Chapter 6

Efficient Fine-tuning Vision-Language Models Based on Knowledge Selection

In Chapter 5, the proposed method utilizes video captioning models to extract semantic representation from each video segment to detect anomaly situations and provide text-based interpretability.

However, limited by the accuracy of video captioning models, the performance of anomaly detection could not be improved significantly. Inspired by the large-scale language models (LLMs), we consider the application of common sense in previous works to help generate correct captions. Therefore, a communication-based interpretability method is introduced in this chapter using LLMs. Following previous work, the high cost of fine-tuning large models is a big challenge before utilizing common sense. Consequently, knowledge selection is proposed to decrease the cost of applying large models. Limited by the resource environment, the knowledge selection method is only employed on image-based visual tasks. In future work, we will apply it to video understanding tasks.

The structure of this chapter is as follows. Section 6.1 presents the background of large models and their fine-tuning methods. Section 6.2 introduces the proposed method, knowledge selection. Section 6.3 demonstrates the performance of knowledge selection-based large models. Section 6.4 offers a conclusion of this chapter.

6.1 Introduction

With the development of large language model (LLM) technology in recent years, many natural language processing tasks have achieved outstanding results using LLMs. These tasks include text translation, semantic analysis, question answering, and more. The success of large language models has also demonstrated their immense potential for ap-

plication in other fields, such as image recognition, document understanding, and video understanding. In addition, due to extensive text training, large models acquire a certain degree of common sense. This capability is crucial for interpretable models, as human cognition of the world is largely based on fundamental common sense. Such common sense is challenging to acquire by training models on typical image datasets. By applying large language models to the visual domain, the interpretability of visual models can be improved.

However, the vast number of parameters in large models poses a significant limitation, as fine-tuning these models requires substantial training samples and time. To reduce the cost of fine-tuning and promote the application of large models, various methods have been proposed. For example, the LoRA method reduces the number of training samples needed and speeds up the fine-tuning process by freezing the parameters of the large model and adding a trainable adapter alongside the original attention layer. Unlike other fine-tuning methods, the knowledge selection method proposed in this chapter improves the performance of large models by first extracting the required knowledge for various domains and then selecting the corresponding knowledge based on the input data. This field-specific knowledge is then input into the large model.

6.1.1 Large Models

Large models have achieved remarkable performance on various tasks across multiple domains. Because they are trained with large-scale data, which provide many insights to help large models understand the world. As a result, large models generally have some common sense and a novel in-context learning capability. For that, they are extended to various tasks in visual domains. In this section, large-scale language models and vision language models are introduced.

■ LLM

The emergence of large language models pre-trained on extensive datasets has introduced a novel in-context learning capability. This allows them to handle a variety of NLP tasks using prompts without the need for fine-tuning. ChatGPT is the first groundbreaking application built on this foundation. This includes capabilities like generating code and invoking tools or APIs of other models for their use.

Large language models (LLMs) have advanced rapidly, recently. They are advanced artificial intelligence models designed to understand and generate text based on vast amounts of training data. These models, such as ChatGPT, are trained on extensive

datasets containing text from various sources, including books, articles, websites, and more.

By leveraging sophisticated deep learning techniques, LLMs can generate coherent and contextually relevant text across a wide range of topics and styles. They have demonstrated remarkable capabilities in natural language understanding, text generation, translation, summarization, and question answering. LLMs are increasingly being used in diverse applications, including content generation, language translation, customer service automation, and academic research.

■ VLM

Vision language models are the product of applying LLMs on vision domain tasks. They combine the capabilities of computer vision and natural language processing to understand and generate content that involves both images and text. These models are generally trained with datasets containing pairs of images and corresponding textual descriptions or captions. By leveraging deep learning techniques, such as convolutional neural networks (CNNs) for image processing and transformer-based architectures for text understanding, vision-language models can analyze visual content and generate coherent and contextually relevant textual descriptions. They enable tasks such as image captioning, visual question answering (VQA), image-text retrieval, and image generation conditioned on text prompts. Vision-language models have shown impressive performance in understanding the semantics of images and generating accurate and meaningful textual descriptions, making them valuable tools for applications in multimedia understanding, and content generation.

6.1.2 Fine-tuning Methods

Since large models have large-scale parameters, fine-tuning them becomes difficult. However, fine-tuning is a necessary process for employing large models on specific datasets. For that, several fine-tuning methods are proposed for large-scale models.

■ Instruction Tuning

Instruction tuning [119] refers to the process of optimizing and fine-tuning instructions provided to a machine learning model to enhance its performance on specific tasks or datasets. This technique involves adjusting various hyperparameters, such as learning rate, batch size, optimization algorithms, and model architecture, to achieve better results.

Instruction tuning is crucial in machine learning because different datasets and tasks may require different configurations to achieve optimal performance. By systematically adjusting these parameters based on experimentation and evaluation, instruction tuning aims to improve the model's accuracy, generalization ability, and efficiency. This iterative process often involves conducting experiments, analyzing results, and iteratively refining the instructions until satisfactory performance is achieved. Instruction tuning plays a vital role in the development and deployment of machine learning models across various domains.

■ LoRA

LoRA [43] reduces the number of trainable parameters by learning pairs of rank-decomposition matrices while freezing the original weights. This vastly reduces the storage requirement for large language models adapted to specific tasks and enables efficient task-switching during deployment all without introducing inference latency. LoRA also outperforms several other adaptation methods including adapter, prefix-tuning, and fine-tuning.

The QLoRA [22] is an efficient fine-tuning approach that reduces memory usage enough to fine-tune a 65B parameter model on a single 48GB GPU while preserving full 16-bit fine-tuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pre-trained language model into Low Rank Adapters (LoRA).

■ P-tuning

P-tuning [66], also known as Prompt Tuning, is a technique where the parameters of a pre-trained model are frozen, and learnable prompts are added to the input of the model for adjustment. The main advantage of this approach is its low computational cost since it only requires updating a small number of parameters. Instead of modifying the model itself, P-Tuning optimizes the prompts to make the model output the best results. The principle behind P-tuning is to overlay a trainable model P on top of an already trained model L. By optimizing P without changing L, the combined output P(L) is optimized. Since the input to L is essentially the prompt, this method is called Prompt Tuning. Prompt Tuning can be implemented manually or achieved through automatic training.

However, P-Tuning performs poorly on some complex natural language understanding tasks. Therefore, P-Tuning V2 [65] is proposed based on P-Tuning and Prefix-Tuning [58]. P-Tuning V2 introduces Deep Prompt Encoding and Multi-task Learning.

6.1.3 Motivation

Large language models have demonstrated impressive performance across a wide range of natural language processing tasks, making them indispensable in numerous domains. However, a significant challenge arises when it comes to fine-tuning these models for specific tasks. Fine-tuning typically demands substantial computational resources and often leads to the forgetting of previously learned knowledge. Despite the emergence of various fine-tuning methods aimed at mitigating this issue, they still possess inherent limitations, particularly in leveraging knowledge from related tasks or domains. Most of them try to prune the model structure or propose a novel adapter. They do not utilize the knowledge from other related samples. Given the impossibility of encompassing all knowledge within a single model, the need to selectively incorporate relevant knowledge for each task becomes paramount. Hence, we propose a knowledge selection method designed to reduce the computational cost associated with fine-tuning while ensuring optimal task performance.

6.2 Knowledge Selection

Fine-tuning a large model requires extensive data and resources, along with some sacrifice in its original accuracy. Due to the continual expansion of knowledge worldwide, selectively leveraging learned knowledge and applying it to corresponding tasks becomes crucial. Therefore, the knowledge selection is proposed to address the problem. The knowledge space composed of various domain knowledge is generated at first. When using the large model for inference, a domain knowledge is selected based on the input data from the knowledge space. The related knowledge is loaded into the large model and improves the performance in a specific domain. The method can reduce the sacrifice of original performance and apply the large model in some specific domains easily.

6.2.1 Knowledge Space

■ Generation of Knowledge Space

The knowledge space KS contains domain knowledge K, support sets S, and queries Q. Each domain knowledge K_i has corresponding support sets S_i and queries Q_i, where i is the index of the knowledge record. A support set is composed of image features related to a specific domain, and queries contain some instructions used to fine-tune the model in a domain.

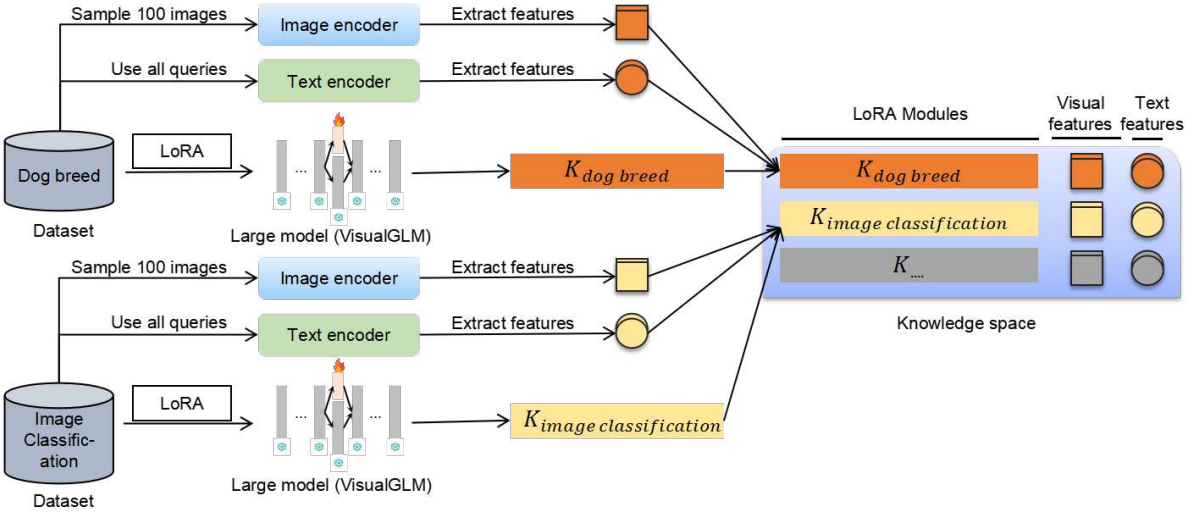


Figure 6.1: ***Generation of Knowledge Space.***

The generation of knowledge space KS is demonstrated in Figure 6.1. A small-scale dataset D_i is prepared to fine-tune the large model based on LoRA method for the generation of related domain knowledge. The small-scale dataset d contains some instructions I_d, some images V_d, and some corresponding annotations G_d. In a bird-related dataset, instructions I_d contains some questions, such as *What type of bird is this?*, *Which species of bird is this?*. The annotation of each image is created automatically using templates, for example, *The species of this bird is* $\{class\ name\}$. The annotated class name would replace {class name} in the templates. When fine-tuning a large model, a question sampled from instructions I_d and an image sampled from V_d are fed into the large model and trained to predict a text that is similar to the corresponding annotation. After fine-tuning, the parameters in LoRA modules are stored as the domain knowledge K_d.

Additionally, dataset features are extracted from dataset D to represent the domain knowledge from visual and semantic perspectives. The knowledge representation is utilized to select related knowledge from the knowledge space in inference. To generate the visual representation of domain knowledge, 100 images are sampled from the dataset D, and ViT extracts visual representation from each image.

The domain knowledge K_d, instructions I_d, and a support set S_d are denoted as a knowledge record K_i in the knowledge space KS. A support set S_i and instructions I_i of each domain knowledge are denoted as the visual representation and semantic representation of the knowledge record. They are used to locate the most related knowledge based on visual similarity and semantic similarity.

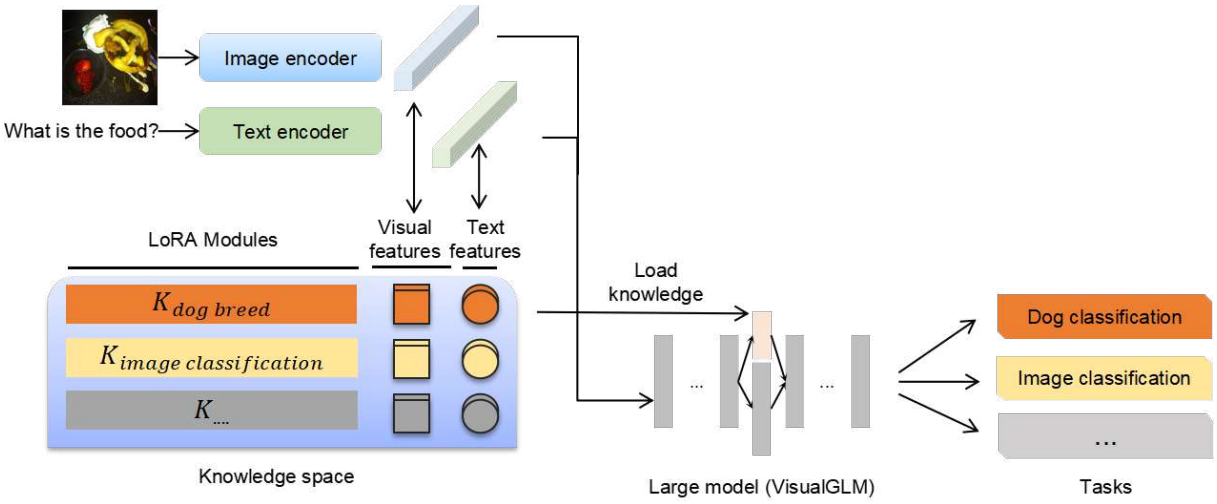


Figure 6.2: ***Overview of Knowledge Selection.***

■ Knowledge Similarity

In the proposed method, the domain knowledge is selected based on the knowledge similarity, which is composed of visual similarity and semantic similarity. Semantic similarity is calculated based on the input question Q_{input} and stored instructions $I = \{I_i | i = 1, 2, ..\}$, which is the semantic representation of all domain knowledge.

$$S_{semantic} = CosSim(Q_{input}, I)$$

If the semantic similarity of top-1 is greater than the similarity of top-2 more than θ, the knowledge related to the instruction with the highest similarity. It is selected and loaded into the base model. If not more than θ, the visual similarity would be calculated based on the similarity of the input image I_{input} and the support sets $S = \{S_{i,j} | i = 1, 2, ...; j = 1, 2, ...\}$ for a further selection.

$$S_{visual} = CosSim(I_{input}, S)$$

The related knowledge is selected according to the support set with top-1 visual similarity.

6.2.2 Model Architecture

The overview of the proposed method is shown in Figure 6.2. An appropriate domain knowledge is selected from the knowledge space based on the input image I_{input} and question Q_{input}. The selected knowledge is loaded into the per-trained large model to predict the answer. With the help of related domain knowledge, the large model can predict better results and improve performance.

In the knowledge space, various domain knowledge is stored. Each domain knowledge contains the domain knowledge LoRA modules k, a support set s, and a query q. The domain knowledge k contains the parameters of the LoRA modules. A support set s includes some images related to domain knowledge k. For example, the support set of dog breed knowledge contains images of different dogs. Similarly, query q contains the questions related domain. For example, *What is the type of this dog?* is a query for dog breed knowledge.

A related domain knowledge is selected based on two similarities: Image Similarity S_i and Text Similarity S_t. Image Similarity S_i is calculated based on the image features. Text Similarity S_t is calculated based on the sentence similarity between the input question and all queries in the knowledge space. The selected knowledge is loaded into the large model to improve the performance of the model. The input image and question are fed into the improved large model to predict a better result for corresponding tasks.

If the input question is related to bird classification, bird-related knowledge would be loaded into the base model automatically. This architecture can help researchers understand the output of the base model and improve it. For example, if the base model cannot reach considerable performance in a specific domain, generating related domain knowledge could improve the performance.

6.2.3 Implementation Details

The VisualGLM [25, 23] is used as the base large model. The second and fourteenth attention layer of VisualGLM is added to a LoRA module respectively. The rank of the LoRA module is 10.

6.3 Experiments

In this section, the datasets and evaluation metrics are introduced first. Some important experiment results and analyses of the results are provided in the following subsections.

6.3.1 Dataset and Evaluation Metric

We use 13 datasets to evaluate the proposed method. For coarse-grained image classification, the Caltech101, Caltech256, and Cifar-FS [7] datasets are used. Stanford cars [53] (Car), CUB-200-2011(Bird), FGVC-Aircraft [72] (Aircraft), Food-101 [8] (Food), Oxford-IIIT Pet [79] (Pet), and Oxford 102 Flower [77] (Flower) dataset are used for fine-grained

Table 6.1: ***Five-shot Image Classification.***

Method	Caltech101	Caltech256	Car	Bird	Aircraft	Food	Pet	Flower	DTD
VisualGLM	6.3	2.2	0	1.2	0	3	1.1	8.2	5.5
VisualGLM[Fine-tuned]	57.6	84	35	4.7	11	68.1	43.8	42.4	46
Ours	**95.1**	**93**	**74.9**	**64.9**	**41**	**72.5**	**60.5**	**86.5**	**56.2**

Table 6.2: ***Five-shot Image Classification comparing with the meta-learning methods.***

Method	CIFAR-FS	Bird
GPICL	41.5	94.5
SNAIL	71.1	92.8
CAML	85.5	**97.1**
Ours	**98.0**	56.2

image classification.

DTD [17] is used for unnatural image classification. DTD is a texture database, consisting of 5640 images, organized according to a list of 47 terms (categories) inspired from human perception.

The effectiveness of the proposed method is evaluated on few-shot image classification. To comprehensively assess the effectiveness of the method, we use the BLEU@1 score and accuracy to evaluate the performance of the proposed method.

6.3.2 Few-shot Image Classification

As presented in Table 6.1, the knowledge selection method could improve the performance of the base model VisualGLM on all few-shot datasets. As shown in Table 6.2, the proposed method reached the best performance on the CIFAR-FS dataset. We consider that common sense stored in the pre-trained large model can help the model recognize objects. However, on the fine-grained image classification tasks, limited by the architecture of the base model, the performance is difficult to improve. The base model uses the pre-trained ViT [1] to extract image features as an image encoder. As the pre-process, images would be resized to $224x224$ images before being fed into the image encoder. It limits the image representation, and results in the poor performance of the base model.

6.4 Conclusion

In this work, knowledge selection is proposed to reduce the cost of fine-tuning large models. Via the proposed method, domain knowledge can be applied flexibly. By introducing targeted knowledge into large models, the performance of large models can be improved. Several public datasets are used to evaluate the proposed method on few-shot image classification.

Chapter 7

Conclusion and Future

7.1 Conclusion

Chapter 2 introduces a variety of video understanding models and categorizes them according to their structure. In order to understand the application of video understanding techniques, Section 2.2 describes video understanding tasks and evaluation metrics. Section 2.3 shows several datasets used for the video understanding task.

In Chapter 3, an interpretable action-spotting model is proposed. Action spotting is a key component in high-level video understanding and aims to locate specific actions from each frame. In order to efficiently process the input video and interpret the predicted results, the self-attention mechanism is utilized to handle the temporal relationships among frame features, and attention scores are used to interpret predictions of the action spotting task. Since an action is composed of several subactions/scenes, the action spotting task is formulated as scene sequence recognition, and a model with multiple scene encoders is proposed to capture scene changes in videos. An input video is divided into multiple subsets to reduce the influence of temporally distant scene contexts, and every subset is fed into a scene encoder to recognize the scene in each subset. Considering the different duration ranges of different action types, the influence of chunk sizes of each action type for action spotting is analyzed. Based on the analyses, the action-aware chunk size is proposed. The experimental results on the public SoccerNet-v2 dataset demonstrate state-of-the-art accuracy and effectiveness of the proposed model and action-aware chunk size. The predictions can be interpreted by visualizing the attention scores.

The frame saliency weighting module is proposed in Chapter 4 to address the efficiency issues in the approach of Chapter 3. Unlike previous models, the frame saliency weighting module uses saliency scores to interpret the prediction results, and it is evaluated in several video understanding tasks. A large number of similar frames can affect the video representation and pose a challenge to the video understanding task. To address the influence of similar frames and improve the efficiency of models, frame saliency is proposed to represent the importance of frames. Frame saliency is calculated on the basis of the

cosine similarity of frame features. It is employed as the weight of frame features in the proposed model to improve video representation and direct the model to focus on keyframes. The proposed model contains two encoders that encode the video context with pre-action and post-action time windows, respectively. On the public SoccerNet-v2 dataset, the method achieves an average mAP of 57.3%, improving over the state of the art. With the frame saliency weighting module, reducing the model size by more than 90% does not significantly affect performance. In addition, extensive experiments validate the design choices and generalizability of the proposed method. The frame saliency weighting strategy is applicable to existing methods that use more generalized action datasets, such as SoccerNet-v1, ActivityNet v1.3, and UCF101.

In Chapter 5, the limitations of score-based interpretability were considered, i.e., the inability to compare the importance scores of different videos. To address the problem, video captioning models are introduced to provide language-based interpretability for the video anomaly detection task. Most video anomaly detection methods are based on non-semantic features, which are not interpretable, and therefore cannot identify the cause of the anomaly. A caption-guided interpretable video anomaly detection framework is proposed to explain the prediction results based on video captions (semantic). It automatically stores representative anomaly prototypes and uses them to guide the model based on similarity with these prototypes. The proposed method generates and updates the memory space during training, and predicts anomaly scores based on the memory similarities between the input video and stored memories. The stored captions can be used as descriptions of representative anomalous actions. The interpretability and reliable detection performance of the proposed method are evaluated through extensive experiments on public benchmark datasets.

In Chapter 6, common sense from pre-trained large models is employed for vision tasks. A knowledge selection method is proposed for large models to improve their performance on few-shot tasks and reduce the cost of fine-tuning. The method utilizes the LoRA method to extract the necessary knowledge for each domain and selects appropriate domain knowledge before inference to decrease the cost of fine-tuning large models and improve large models. The proposed method is evaluated on several public datasets, reach the SoTA performance on the CIFAR-FS dataset, and improve performance of the original large model on other datasets through knowledge selection.

7.2 Future Work

With the development of large-scale models, many foundational models of video understanding have reached the SoTA performance on various tasks. However, to apply large-scale models in real-world applications, fine-tuning is necessary, therefore the knowledge selection has been proposed.

In future work, we will continue to decrease the cost of fine-tuning large-scale models and employ them to video understanding tasks. We believe a large-scale model can be a pace to the AGI era. We will continue to focus on making models more interpretable and efficient.

In future work, one research direction is to reduce the cost of fine-tuning large-scale models and applying them to downstream video understanding tasks. At the same time, improving the interpretability of models based on the dialog capabilities of large models is also a major direction for future research.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Takayoshi Yamashita for the continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Fujiyoshi Hironobu, and Prof. Yamauchi Koichiro, for their insightful comments and encouragement, but also for the hard questions which helped me to widen my research from various perspectives.

My sincere thanks also go to Lecturer Tsubasa Hirakawa for providing kind and careful guidance in my research and giving me a lot of valuable advice for paper writing.

Furthermore, I would like to thank the members in MPRG for their kindness during this work and thank my friends for giving me so many wonderful memories and their encouragement will always be remembered.

Finally, I would like to show my love and deepest gratitude to my parents for their spiritual support, without which I can not finish my study at Chubu University.

Reference

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *International Conference on Computer Vision*, pp. 6836–6846, 2021.

[2] Kumar Ashutosh, Rohit Girdhar, Lorenzo Torresani, and Kristen Grauman. Hiervl: Learning hierarchical video-language embeddings. In *Computer Vision and Pattern Recognition*, pp. 23066–23078, 2023.

[3] Max Bain, Arsha Nagrani, Andrew Brown, and Andrew Zisserman. *Condensed Movies: Story Based Retrieval with Contextual Embeddings*, pp. 460–479. 02 2021.

[4] Satanjeev Banerjee and Alon Lavie. 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, 2005.

[5] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, Vol. 2008, , 01 2008.

[6] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning*, 2021.

[7] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[8] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[9] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

[10] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *CoRR*, 2018.

[11] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *CoRR*, 2019.

[12] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[13] David L. Chen and William B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Association for Computational Linguistics*, June 2011.

[14] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ2-nets: Double attention networks. In *Advances in Neural Information Processing Systems*, Vol. 31, 2018.

[15] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *European Conference on Computer Vision*, 2018.

[16] Kyunghyun Cho, Bart Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. pp. 1724–1734, 2014.

[17] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Computer Vision and Pattern Recognition*, 2014.

[18] Anthony Cioppa, Adrien Deliège, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. A context-aware loss function for action spotting in soccer videos. In *Computer Vision and Pattern Recognition*, pp. 13126–13136, 2020.

[19] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision*, 2018.

[20] Adrien Deliège, Giancola Silvio Cioppa, Anthony, Jacob V. Seikavandi, Meisam Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. Soccernet-v2 : A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *Computer Vision and Pattern Recognition Workshops*, pp. 4508–4519, 2021.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[22] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[23] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems*, Vol. 34, pp. 19822–19835, 2021.

[24] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition*, pp. 2625–2634, 2015.

[25] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Association for Computational Linguistics*, pp. 320–335, 2022.

[26] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. In *Computer Vision and Pattern Recognition*.

[27] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *International Conference on Computer Vision*, pp. 6804–6815, 2021.

[28] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *International Conference on Computer Vision*, pp. 6202–6211, 2019.

[29] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

[30] J. Feng, F. Hong, and W. Zheng. Mist: Multiple instance self-training framework for video anomaly detection. In *Computer Vision and Pattern Recognition*, pp. 14004–14013, 2021.

[31] Antonino Furnari and Giovanni Maria Farinella. Egocentric action anticipation by disentangling encoding and inference. In *Proc. ICIP*, pp. 3357–3361, 2019.

[32] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, Vol. 32, No. 11, pp. 1231–1237, 2013.

[33] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Computer Vision and Pattern Recognition*, pp. 12038–12047, 2019.

[34] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. Soccernet: A scalable dataset for action spotting in soccer videos. In *Computer Vision and Pattern Recognition Workshops*, pp. 1824–1834, 2018.

[35] Silvio Giancola and Bernard Ghanem. Temporally-aware feature pooling for action spotting in soccer broadcasts. *arXiv:2104.06779*, 2021.

[36] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter N. Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *International Conference on Computer Vision*, pp. 5843–5851, 2017.

[37] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Computer Vision and Pattern Recognition*, pp. 6047–6056, 2018.

[38] Pratik Gujjar and Richard Vaughan. Classifying pedestrian actions in advance using predicted video of urban driving scenes. In *International Conference on Robotics and Automation*, pp. 2097–2103, 2019.

[39] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Computer Vision and Pattern Recognition*, pp. 6546–6555, 2018.

[40] Bo He, Xitong Yang, Zuxuan Wu, Hao Chen, Ser-Nam Lim, and Abhinav Shrivastava. GTA: Global temporal attention for video action understanding. *arXiv preprint arXiv:2012.08510*, 2020.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, pp. 1735–80, 12 1997.

[43] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[44] Yang Hu, Yangmuzi Zhang, and Larry S. Davis. Unsupervised abnormal crowd activity detection using semiparametric scan statistic. In *Computer Vision and Pattern Recognition Workshops*, pp. 767–774, 2013.

[45] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *CVPR*, 2019.

[46] Matthew Hutchinson and Vijay Gadepally. Video action understanding. *IEEE Access*, Vol. 9, pp. 134611–134637, 2020.

[47] Y. Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. *Computer Vision and Pattern Recognition*, pp. 1359–1367, 2017.

[48] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *International Conference on Computer Vision*, pp. 3192–3199, 2013.

[49] Qihao Liu Zehuan Yuan Xiang Bai Song Bai Junfeng Wu, Yi Jiang. General object foundation model for images and videos at scale, 2023.

[50] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[51] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[52] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[53] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

[54] Hilde Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: A large video database for human motion recognition. In *International Conference on Computer Vision*, pp. 2556–2563, 2011.

[55] Ang Li, Meghana Thotakuri, David A. Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The ava-kinetics localized human actions video dataset. *ArXiv*, Vol. abs/2005.00214, , 2020.

[56] Jun Li, Xianglong Liu, Mingyuan Zhang, and Deqing Wang. Spatio-temporal deformable 3d convnets with attention for action recognition. *Pattern Recognition*, Vol. 98, p. 107037, 2020.

[57] S. Li, Fang Liu, and Licheng Jiao. Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection. In *Association for the Advancement of Artificial Intelligence*, 2022.

[58] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[59] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *European Conference on Computer Vision*, 2018.

[60] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.

[61] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *International Conference on Computer Vision*, pp. 7083–7093, 2019.

[62] Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. Swinbert: End-to-end transformers with sparse attention for video captioning. In *CVPR*, pp. 17928–17937, 2022.

[63] Jingzhou Liu, Wenhu Chen, Yu Cheng, Zhe Gan, Licheng Yu, Yiming Yang, and Jingjing Liu. Violin: A large-scale dataset for video-and-language inference. In *Computer Vision and Pattern Recognition*, pp. 10897–10907, 2020.

[64] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection – a new baseline. In *Computer Vision and Pattern Recognition*, pp. 6536–6545, 2018.

[65] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, 2021.

[66] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Association for Computational Linguistics*, pp. 61–68, 2022.

[67] Xin Liu, Fatemeh Karimi Nejadasl, Jan C. van Gemert, Olaf Booij, and Silvia L. Pintea. Objects do not disappear: Video object detection by single-frame object location anticipation. In *International Conference on Computer Vision*, pp. 6927–6938, 2023.

[68] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, pp. 1–31, 2020.

[69] Chenxu Luo and Alan Yuille. Grouped spatial-temporal aggretation for efficient action recognition. In *International Conference on Computer Vision*, 2019.

[70] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Association for Computational Linguistics*, 2024.

[71] Farzaneh Mahdisoltani, Guillaume Berger, Waseem Gharbieh, David J. Fleet, and Roland Memisevic. On the effectiveness of task granularity for transfer learning. *CoRR*, 2018.

[72] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[73] William McNally, Kanav Vats, Tyler Pinto, Chris Dulhanty, John McPhee, and Alexander Wong. Golfdb: A video database for golf swing sequencing. In *Computer Vision and Pattern Recognition Workshops*, 2019.

[74] Rajat Modi, Aayush Rana, Akash Kumar, Praveen Tirupattur, Shruti Vyas, Yogesh Rawat, and Mubarak Shah. Video action detection: Analysing limitations and challenges, 2022.

[75] Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. Modeling context between objects for referring expression understanding. In *European Conference on Computer Vision*, 2016.

[76] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021.

[77] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 722–729, 2008.

[78] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics*, p. 311–318, USA, 2002.

[79] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Computer Vision and Pattern Recognition*, 2012.

[80] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. *CoRR*, pp. 5533–5531, 2017.

[81] Arandjelovic Relja, Gronat Petr, Torii Akihiko, Pajdla Tomas, Josef, and Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Computer Vision and Pattern Recognition*, pp. 5297–5307, 2016.

[82] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, Vol. 115, No. 3, pp. 211–252, 2015.

[83] Bashir Olaniyi Sadiq, Bilyamin Muhammad, Muhammad Nasir Abdullahi, Gabriel Onuh, Ali Abdulhakeem Muhammed, and Adeogun Emmanuel Babatunde. Keyframe extraction techniques: A review. *ELEKTRIKA-Journal of Electrical Engineering*, Vol. 19, No. 3, pp. 54–60, 2020.

[84] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, Vol. abs/1610.02391, , 2016.

[85] Seonguk Seo, Joon-Young Lee, and Bohyung Han. Urvos: Unified referring video object segmentation network with a large-scale benchmark. In *European Conference on Computer Vision*, pp. 208–223, 2020.

[86] Gunnar Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016.

[87] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, Vol. 27, pp. 569–576, 2014.

[88] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, Vol. 33, pp. 16857–16867, 2020.

[89] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[90] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *Winter Conference on Applications of Computer Vision*, pp. 625–634, 2020.

[91] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *Computer Vision and Pattern Recognition*, pp. 6479–6488, 2018.

[92] Yuchong Sun, Hongwei Xue, Ruihua Song, Bei Liu, Huan Yang, and Jianlong Fu. Long-form video-language pre-training with multimodal temporal contrastive learning. In *Advances in Neural Information Processing Systems*, NIPS '22, 2024.

[93] Yu Tian, Guansong Pang, Yuanhong Chen, Rajvinder Singh, Johan W. Verjans, and Gustavo Carneiro. Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. In *International Conference on Computer Vision*, pp. 4955–4966, 2021.

[94] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision*, pp. 4489–4497, 2015.

[95] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision*, pp. 4489–4497, 2015.

[96] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *International Conference on Computer Vision*, pp. 5551–5560, 2019.

[97] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *International Conference on Computer Vision*, pp. 5552–5561, 2019.

[98] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

[99] Bastien Vanderplaetse and Stephane Dupont. Improved soccer action spotting using both audio and video streams. In *Computer Vision and Pattern Recognition Workshops*, pp. 3921–3931, 2020.

[100] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, pp. 1510 – 1517, 06 2018.

[101] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[102] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[103] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Computer Vision and Pattern Recognition*, pp. 4566–4575, 2015.

[104] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *Computer Vision and Pattern Recognition*, pp. 1430–1439, 2017.

[105] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. pp. 1430–1439, 2018.

[106] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Computer Vision and Pattern Recognition*, 2015.

[107] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, 2016.

[108] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pp. 20–36, 2016.

[109] Xiang Wang, Shiwei Zhang, Zhiwu Qing, Yuanjie Shao, Changxin Gao, and Nong Sang. Self-supervised learning for semi-supervised temporal action proposal. In *Computer Vision and Pattern Recognition*, pp. 1905–1914, 2021.

[110] Sitapa Watcharapinchai and Nattachai Watcharapinchai. Vehicle detection with sub-class training using r-cnn for the ua-detrac benchmark. pp. 1–5, 08 2017.

[111] Jhih-Ciang Wu, He-Yen Hsieh, Ding-Jie Chen, Chiou-Shann Fuh, and Tyng-Luh Liu. Self-supervised sparse representation for video anomaly detection. In *European Conference on Computer Vision*, 2022.

[112] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Computer Vision and Pattern Recognition*, pp. 5288–5296, 2016.

[113] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *Computer Vision and Pattern Recognition*, pp. 591–600, 2020.

[114] Youngjae Yu, Jongseok Kim, and Gunhee Kim. A joint sequence fusion model for video question answering and retrieval. In *European Conference on Computer Vision*, 2018.

[115] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Association for the Advancement of Artificial Intelligence*, pp. 9127–9134, 2019.

[116] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep net-

works for video classification. In *Computer Vision and Pattern Recognition*, pp. 4694–4702, 2015.

[117] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *International Conference on Computer Vision*, 2019.

[118] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

[119] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 08 2023.

[120] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *Computer Vision and Pattern Recognition*, pp. 9967–9976, 2019.

[121] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. Hacs: Human action clips and segments dataset for recognition and temporal localization. In *International Conference on Computer Vision*, pp. 8668–8678, 2019.

[122] He Zhao and Rick Wildes. Spatiotemporal feature residual propagation for action prediction. In *International Conference on Computer Vision*, pp. 7002–7011, 2019.

[123] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Trajectory convolution for action recognition. In *Advances in Neural Information Processing Systems*, Vol. 31, 2018.

[124] Jia-Xing Zhong, Nannan Li, Weijie Kong, Shan Liu, Thomas Li, and Ge Li. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *Computer Vision and Pattern Recognition*, pp. 1237–1246, 2019.

[125] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *European Conference on Computer Vision*, pp. 803–818, 2018.

[126] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *Association for the Advancement of Artificial Intelligence*, 2018.

[127] Qianyu Zhou, Xiangtai Li, Lu He, Yibo Yang, Guangliang Cheng, Yunhai Tong, Lizhuang Ma, and Dacheng Tao. Transvod: End-to-end video object detection with spatial-temporal transformers. *IEEE transactions on pattern analysis and machine intelligence*, Vol. PP, , 2022.

[128] Xin Zhou, Le Kang, Zhiyu Cheng, Bo He, and Jingyu Xin. Feature combination meets attention: Baidu soccer embeddings and transformer based temporal detection. *arXiv:2106.14447*, 2021.

Research Achievements List

Academic journa

[1] Yuzhi Shi, Takayoshi Yamashita, Tsubasa Hirakawa, Hironobu Fujiyoshi, Mitsuru Nakazawa, Yeongnam Chae, Björn Stenger, "Efficient Action Spotting Using Saliency Feature Weighting" (IEICE), Vol.E107-D No.1, pp.105-114, 2023

[2] Yuzhi Shi, Takayoshi Yamashita, Tsubasa Hirakawa, Hironobu Fujiyoshi, Mitsuru Nakazawa, Yeongnam Chae, Björn Stenger, " Caption-guiding Interpretable Video Anomaly Detection Based on Memory Similarity" (IEEE Access), vol. 12, pp. 63995-64005, 2024

International Conference

[1] Saki Noguchi, Yuzhi Shi, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi, "Embedding Human Knowledge into Spatio-Temproal Attention Branch Network in Video Recognition via Temporal Attention", (BMCV, Poster), 2023.

[2] Tsunemi Nitta, Yuzhi Shi, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi, "Data Drift Detection with KS Test using Attention Map", (ACPR, Oral), 2023.

[3] Yuzhi Shi, Mijung Kim, Yeonnam Chae, "Multi - scale Cell - based Layout Representation for Document Understanding" (WACV, Oral), 2023.

[4] Yuzhi Shi, Hiroaki Minoura, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi, Mitsuru Nakazawa, Yeongnam Chae, Björn Stenger, "Action Spotting in Soccer Videos Using Multiple Scene Encoders," The 26th International Conference on Pattern Recognition (ICPR, Oral), 2022.

National Conference

[1] 野口紗季，史宇植，平川翼，山下隆義, 藤吉弘亘，”動画像認識における ST-ABN を用いた人の知見の組み込みによる説明性の向上と高精度化”, 画像センシングシンポジウム, 2023

[2] 新田常顧, 史宇植, 平川翼, 山下隆義, 藤吉弘亘, ”アテンションマップを用いた KS 検定によるデータのドリフト検知”, 画像センシングシンポジウム, 2023

[3] Yuzhi Shi, Takayoshi Yamashita, Tsubasa Hirakawa, Hironobu Fujiyoshi, “Efficient Action spotting Based on Cosine Similarity to Focus on Keyframes”(MIRU), 2022

[4] Yuzhi Shi, Hiroaki Minoura, Takayoshi Yamashita, Tsubasa Hirakawa, Hironobu Fujiyoshi, “Action Spotting in Soccer Videos via Transformer with Past and Future Encoders”(MIRU), 2021

[5] 史宇植, 岩堀 祐之, 小笠原尚高, 春日井邦夫,“FCN と U-Net による平坦型ポリープのセグメンテーション”(WiNF), 2020