MANCHESTER 1824
The University of Manchester

Anchor-free Pipeline Temporal Action Localisation

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2023

Jiyong Lee
School of Engineering
Department of Computer Science

Contents

Word count: 52059

List of figures

List of tables

Terms and abbreviations

AP	average precision
AR	average recall
AUC	area under the curve
FN	false negative
FP	false positive
FPN	feature pyramid network
FPS	frames per second
HAR	human action recognition
mAP	mean average precision
NMS	non-maximum suppression
PEM	proposal evaluation module
ROC	receiver operating characteristic
TAL	temporal action localisation
TAPG	temporal action proposal generation
TEM	temporal evaluation module
(t)IoU	(temporal) intersection over union
TN	true negative
TP	true positive

Abstract

Temporal action localisation (TAL) has garnered significant attention due to its potential applications across various fields. The primary challenges in this domain involve detecting the start/end times of actions and recognising the action themselves. There are two primary research approaches to address these issues: treating them as separate problems or attempting to solve them simultaneously. In this thesis, the focus will be on the first component of the former approach, which is action proposal generation. This has been chosen because the results can be applied more broadly by not considering the recognition aspect, which is specific to action classes.

The primary challenge in action proposal generation is detecting the start and end of actions without labels. To address this issue, it was done in three steps: (1) identifying factors and challenges affecting TAL performance, (2) implementing anchor-free boundary detection, and (3) conducting a performance comparison. Five possible scenarios were considered to examine factors affecting performance and challenging cases, and the performance was compared using existing algorithms. First, the relationship was analysed between boundary detection and action recognition. In pipeline methods, the outcomes of previous steps influence those of subsequent steps. Consequently, how recognition results vary according to proposal generation outcomes was investigated. Second, to determine the effect of class labels on performance, existing end-to-end methods were modified by transforming multi-label classification into a binary one. This facilitated the evaluation of the influence of class labels on performance. Third, in video clips, actions can sometimes be presented discontinuously due to editing or abrupt viewpoint changes. How these instances affect performance was examined. Fourth, the effects of two commonly used types of data (RGB and Flow) in the literature were studied to understand their influence on performance. Finally, to assess generalisation, cross-corpora tests were conducted to evaluate how a model trained on one dataset applied to another dataset. One key challenge in detecting actions is the unknown and variable length of actions, even within the same category, as it can differ from person to person. To address this issue, some existing methods have adopted predefined temporal spans called “anchors”. However, these methods necessitate repetitive operations at the same location and often result in inaccurate boundaries. To overcome these limitations, an anchor-free method was employed to directly predict the length of the action. This approach is less sensitive to boundary inaccuracies caused by predefined lengths and is more efficient due to the elimination of repetitive operations. However, direct prediction relies on partial information, which may lead to inaccurate boundaries. Therefore, additional refinement is implemented to improve boundary accuracy. Finally, the performance of the proposed method was compared with existing algorithms in challenging situations using the identified factors.

In summary, factors influencing action detection were examined and an anchor-free action proposal generation method incorporating boundary refinement was proposed. Then, the performance of the proposed method was analysed in various settings, demonstrating its effectiveness and potential for addressing the challenges associated with action detection.

Declaration of originality

I hereby confirm that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in The University's policy on Presentation of Theses.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Ke Chen, for his extensive and professional assistance. I would never have completed my PhD course without him.

I would also like to thank many friends in the Machine Learning and Optimization group and the language exchange community. Thanks to them, I was able to endure the COVID-19 pandemic.

I am grateful to all the other people who helped me.

Lastly, I would like to thank the University of Manchester for supporting my research and living through the Dean's Award.

Chapter 1

Introduction

The proliferation of videos on social media and CCTV has made it possible to access abundant visual information in everyday life and special circumstances. Consequently, methods that utilise this visual information have gained considerable attention. Since these videos often contain a significant portion of events or actions focused on people, valuable insights can be derived from analysing human actions. Human actions are employed across various domains, including surveillance, video summarisation, and instructional training. For instance, highlight videos of sports games can be generated by centring on critical actions, and rule violations can be detected by observing athletes' actions. Furthermore, by identifying abnormal behaviour in CCTV footage, it becomes possible to prevent crimes in advance or aid in apprehending criminals. In educational settings, action recognition can serve as a virtual instructor, assisting in training by recognising trainees' motions. To achieve these objectives, it is crucial to detect and recognise human actions in videos. However, manual methods such as police officers watching CCTV videos to identify abnormal events or human editors creating highlight videos of sports events are time-consuming and labour-intensive. Therefore, an automated system is needed, and temporal action localisation (TAL) represents one such attempt to address these challenges.

Action recognition is a task to classify trimmed video clips into one of target labels while action detection refers to a task that detect actions in untrimmed video and recognise the detected actions. TAL is alternative terminology of action detection. Action detection and recognition have advanced alongside recent developments in object and face detection and recognition. Given the similarities between object detection and recognition and action detection and recognition, researchers have drawn on insights from face and object detection and recognition in their investigations. For example, the same object may appear in different sizes and shapes in a 2D image depending on its location and distance from the camera. Similarly, the length and form of actions can vary from person to person along the one-dimensional time axis. However, images and videos have distinct characteristics that make it challenging to apply the same algorithm to both. While an image encapsulates all information within a single frame, a video contains information extended along the time axis. That is, images are two-dimensional data, while videos can be viewed as either one-dimensional or three-dimensional data. If only the temporal axis is considered, videos become one-dimensional data; however, when both spatial and temporal axes are considered simultaneously, videos become three-dimensional data. This means that continuous data along the time axis must be taken into

account. Furthermore, actions exhibit more significant intra-class variation compared to objects. Due to these characteristics, when the same algorithm is applied, the performance of action detection and recognition remains inferior compared to its execution on a single image for either object or face. For example, recent results in object detection and recognition have achieved 65.4% accuracy on the COCO dataset [1], while TAL's accuracy has reached only 42.0% on ActivityNet v1.3 [2]. This discrepancy highlights the significant challenges faced by the field of action detection and recognition but also presents considerable opportunities for researchers. In this thesis, the problem of detecting actions is addressed by considering the unique characteristics of videos that differentiate them from images.

1.1 Motivation and Challenges

In this section, the motivation of this thesis is described by explaining the value of solving TAL. Subsequently, some challenges inherent to TAL are presented.

Availability Across a Variety of Domains, but a Few Works of Anchor-Free Action Proposal Generation
By detecting and recognising human actions in untrimmed videos, valuable tools can be provided to individuals across diverse domains, as most events in everyday life revolve around human activities. For example, police officers may need to spend hours watching CCTV footage to locate a criminal. Likewise, creating highlight reels of sports events like the Olympics often requires watching every match to select crucial scenes. If it were possible to automatically detect specific human actions, identifying suspicious scenes from CCTV or automatically collecting important clips from sports games, this could save a significant amount of time that would otherwise be spent watching videos. Beyond these examples, numerous application domains, such as medical care, tourism, and entertainment, could benefit from advancements in action detection and recognition in videos.

In TAL, many predefined temporal units, known as anchors, are used to handle the variation of action lengths, since the lengths are not known in advance. While these approaches have demonstrated good performance, they also present certain drawbacks. Anchor-based methods can be computationally inefficient due to repetitive computations at all locations needed to evaluate every anchor. Additionally, they are known to generate inaccurate boundaries because of their reliance on predefined anchors. To overcome these limitations, anchor-free approaches have been proposed. However, in the literature, there is some work of temporal proposal generation that employ anchor-free methods.

Large Intra-class Variation
In classification problems that generally exhibit high performance, samples within the same category tend to share similar characteristics. Take object recognition as an example: rigid objects possess fixed shapes and colours and are less prone to deformation. The variation in an object's appearance captured in an image can result from camera position or lighting condition. Even articulated objects, such as humans or robots, have limitations in their gesture

or body shapes due to the limited range of movements imposed by their biological structures. However, actions are composed of components with an unlimited range and space, making them more complex and challenging to classify.

Human Behavioural Habits

Even when performing actions with the same purpose, individuals exhibit different forms and sequences based on their preferences and habits. For example, consider the action of kicking a ball. Depending on a person's preference, they might use a different foot for kicking, kick after running, or kick in place. These variations can make seemingly identical actions appear different, complicating the process of identifying common patterns.

Irrelevant Data in Video

Action detection using visual information is susceptible to interference from scenes unrelated to the target actions. When obtaining data through body-attached sensors, such as motion capture systems, and recognising motion based on the data, the information gathered typically stems exclusively from the target action. However, videos capture not only the person's actions but also background information and the actions of unrelated individuals nearby.

Small Inter-class Variation

There are also instances where human actions appear similar when performed for different purposes. While actions exhibit a high degree of freedom, the human body, when considered in isolation, has limitations in expressing simple actions. Consequently, it is common for certain parts of different movements to appear similar to each other, leading to small inter-class variation. For example, when comparing the actions of throwing a shot put and throwing a javelin, the object and purpose of the throws differ, but both share the common feature of holding an object on the shoulder and then throwing it. In such cases, distinguishing between the two actions can be challenging when relying solely on the shape of the body itself. Visual data offers more useful information compared to other data types, such as motion capture system. With visual data, information about the background in an image or objects held by a person can be obtained. Consequently, even when a person performs similar motion, it becomes possible to differentiate between similar yet distinct actions by considering the surrounding information of the person.

Ambiguous Boundary

Human actions often have ambiguous start and end points. Since specific actions are performed as part of continuous motion, their boundaries can be challenging to define clearly, unlike that of objects. These boundaries may be related to an individual's behavioural habits, and when annotating the start and end points of actions in a video, the resulting annotations might vary based on the subjective judgement of the annotator.

1.2 Research Questions

The research problem to be addressed in this thesis focuses on TAL. Specifically, how to estimate the start and end points of actions in untrimmed videos using a pipeline approach is

investigated. The study begins with a comparative analysis to identify factors affecting performance and challenging cases. Subsequently, a novel framework was proposed for generating action proposals using an anchor-free method. The work presented in this thesis will demonstrate how the proposed model operates in such challenging cases. The research questions to be resolved are therefore as follows:

1. *What are the factors affecting TAL performance, and in what types of video clips are actions difficult to detect?*
   To answer this question, five issues were considered. In each issue, all factors except for the one under investigation are controlled. The five issues and corresponding sub-questions are as follows:

   (a) In the case of pipeline TAL, which consists of multiple stages, temporal action proposal generation has been improved independently. To measure the overall TAL performance, existing action recognition algorithms have been reused. The following sub-questions are therefore needed to be addressed:

   - When the performance of proposal generation improves, will the performance of recognition also improve?
   - Is there any relationship between the performance of proposal generation and that of recognition?

   (b) There are two approaches for TAL in the literature. One is the pipeline method with multiple stages, which addresses action recognition after generating action proposals, which serve as candidates of actions. The other is the end-to-end method, which attempts to solve both action boundary detection and action recognition concurrently. Recent work has favoured the latter approach. The key difference between these two approaches is that the former detects actions in a class-independent manner, while the latter addresses action boundary detection and recognition simultaneously, requiring the use of class labels. Therefore, the following sub-question can be proposed:

   - Are there any advantages of using action labels in terms of boundary detection?

   (c) On the internet, many video clips edited by individuals, resulting in actions that may not be continuous. Sometimes, irrelevant scenes are inserted within a single action. From a method perspective, anchor-based (or boundary-based) methods create proposal-level feature vectors using a set of predefined temporal spans to calculate actionness scores, while anchor-free methods construct multi-scale feature vectors and predict the left and right offsets, enabling method to examine the partial information of actions. The two approaches employ different strategies. This gives rise to two further subsidiary questions:

   - Which approach is more robust to unwanted discontinuous situation?

- If one approach exhibits lower performance than the other, what is the reason behind it?

(d) Two types of data modalities are commonly used as feature vectors: RGB and Flow. RGB is suitable for representing action appearance information, while the Flow captures the trajectory of actors and their surroundings. Generally, overall performance improves by using both modalities. However, for discontinuous actions, discontinuous points can be perceived as boundaries. The question is then:

- Does Flow data always have a positive effect? Is it possible that, in certain situations, Flow data could negatively impact performance?

(e) In machine learning, generalisation is one of the crucial issues. Traditional machine learning algorithms are designed to produce optimal results given a training dataset. Therefore, when using data with characteristics different from the dataset used for training, it may lead to poor results in cases with unintended or different characteristics.
TAL performance was tested using different datasets for action proposal generation. This instigated the following two additional subsidiary questions:

- How will models trained on different datasets perform when tested on a dataset that was not used for training?
- What are the factors contributing to this performance difference?

2. *How can various length of human actions be dealt with?*

Determining the duration of a person's action can be challenging. Objects composed of rigid bodies have their own specific dimensions, even when they undergo slight deformation, they maintain a similar shape, allowing to predict their appearance. In contrast, human actions lack a fixed form. For instance, even when performing a simple action such as walking, the motion ofthe arm varies from person to person, as does the head's orientation. Due to these characteristics, standardising human actions is difficult. The length of the same action may vary according to the speed of the actor's movement; some people might execute the exact same movement quickly, while others do so slowly. This relates to the multi-scale issue, similar to handling different sizes in object detection. Object detection addresses this problem in a two-dimensional image plane, whereas TAL deals with multi-scale issues concerning action length in a one-dimensional time axis. As the anchor-free method is chosen for this thesis, it is necessary to find solutions to the following questions:

- *How to model multi-scale actions?*

  The challenge in addressing this problem of the length of human actions is that since the action's length is unknown, evaluations must be made for all possible time lengths. However, if there are too many temporal spans to be evaluated, there is a disadvantage of increased computational load due to repetitive operations, and

in the opposite case, some expected actions may be missed. How can both short and long actions be handled simultaneously? To achieve this, a method that can efficiently model and evaluate multiple temporal scales is necessary.

- *How to convert actions of different lengths into input vectors to represent them with the same shape?*

  Even if there is a way to model temporal scale efficiently, the amount of information included will vary depending on the action's length. Longer actions will consist of many image frames and therefore contain more data, while shorter actions will consist of less data. However, machine learning algorithms require inputs of the same size. So, how can actions composed of different amounts of data be converted into input vectors of the same size? To achieve this, a video representation method that constructs input vectors of the same size is required.

3. *How can the boundary of human actions be detected?*

   Detecting the boundaries of human actions is challenging as continuous actions on the time axis cannot be accurately separated like object boundaries, and specific actions must be detected while a person is moving. Additionally, the start and end points of an action are often determined by the subjective judgement of the annotator. To effectively model and detect action boundaries, the following questions must be addressed:

   - *How to model the boundaries of actions?*

     Since the boundary of an action is a transient moment, it is impossible to detect a specific time using only data from that particular moment. To overcome this, surrounding information need to be utilised. From this perspective, it is essential to address how much and what quantity of surrounding information to include.

   - *How to deal with the inconsistent boundaries of action determined by people's subjective judgements?*

     Different points may be selected depending on who marks the boundary for the action of the same person. Some motions at the boundaries may be included or excluded from a specific action. Consider a jumping action as an example. Whether or not to include the action of running before jumping depends entirely on the choice of the marker. In addition, even when an additional motion, such as falling after landing, occurs, it is ambiguous whether to include this additional motion in the corresponding action. Therefore, a means to cope with such ambiguity in action boundaries is needed.

1.3 Contributions

To answer the first research question, comparative studies were conducted according to the five investigations. Five factors influencing performance were selected, and performance comparison was carried out in a controlled environment to verify them. To answer the second

and the third questions, a boundary refinement method based on statistics and a complementary anchor-free Temporal Action Proposal Generation (TAPG) model were proposed. The contributions made in this thesis can be summarised as follows.

- Comparative experiments were conducted based on factors. Unlike the ablative study or overall average recall adopted in performance evaluation, this was to see how the factors affect performance. Other factors, except for the one under investigation, were controlled and the corresponding performance was compared. Through this, it was possible to know how a particular factor affects performance.
- An action proposal generation method was proposed using an anchor-free approach. The previously used anchor-based method had the disadvantage of high computational cost due to repeated calculations at each temporal location. However, by directly predicting the offsets of actions, the need for repetitive operations was eliminated.
- An additional boundary refinement method based on statistics was proposed to predict boundaries more accurately. Human-marked labels are difficult to represent precise action boundaries consistently. However, the boundary was refined under the assumption that the actual action boundary will exist near the annotated boundary. Similar to the mean-shift used in object tracking, the boundary was refined using statistical information.
- Based on the five investigations from the first research question, the same protocol for evaluation was applied to the proposed method. It was conjecture that this may provide insights for new methods of evaluating TAL algorithms.

1.4 Thesis Structure

The remainder of this thesis is structured as follows.

In Chapter 2, essential background knowledge for the work presented in this thesis is introduced. First, the chapter starts by reviewing the basic knowledge required for TAL from a general machine learning perspective. Second, a method of converting video into feature vectors is introduced. In the literature, action recognition algorithms are used as the backbone for this purpose. Therefore, some representative action recognition algorithms are introduced as a video representation method. Third, methods are reviewed for making data of the same size from different length of actions, and then, methods are considered for retrieving actions from untrimmed video clips and predicting the lengths of the actions. Fourth, strategies for performing TAL such as end-to-end methods and pipeline methods are introduced. Lastly, the metrics used to evaluate the performance of the algorithm are reviewed for comparison.

In Chapter 3, five hypotheses are established to inspect factors that may affect TAL performance. According to the hypotheses, five corresponding experimental scenarios are constructed and comparative studies are conducted to investigate whether and how performance

is influenced by the identified factors. In each scenario, existing algorithms are selected and modified for the purpose. The five criteria were considered: 1. Performance relationship of detection and recognition, 2. advantage/disadvantage of using multi-class labels, 3. robustness to discontinuous actions, 4. multi-modal test (RGB and Flow), 5. cross-corpora test. In the first scenario, the relationship between the performance of action detection and recognition is examined. In pipeline methods having the ”detection-then-recognition” scheme, the results of detection are used as input for the recognition algorithm. Therefore, whether good detection results lead to good recognition results is investigated. In the second scenario, by the fact that there are few anchor-free action proposal generation methods in the literature, how action class labels affect the detection of actions is examined by comparing results of end-to-end TAL methods and those of their variants for proposal generation. In the third scenario, the experiment is conducted on discontinuous actions caused by abrupt camera movement or editing that can be seen in videos uploaded to personal SNS such as YouTube and Facebook, and how these discontinuous actions affect performance is investigated. In the fourth scenario, a performance comparison is conducted using two commonly used types of data (RGB and Flow). Finally, in terms of generalisation, the same model is trained on ActivityNet and THUMOS14, which are primarily used datasets, and how the model trained on one dataset performs on another dataset are observed.

In Chapter 4, an anchor-free Temporal Action Proposal Generation (TAPG) method is proposed. Action proposals are generated by directly predicting offsets at each location of the feature pyramid, and refine the predicted boundaries using statistical information to achieve more accurate predictions. Since the anchor-free method of detecting actions on the feature pyramid uses only partial information, a proposal evaluation module is introduced to compensate for this disadvantage. Moreover, a temporal evaluation module is introduced to measure the starting and ending probabilities of an action at each temporal location. This helps increase the reliability of boundary detection. Finally, the comparative studies are conducted under the same scenarios as in Chapter 3 to investigate the effectiveness of the proposed method.

In Chapter 5, conclusions are drawn about this research and its shortcomings are discussed. Furthermore, potential future work is explored to address these limitations and enhance the research outcomes.

1.5 Summary

In this chapter, TAL has been introduced. The motivation of this thesis and challenges in this field were described. Based on the motivation and challenges, research questions were made and the contributions of this thesis were enumerated. Lastly, how this thesis is structured was explained.

Chapter 2

Background

TAL is a compelling field in computer vision due to its applicability across various domains, attracting increasing attention over the last few decades. The TAL research field has advanced from various perspectives, including the design of learning algorithm, modelling of temporal information and surrounding context, boundary refinement, and more. Unlike general machine learning issues that aim to solve specific problems, such as feature selection, TAL is a comprehensive topic. For instance, finding accurate boundaries can be viewed as a regression problem, while detecting and recognising actions can be considered a classification problem. Thus, TAL has connections with various machine learning techniques. In this chapter, a thorough literature review of essential techniques related to TAL is presented.

This chapter is structured as follows. Section 2.1 briefly reviews the context of understanding video from a TAL perspective. Sections 2.2 to 2.4 focus more on TAL starting with techniques used to address issues raised in this field (video representation, variable-length action duration modelling, proposal representation, temporal duration handling methods, boundary refinement, and elimination of duplicate results), and then exploring other related topics, such as performance evaluation methods, benchmarking datasets, and supervised learning.

2.1 Video Understanding and Machine learning Overview

Video understanding in the field of computer vision aims to automatically analyse and comprehend the content of videos. This encompasses various tasks, including recognition, detection, anticipation (also known as early recognition in the literature), scene generation, segmentation, retrieval, captioning, recommendation, question-answering, and more. These tasks may necessitate extracting relevant information from the video content and performing additional subsequent tasks. In this regard, handling sequential data presents a greater challenge than tasks involving single data points, such as object recognition in a still image. Among the many tasks required for video understanding, detection and recognition are key topics related to TAL.

The goal of video recognition is to classify trimmed video clips into one of a number of predefined categories. This involves identifying subjects, such as people and objects in video clips, and understanding the event taking place. Specifically, when the object of recognition is a human action, it is called Human Action Recognition (HAR). Various types of data are

used for this purpose. Some approaches utilise data collected from sensors attached to the body, while others employ information obtained from devices in the surrounding environment [3]. Although fusing multi-modal data from diverse sources can yield more accurate results, there are numerous prerequisites, such as setting up an experimental environment. Alternatively, many studies have been conducted using only video data. In this research direction, additional information, such as a skeleton, is extracted from video to achieve the same effect as using a sensor. Furthermore, multi-modal images, such as colour images and depth images, are utilise through the use of specialised cameras [4]. Some researchers have focused on information based on the action's characteristics using only general video data (e.g. a sequence of colour images). They have attempted to analyse human actions using motion flow information and appearance information [5], or by examining the differences between fast and slow motion [6]. Additionally, other researchers have attempted to separate atomic units from high-level abstract actions, and then analyse complex activities by combining these units [7]. These efforts have significantly contributed to automatically identifying the actions presented in videos, as demonstrated by the impressive results in various benchmarking datasets (e.g. UCF101 (top-1 accuracy: 98.64% [8]), Kinetics- 700 (top-1 accuracy: 84.0% [9]), AVA2.2 (mAP: 41.01 [9])). This progress has been widely reused or inspired many advancements in the TAL domain. However, as demonstrated in general classification tasks, these methods classify actions into predefined categories, making it difficult to handle non-defined actions. Moreover, when applied to untrimmed videos, it is challenging to identify the specific part of the video where the action takes place, as the trimmed video clips used in the dataset do not contain boundary information. Additionally, since untrimmed videos often contain many segments without relevant actions, it is crucial to develop techniques to process and account for non-defined actions as well.

Action localisation aims to detect the start and end times of relevant actions in untrimmed videos. This task can be viewed as action detection in videos, similar to object detection in images. It is divided into two main categories: TAL and spatio-temporal action localisation. TAL detects actions by considering only the time axis. As a result, this approach focuses more on how the scene changes and progresses rather than on detailed information, such as the location of people or surrounding objects in the image. Detection is performed using this information. The method of representing videos has primarily depended on action recognition algorithms that use whole frames from videos (which will be addressed in Section 2.3). Video feature vectors are extracted from a specific layer of the action recognition algorithm, and then the starts and ends of actions are detected. In contrast, spatio-temporal action localisation aims to detect actions using both temporal information and spatial information present in each image frame. This approach detects not only the boundaries of the actions but also the position in the video where the action is taking place, resulting in a 3D tube-shaped outcome [10].

In the following sub-sections, TAL will be explained from three perspectives: classification, regression, and generalization. TAL can be seen as a compound task that includes both classification and regression tasks. Therefore, it will be explained from these two as-

pects. Furthermore, the issue of generalization and the efforts made to address it will also be described.

2.1.1 Classification

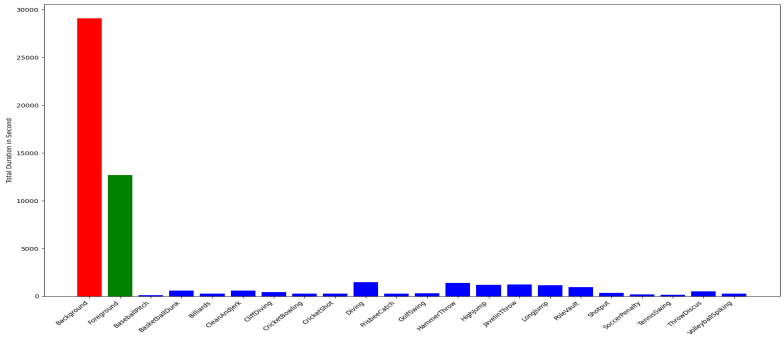
In machine learning, classification refers to assigning a single label to all samples extracted from a problem domain. TAL can be considered to be a classification problem, as it assigns corresponding labels to the actions of people present in the video. However, unlike general classification problems, the video used for TAL includes a class that does not exist in the label set. This occurs because human actions do not take place continuously but intermittently in the video, leading to background segments that do not contain any relevant actions. To address this issue, an additional label called "background" is used. However, instead of performing (N+1)-label classification by directly adding a background label to the label set, this problem is handled indirectly through two types of classification problems: binary classification and multi-label classification. This stratified strategy, in which multi-label classification is performed after initial binary classification, is adopted due to the nature of intermittently occurring actions, the number of samples corresponding to the background accounts for a relatively large proportion, which can cause imbalance in the dataset.

Binary classification is the problem of classifying two target classes. In TAL, it is performed to classify the foreground, which includes an action, and the background, which does not include any action. Table 2.1 displays the total time of the foreground and background of THUMOS14 [11] and ActivityNet 1.3 [2], which are widely used datasets in the literature. Figure 2.1 presents the total time per action class together. The first bar (red) represents the total time in the background, while the second bar (green) represents the total time in the foreground (action). The remaining bars (blue) represent the total time for each action class. As illustrated in this figure, a serious imbalance occurs when the background is treated as a class equal to other actions. Consequently, it is preferably to first classify the background and foreground through binary classification.

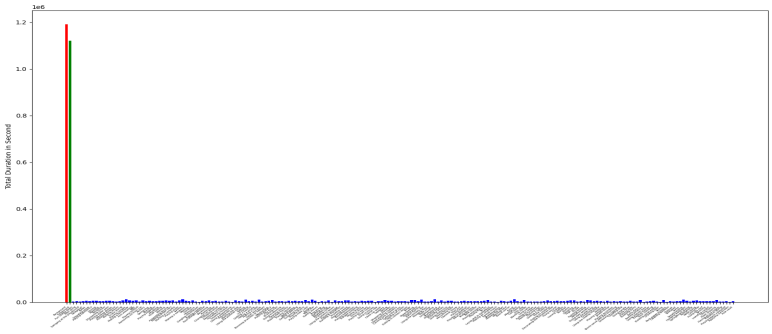
Table 2.1. Total time for foreground and background in THUMOS14 and ActivityNet v1.3 (Unit: s)

Dataset	Background	Foreground
THUMOS14	29087.82	12693.8
ActivityNet v1.3	1190704.18	1120239.74

Multi-label classification is a problem of assigning one or more class labels to samples. In TAL, it is used to assign corresponding action labels to samples classified as foreground through binary classification. Multi-label classification has been used in many fields and has made significant progress. Notably, it addresses the data imbalance problem that exists in various datasets. In general, learning is performed using a cross-entropy loss function, and samples are classified as belonging to one or more of the label sets. Although this method has shown excellent performance, it has been argued that data imbalance could negatively affect performance, and that the cross-entropy loss function may not be able to handle this problem effectively. Collecting the desired data to create a balanced datasets is often not easy. To



(a) THUMOS14



(b) ActivityNet v1.3

Figure 2.1. Histogram of background and actions. First bar (red) represents the total time of background, and the second bar (green) represent total time of foreground, respectively. the remaining bars (blue) represent total time of each class.

address this issue, an improved function called focal loss was proposed in the field of object detection, and this demonstrated improved performance [12]. In particular, creating a dataset for video data requires more effort and time compared to other types of data. Obtaining corresponding samples can be challenging if the frequency of occurrence of the desired actions is low or if risk factors are involved. As a result, there is often an imbalance in dataset used for action analysis. Researchers studying action analysis often borrow ideas from other related fields. Specifically, since work related to either objects or actions have taken similar approaches, many ideas have been adopted from the object analysis field. Focal loss was also adopted to address the data imbalance problem when performing multi-label classification in the field of action recognition. It has been shown to be effective in action recognition as well as object recognition [13]. Since then, numerous studies have attempted to tackle the data imbalance problem, and various ideas have been proposed to supplement the aspect that cannot be addressed by focal loss in the field of action recognition [14, 15].

2.1.2 Regression

Regression in machine learning is a technique that uses independent variables to derive outputs dependent on these variables. TAL can be viewed as a regression problem in that it attempts to find the actual start and end of an action through the context in which the action takes place or the relationship with the action itself. However, there are many cases where

the boundaries of human actions are not clear, and the information of the environment and action, which are independent variables used to find the boundaries, are often inconsistent. Several factors contribute to these difficulties. First, the supervised learning algorithm is heavily influenced by annotations. Annotation work can be done manually by a person or supplemented by a person after being done automatically using an existing algorithm. In the early stage of action analysis, a small number of simple movements, such as running, walking, and jumping, were collected in controlled environments such as laboratories. This allowed for relatively accurate annotation work and made it easy to identify errors [16]. However, recent research trends show that the size of the datasets is gradually increasing, making it increasingly difficult for humans to annotate manually. HMDB51 [17], considered a large dataset in 2011, contains 51 action classes in 6,766 videos. In contrast, current large datasets have grown exponentially in terms of the number of video clips and classes they contain. For example, Kinetics [18] includes 700 human actions in 650,000 video clips, and the Youtube-8M [19] dataset comprises 4,716 action classes in 8,000,000 video clips. The increase in size and number of classes in these datasets makes it nearly impossible to add and modify annotations manually, relying instead on automated algorithms or mass audiences in cloud systems such as Amazon. This change contributes to increased annotation inaccuracy and decreased action boundary precision. Second, it is challenging to find common start and end points in different actions. As mentioned in the Classification (Sub-section 2.1.1), binary classification is performed to separate the foreground and background. In one type of action, the start and end may have similar characteristics. However, finding the same pattern in different types of actions is not easy. In regression problems, outliers exist as factors that negatively impact performance. Outliers are data points that belong to the same class but have different characteristics from other samples, making the overall pattern appear distorted. In TAL, different actions can act as outliers to each other. Figure 2.2 shows three different actions (PoleVault (first row), HammerThrow (second row), and LongJump (third row)). In terms of the direction of movement, they have different characteristics such as up-and-down motion, rotational motion, and linear motion, making it difficult to identify a common pattern from the boundaries of different actions. Thirdly, the action may be interrupted or obscured due to video manipulation or camera movement. In such cases, some scenes that resemble the start and end of actions may appear in the middle of actions, resulting in the generating of false boundaries. Due to countless factors beyond those mentioned above, this aspect has become one of the most challenging parts of TAL. Therefore, researchers have attempted to solve this issue by treating it as an independent problem, often referring to it as the boundary refinement step. They have sought to detect more accurate boundaries by using the results obtained from other studies as prior knowledge. Detailed information on this is provided in Sub-section 2.3.4.

2.1.3 Generalisation

Creating a machine learning algorithm with a data set containing all possible samples is nearly impossible. Therefore, it is ideal for a trained model to perform well even on data not used for

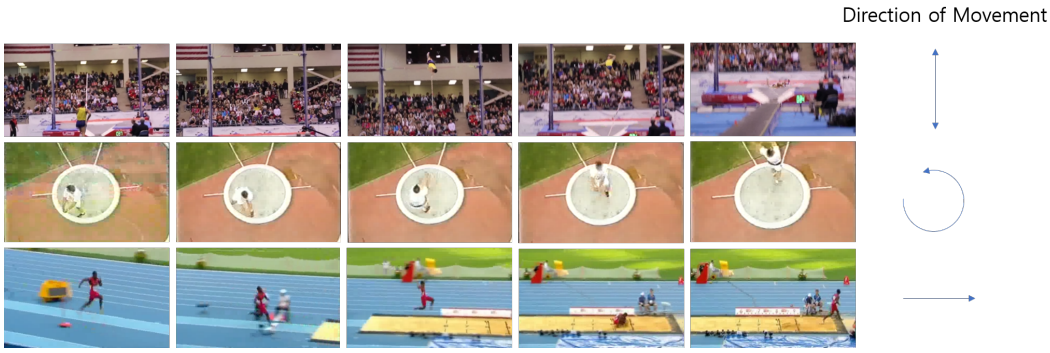


Figure 2.2. Three different actions: PoleVault (first row), HammerThrow (second row), LongJump (third row). In terms of the direction of movement, they are up-and-down, rotational, and straight, respectively.

training. In this regard, the issue of generalisation arises. In machine learning, generalisation refers to a trained model adapting well and showing good performance on new and unseen samples drawn from the same distribution as the training data. To achieve this, it is essential that the training data represents the data distribution in the problem domain well and that the model does not overfit the training data. A commonly used method is partitioning the dataset. During model training, overfitting can be checked by preparing a non-overlapping training set and validation set and running the model trained with the training set on the validation set. In addition to these basic techniques, there have also been efforts on the algorithmic side. One such approach is the use of boosting algorithms. Boosting algorithms began with the idea that a strong classifier can be formed when several weak classifiers are combined. They aim to derive reliable results by applying a weighting scheme to the outcomes of multiple weak classifiers. This idea has been adopted for action recognition. In [20], the authors attempted to detect actions by weighing the results of both temporal and spatial classifiers.

When a learning model performs well on the training data but not on new unseen data, it can be attributed to a potentially biased composition of the training data. Data augmentation techniques have emerged to transform biased data into unbiased data by generating new data using existing ones. Traditional data augmentation techniques include methods such as horizontal flipping and random cropping of images. Although these techniques have been effective, they may not represent uncollected data well. In another approach, there have been attempts to create possible but non-existent data using existing ones. With the advent of generative adversarial network (GAN) [21], which enable the creation of new images,there have been attempts to create new samples in various fields such as fashion [22], music [23] and art [24]. In the field of video understanding, researchers have tried to address the data imbalance problem by using newly generated videos. In [25], video augmentation was performed through GANs, and it was demonstrated that this method can achieve better performance compared to traditional methods when applied to action recognition. Additionally, efforts have been made to prevent overfitting by the algorithm itself and to secure generalisation. In traditional machine learning, regularisation is used, which prevents overfitting by constraining parameter values to satisfy specific conditions. Recently, with the advancement of deep learning in the field of machine learning, special layers such as dropout have been employed to address the generalisation problem. In the field of action detection and recognition, bench-

marking datasets for performance comparison have been created and provided by numerous research groups. However, it is still challenging to find algorithms that yield consistently good results across all datasets every year. Researchers often report results obtained for each individual dataset. To investigate this phenomenon, a cross-corpora test as conducted which is reported in Section 3.5.

2.2 Video Representation

In order to analyse a video and comprehend its contents, it is important to determine how to represent the video in a way that makes the extracted information useful. The amount of information available may vary depending on the video representing method employed. Pixels, which represent low-level information, play a significant role in conveying the colour information of regions of interest (e.g. objects or humans). However, humans can perceive more than just colour when viewing a video. A wealth of information can be gleaned, such as what is happening, who and what is present, where events are occurring, and so forth. What might happen in the near future can be predicted. This implies that more abstract and meaningful information can be derived from videos than simply colour. So, how should a machine process videos in order to extract such information, emulating human abilities? Initially, researchers have attempted to represent videos using human knowledge. This method is known as a hand-crafted representation. In this approach, a region of interest is first identified by a detector. The detected regions are then expressed using feature description methods designed by human experts. Since this method is based on human knowledge, it is easy to interpret and intuitively understandable. However, valuable information may exist in areas where people are not aware of it. This can be missed even by experts when devising a descriptor based on their experience and knowledge. To address this, researchers have come to rely on learning-based representations. Learning-based representations utilise specialised learning algorithms to automatically extract and express the necessary information from raw data in order to achieve a specific goal. Although these methods have resulted in significant improvements in performance, it may not be easy to intuitively understand which part of the data the information originates from or how the information takes its final form. In TAL, video representation derived from the field of action recognition is utilised as input for boundary detection. To provide context, hand-crafted representations and learning-based representations are briefly reviewed for action recognition. Following that, the methods predominantly used in TAL among learning-based representations will be discussed in greater detail.

Hand-crafted Representation

Hand-crafted video representations can be categorised into local descriptor and global descriptor based on the area from which information is extracted. Local descriptors are expressed using local information, such as texture and colour, from a limited area such as interest point or cuboids determined by a feature detector. In contrast, global descriptors involve expressing an action using global information, such as illumination changes, phase shifts, and

speed variations, that may appear throughout the entire screen while an action is taking place. Once the area for information extraction is determined, hand-crafted video representations can be further divided into spatio-temporal representations and appearance-based representations based on the information extracted from the designated area. Spatio-temporal representation involves expressing information in a volume-shaped space by concurrently considering both spatial and temporal axes. In [26], two 2D images, binary Motion-Energy-Image (MEI) and Motion-History-Image (MHI), are utilised to represent actions in a volume-shaped space for template matching between actions. In [27], a Volume Motion Template (VMT) and projected Motion Template (PMT) are proposed for view-independent HAR, using images obtained from a stereo camera. Actions in 3D space can also be represented by the trajectories of human body joints. In [28], the performance of motion-based descriptors (e.g., histogram of optical flow (HOF), motion boundary histogram (MBH)) is significantly improved by eliminating camera motion, using local feature descriptors such as speeded up robust feature (SURF) [29], while obtaining motion trajectory information from dense optical flow. In [30], trajectories are composed of a sequence of skeleton shapes acquired using a camera, such as a Kinect sensor, which can capture depth images in real-time. The aforementioned method is known to be computationally efficient by using Transported-Square Root Vector Fields (TSRVFs) of trajectories and the standard Euclidean norm. It is also robust to various execution rates. Appearance-based representation is a method designed to efficiently use the shape and appearance of an action. In [31], actions are represented by dividing the human silhouette into a fixed number of cells. In [32], contour points on multi-view key poses are used to represent actions for view-invariant action recognition.

Learning-based Representation

Learning-based representation is a method that automatically learns features from raw data, and it can be divided into two types: non-deep learning-based representation and deep learning-based representation. Among non-deep learning-based representations, dictionary learning is the most widely used method. It learns a sparse representation called an atom from a large number of samples, and an action is expressed as either an atom or a combination of these atoms. In [33], extreme points are obtained from the human body, and local motions are defined by a local motion descriptor using the changes in these points within a cubic area of a certain size. Three over-complete dictionaries are composed of these spatio-temporal descriptors. In [34], a hierarchical representation is proposed. After detecting interest points, motion and appearance features are created for spatial information. In the case of temporal ordering information, a motion segment descriptor is proposed, which is created by concatenating features. In [35], transferable dictionary learning is performed for cross-view action recognition. By constructing dictionaries in each view and making them transferable between dictionaries in different views through a learning method, actions captured in multiple views share a common dictionary. Besides dictionary learning, evolutionary methods have also been employed. In [36], genetic programming is used to learn descriptors that can adapt between different datasets. Using scale and shift invariant features extracted from a sequence of colour and optical flow images, spatio-temporal motion features can be learned

automatically with the aid of evolutionary methods. Non-deep learning-based representation can automatically learn features and represent actions efficiently compared to hand-crafted representations. However, to create the input data necessary for learning, additional work is required, including tasks that detect either interest points or extreme points. With the advancement of deep learning, recent research trends have focused on representing videos using deep learning for action recognition, resulting in significant improvements in performance.

In the remainder of this section, further details are discussed concerning the deep learning-based representations primarily used for temporal action detection.

2.2.1 3D Convolution Network

Due to the success of deep learning in image representation for object recognition, there have been attempts to employ deep learning for video representation for action recognition. Numerous 2D convolution networks have been proposed for image representation [37, 38, 39]. However, several challenges have arisen when applying these networks to videos. While an image is a 2D plane, a video is expressed in a 3D space, including the time axis. Consequently, using such a network can lead to loss of temporal information about motion; the primary concern is therefore how to effectively preserve this information using deep learning. To address this issue, the convolutional 3D (C3D) network [40] employs convolution layers with 3D filters instead of 2D filters. By doing so, it is possible to learn spatio-temporal information of actions using the same architecture that has previously demonstrated good performance in object recognition. Although this method is simple and effective, it has the drawback of requiring a large amount of computation due to the use of 3D convolutions. As a result, numerous subsequent work has been conducted to mitigate this disadvantage while retaining temporal information. Firstly, the use of continuous 3D convolution was considered a factor that hindered computational efficiency. To address this, 2D convolutions and 3D convolutions were combined. In [41], the authors aimed to increase computational efficiency by reducing the size of the output. They executed 2D convolutions and 3D convolutions in parallel in each layer and applied 2D convolutions to the result once more. Another line of research attempted to achieve the same effect by decomposing the 3D convolution into several more efficient operations. In [42], the researchers approximated 3D convolutions by performing spatial convolutions and temporal convolutions separately. Instead of the 3x3x3 size filter used in C3D, they employed 1x3x3 and 3x1x1 size filters. The first filter was used to learn spatial information by producing the same effect as 2D convolution, while the second filter was used to learn temporal information. Similarly, [43] attempted to implement an approximated 3D convolution, called a (2+1)D convolution, using a 2D convolution followed by a 1D convolution. Lastly, they sought to learn video representation by repurposing networks designed for image representation. In [44], the authors attempted to achieve the effect of 3D convolution using only 2D convolution by manipulating the input data. To encode the temporal evolution into a single image, a 2D image called a dynamic image containing temporal information was created from multiple images using the rank pooling concept. This method has the advantage of being able to reuse networks that have successfully performed image

recognition without losing temporal information, by converting the input form from video to image.

2.2.2 Two-stream Network

Unlike the 3D convolution network, the two-stream network is a method of adding additional information by creating another network alongside the 2D convolution network. Having a network that uses 2D convolution means that any network used for image representation can be reused. Additionally, it has the advantage of allowing any type of additional information to be inserted through the network of another stream. Among these two-stream networks, there is a model designed to learn temporal information through the second stream. This is an effort to obtain a spatio-temporal representation by adding the temporal relationship between adjacent frames, which cannot be learned in a 2D convolution network that uses still images as learning data. In [45], motion information along the time axis is learned through optical flow images. The stacked optical flow is calculated using the neighbouring images around each colour image, and both the colour image and corresponding optical flow images are used as input data for the two-stream networks. The output of the two streams incorporates both spatial and temporal information. In the inflated 3D convolution (I3D) network [46], the authors believe the original two-stream network is overly simplistic, so they modify the existing model to enhance performance. In the earlier version of the two-stream approach, a 2D convolution stream for colour image and a 3D convolution stream for stacked optical flow were employed. However, they opt to use two 3D convolution streams by inflating the 2D convolution stream. To reduce computational cost, an inception module is introduced. In [6], the authors express concerns about the difficulty of integrating optical flow computation into a two-steam network in an end-to-end manner and propose learning motion information without such computation. They developed two pathways with distinct frame rates: a slow pathway operating at a low frame rate captures spatial semantics, while a fast pathway operating at a high frame rate captures motion. Additionally, they introduced lateral connections between the two streams to fuse the different types of information effectively.

2.2.3 Attention-based Network

In psychology, it is well-established that when people process visual information, they do not pay equal attention to all visual stimuli but selectively focus on specific parts of the information [47]. In machine learning, one technique that emulates human visual cognitive processing is the attention mechanism. The primary challenge in the attention mechanism is determining the attention weights that represent the areas to focus on and those to ignore. The simplest method for calculating these weights involves using a linear model. In [48], the authors proposed an input-dependent module that utilises a fully connected layer with a tanh activation function to obtain weights for both spatial and temporal attention from human skeleton data. The module, proposed in Action Attention Recalibration Module (AARM) [49], generated two attention maps using convolution operations, creating an inter-channel

attention map and a spatio-temporal attention map to identify informative parts in a video. For action recognition, the authors improved classification performance by calculating attention weights for the final activation function. Non-local networks [50] introduced a non-local block in the form of self-attention, which can be incorporated into existing networks. This module enables the capture of specific, important information related to a person's action and the surrounding context. One of the most notable developments in attention mechanisms is the transformer [51]. A transformer is a network composed of encoders and decoders that eliminates recurrences and convolutions found in existing networks, relying solely on attention mechanisms. The attention mechanism was further systematised by introducing scaled dot-product attention and multi-head attention. This method demonstrated excellent performance for sequential data and has been used with remarkable success in various fields, particularly language-related tasks. Inspired by this method, the video action transformer network [52] adapted the transformer architecture for video applications, proving that the transformer architecture can also be used for spatio-temporal data such as videos. ViViT [53] proposed a more efficient approach for applying the encoder part of the transformer to video classification. The authors factorised the different components of the transformer encoder to address the spatial and temporal dimensions of the video separately.

2.2.4 Others

As previously discussed, numerous action recognition networks have been proposed, with some being adopted for TAL in video representations. However, some argue that action recognition is not well-suited for localisation tasks because it primarily focuses on the video's content. Temporally-Sensitive Pretraining (TSP) [54] contends that existing networks are specialised solely for classifying positive samples, which negatively impacts the accuracy of the localisation task. To address this issue, researchers are exploring ways to incorporate the presence or absence of information in video representations.

2.3 Temporal Action Localisation

TAL aims to achieve two goals: boundary detection and action classification. Boundary detection identifies the start and end timestamps of an action in an untrimmed video, while action classification recognises the specific action within the video. Depending on the strategy employed to perform these tasks, TAL is divided into one-stage temporal action localisation (end-to-end method) and two-stage (or multi-stage) temporal action localisation (pipeline method). One- stage TAL simultaneously addresses both tasks within a single integrated network, while two-stage (or multi-stage) TAL tackles them sequentially by dividing them into several independent tasks. In this thesis, a two-stage TAL method is employed. Thus, the related background will be reviewed. Two-stage (or multi-stage) TAL breaks down the whole task into smaller, independent ones. Following the "detection-then-recognition" scheme in object recognition, the boundary of an action is first detected, and then the seg-

ments of the detected action are used as input for recognition. The step of detecting action boundaries, known as action proposal generation, has emerged as an independent research area. This field generally involves a three-step process: feature extraction, proposal generation, and post-processing. In the feature extraction step, feature vectors used for the localisation task are created through video representation methods. The proposal generation network uses these extracted features to distinguish between the background and foreground in an untrimmed video, subsequently detecting action candidates. Finally, in the post-processing step, proposals with significant overlap are removed to reduce redundant detection.

This section presents an overview of action proposal generation and its relevance in the context of Temporal Action Localization (TAL). The primary objective is to describe the adopted methodology for handling actions of varying durations and proposal representation methods that can effectively capture the essence of these actions in a consistent manner. Additionally, the section introduces approaches for efficient search of lengthy videos, refining action boundaries to improve precision, and eliminating redundant outcomes.

2.3.1 Variable-Length Action Duration Modelling

Actions occur in various forms, depending on the context and the individual performing the action. Even when repeating the same action, it might be performed differently each time. To process these variable-length actions within a single network, a consistent representation method is required. In this section, commonly used methods will be examined found in the literature.

Grouping

Grouping is a method used to evaluate the length of an action in a bottom-up approach, addressing the variable length of actions. This technique assesses whether something is an action within the smallest temporal unit and predicts the length of various actions by combining units that achieve consecutive high scores into a single action. A classifier is applied to determine if an individual snippet, represented by a feature vector, is an action or not [55]. Based on the actionness scores, snippets with a high probability of being actions are grouped together to establish the action's length. The watershed algorithm [56] is employed as a grouping method in this process. Proposals are generated by connecting snippets that merge into one while the water level is raised.

The length and position of an action can sometimes be represented using parameters of a specific kernel. In [57], Gaussian kernels are applied temporally to represent the dynamic temporal scales of an action. To achieve this, the model is trained to predict the centre of the action and a specific interval in each cell of each feature map. Action proposals are then expressed with a Gaussian kernel, where the centre of the action is represented as the kernel's mean, and the action interval is represented as the standard deviation. Grouping is performed based on the relationship between these kernels. Kernels with a high degree of overlap are considered to represent a single, longer proposal and are merged into one kernel.

Grouping offers the advantage of efficient prediction without limiting the evaluation length. As result, it is considered an appropriate method for detecting actions, taking into account the unpredictable length of an action and the algorithm's scalability. However, a challenge arises when using small snippet units to determine if an action is present for grouping purposes. Since snippets are the smallest temporal segments, they may only capture incomplete parts of actions. Relying solely on partial information to determine if it is a positive sample reduces the accuracy of the decision. If a false positive sample occurs in the middle of an action, a single action may be split into multiple actions. To address these drawbacks, complementary methods have been proposed in the literature [58].

Anchor-based Method

As a top-down approach, predefined temporal segments called "anchors" are used to evaluate the completeness of actions. Since each anchor has a fixed-length value, it can only represent one action length. Therefore, multiple anchors are employed to accommodate various lengths. In [59], anchors were expressed using a Recurrent Neural Network (RNN) rather than being directly defined. A many-to-many RNN configuration is used, where the output from each time step evaluates the length from the beginning of the input sequence to the corresponding output time. The action length is determined based on the number of time steps set in the model.

In contrast, other work utilising anchors employs a set of predefined anchors. Each anchor is evaluated to predict whether the corresponding temporal span represents a complete action or not. These approaches can be found in various literature sources [58, 60, 61, 62].

The performance of anchor-based methods depends on the configuration of the anchors. Anchor settings can be established through empirical knowledge. By using a large number of anchors to cover various possible action lengths, performance can be improved. Moreover, if the length is determined based on expertise in the field, a more efficient set of anchors for detection can be configured. However, anchor-based methods have some drawbacks. Since it is necessary to evaluate whether all anchors represent an action at every time location, the computational cost increases due to the repetition of the same operation for each anchor. Additionally, since anchors have a fixed length, they may not accurately detect action boundaries if their lengths differ slightly from the actual action. As a result, an additional boundary refinement process is needed to address this issue.

Anchor-free Method

The anchor-free method was developed to address the disadvantages of anchor-based approaches. In the field of object detection, a technique for detecting objects by directly predicting the offsets of the bounding box, without using an anchor box, was proposed and demonstrated excellent performance [63]. Inspired by this success, researchers have attempted to detect actions without using anchors.

To resolve the high computational cost associated with using a large number of anchors, a problem in anchor-based approaches, the use of anchors was eliminated. Instead, researchers

directly predicted the left and right offsets of the action from the input vector, streamlining the detection process. Additionally, since the boundary is determined by the prediction method rather than using fixed-length information, it overcomes the limitation of generating inaccurate boundary values in the anchor-based method. However, this approach has to predict offsets from a single feature vector, which may contain incomplete information about actions, making it challenging to obtain accurate offsets. As a result, an additional structure is needed to incorporate multi-scale temporal information into a single vector representation.

To address this issue, various methods have been proposed (which will be discussed in Sub-section 2.3.3). With the assistance of these methods, action detection has been performed in an anchor-free manner, as seen in multiple studies [64, 65, 66].

Boundary-based Method

The ultimate goal of action detection is to accurately detect the boundaries of an action. Several existing algorithms have indirectly predicted action boundaries by evaluating the probability that a specific part of a video represents an action and the completeness of the action. In contrast, boundary-based methods directly compute the probabilities of the start and end of an action from the input sequence.

In [67], the boundary scores are calculated directly from the input data, and these scores are used to create boundary pairs (start and end) of actions for proposal generation. This approach shifts the focus to identifying action boundaries more explicitly, potentially leading to improved detection performance. The action segments created in this manner are used to calculate the actionness score, as previously described. Extending this idea further, methods such as BMN [68] and DBG [69] develop a more systematic approach to creating action pairs, which was done using heuristic rules in earlier approaches. Rather than creating pairs directly, these methods generate a score map in the form of a table for all possible temporal pairs. With the horizontal and vertical axes of the map representing starting and ending times, respectively, the score map contains scores for all potential pairs. These obtained score maps are then used to determine the ranking of proposals, further refining the action detection process. In [69], researchers addressed the issue of noisy boundaries affecting the calculation of starting and ending scores. They posited that the starting point of an action can be seen as the ending point when the action proceeds in the opposite direction. By leveraging this idea, they aimed to generate more stable boundary scores in a complementary manner. In [70], researchers accepted the idea of the previous boundary-based method but noted that boundary scores predicted by convolution operations might only capture local temporal relationships between nearby frames. This is due to the limited ability to model long-term temporal information when using information constrained by the kernel size. To tackle this issue, the researchers introduced a transformer architecture to better capture long-range temporal relationships and improve action detection performance.

The boundary-based method offers the advantage of being less likely to miss an action, thanks to its dense evaluation of all possible pairs. However, calculating scores in the form of a map to evaluate all possible cases results in a high computational cost and significant memory

requirements to handle these values. This trade-off between increased detection accuracy and computational complexity must be considered when employing boundary-based methods for action detection.

Graph-based Method

In the feature extraction step, a video is represented as a single vector in snippet units. The graph-based method is an approach that constructs a graph to determine which vectors come together to form a single action. Each vector constitutes a node in the graph, and the edges represent the relationship between the nodes. This structure can also be used for convolution operations. Unlike conventional convolution operations, the number of nodes participating in the convolution is not fixed, allowing for more flexible model training. By leveraging the graph-based structure, this method aims to better understand and model the relationships between action components in a video sequence.

In [71], the Graph Convolutional Network (GCN [72]) was used to address the action localisation task as a problem of finding target sub-graphs related to the action. The graph consists of snippet nodes, temporal edges to maintain the temporal order of the video, and semantic edges learned from the nodes. Through multiple convolutional operations, the semantic context is encoded into semantic edges, and actions are detected by evaluating sub-graphs defined by anchors (a fixed number of nodes). In [73], a Graph Reasoning Module (GRM) is proposed. This approach configures each node of the graph as a snippet and connects two nodes as a result of learning when the two nodes indicate the start and end of an action. Consequently, nodes connected to each other in the constructed graph represent action proposals.

The graph-based method offers the advantage of handling actions of various lengths because it allows for random access to all nodes and can express the relationship between nodes through edge weights. By leveraging this flexible structure, graph-based approaches aim to adapt to diverse action lengths and better capture the relationships between different parts of an action sequence.

2.3.2 Proposal representation

In Sub-section 2.3.1, methods for predicting the lengths of actions with varying durations are examined. These predicted actions generate proposals comprised of several feature vectors. Proposals of different lengths consist of different numbers of vectors, but in order to evaluate whether they represent actions, a unified representation method is needed to perform classification and regression in a consistent algorithm. This representation is mainly constructed using pooling and concatenation. Pooling summarises the feature maps generated by the convolutional layers. This process has the effect of reducing the computational cost by decreasing the parameters of the network and the dimension of the feature map. By utilising the pooling effect, actions composed of various numbers of vectors can be expressed as a vector of the same size. Common pooling methods include max pooling, which replaces an area with its

largest value, and average pooling, which uses the average of the area to represent it. Additionally, to emphasise specific parts of the data, weighted average pooling is applied using a specialised weighting scheme. Concatenation is a method of creating a single, meaningful vector from a series of data. While the pooling operation can produce a vector of the same size from various numbers of data, the loss of detailed information, such as temporal ordering, occurs when creating a single vector composed of several vectors. On the other hand, concatenation is a method that retains more information but leads to an increase in dimensionality by connecting several vectors. These two methods can be used separately or together. To apply both methods to a sequence of vectors, it is necessary to decide which vectors are used to create a proposal-level representation. According to this approach, proposal representation can be divided into dense sampling-based representation and sparse sampling-based representation.

Dense Sampling-based Representation

Dense sampling-based representation involves using all feature vectors related to each proposal. It is chosen to include all information without discarding any data. This approach has been widely used to express video-level representation in the field of video classification.

The most commonly used method in dense sampling-based representation is average pooling, which is employed to represent the region of action defined by an anchor at each temporal location. Unlike action recognition, in the context of TAL, average pooling is applied to three parts of proposals to incorporate local context information. This method defines the internal area between the starting time and the ending time, representing the action itself, and defines a certain area before and after the internal unit as the context area. This inclusion of the situation before and after the action occurs within the action representation helps provide more comprehensive information. After average pooling is applied to the internal area and the context area, which are expressed as three vectors, they are concatenated to create the complete action representation. This approach has been adopted in many cases, with the only difference being the context-setting method [60][74].

In [57], actions were composed of multiple Gaussian kernels, and these kernels were weighted averaged to represent action proposals. This method, called Gaussian pooling, generated vectors for subsequent tasks. The authors believed that the data at the centre of the action was more informative than the data at the border, so they wanted to apply different weights according to their positions within the action. These weights corresponded to the Gaussian kernel representing the actions. Moreover, since the standard deviation value expresses the length from the centre of the action to the boundary, the context information is naturally included in the proposal presentation in the tail part of the Gaussian kernel. As a result, there was no need to set a separate area for including context information.

Sparse Sampling-based Representation

Sparse sampling refers to a technique that reduces the amount of data collected or processed while still maintaining sufficient information. Instead of capturing or processing data

at regular intervals or densely sampling the entire dataset, sparse sampling strategically selects a subset of representative samples. In [55], inspired by [75], the authors proposed a structured temporal pyramid to represent actions. [75] used a proposal-level representation based on sparse sampling instead of the video-level representation based on dense sampling, which is commonly used for video classification. To achieve this, they created an extended proposal that included the context area before and after the action. This extended proposal was divided into nine sections, and one vector from each section was randomly sampled and concatenated to complete the action representation. In [58] and [67], the proposals were divided into three parts: one for the proposal itself and two for the context areas (before and after the action). The two context areas in these methods consist of frames surrounding the action boundaries, which means they include a small portion of the action itself. By performing sparse uniform sampling in these three areas, one proposal unit and two context units are created and concatenated to represent the proposal. In [68], the authors pre-calculated a weight mask in the form of a matrix to make sparse sampling more efficient. Since the sampling position appears as a decimal point, the coefficients multiplied by the vectors are pre-calculated using linear interpolation. This approach leads to significant computational cost savings while still effectively representing action proposals and their surrounding context.

Using sparse sampling provides several advantages in action detection algorithms. Firstly, the computational cost is greatly reduced, as redundant information is removed, and only necessary data is used. This allows for more efficient processing and analysis of the video data. Secondly, since the same number of samples is selected from all proposals, it is always possible to represent them in the same size, regardless of the length of the proposal.

2.3.3 Temporal Duration Handling Methods

Detecting actions in untrimmed videos, which can vary in length from a few minutes to a few hours, presents unique challenges compared to action recognition in trimmed clips. To efficiently traverse videos and detect actions, various strategies can be employed. Here, some of these strategies are discussed.

Sliding Window

The sliding window strategy processes sequential data by sliding a fixed-size window over the sequence and evaluating the data within the window at each position. The performance of the sliding window method can be affected by the size and stride of the window. However, there is no theoretical method to determine these parameters, and they are usually chosen based on practical experience and the nature of the data. The sliding window technique is commonly used with top-down proposal evaluation methods such as anchor-based approaches, or when processing long sequential data by dividing it into segments.

Anchors for action detection can be viewed as sliding windows with varying settings. A small window size is utilised to detect relatively short actions, as it enables finer searching, while a large window size is ideal for detecting longer actions by processing a substantial

amount of data simultaneously. When employing the sliding window approach, a crucial hyperparameter must be determined: whether all anchors share the same stride value or each anchor has a unique stride value. Using the same stride for all windows simplifies implementation since it can be controlled with a single value. However, adopting a small value might be suitable for short actions, but it can lead to inefficient calculations due to repetitive redundancy in longer windows. Alternatively, the overlap ratio of windows at adjacent locations can be considered. This parameter represents a trade-off between computational efficiency and detection accuracy.

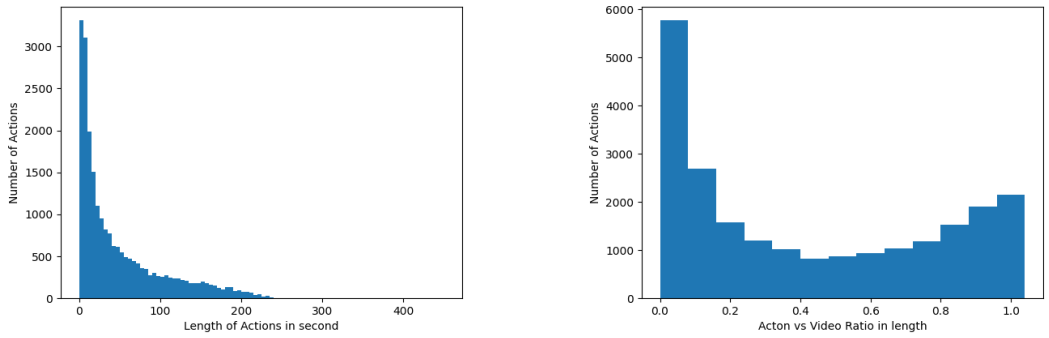
In the literature, the parameter selection is often determined empirically based on the length of the target action class present in the dataset and the overall video length. In [76], the authors implemented a sliding window strategy to run the RNN on untrimmed videos, requiring only one window size. To determine this size, they selected one that could cover 98% of the action by referring to the ground-truth annotations and validated the setting experimentally. This configuration has since been adopted by several researchers [68, 69, 77]. Regarding the stride of the sliding window, many cases involve an overlap of 75% [58, 60, 78].

Video Resizing by Linear Interpolation

In a dataset, videos have varying lengths. After the feature extraction process, each video is represented as a set of vectors with different numbers of vectors. However, by applying linear interpolation, it is possible to make each video have the same number of vectors. In the literature, this method has been employed specifically for handling the ActivityNet dataset. As depicted in Figure 2.3, the distribution of action lengths within the dataset is skewed. However, when expressed as a ratio of action length to video length, the distribution appears relatively even. The use of linear interpolation to rescale each video can help alleviate data imbalance in the dataset. This approach is particularly beneficial for efficiently handling long videos. To detect long actions within long videos, it is necessary to have a means of processing these videos within the network model. However, doing so demands significant memory and increases computational costs. By rescaling videos to a more reasonable length using linear interpolation, long videos can be processed with reduced memory requirements and lower costs. For this purpose, in studies such as [67, 68, 69, 77], videos were rescaled to a length that could be processed in a single pass, allowing the dataset to be used without the need for repetitive operations like sliding windows.

Multi-scale Pyramid

Pyramid structures, as described in [79], are designed to process images in a scale-invariant manner. These structures have been introduced to tackle multi-scale challenges in performing detection tasks and have demonstrated positive results [80][81]. The pyramid structure stacks images of varying resolutions. The lower layer, with a higher resolution, contains detailed information and is therefore well-suited for handling small objects. Conversely, the upper layer, with a lower resolution, is better equipped to manage larger objects.



(a) Action Length in Second

(b) Length Ratio of Action to Video

Figure 2.3. Histogram of action length in ActivityNet 1.3 dataset. The dataset contains a large number of short actions, resulting in an imbalanced distribution when looking at the length of action in seconds. To alleviate this, a sequence of feature vectors can be rescaled using linear interpolation.

In further development, [82] discovered that data at each scale contributed to performing tasks at different scales, and incorporated links allowing data at each scale to receive information from other scales. This approach demonstrated outstanding performance in object detection tasks. Owing to this success, the pyramid structure has been employed in action detection to manage actions of varying lengths from a multi-scale perspective.

Feature pyramids can be created through convolution and pooling operations. To generate a layer with a lower resolution from one with a higher resolution, the data can be processed either by reducing the number of output filters in a 1D convolution or by applying a pooling operation to the feature map. In [65], a temporal reduction unit consisting of four convolution layers was proposed to create a subsequent feature map with a larger receptive field in a pyramid structure. The first three convolutions operate with a stride of 1, while the last convolution used a stride of 2 to create a feature map with half the scale. [64] introduced a U-shaped architecture to convey both high-level semantic information and low-level detail information. After applying the convolution operation, a deconvolution operation was performed to create a U-shaped pyramid architecture. Lateral connections were then inserted between the layers to enable information fusion. In [83], a relation-aware module was proposed, similar to the self-attention mechanism, to address the long-range dependency problem in U-shaped architectures. In [84], a pyramid non-local block was proposed to handle multi-scale challenges. A pyramid with different scales was created using average pooling, and a non-local operation was applied to obtain long-range context information in each layer. Subsequently, a feature representation with multi-scale information was generated through upsampling, convolution, and concatenation operations.

The pyramid structure is primarily employed in TAL algorithms for detecting actions using an anchor-free method. As the anchor-free method predicts offsets with a single feature vector, these vectors need to contain scale information corresponding to the action being detected. It is expected that each layer of the pyramid structure will detect actions with lengths similar to their corresponding scales, allowing for more precise action localisation across different lengths.

2.3.4 Boundary Refinement

In TAL, detecting the exact boundary of an action is a crucial task. However, accurately predicting boundaries is challenging due to the inherently difficult-to-define nature of action boundaries. Researchers often perform boundary refinement to readjust the detected action boundaries, making them more accurate. In this section, the proposed methods are reviewed for refining action boundaries. In the literature, boundary refinement typically adopts one of two strategies: coarse-to-fine refinement or progressive refinement. Coarse-to-fine refinement mainly involves using dedicated modules to enhance the accuracy of detected boundaries, while progressive refinement iteratively employs the same boundary detection module to refine the boundary with increasing precision.

Coarse-to-fine refinement initially performs a coarse detection to obtain preliminary action detection results. These results are then refined further using additional modules to enhance their accuracy. In [85], the authors addressed this problem by designing a novel type of boundary feature for refinement and performing regression once more. Recognising that pooling methods such as mean pooling and Gaussian weighted average do not adequately represent action boundary information, they proposed a boundary pooling method that leverages salient boundary features. Based on the start and end points of the action detected by coarse detection, a certain region is designated as a boundary area. Large values, such as those obtained through max pooling, are considered salient features of the boundary. The start and end features composed of these salient values are concatenated to obtain a salient boundary feature. Utilising the salient boundary feature, boundary regression was performed again to readjust the boundary. In [86], the authors introduced a structure resembling an inverted pyramid to refine the proposals detected in the feature pyramid. The detected proposals from the coarse detection step were refined by incrementally increasing the temporal resolution and merging features from the feature pyramid. Moreover, frame-level features were generated from the output of the reverse pyramid, and the final boundary was determined using dilated convolution.

On the contrary, progressive refinement iteratively uses the same module to determine increasingly more accurate boundaries, presenting a trade-off between the number of iterations and accuracy. In [74], videos are extracted using snippet-level representation. The authors believed that although frame-level coordinate regression might yield more accurate results, snippet-level coordinate regression was easier to learn since frame-level features were not discriminative enough for regression. However, unit-level regression lacked accuracy, so boundary refinement was performed. More refined results were obtained by using the regression and classification outcomes from the previous step as input. The number of repetitions was determined experimentally. In [87], cascaded boundary refinement was conducted based on the same concept. Specifically, they aimed to improve the boundary in the boundary-based approach. The boundary-based approach involves detecting an action among all possible temporal location combinations in a video with a uniform temporal interval, and the precise boundary is influenced by how the temporal interval is configured. However, it was argued

that small intervals increase the computational cost, making the entire process inefficient. To address this issue, additional boundary refinements were performed to obtain accurate boundaries without increasing computational complexity. Ultimately, a cascade boundary refinement (CBR) module was proposed, and it was executed iteratively to output finer boundaries for the proposal. In [88], the authors attempted to address the refinement problem by utilising the dependency between feature vectors. To achieve this, local and global dependencies were calculated using cosine similarity. Local dependency is determined using similarity within a specific window, while global dependency is calculated using similarity with the entire input. Each feature vector is encoded to include both local and global information. With this encoding method, a regression module predicted more precise boundaries, where frame-level and segment-level boundary regression work complementarily. Lastly, complementary and progressive boundary refinement was performed by repeatedly applying this method.

Both methods have been experimentally proven to detect more accurate boundaries; however, they have their respective challenges. The coarse-to-fine method necessitates additional model design, while the progressive method requires determining the optimal number of iterations for refining the boundaries.

2.3.5 Elimination of Duplicate Result

Many temporal segments with only slight differences in location share numerous similar characteristics. These areas have comparable scores during the inference step, resulting in the generation of many similar outputs. To address this issue, numerous researchers implement additional post-processing techniques or incorporate modules that prevent the learning algorithm from producing redundant outputs.

Post-processing The most widely used method for eliminating redundant detection in TAL is non-maximum suppression (NMS). NMS is an algorithm that selects a single entity by retaining only the entity with the maximum value among those in similar positions while removing the rest. Traditionally, it has been employed in computer vision for edge detection tasks, where it is used to make bold edges thin [89]. The application domain of this algorithm has expanded to cover multiple detection tasks, effectively removing redundant detection results. Specifically, in the field of object detection, it has been used to select one of the overlapping bounding boxes, exhibiting excellent performance. In action detection, the same method is adopted to address the issue of redundant detection.

NMS is primarily categorised into two types: hard NMS and soft NMS. Hard NMS represents the original form of this algorithm, retaining only the entity with the maximum score and suppressing all other entities. Upon selecting an entity with the maximum score, highly overlapped entities are removed using a predetermined threshold applied to the remaining entities. During this process, intersection over union (IoU) is used as an evaluation criterion for object detection, while temporal intersection over union (tIoU) is employed for action detection. However, due to the algorithm's nature of removing overlapped entities using a predefined threshold, a problem arises when closely existing objects or actions are removed,

leading to misses. To address this issue, soft NMS has been proposed [90]. Soft NMS holds duplicate entities by decaying their corresponding scores instead of completely removing them.

NMS is a simple yet effective method that is widely employed in both object detection and action detection [70].

No post-processing

While NMS demonstrates excellent performance in removing redundant detection, it has the drawback of necessitating a separate pipeline. In deep learning, end-to-end networks offer the advantage of consolidating learning and testing within a single network, eliminating the need for separate processes. However, integrating NMS into learning is challenging, and it is typically employed as an additional step during inference time. To overcome these limitations, methods have been proposed for conducting learning without generating duplicate entities.

These efforts were first attempted in the field of object recognition. In one study [91], researchers tried to directly implement NMS as a convolutional network. They proposed a module that generated pairwise representations for all detection results and re-scored the current detections using them. Another study [92] addressed this problem by formulating it as a classification problem that distinguished between correct and duplicate detections. Similar to a network module primarily performing object detection, a classification module was inserted and generated scores indicating whether each detection was correct. The initial detection scores were then decayed by multiplying the scores. In these approaches, the goal of creating end-to-end trainable networks was achieved by rescaling scores during training, similar to soft NMS.

Alternatively, the problem was treated as an assignment problem that assigns one entity among several redundant detections to a single ground-truth annotation. In one study [93], the Hungarian algorithm [94] was utilised. Direct prediction was performed without post-processing, such as NMS, by applying a loss function based on bipartite matching between ground truth and prediction. Adapting this idea for action detection, another study [66] addressed the duplicate detection problem through bipartite matching as well. However, unlike object detection, multiple detected action proposals were permitted to be assigned to a single ground truth in action detection. The reason for this is that generating temporal action proposals is more ambiguous than defining bounding boxes for an object, as actions may occur intermittently in a video or span the entire video, making it difficult to assign a single entity to a ground truth. The researchers believed that forcing proposals to match only one ground truth could have a detrimental effect on obtaining a stable solution.

Indeed, the two methods mentioned above appear to be attempts to implement the concept of NMS as an end-to-end approach. Rescaling the scores during training is akin to soft NMS, while retaining a single detection result through matching is a form of hard NMS.

2.4 Miscellaneous on Temporal Action Localisation

Action detection in untrimmed video is typically performed using the techniques described in Sections 2.2 and 2.3. For example, anchor-based methods employ a sliding window strategy to traverse a long video using predetermined anchors, while anchor-free methods use a multi-scale structure such as a feature pyramid for direct offset prediction on a single vector. Additionally, boundary-based methods rescale the entire length of the video using linear interpolation to handle long videos, and evaluate actions for all possible combinations of temporal spans. These techniques provide ways to handle untrimmed video and model action segments of varying lengths to detect actions.

In this section, an overview is provided of how the techniques mentioned above are utilised in practice for action detection in untrimmed videos. Additionally, some additional topics will be reviewed that were not covered in the previous sections, such as how to utilise context information outside of the action, strategies for implementing TAL, learning supervision, evaluation metrics used to compare performance, and widely used datasets.

2.4.1 Temporal Proposal Generation

Initially, anchor-based methods were the primary focus in action detection, thanks to the success of using anchor boxes in object recognition. In one study [60], action detection based on a sliding window was performed using multiple anchors. To compensate for the inaccuracies of boundary detection based on the stride of sliding windows and the configuration of anchors, boundary regression was performed in addition to binary classification. In contrast, another study [55] utilised grouping-based proposal generation using actionness scores. After generating initial proposals based on the scores and creating a proposal-level representation through structured temporal pyramid pooling, two types of classifiers were employed: one for predicting the class of each proposal and another for determining the completeness of each class. Furthermore, the class-dependent boundary regression module was employed to obtain more precise results. The above-mentioned methods utilise top-down and bottom-up approaches, respectively. For fine-grained detection, it would be beneficial to perform bottom-up detection using frame-level or snippet-level representations. However, this representation method contains only a small amount of information, leading to unstable detection results. Conversely, top-down methods such as anchor-based methods can obtain more stable results by performing detection in units of temporal segments, but this approach has the disadvantage of inaccurate boundary detection. In another study [58], researchers attempted to compensate for the limitations of each approach by leveraging their respective strengths. First, two types of proposal generation were performed using both anchor-based and grouping-based methods. As these two types of results can detect actions that may be missed in each method, both types of results were merged and used as input for subsequent steps, such as action recognition and boundary refinement.

The methods mentioned above treat all data equally when creating a proposal-level rep-

resentation. However, in recognising an action, there may be more or less important parts of the data. In one study [62], an attention mechanism was applied to the anchor-based method. Temporal attention was applied to suppress insignificant parts within the action sequence, thereby improving classification performance. In another study [84], a different approach was taken to the bottom-up method. The researchers believed that the feature vectors constituting one action would be related to each other and attempted to learn this relationship. Pyramid pooling was utilised to address multi-scale issues, and channel-wise and temporal-wise non-local operation blocks were proposed to discover the relationship using the attention mechanism.

In general, anchor-based methods evaluate whether a corresponding region has a high probability of containing an action in units of temporal segments. Boundary-based methods, on the other hand, additionally calculate the probability of a corresponding region being a boundary of the action. For instance, BSN [67] predicted the boundary score and used it to generate proposals. The boundary scores were then multiplied by the actionness score during proposal ranking to increase reliability. This method was further improved upon in BMN [68]. In BSN, action proposal generation was divided into several stages and was not consolidated into a single model. To address this limitation, a score map was created in BMN, which allowed for the calculation of the actionness score before setting proposal pairs. This allowed the entire process to be carried out in a single unified model. In another study [77], while the previous two studies concatenated the two types of data (RGB, Flow) and treated them as a single data stream, better results were achieved by handling each data type independently and performing proposal generation through late fusion. In yet another study [70], to address the issue of unstable boundary scores, the scores were supplemented by utilising boundary scores detected in the reverse direction. The utilisation of boundary scores significantly contributed to improving the performance. However, these approaches have the disadvantage of requiring a large amount of memory due to the output in the form of a map.

The methods mentioned above rely on the data surrounding the boundary to detect the context before and after the action, which can capture the local context. However, this approach may be insufficient for detecting the boundary due to the ambiguity of the action boundary. To supplement this, global context was utilised. The success of transformer architecture in the field of natural language processing has made it possible to efficiently apply attention mechanisms. This is useful for creating global context feature vectors since it can compute attention weights for the entire input sequence. In [66], various techniques were combined to comprehensively address existing limitations. The researchers used a transformer architecture for efficient context aggregation and predicted offsets directly, without utilising anchors. Additionally, in order to eliminate the need for post-processing, duplicate results were not produced by using Hungarian matching. While this method greatly improved performance, it still exhibited poorer performance compared to object detection, indicating that many potential issues still exist in TAL.

2.4.2 TAL Strategy

In TAL, there are two main approaches to achieving the ultimate goal of detecting and recognising actions in untrimmed video: the pipeline method and the end-to-end method.

The pipeline method, also known as the multi-stage method, solves TAL by dividing it into several stages. The most representative of these is a two-stage approach that proceeds by dividing it into action detection and action recognition. Studies that deal with Temporal Action Proposal Generation (TAPG) in the literature can be said to belong to this approach, as TAPG is the first step of the two-stage approach. This step does not use class labels because it only deals with action detection in a class-independent way, detecting actions by distinguishing between background and foreground. The detected action proposals are used as input for the next step, action recognition, and are refined more precisely while assigning action labels. For the second step, any algorithm can be used. For example, in UntrimmedNet [95], each detected action was treated as a single video clip and classified using a video-level classifier. In p-SCNN classifier [96], direct recognition of action proposals is performed using a proposal-level classifier. By separating the first detection step from the recognition step, the pipeline method becomes more versatile. The results of proposal generation can be used as input for an action recognition algorithm or other algorithms for other purposes, increasing the reusability of existing recognition algorithms. Additionally, since the pipeline method is divided into several steps, the overall performance can be increased by replacing a specific step, or the problem can be divided into smaller units using a divide-and-conquer strategy. However, there is a disadvantage to this method, which is that each step must be separately prepared and tested, which can be inconvenient.

The end-to-end approach, also known as the one-stage method, solves the goal of TAL, action detection and action recognition, with one learning algorithm. With this approach, the action detection and recognition result can be obtained using one learning model without the need for a separate additional process, making the preparation process simpler and the final result more immediate. Due to these advantages, recent research trends have increasingly focused on end-to-end approaches. In the past, the performance of end-to-end approaches was worse than the pipeline approach, but recent studies have shown more advanced results due to various efforts. As a result, attempts to achieve better performance with simple methods are continuously being made.

2.4.3 Supervision Learning

Ground-truth labels are provided for datasets used for training. Depending on the type and number of labels provided, there are four types of learning: supervised learning, semi-supervised learning, weakly-supervised learning, and unsupervised learning. From the perspective of TAL, the labels included in the dataset are temporal annotations indicating the start and end points of actions, and class labels indicating what the action is.

(Fully) supervised learning is applicable when both temporal annotations and class labels

are provided for all available data. All the studies mentioned in this chapter fall under this category. However, annotating untrimmed videos can be a tedious and time-consuming task, leading to increased attention towards other learning supervision methods. Semi-supervised learning is a scenario where temporal annotations and class labels are provided only for a part of the dataset [97]. With the recent increase in the amount of available datasets and videos on the internet, this method has been developed to further utilise unlabelled data based on previously labelled datasets. This method relies on accurately assigning pseudo-labels to unlabelled data based on existing labels, and then performing the learning process to further advance the model like in fully supervised learning. Weakly-supervised learning is a technique used when action labels are available in videos, but corresponding temporal annotations are unknown [98]. In this approach, similar segments are identified by calculating similarity between videos, separated into actions of the same category, and appropriate labels are assigned to the corresponding segments. Unsupervised learning is used when no annotation information is provided [99]. In such cases, it is impossible to recognise a specific action as there is no action label information available. Instead, actions in the same category are detected by using similarity matching between videos.

Depending on the amount and type of labels provided, the supervision used for learning changes, and the fewer labels of ground-truths are available the more challenging it is. Alternatively, there is also a method called self-supervised learning. This method may look similar to semi-supervised learning. The difference is that self-supervised learning uses labelled data to create additional data and assigns the original data labels to this data to enrich the dataset.

2.4.4 Evaluation Measures

In this section, the metrics that are used to evaluate the performance of information retrieval are explained. First, the basic concepts are introduced, and the individual evaluation metrics that are frequently used in TAL are considered.

- **Basic Concepts**

  For the convenience of explanation, a binary classification example will be considered. In a classification problem, the cases can be divided into the following confusion matrix (Table 2.2).

Table 2.2. Confusion matrix of four outputs in binary classification.

		Actual	
		True	**False**
Predicted	**True**	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

  TP represents a true positive that predicts a value that is actually true as true, while FP represents a false positive that predicts a value that is actually false as true. FN is a false negative that predicts an actual true value as false, and TN is a true negative that predicts an actual false value as false. In other words, TP and TN are correct predictions, while

FP and FN are incorrect predictions. These four parameters are used to evaluate the performance of different models in solving classification problems.

Recall

Recall, also known as sensitivity or hit rate, refers to the probability of correctly classifying the ground truth, which is the proportion of actual true values predicted out of all true values. The formula can be represented as follows:

$$Recall = \frac{TP}{TP + FN} \tag{2.1}$$

Recall has a value between 0 and 1, and a high recall score indicates that most ground truths are predicted correctly. However, a high recall score does not necessarily indicate the most ground truths are predicted correctly since it does not deal with TNs. For example, if a dataset with a total of 200 samples has only 10 true samples, and the classifier predicts 100 samples as true, including these 10 samples, it cannot be considered a classifier with good performance. This is because the remaining 90 samples are incorrect, despite the recall score being 1.

Precision

Precision, also referred to as the positive predictive value, represents the proportion of true positive predictions out of all predicted positive values. It measures the accuracy of positive predictions and is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Precision, like recall, has a value between 0 and 1, and a high precision score means that most of the results predicted to be true actually have a true value. This metric does not detect misses. For example, if there were 50 predictions that were predicted to be true in a dataset containing a total of 45 true samples, but only 40 of them were actually true, it would have a high precision score of 0.8, but it would not indicate that five true samples were missed.

Accuracy

Accuracy is the percentage of all samples that are predicted correctly, regardless of whether they are true or false. While recall and precision focus only on correctly predicting true samples, accuracy also considers the correct prediction of false samples. The formula for accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.3}$$

The accuracy metric is commonly used to evaluate the performance of a classifier In a straightforward manner. However, it can be influenced by class imbalance in the dataset, where the dominant class can dominate the evaluation. Hence, a metric that can compensate for class imbalance is required.

F1-score

The F1-score is a metric that can deal with the class imbalance problem and make up for the disadvantages of precision and recall mentioned above. It is a harmonic average of precision and recall. The formula for F1-score is:

$$F1\_score = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

Intersection over Union (IOU)

IoU, also known as the Jaccard Index, is a metric that measures the overlap between the predicted bounding box and the ground-truth bounding box. It is widely used in object detection tasks, including TAL. The IoU score represents the accuracy of detection, where a higher value indicates a better match between the predicted and ground-truth bounding boxes. The IoU can be calculated using the following formula:

$$Intersection\_over\_union(IOU) = \frac{predicted\_box \cap ground\_truth}{predicted\_box \cup ground\_truth} \tag{2.5}$$

In the detection task, if the IoU value is greater than a certain threshold value, it is considered to be correctly detected. TAL utilises tIoU, which is a version of IoU that is applied temporally.

- **Evaluation Metrics for TAL**

  In literature, TAPG and TAL are evaluated using different metrics. Average Recall (AR) and Area Under the Curve (AUC) are used for TAPG while mean Average Precision (mAP) is used for TAL. These metrics will be explained.

  Average Recall (AR)

  This metric is used to evaluate the performance of TAPG and represents the recall score when N_p proposals are given. The networks for action proposal generation generate numerous proposals. Even if duplicates are removed using techniques such as NMS, a considerable number of proposals still remain. By limiting the number of proposals, it is possible to evaluate whether accurately detected proposals receive a high-ranking score. This metric is denoted as AR@AN (average recall given the average number of proposals). For instance, when the performance of two models is expressed as AR@10 and AR@1000, it means that the model with a high recall score in AR@10 assigns a high score to the correct proposal. On the other hand, a model with a high AR@1000 indicates that it can detect more ground truth when a large number of proposals are given.

$$AN = \frac{\text{total number of proposals}}{\text{total number of videos in the testing subset}} \tag{2.6}$$

$$AR_{N_p} = \frac{\text{sum of recall per video given } N_p \text{ proposals}}{\text{total number of videos}} \tag{2.7}$$

  Area Under the Curve (AUC)

Among the indicators used to evaluate the performance of a classification model, the plot measuring the performance of the model at various threshold values is called the AUC. The receiver operating characteristic (ROC) is a graph that shows the performance of a classification model at all threshold values, and the AUC refers to the area under the ROC curve. In the context of TAPG, the x-axis of the ROC curve represents the average number of proposals per video (AN), while the y-axis represents the AR. The ROC curve can draw one curve according to a specific threshold and calculate the corresponding AUC. Instead of calculating AUC for a specific IoU value, AUC for a range of IoUs is calculated. In general, for the detection task, it is judged that a result has been correctly detected if the IoU value is 0.5 or greater. Therefore, AUC is calculated using an IoU threshold in the range [0.5, 1] and used to evaluate the overall performance of the model.

mean Average Precision (mAP)

The mean average precision (mAP) is the average of the Average Precision (AP) scores for each class, and it is the most commonly used metric for evaluating the performance of TAL models. Specifically, when a single video is given, the precision of a particular class can be calculated as follows.

$$P = \frac{TP}{TP + FP} = \frac{\text{number of correctly predicted proposals}}{\text{total number of predicted proposals}} \tag{2.8}$$

However, since one video may include several action classes, precision can be calculated for each class. Using these precision values, the AP can be calculated as follows:

$$AP = \frac{\text{sum of precision of all classes}}{\text{total number of classes}} \tag{2.9}$$

The final performance evaluation of TAL in a dataset, which typically contains a large number of videos, is measured and compared using mAP.

$$mAP = \frac{\text{sum of average precision of all classes}}{\text{total number of videos in testing set}} \tag{2.10}$$

In general, when tIoU is 0.5, mAP is used for performance comparison.

2.4.5 Benchmark Datasets

Although there is no official benchmark for TAL, many researchers use THUMOS14 and ActivityNet 1.3 to report algorithm performance. THUMOS14 is a dataset that was used in the THUMOS Challenge 14, held at the European Conference on Computer Visoin (ECCV) 14, and has since been used for benchmarking with the evaluation setting still being adopted. Similarly, the ActivityNet dataset was used for the international challenge on activity recognition held together with the Computer Vision and Pattern Recognition (CVPR) conference. After releasing v1.2 in 2015, it was expanded and v1.3 was released in 2016. Since then, it has been widely used as a benchmark along with THUMOS14. In this thesis, ActivityNet

v1.3 and THUMOS14 are mainly used as benchmark datasets. There are also other large-scale datasets such as MultiTHUMOS, Charades, and HACS. These are all introduced in this section.

THUMOS14[11]

The THUMOS14 dataset was created for the purpose of action recognition and temporal action detection. It includes 254 hours of video data and 25 million frames, and is divided into a training set, a validation set, and a test set. The training set uses the UCF101 [100] dataset, which consists of 13,320 temporally trimmed videos for 101 action classes. The validation and test sets consist of 1,010 and 1,574 temporally untrimmed videos, respectively. The THUMOS14 dataset provides video-level annotation for action recognition on its sets, and temporal annotation for only 20 sport-related classes for temporal detection. The validation set and test set each contain 200 and 213 videos respectively, corresponding to these 20 classes. Since the training set lacks temporal annotation and untrimmed videos, researchers use the validation set for training models to leverage untrimmed videos and evaluate performance on the test set.

ActivityNet v1.3[2]

ActivityNet is the second largest dataset for temporal action detection. It was created to target 100 action classes in 2015 and was later expanded to 200 classes in 2016. The dataset comprises 19,994 untrimmed videos with a total duration of 849 hours. It consists of 10,024 training videos, 4,926 validation videos, and 5,044 testing videos. It provides temporal annotations for a total of 23,064 instances, with each video having an average of 1.41 action instances. However, ground-truth annotation is not provided for the test set. Hence, many researchers train the algorithm on the training set and report performance using the validation set in the testing phase.

HACS[101]

The HACS dataset is composed of two parts: HACS Clips for action recognition and HACS Segments for temporal action detection. HACS Clips contains 1.5 million video clips, while HACS Segments provides 139,000 temporal annotations on 50,000 untrimmed videos. Like ActivityNet, the dataset targets 200 action classes.

MultiTHUMOS[102]

The MultiTHUMOS dataset was created by extending the THUMOS14 dataset to include multi-label classification. It consists of 400 videos with a total length of 30 hours and includes 38,690 temporal annotations for 65 action classes. In contrast to the previous dataset, each action instance has an overlap, with an average of 1.5 labels per frame, and an average of 10.5 action classes per video.

Charades[103]

The Charades dataset is a large-scale dataset that targets common indoor activities. It comprises a total of 9,848 untrimmed videos and 157 actions that have been collected from

267 people. Each video is about 30 seconds long, providing 66,500 temporal annotations with each instance being 12.8 seconds long on average. Additionally, the dataset provides 41,104 labels for 46 objects and 27,847 descriptions for each video.

2.5 Summary

In this chapter, the task of TAL was reviewed. TAL is a crucial component of video understanding. Unlike previous survey papers that classify algorithms based on their characteristics, in this literature review, the adopted approach was to examine the different components required for research in this field and the corresponding techniques. The field of TAL consists of multiple tasks, including regression to accurately find the boundaries of an action and classification to recognise the action itself. The process of action detection begins with converting the video into a vectorised feature representation. To achieve this, action recognition models are utilised as a backbone for representing video, and researchers are working towards creating more suitable backbones by modifying or fine-tuning pre-trained networks specifically for action detection.

TAL involves detecting actions of various lengths efficiently in long videos. To achieve this, techniques such as sliding window or video rescaling are used to search long videos efficiently, and anchors, which are predefined temporal segments of a specific length, are used to evaluate actions of various lengths. Alternatively, actions can be detected through grouping, or through the direct prediction of left and right offsets in an anchor-free manner using a pyramid structure with multi-scale features. The boundary-based method predicts the probability of being a boundary to detect action boundaries, and action detection problems of various lengths is addressed by learning the relationship between feature vectors using a graph structure.

In addition to detecting action boundaries, TAL also requires the ability to perform other tasks such as classification and regression. To achieve this, a method of representing proposals as action candidates is needed. Typically, a proposal-level vector is created by applying a convolution or pooling operation to the feature vectors between the starting and ending points of the proposal. To improve efficiency, sparse sampling is used to reduce the number of feature vectors, and structured pooling is used to preserve the temporal ordering of the features. Additionally, a weighted pooling approach can be used to suppress unimportant information through an attention mechanism.

Due to the ambiguity of the action boundary, boundary refinement is optionally performed for more accurate detection. This problem is addressed in the regression branch of the action detection model, or additional refinement is performed using the initial proposals. After completing action proposal generation, duplicate detection results are removed through non-maximum suppression. Although attempts have been made to eliminate this post-processing process, it still relies on NMS.

In the literature, there are many benchmark datasets available for TAL. However, even al-

gorithms that currently report excellent performance use different hyperparameters depending on the dataset's characteristics and traverse the video in different ways. Furthermore, compared to the performance of other similar fields, such as object detection, the performance of TAL is not yet sufficient. This implies that there are still many potential problems in the field of TAL.

Chapter 3

Investigation on Challenges in Pipeline TAL – Comparative Study

New research papers are consistently published every year, and they report their performance on benchmarking datasets to demonstrate the effectiveness of their work. They achieve better results by either enhancing existing algorithms or proposing new methods that are more suitable for action detection. However, these papers often claim their superiority by comparing the AR value with the state-of-the-art. As a result, it can be challenging to identify potential limitations that may impede further advancements, despite the proposed methods improving overall performance. In this chapter, factors that can positively or negatively affect overall performance will be examined, rather than discussing concepts or ideas that can contribute to performance improvement.

There can be many factors that affect the performance of machine learning algorithms. These can include the distribution of data within the dataset, as well as the specific methods used to solve a problem. If a dataset is composed of samples that accurately represent the task domain, even a small dataset can yield excellent results. Conversely, a dataset with biased characteristics, even if it contains a large number of data, may yield low performance when training a model. To identify potential factors that affect performance, this chapter explores five scenarios through experiments. Although these factors are not necessarily fatal problems, they should be considered in future research. The five investigations are as follows:

1. The relationship between detection and recognition performance
2. The effect of using multi-class labels on learning
3. The robustness to discontinuous actions
4. The relationship between multi-modal data and performance
5. Cross-corpora testing

3.1 The Relationship Between Detection and Recognition Performance

Studies in object detection have adopted a detection-then-recognition scheme, which first detects an object and then recognises the detected object. This method has consistently

demonstrated excellent performance over an extended period of time. Consequently, this scheme has also been applied to multi-stage methods in TAL. In particular, the two-stage method can be executed using the same procedure as object detection, making the detection-then-recognition scheme a suitable approach for performing pipeline methods in TAL. In TAL, proposal generation (corresponding to the detection phase in object detection) aims to find the boundaries of actions in untrimmed videos, while recognition classifies the detected actions into specific classes. Algorithms in multi-stage methods utilise the results of previous steps as inputs for subsequent steps. As a result, it is reasonable to expect that the outcome of action recognition will be influenced by the performance of proposal generation. In the first step of the two-stage method, TAPG, significant efforts have been made to improve detection performance with the expectation that advancements in this area will lead to overall improvements in TAL. This is because action recognition has made more progress than TAPG, the results of which can be used as input for action recognition in TAL. The input data for action recognition consists of trimmed video clips containing a single action. This data does not include background information unrelated to the action, and almost all the clips start at the beginning of the action and end at the final boundary, revealing that actions are arranged in units of video clips. In contrast, in TAL, the recognition process commences after an action has been detected in untrimmed video. Therefore, the outcome of action detection can be viewed as a step to create trimmed video clips used in action recognition. As the results in the detection step become more accurate, the detected actions will yield well-aligned action segments, similar to trimmed video clips. This, in turn, can lead to improved performance of the entire TAL process by enhancing the results of action recognition.

In this sections, the performance between TAPG and TAL will be compared to verify TAL is affected by TAPG. In addition, to find out the relationship between the performance of TAPG and that of TAL, the result of the experiments will be further investigated.

3.1.1 Research Question

It is easy to assume that the accuracy of action detection in the two-stage method can lead to improved action recognition results. If so, the enhancement in action detection performance would consistently result in better action recognition. Among several TAPG algorithms, can it be said that the algorithm with the best performance consistently exhibits superior recognition performance? Based on this, the following questions can be asked:

- For 2-stage TAL, TAPG has been improved for better detection performance, and the overall TAL performance is measured by the output of the subsequent recognition.
  - When the performance of proposal generation is improved, will the performance of recognition be also improved?
  - Is there any relationship between the performance of proposal generation and that of recognition?

The above question aims to determine how the results of action detection impact the outcomes of action recognition. Due to the ambiguity of action boundaries, accurately detecting actions is challenging. Consequently, it is also quite difficult to create data, such as trimmed video clips, through proposal generation. In this context, it is believed that the influence of the proposal generation results on the overall TAL outcomes will vary according to the performance of the algorithms. Some proposal generation algorithms may exhibit high overall performance (AR), but they might miss target instances under stricter criteria. Due to these characteristics, it is meaningful to explore the relationship between the results of action detection and action recognition. In this section, the aim is to uncover this relationship by examining how action recognition outcomes differ when using proposal generation results with varying characteristics, produced by several existing temporal action generation algorithms.

3.1.2 Experimental Settings

To investigate the relationship between proposal generation and the subsequent task (recognition), different sets of proposals are required to display various aspects, as well as action recognition algorithms. To prepare these sets of proposals, four existing TAPG algorithms were employed: BSN [67], BMN [68], DBG [77], and BSN++ [69]. All four of these algorithms belong to the boundary-based method and exhibit differing performances. Boundary-based models have reported better performance than other models, so among them, four models having similar TAL performance were chosen. For recognition, the proposal-level classifier p-GCN [104] was used. This accepts proposal-level information as input, so it is suitable for the pipeline TAL task. Once initial proposals are provided, this recognition algorithm generates and recognises proposal-level feature vectors from feature sequences.

In this experiment, action proposals obtained from the four TAPG algorithms mentioned earlier were used as initial proposals for recognition. The investigation was to how recognition results vary based on different initial proposals. To obtain more diverse results from each TAPG algorithm, two feature extraction algorithms were employed: the TSN [105] and I3D [46]. All currently published TAPG algorithms produce varying results depending on the feature extraction method used. This is because the information contained in features differs according to the chosen feature extraction methods. In published papers, C3D[40] and the 2-stream network are widely used for feature extraction. However, the feature extraction method currently favoured by many researchers is the 2-stream network (e.g. TSN, I3D). Thus, for a more general comparison, feature vectors generated by the 2-stream network were used. As different results can be obtained depending on the type of feature, more diverse initial proposals can be acquired by utilising feature vectors generated by various feature extraction methods.

The experiment was conducted on the THUMOS14 dataset. This dataset provides a training set, validation set, and test set; however, the training set does not include temporal annotation, as it consists of trimmed video clips. Following a convention in the literature, the TAPG model was trained on the validation set and evaluated on the test set. Feature vectors was ex-

tracted every 5 frames. To use different types of input vectors, the TAPG model was trained on the two types of feature vectors (TSN and I3D). Since the goal was to investigate how recognition results vary according to the proposal generation results, the parameters were not optimised for training the TAPG model and instead used the parameters mentioned in each paper. A sliding window technique was employed to retrieve actions in the THUMOS14 dataset. During this process, the window size (l_w) was set to 128, and the overlap ratio is 0.5. This means that the window has a stride value of 64 and detects the action while moving across the untrimmed video. This configuration covers the length of 98% of action instances. all models were trained from scratch using the Adam optimiser, with a batch size of 16. The learning rate was set to 0.001 for the first seven epochs of BSN++ and the first ten epochs of the others. It was then decayed to 0.0001 for another ten epochs (BSN, BMN), two epochs (DBG), and three epochs (BSN++). Soft NMS was performed to address redundant results. For action recognition, the P-GCN classifier is used. This is a proposal-level classifier that can accept proposals (the starts and ends of boundaries) as input. This classifier has its own configuration, distinct from that of proposal generation. The configuration mentioned in its original paper is used without changes. Each input video is uniformly divided into 64-frame segments, and features are extracted using a two-stream inflated 3D ConvNet (I3D) model pre-trained on the Kinetics [18] dataset. This model has different learning rates for the two streams: 0.001 for the RGB stream and 0.01 for the Flow stream. This learning rate is divided by 10 every 15 epochs. In the NMS stage, the total number of proposals was controlled such that it does not to exceed 1500 per video.

3.1.3 Experiment Results and Analysis

The results of the proposal generation used as input are shown in Table 3.1. This table displays eight different sets of proposals generated using two feature types (TSN, I3D) and four types of TAPG algorithms (BSN [67], BMN[68], DBG[77], BSN++[69]). The results of the eight proposals are presented in the form of AR, given 50, 100, 200, 500, and 1000 proposals. As evident in this table, the eight sets of proposals exhibit different characteristics. The most notable observation is that even when the same algorithm is employed, the performance varies depending on the feature used.

The results of the action recognition using P-GCN with the initial proposals having eight different properties are shown in Table 3.2. When training P-GCN, the I3D feature vectors are generated using a different configuration from that of proposal generation, so the recognition results are affected only by the initial proposals. Much work on two-stage TAL has focused on improving the performance of proposal generation, assuming that better accuracy in proposal generation leads to better performance In action recognition. However, the results in Table 3.2 indicate that this assumption may not hold true for action recognition. Figure 3.1 is a plot designed to facilitate the comparison of proposal generation results and recognition results. Since the total number of generated proposals from BSN is different from the others, the results were compared excluding BSN to maintain equal conditions.

Table 3.1. Results of temporal action proposal generation: Four different TAPG algorithms were evaluated on the THUMOS14 dataset, and the average recall at different numbers of proposals (AR@AN) was calculated for each method. The resulting proposal sets were then used to train the P-GCN classifier. (Unit: %)

	Feature type	AR@50	AR@100	AR@200	AR@500	AR@1000	AR@1500	AR@AN
BSN	TSN	37.17	46.12	53.57	60.49	64.39	-	-
	I3D	32.33	40.18	47.63	57.17	61.77	-	-
BMN	TSN	37.33	46.01	53.27	61.28	**66.36**	**69.09**	**61.46**
	I3D	37.23	45.43	52.69	60.30	65.35	68.10	60.54
DBG	TSN	36.24	45.29	52.43	59.83	63.77	65.61	59.45
	I3D	**38.26**	**47.20**	**54.50**	**61.76**	65.44	67.14	61.21
BSN++	TSN	37.39	44.71	51.22	58.67	63.16	65.63	58.81
	I3D	33.40	40.63	46.56	53.87	58.87	61.38	54.36

Table 3.2. Results of action recognition with P-GCN on the THUMOS14 dataset. The performance was reported as average precision (AP). (Unit: %)

	Feature type	tiou=0.1	tiou=0.2	tiou=0.3	tiou=0.4	tiou=0.5	tiou=0.6	tiou=0.7	tiou=0.8	tiou=0.9	mAP
BSN	TSN	57.36	55.23	**52.14**	**46.23**	37.04	25.48	13.8	4.92	0.81	32.56
	I3D	56.62	53.87	50.94	44.18	35.44	24.71	14.24	4.71	0.7	31.71
BMN	TSN	51.73	47.33	41.11	33.58	24.21	14.66	6.95	2.16	0.29	24.67
	I3D	54.65	53.09	49.41	44.3	36.65	25.74	15.13	6.07	0.91	31.77
DBG	TSN	57.61	55.47	51.12	45.43	36.57	25.63	13.96	5.9	1.11	32.53
	I3D	**57.62**	**55.48**	51.13	45.39	36.56	25.56	13.98	5.96	1.1	32.53
BSN++	TSN	50.95	49.64	47.23	42.6	34.32	23.15	12.36	3.91	0.43	29.40
	I3D	55.59	54.04	50.38	45.30	**37.28**	26.65	**15.33**	6.14	**0.92**	32.40

According to Figure 3.1, AR@AN is the highest for BMN using TSN and the lowest for BSN++ using I3D. However, the recognition results were the lowest in the average mAP for BMN with TSN, while the recognition performance of BSN++ with I3D is relatively excellent. To analyse these phenomena, the distribution of positive samples included in the initial proposals was examined. Table 3.3 displays the number of positive samples included in each set of initial proposals. The performance of BSN++ is noteworthy. In Table 3.1, AR@AN for BSN++ is lower than those of BMN and DBG. However, in Table 3.2, the recognition results show a similar level to those of BMN and DBG. As demonstrated in Table 3.3, even if the TAPG performance is relatively low, the recognition performance can be improved if the number of positive samples accounts for a large proportion of the proposal set. When evaluating proposal generation, all generated proposals are sorted according to their confidence scores, and then the AR is calculated. However, when using initial proposals for the recognition task, the TAPG scores were not taken into consideration, which may undermine the assumption that better proposals lead to better recognition results. In other words, high-quality proposals might have low confidence scores. This can also be verified through the results of BSN. The proposal set created using TSN is found to contain more positive samples than the one generated using I3D. Consequently, the corresponding result of action recognition with I3D is also better than that using the proposal set with TSN.

This experiment demonstrated that the performance of the final TAL when using the two-stage method is influenced not only by the performance of proposal generation but also by the number of positive samples included in the proposal set.

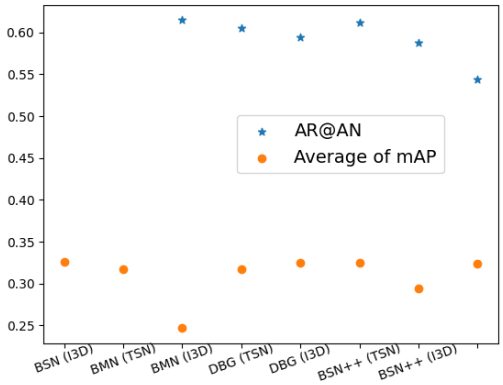


Figure 3.1. Average Recall (AR) and mean Average Precision (mAP) Results of proposal generation and recognition using THUMOS14.

3.1.4 Summary and Discussion

In this section, how TAPG results affect the recognition of the subsequent task in the two-stage (Pipeline) method was examined. To answer the first question, “When the performance of proposal generation improves, will the performance of recognition also improve?”, experiments were conducted using four types of TAPG algorithms and two types of features, producing proposal sets with eight different average recalls. The experiment was performed using the P-GCN classifier. In general, for pipeline methods, it was suggested that better re-

Table 3.3. Number of positive samples in six kinds of initial proposals.

	Feature type	tiou=0.1	tiou=0.2	tiou=0.3	tiou=0.4	tiou=0.5	tiou=0.6	tiou=0.7	tiou=0.8	tiou=0.9	tiou=1.0
BSN	2-stream	47436	50361	39168	28568	20258	13731	9894	6047	3970	2043
	i3d	43875	47724	36368	26478	18061	11916	8534	5120	3066	1364
BMN	2-stream	60937	64198	49938	37578	29673	22636	18288	13705	10005	4665
	i3d	63076	64161	49152	37086	28227	20997	16703	12280	8838	3987
DBG	2-stream	103774	76047	44168	28942	19598	13498	9477	5904	4044	1754
	i3d	91380	75511	46753	31157	21494	15091	10962	6845	4619	2148
BSN++	2-stream	111302	135806	87589	52622	41116	28676	22135	15982	12314	5360
	i3d	127115	158303	97309	55435	37590	27263	20414	14427	10826	5046

sults in the previous stage lead to better results in the next stage. According to the results, it holds true in general. However, in the two-stage TAL, it cannot be guaranteed that the action recognition results are directly proportional to the TAPG results. TAPG is evaluated after sorting the generated proposals, but there is a possibility that high-quality proposals have low confidence scores. When training the classifier for the subsequent task, the scores of initial proposals are not taken into account. As the answer of the second question, "Is there any relationship between the performance of proposal generation and that of recognition?", the experiment in this section demonstrated that the more positive samples are in the proposal set, the better the performance of the whole TAL (proposal generation + classification) when the same number of proposals are provided.

In these experiments, it can be observed that high-quality proposals may have low confidence scores. This implies that there are still undiscovered features and information hidden in action detection, and there is much to be studied and improved in the future.

3.2 The Effect of Using Multi-Class Labels on Learning

In TAL, the labels used in the one-stage method and the multi-stage method contain different levels of information. Since the one-stage method has the final goal of detecting an action from untrimmed video and classifying it as a specific action, the ground-truth label includes the timestamps of the start and end of the action, as well as the action name indicating the class of the action. On the other hand, the labels in multi-stage methods include only target information to be obtained at each stage. In particular, in the two-stage method, since the goal in the first stage is to classify the foreground action from the background, the ground-truth label set includes the timestamps of the start and end of the actions, but does not include the specific action class names. Instead, the model is trained using a label indicating whether each temporal segment is foreground or background. Until about three years ago, two-stage methods were dominant in the TAL field, and TAPG was included as one of the challenges in the ActivityNet challenge held at CVPR 2019. However, since 2020, the TAPG task has been removed, and only the TAL task has remained, requiring researchers who plan to participate in the competition to submit action timestamps and class labels simultaneously. Influenced by this change, a number of papers have begun to be published using the one-stage method.

It is not that there were no studies adopting the one-stage method before 2019. There have been attempts to solve the detection and recognition of actions simultaneously by proposing an end-to-end model [106, 107]. However, since then, the multi-stage method has been recognised as superior to the one-stage method in terms of performance. Recently, studies adopting one-stage methods have begun to be dominant, and they are reporting similar or superior results to those of multi-stage methods [85, 108].

With the advancement of the one-stage method and the removal of the TAPG task in the ActivityNet challenge, it can be assumed that the one-stage method will gradually dominate the research direction in the TAL field. As long as the one-stage method is superior in terms of

performance, this method will be preferred over the multi-stage method because it is easier to use and the results can be obtained immediately. It has been recognised that solving multiple tasks at once in one model is more difficult than solving each problem separately after dividing the whole task. So, in an effort to deal with the TAL problem by dividing the whole task into easier, independent units, multi-stage methods have been preferred. However, the multi-stage method requires a separate configuration of the experimental environment to solve each task, and it cannot be guaranteed that the improvement of the result in the intermediate stage will improve the result in the final stage. It also makes it more difficult for information used to solve a problem in an earlier stage to be reused in a later stage.

3.2.1 Research Question

The one-stage method presented in the literature is not conceptually very different from the two-stage method. However, when comparing the TAPG – which is the first stage of the two-stage method – with the one-stage method, there is a noticeable difference. The one-stage method performs multi-label classification and directly predicts the corresponding class, while the TAPG performs binary classification to classify foreground and background. Hence, the only difference in the output is the inclusion of specific action labels. Various technical skills, such as multi-scale modelling and proposal-level representation, can be shared for both methods. Furthermore, the pre-trained model for TAPG is leveraged to initialise the parameters of the one-stage method, and the modified one-stage model can be utilised for other video understanding tasks, besides TAL. It would be interesting to examine the impact of action labels on the performance of one-stage methods versus two-stage methods, which share common internal modules. In light of this, the following question can be posed:

- Two-stage methods do not use action labels for action proposal generation, while one-stage methods employ both temporal annotations and action labels.
  - Are there any advantages of using action labels in terms of boundary detection?

The above question arises from the assumption that it is easier to extract features for each action than to find common features for all actions. Different actions have unique characteristics, and if treated as one class, various actions may be mistakenly classified as background. However, if considered separately, the same action may be expressed in seemingly different forms depending on the performer. Nonetheless, certain motions occur more commonly, making classification easier. To verify the effect of action labels on boundary detection, an experiment using an existing one-stage method was conducted.

3.2.2 Experimental Settings

The primary focus of the work presented in this section was to examine the impact of using action labels on boundary detection. To achieve this objective, two experimental environments were required. The first involved completing the TAL task using an end-to-end model,

such as a general one-stage method. The second involved generating temporal action proposals using a label that indicates whether it is foreground or background, without specific action labels. All other conditions remained constant. The experiments were performed using the THUMOS14 dataset.

For the case of using labels, the study employed existing one-stage methods, namely ActionFormer [109] and AFSD [85]. AFSD is a model that was introduced in 2019 and demonstrates comparable performance to multi-stage methods in an anchor-free manner. Meanwhile, ActionFormer, which is based on transformer architecture, demonstrates superior performance to multi-stage methods and is considered a more advanced form. The investigation focuses solely on the effect of using labels on boundary detection, so Average Precision (AP), which is used to evaluate the entire TAL task, was not used. To evaluate detection performance exclusively, the study employs AR instead of AP. To compare boundary detection performance exclusively, the classification results of each class were disregarded, and the output from using multi-label classification was treated the same as that of TAPG. In the absence of labels, the two algorithms mentioned above were adjusted. Rather than multi-label classification, binary classification was used solely for separating foreground from background.

3.2.3 Experiment Results and Analysis

Table 3.4. Results of temporal action proposal generation either with labels or without labels. Binary classification without using specific class labels yields better outcomes in terms of boundary detection.

	AFSD[85]		ActionFormer[109]	
	with labels	without labels	with labels	without labels
AR@5	14.25	14.97	15.49	**15.71**
AR@10	26.98	**27.79**	26.75	26.70
AR@50	53.02	54.32	55.68	**56.31**
AR@100	57.74	58.49	61.86	**62.72**

Table 3.4 presents the results obtained using AFSD and ActionFormer. Recall is calculated at 5, 10, 50, and 100 proposals. The first two columns of the table display the results from AFSD's algorithm, while the last two columns show the results from ActionFormer. The findings reveal that binary classification without class labels yields better outcomes. Upon closer inspection, marginally better results are obtained consistently when class labels are not used, likely due to the added complexity of the problem when action labels are incorporated. With action labels, the network model is tasked with solving both boundary detection and action classification problems simultaneously, which is more complex than binary classification without class labels.

3.2.4 Summary and Discussion

This section has presented an investigation of the impact of action class labels on performance. The one-stage method, which is an end-to-end method, handles both boundary detection and action classification of action classes in a single model, while the multi-stage

method, which is a pipeline method, divides complex problems into simpler problems and solves them step by step. Due to this difference, the one-stage method classifies actions through multi-label classification, while the multi-stage method (particularly the two-stage method) separates foreground actions from background scenes and assigns one of the action labels to detected action segments. Based on this difference, the study raises a question regarding the effect of using class labels on detection performance.

Intuitively, using class labels provides an advantage in that common characteristics among the same classes can be used to distinguish between different actions. However, the experiments presented in this section indicated that using multi-class labels is not helpful in detecting action boundaries. Since multi-label classification is conducted simultaneously with boundary regression, the complexity of the problem increases. In contrast, when class labels are not employed, multi-label classification is simplified to binary classification, reducing the complexity of the problem.

The experiments presented in this section indicated that considering only boundary detection, results for a simpler problem outperforms those of complicated one, and the complexity of the problem can have a negative impact on detection performance. However, treating all types of actions as one class prevents the utilisation of the unique characteristics of each action. Moreover, employing several stages increases the complexity of the preparation process and the procedures for obtaining the final result. This suggests that there is still room for research on a method that can perform multi-label action classification without adversely affecting the final detection result and an appropriate approach should be selected according to a purpose.

3.3 Robustness to Discontinuous Actions

Due to the proliferation of the internet and the influence of various forms of mass media, countless videos are encountered every year. Scenes in these videos may be enlarged or reduced to create a dramatic effect or enhance viewers' understanding, and multiple scenes can be overlapped. Moreover, as images are captured from multiple angles using multiple cameras, a scene captured by one camera may transition to the viewpoint of another camera. While such directing can be helpful in comprehending an event or context, it can pose a significant obstacle in analysing human behaviour. An action consists of a continuous flow of motions aimed at achieving a specific goal. Since it is impossible for a person to instantly move from one place to another, human behaviour, in reality, does not occur discontinuously. However, in videos, the continuity of actions may be disrupted due to video editing or abrupt camera movements. Furthermore, the vast amount of information has led to the utilisation of large-sized datasets in the field of machine learning. As a result, it is nearly impossible to directly create well-designed datasets. Consequently, video clips from films, TV programmes, and social media are collected to construct these datasets. For this reason, they inherently contain discontinuous action instances.

The continuity of action instances is valuable information for analysing and recognising human behaviour. An individual's next action can be predicted because they often adopt a similar approach when pursuing a specific goal. Although each person may employ a different method to achieve the same goal, what will happen next can still be anticipated based on experiential knowledge. Furthermore, in sports activities like kicking or throwing a ball, it is possible to predict the subsequent action by observing an individual's movement at a specific moment. Due to the human body's structure, its range of motion is limited, and it is believed that the next movement will smoothly follow the previous one, owing to the continuity of action. This information is used to recognise human actions between consecutive images. As a person moves, pixel colour values in the image are updated, and their movement path may be predicted using the motion information of the same colour.

It can be reasonably expected that videos with disrupted action continuity may pose challenges in detecting and recognising actions. Consequently, addressing this issue will be a crucial factor in enhancing performance. This section reports on an investigation directed at identifying which method is more effective among the existing methodologies, specifically comparing the anchor-based method and the anchor-free method.

3.3.1 Research Question

On the internet, there are numerous video clips edited by individuals. In such situations, actions may not be continuous. Sometimes, irrelevant scenes are inserted within a single action. On the other hand, from a methodological perspective, anchor-based (or boundary-based) methods generate proposal-level feature vectors to calculate actionness scores, while anchor-free methods construct a multi-scale feature structure and predict the left and right offsets, allowing them to examine the partial information of actions. These approaches employ different strategies. The following research question was therefore suggested:

- Which approach is more robust to unwanted discontinuous situation?
- If one approach exhibits lower performance than the other, what is the reason behind it?

Since the anchor-based method evaluates temporal segments using all proposal unit information, if there is a discontinuous point within the action, this part is inevitably included in the evaluation. In contrast, anchor-free methods detect actions by predicting offsets while creating multi-scale features. As a result, the action is detected based on the incomplete information of the action. One consideration is that although using incomplete information could be a disadvantage, the likelihood of avoiding discontinuous parts may increase. From this perspective, the investigation was to know whether the different action detection methods, namely the anchor-based method and anchor-free method, are more robust in detecting discontinuous actions.

3.3.2 Experimental Settings

For the experimental set-up, publicly available anchor-based (or boundary-based) methods and an anchor-free method were selected. The difference between anchor-based and boundary-based methods is that anchor-based methods utilise a set of temporal segments with predefined lengths, while boundary-based methods divide the video into fixed intervals and use all possible temporal pairs. In other words, anchor-based methods employ the same set of anchors at all locations, regardless of video length or time location conditions. Meanwhile, boundary-based methods use only the temporal pairs that are actually possible, ensuring that at the beginning and ending of a video, they do not exceed the video's bounds. However, both temporal pairs in anchor-based and boundary-based methods can be regarded as fixed-length temporal segments. Temporal pairs in the boundary-based method can be seen as utilising only the anchors that are actually possible. Therefore, in this experiment, the boundary-based method is used instead of the anchor-based method algorithm, and the two terms will be used interchangeably.

In the literature, only one anchor-free TAPG method has bee found, called RTD [66], so RTD was used as an anchor-free method. BMN and BSN++ were used as boundary-based methods. The experiments were conducted on the THUMOS14 dataset, and discontinuous action segments were manually selected from the ground-truth action labels.

3.3.3 Experiments Results and Analysis

Figure 3.2. Three samples of discontinuous actions in THUMOS14 dataset caused by viewpoint change (first row), occlusion (second row), and pause and insertion of text (third row).

Manually selected discontinuous actions are used to find out which method is more robust against discontinuous actions between the boundary-based method and the anchor-free method. Examples of these actions are shown in Figure 3.2. In this figure, the first row shows discontinuous actions due to a sudden change in viewpoint, and the second row displays cases of occlusion caused by a passer-by. The third row is the case where the action pauses while the action is taking place, and the character is inserted, disappears, and proceeds again. Video

clips with disrupted action continuity will likely exist in datasets collected from the internet, potentially hindering the analysis of continuous actions.

To determine which method is more robust in these discontinuous situations between the boundary-based method and the anchor-free method, the AR for only discontinuous actions is calculated. These results are shown in Table 3.5. In this table, the top two rows are from the boundary-based methods, while the last row is obtained from the anchor-free method. Based on these results, it can be inferred that the anchor-free method is more robust against discontinuous action instances. Several reasons can be considered for this outcome. Firstly, the boundary-based method evaluates whether an action is complete or not by using all information included in the given temporal segments when performing proposal-level evaluation. Consequently, even if discontinuous scenes are embedded within the action, there is no way to avoid such scenes. On the other hand, anchor-free methods predict action segments using a structure similar to a feature pyramid network. Each feature vector present in the feature pyramid is likely to contain only partial information, rather than all information about the action. Due to this, it is possible to avoid discontinuous scenes. Another consideration is whether the impact of discontinuous scenes can be reduced using an attention mechanism. Both BSN++ and RTD are algorithms based on the attention mechanism. Even when comparing the results of these two algorithms, the anchor-free method appears to be superior in this experiment.

Table 3.5. Average recall of discontinuous actions.

	Method	AR@50	AR@100	AR@200	AR@500
BMN	Boundary-based	38.15	44.29	50.09	57.43
BSN++		36.40	45.93	52.68	59.33
RTD	anchor-free	**42.85**	**51.55**	**57.72**	**65.61**

3.3.4 Summary and Discussion

This section has reported on conducted experiments to determine which method is more robust against discontinuous action instances among anchor-based (boundary-based) methods and anchor-free methods, which are approaches widely used for TAL in the literature. According to the experimental results, the anchor-free method appears to be more robust to unwanted discontinuous situations. The reason for this is that the anchor-based (boundary-based) method evaluates actions using all features within the given temporal segments, making it impossible to avoid unexpected discontinuous scenes. On the other hand, the anchor-free method may be more likely to avoid these discontinuous scenes because action instances are predicted using a multi-scale feature representation. However, this characteristic of the anchor-free method can also be a disadvantage. Multi-scale representation is highly likely to lack complete action information. As a result, making predictions using incomplete information may yield unstable results. For this reason, in the literature, anchor-based (boundary-based) methods often outperform anchor-free ones in TAPG.

Three algorithms were used for the experiment. Due to the limited number of algorithms

utilised, the results obtained here may not be representative of all anchor-based (boundary-based) and anchor-free methods. However, the performance of each methodology in unexpected situations may provide insight into future research directions. It is anticipated that more advanced results can be achieved if TAL is performed, taking into account the characteristics of each methodology.

3.4 The Relationship Between Multi-modal Data and Performance

A person can easily determine what action is taking place by watching a video. This is accomplished by combining latent knowledge learned from past experiences and the information seen and heard from videos, while unconsciously processing complex information. Although this complex process can be easily performed by humans, it is not simple to enable automatic execution through algorithms. To reason based on this complex information, different data modalities would be used to analyse and learn human behaviour. The use of different types of data is due to each type containing distinct information. Information that can be directly utilised from videos is obtained from RGB channels. This represents the actions occurring in the video or the background scenes. Objects used in daily life may have a unique colour or a specific pattern. All of this information is expressed through the RGB channels. However, while the person who is the subject of the action cannot be considered to have a unique colour, the colour is arranged in a shape similar to that of the human body, which provides valuable information for analysing human behaviour. However, RGB channels also have the disadvantage of not being able to express changes over time. Although the appearance of the person performing the action and the objects interacting with the person is important, an action is not a static object. Therefore, it is necessary to understand the change over time in how the target to be detected and recognised moves.

An action is a set of consecutive postures of a person. The set of positions and postures of people and objects over time is important information about the progression of the current action. Therefore, motion information of a person has been obtained using equipment such as motion capture or the trajectory of specific parts, such as a person's joints, and used as motion information. In the TAL field using video, optical flow images are used for motion information. Optical flow is an image that expresses the movement of the same pixel between adjacent image frames. Since the amount of change in motion can be expressed as the position difference between the same pixels in an image, it is suitable for conveying motion information. In an environment where the camera is fixed, it can be used to clearly represent moving objects in an image. Due to the continuity of action, optical flow serves as a useful means to obtain motion information for video understanding purposes. However, many of the videos included in the dataset contain numerous discrete instances of action. How the performance of TAL is affected by this situation may be a subject of inquiry.

Does using both types of data always benefit TAL's performance? In this section, how each type of data impacts performance will be examined.

3.4.1 Research Question

Two types of modality are used as feature vectors (RGB and Flow). RGB is suitable for conveying appearance information, while Flow represents the motion of actions. Generally, many papers have reported that overall performance is improved by using both types of modality. However, discontinuous actions may appear as boundaries in the Flow data.

- Does Flow data always have a positive effect? Is it possible that, in certain situations, Flow data could negatively impact performance?

Since the two modalities of data contain different information, they can be considered helpful in improving performance through a complementary relationship. However, in the case of discontinuous actions, motion information may not provide accurate information. In particular, optical flow represents the continuous movement of humans. Therefore, it has been considered as useful information based on the assumption that one action consists of a series of postures. However, in the case of discontinuous actions, flow images cannot express the continuity of actions. Actions do not progress smoothly at points where video editing or a sudden change in camera viewpoint occurs, making it easy to observe a sudden change of context or a different event. Therefore, these points are likely to be recognised as the boundaries of actions. In these discrete actions, the results obtained from the utilisation of RGB and Flow data separately versus their combination will be compared, and the influence of each data type on performance will be examined.

3.4.2 Experimental Settings

To conduct experiments on multi-modal data, three TAPG models were used (BMN, DBG, BSN++) on the THUMOS14 dataset. Feature extraction was performed by two types of two-stream network (TSN, I3D). For feature extraction, pre-trained models were used. These can be downloaded from the internet. Without any modification, the models were used to extract the two types of data, RGB and Flow.

For both feature types, feature vectors can be obtained from RGB and Flow channels, respectively. Firstly, three cases were set up to examine the performance of TAPG using features obtained from different channels: 1) only RGB data was used, 2) only Flow data was used, and 3) both channels were used. Additionally, to determine how these two types of data affected performance in the case of discontinuous actions, the performance evaluation of the discontinuous action used in Section 3.3 was also performed.

3.4.3 Experiment Results and Analysis

The results for the three cases on the test set of the THUMOS14 dataset are shown in Table 3.6. In each model, only 1500 proposals are kept after the post-processing process. AR

is calculated when 50, 100, 200, 1000, and 1500 action proposals are given for the six cases using two feature types and three models. According to this result, performance is generally better when using Flow data than when using RGB data. This can be seen as Flow data better represents the characteristics of action over time compared to RGB data. When extracting feature vectors using RGB frames, the entire image is used. Using RGB frames for feature extraction includes not only the parts related to the action but also many unrelated parts such as the surroundings. The THUMOS14 dataset comprises sports events scenes where athletes are often shown in small sizes to capture the entire scene. As a result, more unrelated parts are included in the image than action-related ones. In contrast, Flow data captures only the moving parts in the video. Hence, if the camera is fixed, even if the actor doing the action is small, the Flow data can capture their information well.

The above description indicates that both RGB and Flow data have their strengths and weaknesses, and by combining them, they can compensate for each other's limitations and improve performance. Therefore, it is beneficial to use both types of data when performing TAPG. However, it should be noted that in the case of discontinuous actions, the performance improvement effect of using both data types is not significant compared to the case of continuous actions. This is because in the case of discontinuous actions, the Flow data used to capture motion information may not accurately represent the action due to sudden changes or camera movements.

The overall performance in Table 3.6 shows that using both data types is more beneficial. Next, whether this holds true for discontinuous actions was tested. The results in Table 3.7 show a similar trend to the previous result, with Flow data outperforming RGB data even for discontinuous actions. This is somewhat unexpected, as it had been assumed that Flow data would not perform as well as RGB data in discontinuous actions. However, there are a few cases where using both types of features did not result in better performance than using only Flow data. Specifically, in the case of BMN with TSN and BSN++ with I3D, the AR@AN using both types of features was lower than that of using only Flow data. It was speculated that this may be due to the boundary-based models used in this experiment, which make proposal-level feature representations through sparse sampling and may skip discontinuous points.

Despite the unexpected results, it can be concluded that discontinuous actions have an impact on detection performance. cases where using both types of features did not ensure performance improvement in discontinuous actions were observed. Moreover, since only boundary-based methods were used in the experiments, different results may be obtained with anchor-free methods. In Chapter 4, this will be further investigated by using the proposed framework.

3.4.4 Summary and Discussion

In this section, how two modalities of data, RGB and Flow, which are primarily used to perform TAL, affect overall performance was investigated. RGB data contains colour infor-

Table 3.6. Result of using either RGB channel or Flow channel or both in all actions (THUMOS14)

	Feature type	Channel	AR@50	AR@100	AR@200	AR@500	AR@1000	AR@1500	AR@AN
BMN	TSN	RGB	23.87	30.88	37.94	46.91	53.81	57.36	48.13
		Flow	37.09	44.90	51.67	59.29	63.86	66.21	59.33
		Both (RGB+Flow)	**37.35**	**46.03**	**53.28**	**61.32**	**66.36**	**69.10**	**61.46**
	I3D	RGB	31.69	39.52	46.73	55.27	61.23	64.32	55.89
		Flow	34.46	42.97	50.34	58.46	63.87	66.46	58.80
		Both (RGB+Flow)	**37.23**	**45.43**	**52.69**	**60.30**	**65.35**	**68.10**	**60.54**
DBG	TSN	RGB	25.17	32.49	39.52	47.99	54.22	57.46	48.87
		Flow	**38.16**	**46.45**	**53.27**	**60.57**	**64.05**	**65.99**	**60.00**
		Both (RGB+Flow)	36.26	45.27	52.39	59.82	63.74	65.61	59.45
	I3D	RGB	32.93	40.68	48.25	56.31	61.04	63.61	56.33
		Flow	34.88	43.04	50.91	58.48	62.60	64.65	58.10
		Both (RGB+Flow)	**38.26**	**47.19**	**54.50**	**61.76**	**65.44**	**67.14**	**61.21**
BSN++	TSN	RGB	23.23	29.56	35.88	44.55	51.63	55.46	46.08
		Flow	34.19	41.42	47.70	55.76	61.08	63.56	56.22
		Both (RGB+Flow)	**37.38**	**44.71**	**51.22**	**58.67**	**63.16**	**65.63**	**58.81**
	I3D	RGB	28.44	35.17	42.27	50.32	56.43	59.74	51.34
		Flow	32.61	**40.67**	**47.74**	**55.86**	**61.41**	**64.14**	**56.30**
		Both (RGB+Flow)	**33.40**	40.63	46.56	53.86	58.86	61.38	54.36

Table 3.7. Result of using either RGB channel or Flow channel or both in discontinuous actions (THUMOS14)

	Feature type	Channel	AR@50	AR@100	AR@200	AR@500	AR@1000	AR@1500	AR@AN
BMN	TSN	RGB	24.01	29.17	36.91	47.13	53.72	56.85	47.74
		Flow	**38.81**	**47.77**	**53.45**	**60.33**	64.18	67.91	**60.27**
		Both (RGB+Flow)	35.28	44.19	50.74	58.56	**65.62**	**68.13**	59.52
	I3D	RGB	33.77	43.19	49.36	56.39	61.29	63.91	56.54
		Flow	32.49	39.34	45.14	55.46	60.84	63.47	55.58
		Both (RGB+Flow)	**38.15**	**44.28**	**50.09**	**57.43**	**62.81**	**66.21**	**58.15**
DBG	TSN	RGB	29.04	39.81	45.03	52.83	57.61	60.29	52.97
		Flow	**36.49**	**49.94**	**57.97**	**63.85**	**67.31**	**68.29**	**62.82**
		Both (RGB+Flow)	31.36	46.81	53.82	59.75	62.78	65.61	59.11
	I3D	RGB	32.64	43.09	51.58	59.97	64.78	65.99	59.42
		Flow	34.94	46.64	56.54	**63.58**	66.76	68.02	62.28
		Both (RGB+Flow)	**35.92**	**47.68**	**57.54**	63.38	**67.31**	**68.35**	**62.68**
BSN++	TSN	RGB	32.66	39.83	45.34	54.27	59.64	61.83	54.32
		Flow	36.78	**45.69**	50.84	59.09	64.22	67.36	59.17
		Both (RGB+Flow)	**39.06**	45.64	**53.87**	**61.08**	**64.68**	**68.51**	**60.76**
	I3D	RGB	30.10	40.43	46.48	57.61	61.88	64.95	57.03
		Flow	33.39	45.09	52.57	**59.51**	**64.75**	**66.26**	**59.70**
		Both (RGB+Flow)	**36.40**	**45.93**	**52.68**	59.33	64.57	**66.26**	59.53

mation that reveals the appearance of an action, while Flow data indicates how the scene has changed by showing the movement of the same pixel between successive image frames. These two data types are known to complement each other to improve performance. This was validated using three TAPG algorithms. The experimental results demonstrated that using both types of data is more effective in enhancing performance than using only one type of data.

Also, to analyse the effect in other aspects, the same experiment was conducted on discontinuous actions. Discontinuous actions were expected to have a negative effect on performance as the continuity of actions is disrupted. In particular, since Flow data expresses the movement of pixels, it had been assumed it would not be helpful in detecting actions when an unexpected scene occurred or was inserted. However, the results were reversed, and the assumption held true in only one case. Rather, there were cases where the results were good when only Flow data was used. This is because proposal-level representation was made with sparse sampling in the models, so discontinuous scenes may be ignored.

Although Flow data made a positive impact on performance in general, it was confirmed that performance was influenced by the discontinuous actions. An case where RGB data is better than Flow data was observed, so it can be concluded that Flow data is not always helpful and negatively influenced by discontinuous actions. This will be further investigated with the proposed model, and using both types of features tended to show enhanced performance.

3.5 Cross-Corpora Testing

Ideally, many researchers aim to develop algorithms that perform well in any given situation. However, it is almost impossible to collect all possible samples from domains related to problems to be solved using data-driven methods such as machine learning. Thus, the issue of generalisation becomes important in the field of machine learning. Even when addressing tasks such as TAL through learning-based methods, generalisation remains a significant challenge. Many papers in the literature report performance on benchmarking datasets that are widely used by others to demonstrate the excellence of their research results. There are various datasets available to detect and recognise human actions, as discussed in Subsection 2.4.5. These datasets differ in their target classes and class characteristics, making them suitable for different purposes. For instance, THUMOS14 and ActivityNet are commonly used benchmarking datasets in this field. However, when reporting performance using these datasets in most papers, different hyperparameters and configurations are set, and performance is reported separately. This situation is not ideal, and it raises doubts about the possibility of detecting action classes that were not used during training in a unified experimental environment.

In the TAPG step of the two-stage (pipeline) method, action proposals are generated without utilising class labels. This setting implies that there may be shared features that can be used for classification across different actions. If this is the case, it raises the question of

whether there are common features between unseen classes and seen classes. This section reports on experiments conducted to examine this issue.

3.5.1 Research Question

To assess the generalisation performance of the model trained with seen data, the model is evaluated on unseen data that share target categories with the seen data. The corresponding results are used to demonstrate excellence in terms of generalisation. However, what if the seen and unseen data have different target categories? Can the problem of generalisation in such extreme cases be addressed? The datasets commonly used in TAL consist of different target classes. Hence, the issue of generalisation between seen and unseen classes can also be addressed. Therefore, the following research question can be formulated:

- How will models trained on different datasets perform when tested on a dataset that was not used for training?
- What are the factors contributing to this performance difference?

This problem has already been addressed in a research topic called Transfer Learning. Specifically, transfer learning is utilised to solve a problem by leveraging the learned knowledge or information used to solve other or related problems. While this approach can be used to solve new problems using learned knowledge, it requires appropriate updates for new data through fine-tuning to acquire new knowledge. Therefore, if only the dataset is different, can't the trained model be used without additional procedures? To address this question, experiments with models trained on different datasets was conducted.

3.5.2 Experimental Settings

To conduct cross-corpora tests, three types of TAPG algorithms (DBG, BMN, BSN++) were used, all designed to solve the same problem. For this purpose, two datasets were prepared to evaluate the performance of models trained on different datasets. ActivityNet and THUMOS14 are the most commonly used datasets for TAPG in the literature; hence experiments were conducted using these two datasets. First, each model was trained using one dataset. The released versions of the models were used without any modification, and untrimmed videos were traversed differently depending on the dataset. For the ActivityNet dataset, a sequence of feature vectors from the video was resized using linear interpolation, and the resized length of videos was set to 100. For the THUMOS14 dataset, a sliding window scheme was utilised. A window size (l_w) of 100 was used, and a stride value of 50 was set to have an overlap ratio of 0.5 between neighbouring windows. This configuration is compatible with that of the ActivityNet dataset.

The testing set of the ActivityNet dataset does not include temporal annotations. Therefore, following the reporting convention of other research work, the model was trained using

the training set, and its performance was evaluated on the validation set. The training set of the THUMOS14 dataset consists of trimmed video clips, making it impossible to obtain temporal annotations or information about the context before and after the action. For this dataset, following the convention used in other work, the model was trained using the validation set, and its performance was evaluated on the testing set. The experiment was conducted according to the following procedure.

- The three models were trained on each dataset.
- Inference was executed using both a testing set included in the same dataset and a testing set included in a different dataset using the trained model. For instance, a model trained on the training set of the ActivityNet dataset was run on the validation set of ActivityNet and the test set of the THUMOS14 dataset. The same approach was used for the other cases.
- The performance was evaluated using the recall metric for the output of each case, and a comparison was made.

3.5.3 Experiment Results and Analysis

The results of the cross-corpora test are presented in Tables 3.8 and 3.9. Table 3.8 shows the results obtained from models trained on the training set of ActivityNet v1.3 and tested on the validation set of the same dataset (i.e., unseen data of seen classes) and the testing set of THUMOS14 (i.e., unseen data of unseen classes). As shown in this table, it can be observed that the models were able to detect actions even in datasets that were not used for training. There could be two reasons for this. Firstly, some actions in ActivityNet v1.3 are similar to those in THUMOS14 (as shown in Appendix A.1). From these similar classes, it can be inferred that the classes belonging to the unseen dataset could have been learned. Secondly, this could be attributed to the assumption that there are latent common characteristics between actions. TAPG is performed in a class-independent way, which means that the algorithm is designed to distinguish between background and foreground. Detecting an action in a dataset that includes unseen classes would indicate that the unseen classes also share the same characteristics that distinguish them from the background.

Table 3.9 shows the results obtained from models trained on the validation set of THUMOS14 and tested on the testing set of the same dataset (i.e., unseen data of seen classes) and the validation set of ActivityNet v1.3 (i.e., unseen data of unseen classes). Even in this case, it can be observed that detecting unseen classes is possible, as evident from the value of AR@1000. However, it is noticeable that the performance is poorer than that of models trained on the ActivityNet v1.3 dataset. This can be attributed to the size of the dataset. The ActivityNet v1.3 dataset comprises 10,024 untrimmed videos containing 200 action classes in the training set, including various actions such as leisure, sports, and daily life actions. In contrast, the THUMOS14 dataset comprises 200 videos containing 20 classes and is mainly composed of sports activities. Hence, the models trained on the ActivityNet v1.3 dataset

Table 3.8. Result of cross-corpora Test using ActivityNet dataset. The models are trained on the training set of ActivityNet v1.3 and tested on the validation set of ActivityNet v1.3 and the testing set of THUMOS14.

	Feature type	Test Set	AR@1	AR@5	AR@10	AR@100	AR@AN
BMN	TSN	ActivityNet	33.64	49.95	57.03	75.29	67.39
		THUMOS14	4.87	13.92	20.32	54.12	39.03
	I3D	ActivityNet	33.52	49.39	56.96	75.60	67.56
		THUMOS14	10.99	27.36	33.89	54.80	45.60
DBG	TSN	ActivityNet	30.46	49.38	57.04	76.57	68.15
		THUMOS14	4.35	11.46	18.09	56.53	40.25
	I3D	ActivityNet	31.75	49.12	56.92	76.28	67.82
		THUMOS14	5.77	15.52	24.35	61.27	47.22
BSN++	TSN	ActivityNet	33.98	47.65	54.85	74.80	66.28
		THUMOS14	5.96	19.09	29.05	64.96	51.15
	I3D	ActivityNet	33.26	46.23	53.33	74.21	65.32
		THUMOS14	22.69	38.04	44.19	65.44	55.79

Table 3.9. Result of cross-corpora test using THUMOS14. Models are trained on validation set of THUMOS14 and tested on validation set of ActivityNet 1.3 and the testing set of THUMOS14.

	Feature type	Test Set	AR@50	AR@100	AR@200	AR@500	AR@1000
BMN	TSN	ActivityNet	4.78	7.56	11.49	18.37	24.65
		THUMOS14	37.33	46.01	53.27	61.28	66.36
	I3D	ActivityNet	2.73	4.64	8.89	19.63	33.23
		THUMOS14	37.23	45.43	52.69	60.30	65.35
DBG	TSN	ActivityNet	17.12	23.61	32.42	45.46	54.51
		THUMOS14	36.24	45.29	52.43	59.84	63.77
	I3D	ActivityNet	4.68	8.31	14.10	28.67	41.83
		THUMOS14	38.26	47.20	54.50	61.76	65.44
BSN++	TSN	ActivityNet	2.51	3.89	6.27	10.11	14.84
		THUMOS14	37.39	44.71	51.22	58.67	63.16
	I3D	ActivityNet	1.74	2.34	2.95	3.78	5.07
		THUMOS14	33.40	40.63	46.56	53.87	58.87

demonstrate better performance in the cross-corpora test as they could observe a more extensive range of actions than the models trained on the THUMOS14 dataset.

3.5.4 Summary and Discussion

In this section, a cross-corpora test was conducted to analyse the model's performance in terms of generalisation. Unlike transfer learning, which utilises learned knowledge for other or related tasks, the performance was evaluated without any further processes like fine-tuning when using different datasets for the same task. Experiments were conducted using the ActivityNet v1.3 and THUMOS14 datasets, which are two widely used datasets for TAPG task.

The experiments conducted in this section indicates that a model does not perform well on an unseen dataset as well as on a seen dataset. The performance varied depending on the dataset used for training, which is proportional to the amount of information available from the dataset. It is found that the more classes and numbers of actions included in the training set, the higher the probability of detecting the unseen class by distinguishing it from the background.

Based on these results, new research directions can be suggested. One possible research direction could be to enrich background and foreground data based on data augmentation using generative models, and another one will be a zero-shot TAPG task by inspiring the detection of unseen classes.

3.6 Summary

This chapter has considered five investigations to identify factors that may impact the performance of TAL and conducted experiments to investigate how they affect that performance. These investigations include: 1) Examining performance relationship between detection and recognition, 2) Investigate the effect of using class labels on boundary detection, 3) Verifying a robust methodology for discontinuous actions, 4) Investigating the relationship between multi-modal data and TAL performance, and 5) Conducting a cross-corpora test.

The first investigation was related to the performance relationship between detection and recognition in the pipeline method. In particular, the two-stage method detects an action based on a detection-then-recognition scheme. In this approach, the result of the previous step is used as an input to the next step, and the accuracy of the previous step can influence the performance of the next step. This approach is widely used to improve performance in various fields including TAL. Two research questions were asked: 1) "When the performance of proposal generation improves, will the performance of recognition also improve?" 2) "Is there any relationship between the performance of proposal generation and that of recognition?". As the answer of the first question, better performance of proposal generation produced better results of recognition. As the answer of the second question, it was discovered that including

more positive samples in the action proposals used as input leads to better recognition performance. This is a factor that cannot be determined by AR alone. Therefore, even if the AR is high, the overall performance of TAL, including action recognition, cannot necessarily be considered excellent.

Secondly, the effect of using class labels on proposal generation was examined. It was assumed that using specific labels would be more beneficial in classifying actions by utilising the unique characteristics of each class. The subsidiary research question was "Are there any advantages of using action labels in terms of boundary detection?". However, contrary to expectations, there was no advantage in using specific labels, and the results of proposal generation were not as good as when training the models in a class-independent way. It was suggested that this was due to the complexity of the problem. This does not mean that one-stage methods are worse than pipeline methods. There is therefore a need for research that can utilise the unique characteristics of each class in a unified model.

In the third experiment, inspired by the fact that online video data often contains discontinuous actions due to editing, occlusion, sudden changes in viewpoint, and other factors, the anchor-based (boundary-based) method was compared with the anchor-free method for detecting discontinuous actions. Two research questions were asked: 1) "Which approach is more robust to unwanted discontinuous situation?" 2) "If one approach exhibits lower performance than the other, what is the reason behind it?". It was found that the anchor-free method is more robust to discontinuous actions than the anchor-based (boundary-based) methods. The anchor-based (boundary-based) method includes discontinuous points in the action because boundary detection is done in proposal units to calculate the score. On the other hand, the anchor-free method predicts boundaries using incomplete information, which could avoid discontinuous scenes. It should be noted that this experiment does not imply that the anchor-free method is superior in all cases. In fact, the results obtained with the anchor-free method are typically worse than those reported in the literature. However, this experiment provides an opportunity to find a way to leverage the unique advantages of the anchor-free method, which can be further explored in future research.

The fourth experiment investigated the relationship between multi-modal data and performance. The question was "Does Flow data always have a positive effect? Is it possible that, in certain situations, Flow data could negatively impact performance?". It is generally believed that combining RGB and Flow data improves performance, and through the experiments, it was confirmed that data fusion does indeed help improve performance, as reported in many other studies. However, for discontinuous actions, a case in which using RGB data is better than using Flow data was noticed, which means Flow data is negatively affected by the discontinuity. Nevertheless, it was also confirmed whether this relationship between multi-modal data and performance holds true for discontinuous actions as well.

Finally, a cross-corpora test on TAPG was conducted. The question was "How will models trained on different datasets perform when tested on a dataset that was not used for training?". As is well-known, the performance of TAPG models differs depending on the dataset used

for training, and the larger the difference between the classes included in the training set and the classes included in the test set, the lower the performance. It was found that models trained on the ActivityNet v1.3 dataset did not show significant performance degradation on the THUMOS14 dataset. This suggests that the larger the size of the dataset and the more diverse classes it contains, the smaller the performance degradation in the TAPG task.

While some of the experiments performed in this chapter may already be established findings, they are still meaningful in that they provide a rigorous investigation into these factors and suggest areas that can be further explored in future research. By clarifying these factors through experiments, the impact on TAL performance can be better understood and this knowledge can be used to inform the development of more effective TAL systems in the future.

Chapter 4

Anchor-free Pipeline Temporal Action Localisation

There are many anchor-based (boundary-based) methods for TAL in the literature. These methods allow for easy adjustment of the evaluation interval according to hyperparameter settings, which can prevent the omission of specific temporal segments during evaluation and reduce the possibility of missing action detection. However, this method can also cause computational inefficiency. When an untrimmed video is converted into feature vectors through the feature extraction process, a large number of vectors are created. Each vector becomes a temporal location to be evaluated for the presence of an action. At this time, the length of the action is unknown, and a set of anchors composed of several predefined lengths is used to evaluate the action length. The number of anchors included in this set can be adjusted as needed. This approach involves creating a large number of anchors to cover as many cases as possible. The resulting anchor set is used to evaluate the presence of an action in a similar way to block matching in computer vision, predicting the length of actions at each temporal location. This method requires repeated operations for each anchor at each temporal location, leading to computational inefficiency. Alternatively, boundary-based methods replace anchor sets with all possible combinations of temporal location pairs, with these pairs acting as anchors. However, since a specific temporal location may be included in multiple pairs, these two methods ultimately evaluate the length of the action in the same way, except for the method used to determine the set of temporal segments with a predefined length, resulting in inefficient calculation due to repetitive operation. Additionally, setting the anchor lengths using an empirical criterion may not cover the length of all possible actions. If the maximum number of anchors is set, the possibility of covering the length of possible actions increases, but it also increases the amount of computation. If the number of anchors is reduced for computational efficiency, it becomes difficult to cover the length of possible actions, resulting in lower boundary accuracy. Thus, anchor-based methods are known to be computationally inefficient and the detected boundary may not be accurate.

In contrast, anchor-free methods are mainly performed on feature pyramids consisting of multi-scale feature representations. Each layer consists of vectors containing information of different scales, and actions are detected by predicting left and right offsets using these vectors. This method has the advantage of not having to perform repetitive calculations at each temporal location by directly predicting the offsets of an action without using an anchor.

In addition, because it does not use predefined temporal segments, it can theoretically be free from the problem of inaccurate boundaries. However, it cannot be assumed that the feature pyramid's vectors contain all the necessary information about the actions present in the video. Each layer consists of vectors that possess information of the same scale. These vectors hold information divided into different scales of the video, but it is not guaranteed that one vector contains all the information about a complete action. In other words, offset prediction is conducted by analysing only a portion of the action, which is incomplete information. Therefore, although it is theoretically free from the inaccurate boundary issue, it cannot be conclusively stated that its accuracy in detecting an action is superior to that of the anchor-based method in practical applications.

In this chapter, the research question 2 and 3 are mainly handled: "How can various length of human actions be dealt with?" and " How can the boundary of human actions be detected?". To answer the questions, the proposed model, the Complementary Anchor-free network (CAFN), will be described. After introducing the motivation, the components of CAFN are explained in detail. Then, the performance is investigated by comprehensive ablation and comparative studies.

4.1 Motivation

Based on the strengths and weaknesses of anchor-based and anchor-free methods, there is a possibility of performing TAL while taking advantage of these strengths and compensating for the weaknesses. The repetitive computation of anchor-based methods caused by using anchors can be eliminated by directly predicting left and right offsets as in anchor-free methods, and action detection using incomplete information in the anchor-free method can be supplemented by checking the completeness of the detected action by performing proposal-level evaluation as in the anchor-based method. By combining these two methods, a more efficient and accurate TAL method can be developed. A complementary anchor-free network (CAFN) is proposed that utilises the strengths of both methodologies and compensates for their weaknesses. As mentioned earlier, the aim is to address each other's shortcomings with their respective strengths. In the anchor-based method, the length of the action is unknown, and a large number of anchors are used to cover all possible cases, leading to computational inefficiency. To address this, a set of anchors are generated by predicting them using an anchor-free method. By using predicted anchors instead of predefined ones, the number of repeated operations can be reduced at all temporal locations. On the other hand, proposal-level evaluation is performed to compensate for the unstable prediction caused by feature vectors in the anchor-free method, which contain incomplete information. This way, each other's weaknesses can be complemented with their strengths.

In addition to the challenge of defining the start and end of an action, inaccurate prediction of action boundaries may also arise from the characteristics of the two methodologies. There could be several reasons for this. The learning approach used here is supervised learning, which relies on the accuracy of the given ground-truth annotations. However, since human

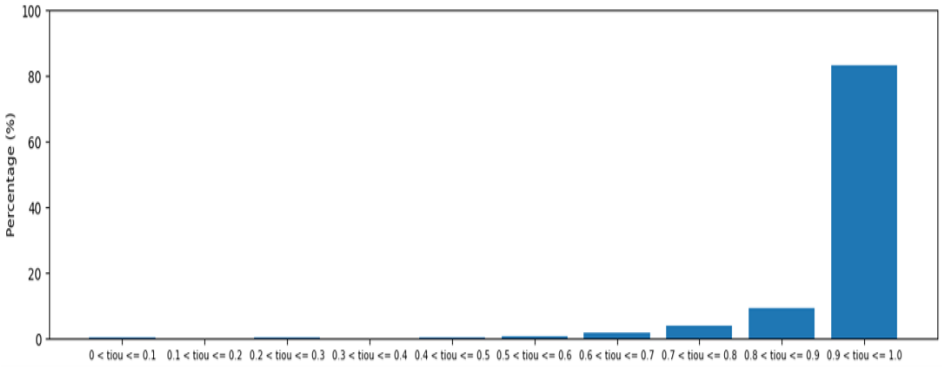
judgement is subjective, even actions of the same category may have inconsistent boundary information. Additionally, the feature vectors used in the anchor-based method are composed of a fixed number of consecutive image frames. The temporal interval for evaluating actions depends on the number of images and strides used for feature extraction. Real boundaries may be missed if they fall between consecutive temporal locations. While the anchor-free method can avoid missing boundaries due to direct boundary prediction, inaccurate boundaries may still be predicted due to the use of feature vectors containing incomplete actions. To address this issue, a boundary refinement module is proposed. Given the inconsistency of temporal annotations, it is nearly impossible to obtain exactly the same results as ground-truth labels. However, there is likely an implicit consensus on the human-perceived boundaries of actions, despite annotation inconsistencies. Therefore, boundaries can be refined using error information between the prediction results and ground-truth annotations. By statistically analysing the errors between the temporal annotations and prediction outputs, a point where people commonly agree on the boundary of an action can be identified.

4.2 Limitation of Anchor-free Method Compared to Boundary-based One

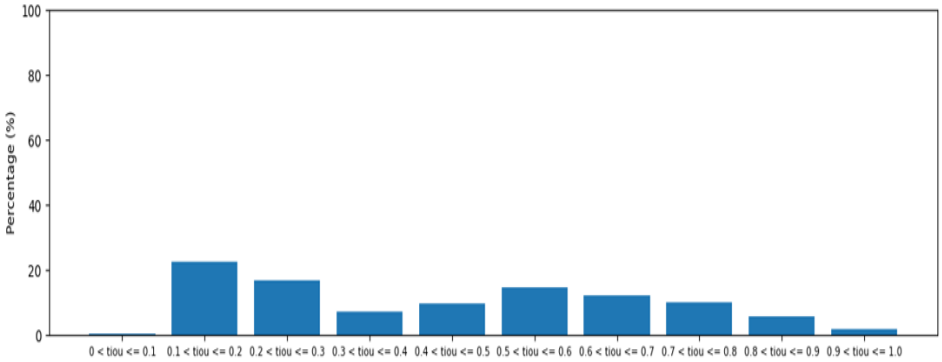
Before performing anchor-free TAL, why anchor-based (boundary-based) methods generally perform better than anchor-free methods was analysed. To achieve this, the temporal span covered by the feature vectors used for action boundary prediction were investigated. The construction of vectors used for action proposal generation was examined in a model that has demonstrated excellent performance in the literature, and the tIoU (temporal Intersection over Union) was calculated between the temporal span covered by each vector and the ground-truth segments. To do so, the boundary-based method (BMN[68], DBG[77], BSN++[69]) and the anchor-free method AFSD[85] were used to construct feature vectors on the ActivityNet v1.3 dataset.

The results are presented in histogram format in Figure 4.1. Based on the analysis, it was found that in the boundary-based method, approximately 90% of the ground-truth actions have a tIoU value of over 0.9 with one of the temporal spans used as an action evaluation unit. This indicates that the action is evaluated using a highly overlapped temporal span.

In contrast, it can be observed that there are few multi-scale feature vectors with high tIoU values that correspond to the ground-truth actions in the feature pyramid used in the anchor-free method. This implies that the vectors employed for predicting left and right offsets only capture a portion of the action. This difference arises from the way each method addresses the multi-scale issue. Figure 4.2 illustrates how the feature vectors span temporal extents in each method. For the sake of simplicity in explanation, it is assumed that the video comprises T feature vectors and that the input for each method accepts T vectors simultaneously. In the table at the bottom left of this figure, temporal pairs featuring feature vectors within the boundary-based method are marked with black circles. All temporal segments represented by pairs whose start time precedes the end time serve as evaluation units for action detection. In the boundary-based method, $(T + 1)T/2$ temporal pairs are evaluated, enabling a dense



(a) tIoU with ground truth in boundary-based methods (BMN[68], DBG[77], BDN++[69])



(b) tIoU with ground truth in feature pyramid of anchor-free method (AFSD[85])

Figure 4.1. tIoU Histogram of temporal segments with ground truth (GT) in ActivityNet 1.3 dataset.

search for actions.

On the other hand, the table at the bottom right of Figure 4.2 depicts the temporal spans in the feature pyramid. Compared to the boundary-based method, it demonstrates a very sparse distribution. In the anchor-free method, the presence of actions is evaluated using feature vectors of $\sigma_{T/2^{L-1}}$. In each algorithm of boundary-based methods, T is set to 100. Therefore, (100 + 1)100 / 2 = 5,050 temporal segments are evaluated to detect actions in one video. In contrast, in ASDF (anchor-free method), feature vectors are obtained from layers C4 and C5 of the I3D network. The C4 layer generates 192 vectors, and the C5 layer generates 96 vectors. If a feature pyramid was constructed using this, (192 + 96 + 48 + 24 + 12 + 6) = 378 vectors would be present. As such, since the two methods differ in the unit of information used for action detection, the anchor-based method generally exhibits superior performance compared to the anchor-free method. However, using a larger number of vectors necessitates more computing resources. These two methodologies can be said to have a trade-off between computing resources and performance.

4.3 Problem Formulation

The goal was to generate action proposals given a set of untrimmed videos. More specifically, it can be described as follows. Let a video dataset be denoted as $D = \{V^{train}, V^{test}\}$, where V^{train} and V^{test} are sets of untrimmed videos for training and testing, respectively. Each data $V^{train}, V^{test} = \{X, \Psi_g\}$ contains a frame sequence $X = \{x_i\}_{i=1}^{l_v}$, where x_i repre-

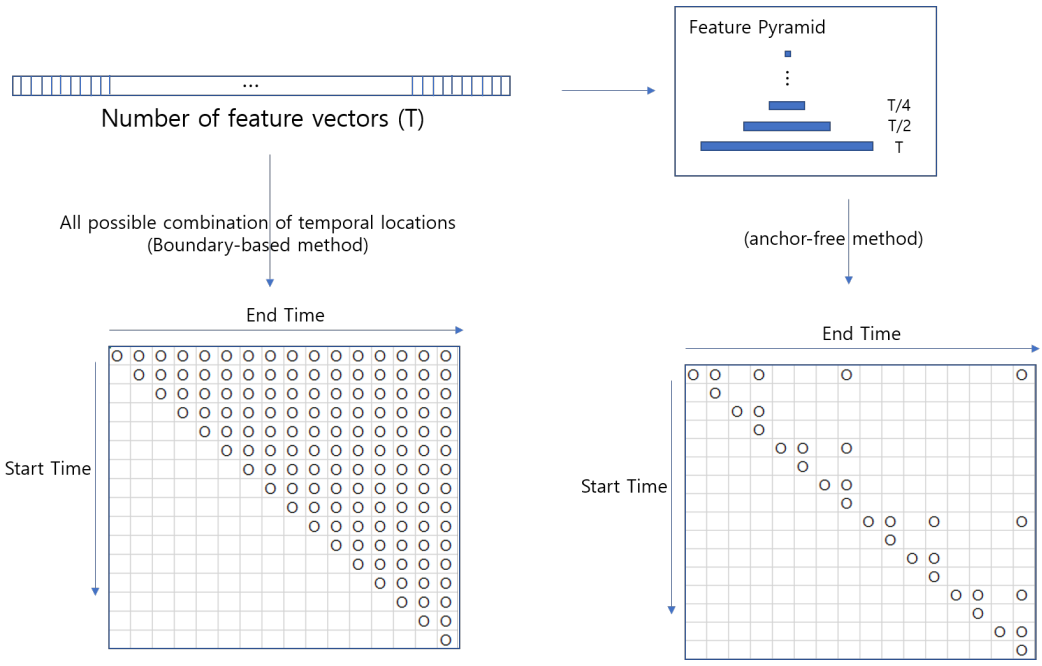


Figure 4.2. Temporal span of features in both boundary-based method and anchor-free method.

sents the i-th RGB frame in the video of length l_v. The corresponding temporal annotation Ψ_g of X consists of temporal action instances $\Psi_g = \{\psi_{g,i} = (t^s_{g,i}, t^e_{g,i}, l^c_{g,i})\}_{i=1}^{N_g}$, where N_g is the number of ground truth action instances in video V, and $t^s_{g,i}$, $t^e_{g,i}$, $l^c_{g,i}$ are the starting time, ending time and class label of the action instance ψ_i, respectively. During the training phase, the model is trained on Ψ_g. Unlike in the whole TAL, the categories of action instances are not included in ground-truth labels. During the inference phase, the trained model will generate action proposals $\hat{\Psi}_p = \{\hat{\psi}_i = (\hat{t}^s_i, \hat{t}^e_i, p_i)\}_{i=1}^{N_p}$, where $\hat{t}^s_i, \hat{t}^e_i$ are the starting and ending times of the predicted action proposals, and p_i is the confidence score of $\hat{\psi}_i$. The proposals $\hat{\Psi}_p$ should cover the ground truth action instances Ψ_g with high recall and high temporal overlap.

4.4 Pipeline of Complementary Anchor-free Network (CAFN)

A complementary TAPG model was proposed that leverages the strengths of both anchor-based and anchor-free methods. The entire proposed pipeline is depicted in Figure 4.3. A video representation network was employed to encode the spatial and temporal information in the input video. A type of two-stream network, called the inflated 3D network (I3D), proposed by [46], was used, and its output scores served as RGB and Flow feature vectors. These feature vectors were then fed into the proposed TAPG model.

The TAPG model was divided into two main parts: proposal generation in an anchor-free manner and verification of predicted proposals, similar to the approach used in boundary-based methods. In proposal generation, a feature pyramid is created to accommodate the various action lengths using the extracted I3D feature vectors. The feature pyramid comprises vectors containing information of different scales in each layer, making it suitable for handling multi-scale issues. This feature pyramid is considered as the base module of the framework. It is known that fusing vectors with those in adjacent layers is helpful for improving detection

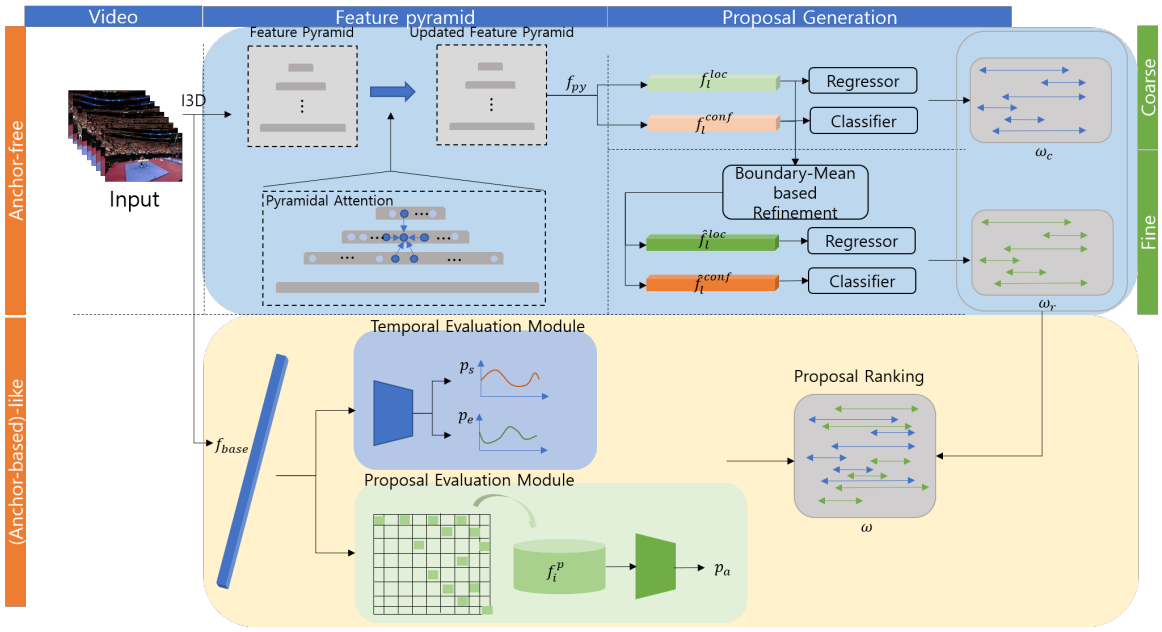


Figure 4.3. Overview of proposed approach (CAFN), which consists of two parts: 1) proposal generation in an anchor-free manner, and 2) proposal verification in an anchor-based manner. The proposal generation part includes both coarse and refined prediction steps, while the verification part consists of a temporal evaluation module and a proposal evaluation module.

performance. To incorporate these aspects, a pyramid-type attention mechanism is utilised to update the feature pyramid. Following this, the left and right offsets of action proposals are predicted. This step consists of two modules. First, proposals are generated using a simple regressor and classifier. This step is referred to as coarse prediction. Subsequently, the boundaries of the generated proposals are refined further. To achieve this, refined prediction is performed by creating boundary feature vectors using statistical sampling techniques at the boundaries of the generated proposals. Since the vectors present in the feature pyramid contain only a portion of the action information, the result cannot be considered stable. To compensate for this, proposal-level actionness scores, as well as starting and ending scores, are calculated similarly to the boundary-based method. This is carried out in the proposal evaluation module and the temporal evaluation module, respectively. Instead of using either all possible temporal pairs or predefined anchors, the generated proposals are utilised as anchors or temporal pairs to predict the starting and ending scores, along with actionness scores. The ranking of generated proposals is determined using the scores from the coarse prediction, refined prediction, proposal evaluation module, and temporal evaluation module. Finally, soft NMS is performed to remove duplicate results.

4.4.1 Video Feature Encoding

Following previous proposal generation methods [59, 60, 67, 76], the proposed model was built on visual feature sequences extracted from raw video. In this work, the inflated 3D network (I3D) were employed for video encoding. This method has achieved high precision in action recognition and has been widely used in numerous video understanding studies [55, 74, 110]. A pre-trained I3D model trained on the Kinetics dataset was used. Image frames $X = \{x_i\}_{i=1}^{l_v}$ were extracted from the untrimmed video at a controlled frame rate (details

will be explained in Appendix B), and the set of images is partitioned into snippets sequence $S = \{s_i\}_{i=1}^{l_s}$ by a regular interval δ, where $l_s = l_v/\delta$. The snippet s_i consists of five RGB images and five stacked optical flow images. The outputs were collected from the layer right before the last fully connected (FC)-layer of the two-stream network to represent a video with a set of 3D features $F = \{f_i^{rgb}, f_i^{flow}\}_{i=1}^{l_s} \in \mathbb{R}^{T\times C\times H\times W}$, where T, C, H, and W represent the time step, channel, height, and width, respectively.

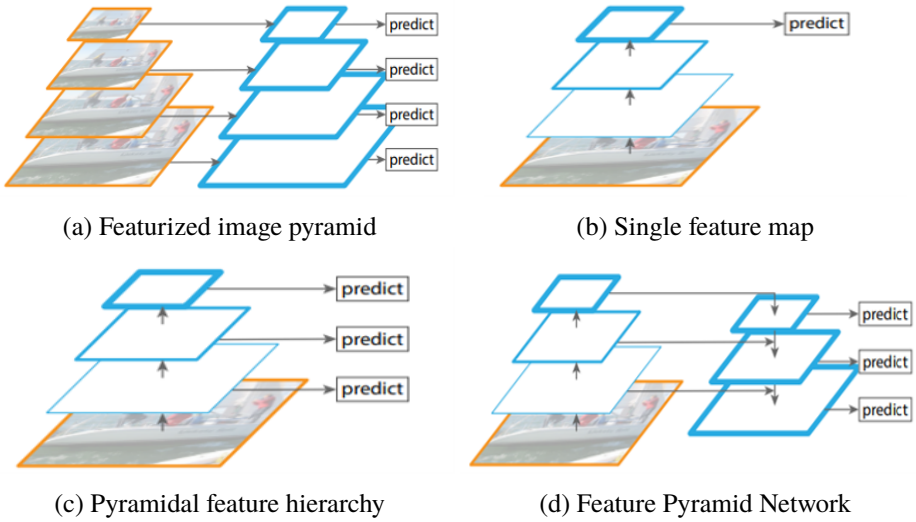
4.4.2 Base Module

Many anchor-free methods utilise the feature pyramid structure to handle multi-scale issues. The feature pyramid was used as a base module to address the same problem as in previous studies. Once 3D features $F = \{f_i^{rgb}, f_i^{flow}\}_{i=1}^{l_s} \in \mathbb{R}^{T\times C\times H\times W}$ have been obtained in the feature extraction step, these two types of feature vectors are concatenated. In the proposed framework, only the time axis is taken into consideration. Therefore, convolution is used to shrink the dimension of H and W to size 1, and these dimensions are subsequently dropped out (4.1). In this way, the base feature is converted to $\hat{F} = \{\hat{f}_i\}_{i=1}^{l_s} \in \mathbb{R}^{T\times 2C}$, and this is used as input to construct the feature pyramid.

$$\begin{aligned}\hat{F} &= \{\hat{f}_t\}_{t=1}^{l_s} \\ &= ReLU(GN(Conv3D(concat(f_t^{rgb}, f_t^{flow}), C_{in}, C_{out}, ks))) \in \mathbb{R}^{T\times C},\end{aligned} \tag{4.1}$$

where C_{in}, and C_{out} are the channel size of input and output respectively, GN stands for group normalisation, and ks is the kernel size, which is set as (1,3,3).

Feature Pyramid



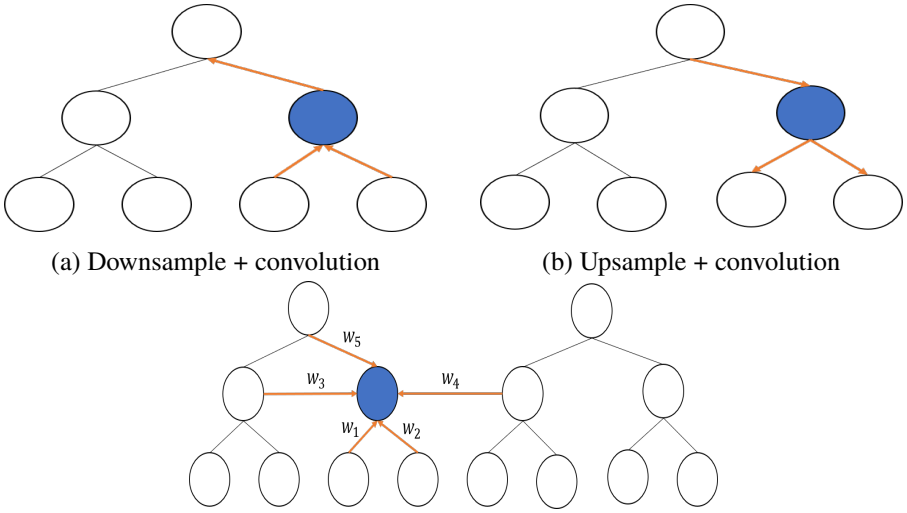
(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 4.4. Four types of pyramids from [82]. In the proposed framework (CAFN), FPN was constructed based on (d), then it was updated a pyramidal attention instead of downward connection.

Given the base feature $\hat{F}$, a feature pyramid is formed by stacking layers with progressively wider receptive fields. This technique was initially introduced to address multi-scale problems in object detection [82], and later adapted to a one-dimensional format to tackle similar issues in action detection. According to [82], there are four methods for constructing a pyramid-shaped feature structure. Figure 4.4 illustrates the four types of pyramids introduced in [82]. Figure 4.4a represents a method that generates features of different scales independently by directly adjusting the input resolution, with each layer detecting targets at its corresponding scale. In Figures 4.4b and 4.4c, features in lower resolution layers are generated using those from higher resolution layers. This is based on the observation that features from neighbouring higher resolution layers are beneficial for tasks performed in lower resolution layers. In Figure 4.4d, a lateral connection is added from lower resolution layers to the higher resolution layer, further enhancing the model's capability. This configuration is the most commonly used form of feature pyramid network for addressing multi-scale problems. These pyramid-shaped variations demonstrate that features from different layers can assist with tasks performed across other layers at various scales. Depending on how features from other scales are fused with those of the current scale, various types of feature pyramids can be created. To facilitate this fusion, a method of combining features from different layers using an attention mechanism has been introduced.

Attention-based Feature Fusion



(a) Downsample + convolution (b) Upsample + convolution

(c) Pyramidal self-attention

Figure 4.5. Data propagation in Feature Pyramid Network (FPN). The lateral connections in FPN allow features to be mixed in one direction ((a), (b)), while pyramidal attention allows features to be fused in both directions as well as from sibling nodes (c).

There are several ways to fuse features from adjacent layers in a Feature Pyramid Network (FPN). Among them, the most widely used method involves a combination of upsampling (or downsampling) and convolution to match different temporal dimensions between neighbouring layers. Features from the two layers are then combined using operations such as weighted average or summation. This method is implemented by inserting lateral connections in the previously mentioned pyramid-shaped structure. However, when convolution is used, the

same weight value is applied to all locations because the same convolution filter is utilised when fusing feature information from other layers in each feature. Figure 4.5 illustrates this situation. To simplify the explanation, a feature pyramid is considered as a graph. Figures 4.5a and 4.5b illustrate the data propagation directions when a feature pyramid is created using a convolution operation. In the two top figures, the blue circle represents the feature vector to be updated. In Figure 4.5a, inserting a lateral connection from a lower layer with high resolution to a higher layer with low resolution can be implemented using a combination of downsampling and convolution. The features of the lower layer are weighted with the convolution filter. The same filter is also applied when updating its sibling nodes. Similarly, in Figure 4.5b, the lateral connection from a higher layer with low resolution to a lower layer with high-resolution features can be implemented using a combination of upsampling and convolution. The features are fused in a similar manner. In both cases described above, it is important to note that only one-way fusion is possible. To implement a bidirectional lateral connection, as suggested in Figure 4.4d, one approach would be to create a feature pyramid with bottom-up lateral connections and then calculate it again with top-down lateral connections. However, such repetitive execution may not be regarded as an efficient computational method. To address this issue, an attention mechanism is introduced. Figure 4.5c demonstrates the data propagation in the lateral connection using a pyramidal attention approach. The blue circle receives feature information from parent nodes, child nodes, and sibling nodes, and data fusion occurs according to each weight. Such an operation can be implemented using self-attention with neighbouring nodes.

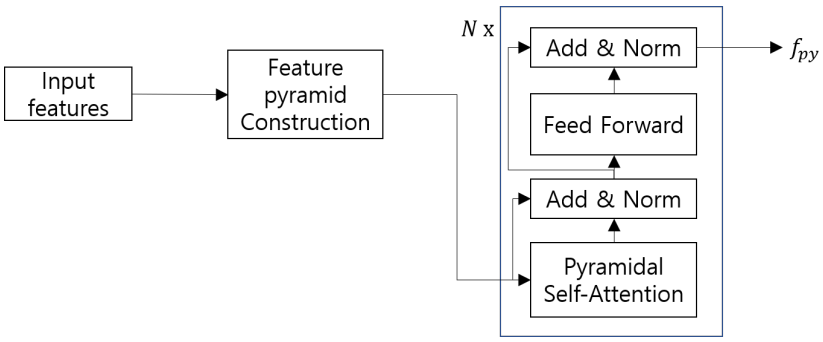


Figure 4.6. Updated feature pyramid using pyramidal attention.

A feature pyramid network has a graph-like form, with feature vectors in lower layers used to construct feature vectors in higher layers, creating a parent-children relationship in the graph. The method of implementing the self-attention module in FPN is first proposed in the Pyraformer [111]. In this work, long-range data is formed into a pyramidal structure and used to anticipate future events. Inspired by this work, a self-attention module is implemented in the form of a Pyraformer. Figure 4.6 illustrates the process of updating the feature pyramid using pyramidal attention. Given input feature vectors, a feature pyramid is created using convolution. The attention mechanism is then performed in the same way as the transformer's self-attention module. The key difference is that the pyramidal self-attention module is used instead of the standard self-attention module.

Before explaining pyramidal attention, the self-attention mechanism of the transformer

architecture will be discussed. Assume the inputs and outputs are X and Y. Initially, X is converted into three different vectors (query $Q = XW_Q$, key $K = XW_K$, and value $V = XW_V$) through linear operations, where W_Q, W_K, and $W_V \in R^{L \times D_K}$. The attention weight for the i-th input vector is calculated using the scaled dot product of the i-th query q_i and the key K. In other words, a row vector q_i in query Q can interact with all keys. By applying the softmax function to the calculated value, the final attention weight vector is computed. The output y_i is then determined as a weighted sum of all row vectors of V by multiplying the weight by value V. This can be expressed as follows.

$$y_i = \sum_{l=1}^{L} \frac{exp(q_i k_l^T / \sqrt{d_K} v_l)}{\sum_{l=1}^{L} exp(q_i k_l^T / \sqrt{d_K})} \tag{4.2}$$

Standard self-attention determines attention weights through similarity calculation with all input vectors. This process requires a large amount of computation, as the attention weight is calculated by the query participating in all keys. Many studies have been conducted to efficiently calculate this part [112]. Pyramidal attention can be considered as one of the results of these efforts. Pyramidal attention aims to enable more efficient calculation by allowing only specific vectors to participate in calculating the attention weight. Additionally, to address the multi-scale issue, it is performed on the feature pyramid. The feature pyramid can be viewed as a C-ary tree. In the feature pyramid, all nodes except for the leaf nodes and root nodes have C children and one parent. Therefore, instead of participating in all nodes, each node can calculate the attention weight by participating in a limited set of directly connected nodes, as shown in Figure 4.5c. Let $n_i^{(s)}$ be the i-th node of scale s. This node can participate in neighbouring nodes of three scales: the adjacent nodes ($\mathbb{A}_i^{(s)}$) of the scale including itself, the parent node ($\mathbb{P}_i^{(s)}$), and the children nodes ($\mathbb{C}_i^{(s)}$). In other words, the nodes participating in calculating attention weights at each node i can be defined as follows:

$$\begin{aligned}
\mathbb{N}_i^{(s)} &= \mathbb{A}_i^{(s)} \cup \mathbb{C}_i^{(s)} \cup \mathbb{P}_i^{(s)} \\
\mathbb{A}_i^{(s)} &= \{n_j^{(s)} : |j - i| \leq \frac{A-1}{2}, 1 \leq j \leq \frac{L}{C^{s-1}}\} \\
\mathbb{C}_i^{(s)} &= \{n_j^{(s)} : (i-1)C < j \leq iC\} \quad if\ s \geq 2\ else\ 0 \\
\mathbb{P}_i^{(s)} &= \{n_j^{(s+1)} : j = \lceil \frac{i}{C} \rceil\} \quad if\ s \leq S-1\ else\ 0
\end{aligned} \tag{4.3}$$

After determining the nodes to participate in this way, the output Y for the input X in pyramidal attention can be rewritten as:

$$y_i = \sum_{l \in \mathbb{N}_i^{(s)}} \frac{exp(q_i k_l^T / \sqrt{d_K} v_l)}{\sum_{l \in \mathbb{N}_i^{(s)}} exp(q_i k_l^T / \sqrt{d_K})} \tag{4.4}$$

In Figure 4.6, the output Y is denoted as f_{py}.

4.4.3 Coarse Proposal Prediction

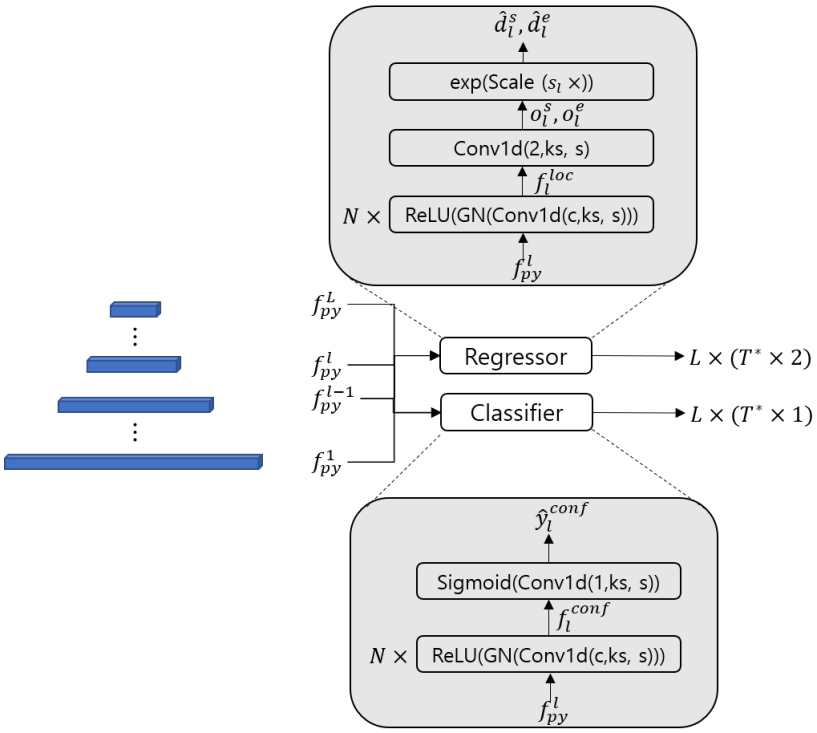


Figure 4.7. Proposal prediction heads. To predict boundary offsets and classification scores, two simple modules are used as a classifier and a regressor.

After obtaining f_{py} in the step of creating a feature pyramid, the model performs classification to distinguish the foreground and background of an action and regression to detect the boundary of an action by predicting left and right offsets. To accomplish these tasks, separate branches are created that use shared feature vectors. Figure 4.7 shows a diagram of the headers for performing each task in these two branches. Each header has a similar structure. First, ReLU(GN(Conv1d(c, ks, s))) is used repeatedly N times to convert the output of the feature pyramid into vectors in the latent space suitable for each task, where c is the dimension of the output, ks is the kernel size, and s is the stride. Two separate small networks with the same structure are used to learn the necessary information and output vectors f^{conf} and f^{loc} in the latent space. For the classification task, as binary classification is performed, f^{conf} is used as input, and Conv1d(1,ks,s) is applied, followed by the sigmoid function. The same classifier is used for all feature vectors in all layers. For the regression task, Conv1d(2,ks,s) is performed using f^{loc} as input. The aim of the regression branch is to predict the left and right offsets at each time location. To account for the different target scales in each layer of the feature pyramid, the regression branch uses an output dimension of 2. In order to share one head for all layers, a learnable scale factor s_l is multiplied at the end. As mentioned in [113], the target value in the regression branch is always a positive, the output value of the regression branch is transformed by applying the exponential function, which maps the output to any real number in the range of $(0, \infty)$. In the FPN, the action proposal $\omega_c^{l,i} = \{\hat{t}_{l,i}^s, \hat{t}_{l,i}^e\}$ at the i-th time location of the l-th layer can be obtained as follows:

$$
\begin{aligned}
\hat{t}^s_{l,i} &= i * 2^l - \hat{d}^s_{l,i} \\
\hat{t}^e_{l,i} &= i * 2^l + \hat{d}^e_{l,i} \\
\hat{d}^s_{l,i} &= exp(s_l \times o^s_{l,i}) \\
\hat{d}^e_{l,i} &= exp(s_l \times o^e_{l,i}),
\end{aligned}
\tag{4.5}
$$

where $\hat{t}^s_{l,i}, \hat{t}^e_{l,i}$ are the starting and ending points of the predicted proposal, and $\hat{d}^s_{l,i}, \hat{d}^e_{l,i}$ are the left and right offsets.

4.4.4 Two-Stage Boundary Refinement (Refined Prediction)

The proposed framework performs boundary refinement in two steps: indirect and direct. Indirect refinement aims to predict more accurate boundaries by improving backbone features used as inputs to the action proposal generation, while direct refinement aims to refine the boundaries of predicted proposals based on boundary information. In Figure 4.3, two types of action proposal generation are performed in the proposal generation stage: coarse prediction and refined prediction. In the coarse prediction, proposals are generated on the feature pyramid updated by pyramidal attention. However, as mentioned in Section 4.2, the feature pyramid network used in previous studies of the anchor-free method is composed of feature vectors that only contain partial information of the target actions. These incomplete features may adversely affect the performance of the model. The pyramidal attention mechanism widens the receptive field of each feature vector by incorporating information from neighbouring nodes, leading to indirect boundary refinement and improving the backbone structure. In the refined prediction stage, direct boundary refinement is performed using the proposals obtained in the coarse prediction. In this step, residual offsets are calculated instead of absolute offsets. The boundary refinement is based on the assumption that creating features for boundary refinement in regions aligned with actual boundaries will allow for more accurate boundary prediction. Previous work has created boundary features for refinement using vectors with a constant ratio before and after the predicted boundaries. In some cases, this approach may result in boundary features that are slightly off from the actual boundaries, leading to inaccurate information being provided to the model. While previous work, such as [85], applied max pooling to features sampled from the boundary area to obtain salient values, this method may still not provide accurate boundary information for learning. To address this issue, a novel method for creating boundary feature vectors is proposed using a learning-based alignment and sampling technique.

Gaussian Mean Learning

Once the proposals ω_c generated from coarse proposal generation have been obtained, a module is needed to align the predicted boundaries with the real boundaries. Figure 4.8 shows a module that learns parameters for creating boundary features. It is hypothesized that the

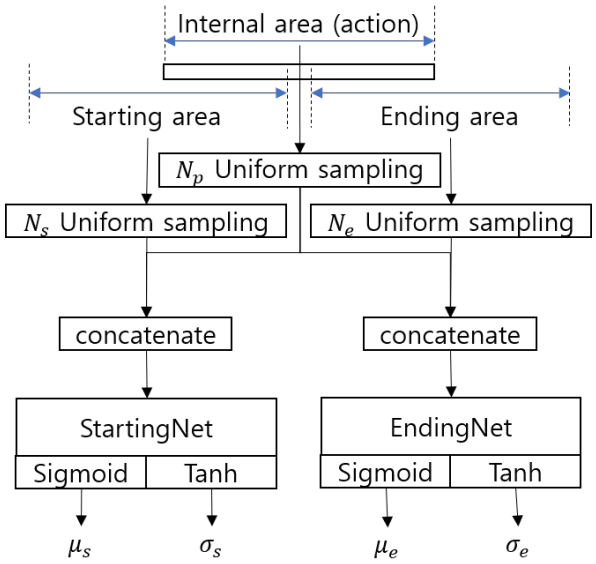


Figure 4.8. Mean and standard deviation learning. The StartingNet and EndingNet predict the means and standard deviations of real boundaries with respect to predicted proposals in coarse prediction.

information required to align the boundary can be modeled as a Gaussian function, and the purpose of this module is to find the mean and standard deviation of the Gaussian function. As in other work, the starting area, ending area, and internal area are defined as follows within the predicted proposals.

$$\begin{aligned} r_s &= \{t^s - l_a/k, t^s + l_a/k\} \\ r_e &= \{t^e - l_a/k, t^e + l_a/k\} \\ r_a &= \{t^s, t^e\} \\ l_a &= t^e - t^a, \end{aligned} \tag{4.6}$$

where r_s, r_a, r_e are the starting, internal, ending regions respectively, l_a is the length of predicted proposal, k is the ratio factor to set the starting and ending region based on the length of the predicted proposal; $k = 2$ was used.

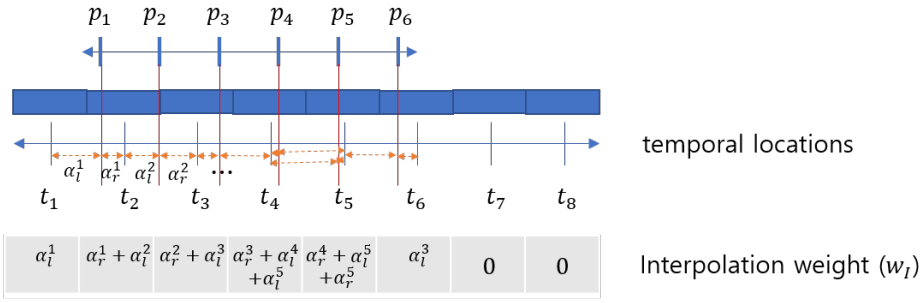


Figure 4.9. Weights for linear interpolation. After calculating weights for linear interpolation in the form of vectors, uniform sampling is done by multiplying the feature vectors by weight vectors.

Uniform sampling is performed using linear interpolation in each region to create input vectors for predicting Gaussian parameters. Specifically, N_s points are sampled in the starting region r_s, N_a points in the internal region r_a, and N_e points in the ending region r_e. In each region, a single feature vector is generated by taking a weighted average of the sampled

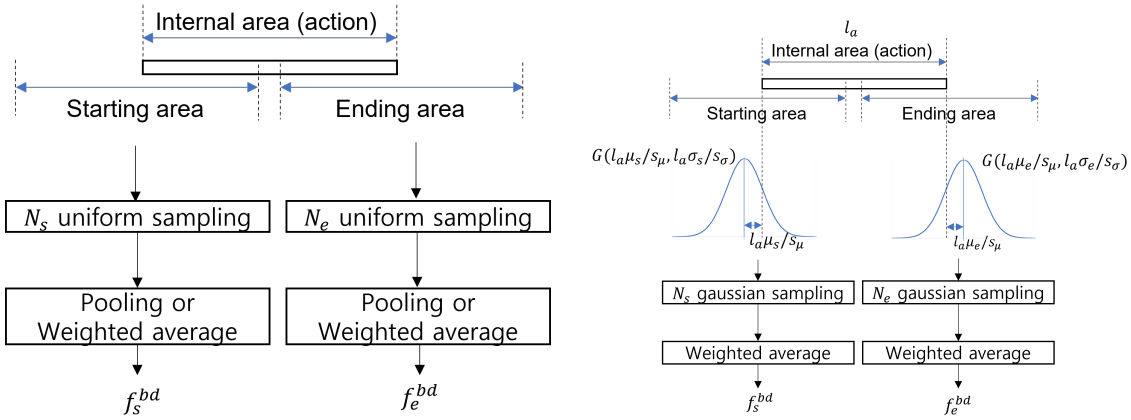
vectors. The weights are determined by linear interpolation and a mask matrix composed of these weights is used to calculate the weighted average efficiently. This weight vector is illustrated in Figure 4.9. To illustrate, let $P = \{p_i\}_{i=1}^{i=6}$ be the points that need to be sampled and $V = t_i{}_{i=1}^{i=8}$ be the corresponding feature vectors. If the first point p_1 lies between t_1 and t_2, it can be expressed as $p_1 = \alpha_l^1 t_1 + \alpha_r^1 t_2$ resulting in single weight vector. Other points are expressed in the same way, leading to the mask matrix composed of multiple weight vectors. The sampling is done using the mask matrix composed of weight values, which is created based on the interpolation weights for each point. The resulting feature vectors f_s, f_a, and f_e are used as inputs to StartingNet and EndingNet to obtain the mean and standard deviation of the boundary through concatenation as follows:

$$\begin{aligned} f_{in}^{start} &= concatenate(f_s, f_a) \\ f_{in}^{end} &= concatenate(f_e, f_a), \end{aligned} \tag{4.7}$$

where f_{in}^{start} and f_{in}^{end} are the input to StartingNet and EndingNet, respectively.

StartingNet and EndingNet are constructed with three layers of convolutional layers followed by Group Normalisation and ReLU activation functions. The last layers of StartingNet and EndingNet use the sigmoid function to output values between 0 and 1 for the mean, and the tanh function to output values between -1 and 1 for the standard deviation.

Boundary Feature Construction



(a) Boundary feature construction based on uniform sampling

(b) Boundary feature construction based on Gaussian sampling

Figure 4.10. Illustration of various techniques of constructing boundary features using different sampling methods.

Boundary features are created for boundary refinement using the outputs of StartingNet and EndingNet. Figure 4.10 shows different ways of constructing boundary features, using either uniform sampling (on the left) or Gaussian sampling (on the right). With uniform sampling, the sampled points have equal weights or learned weights using the convolution layer. With Gaussian sampling, the parts considered as real boundaries receive more weight, while the neighbouring areas receive lower weights. Specifically, given μ_s and σ_s, N_s points

are sampled via the Gaussian function $G(l_a\mu_s/s_\mu, l_a\sigma_s s_\sigma)$ in the starting area, where s_μ and s_σ are the scale factors for the mean and standard deviation, respectively. For the mean, it is set to $s_\mu = 1/2$ so as not to deviate from the starting and ending areas, and it is known that $3 \times \sigma$ in the Gaussian distribution includes 99.7%, so $s_\sigma = 1/3$ was set so that sampling is done within each area. To perform Gaussian sampling, first, N_s points are sampled using uniform sampling. Then, the index for Gaussian sampling is obtained by using the sampled index as an input to the ICDF for a given Gaussian function, as shown in Figure 4.9. Similar to the process shown in Figure 4.9, the sampled points are used to create a starting boundary feature f_s^{bd} using linear interpolation. The same process is applied to the ending area to create a corresponding ending boundary feature f_e^{bd}. These features, along with the internal feature vector f_a, are used for boundary refinement.

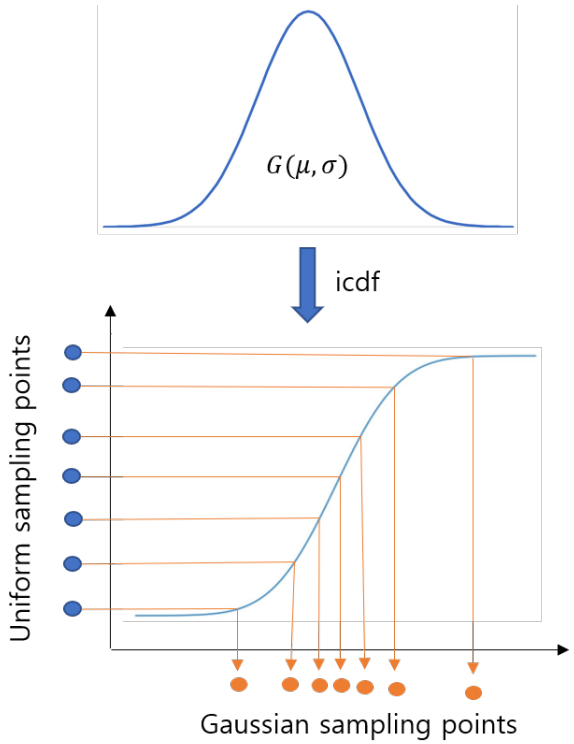


Figure 4.11. Gaussian sampling. After sampling points uniformly, its indices are used to get the indices of Gaussian sampling via inverse of cumulative distribution function (ICDF).

Boundary Refinement

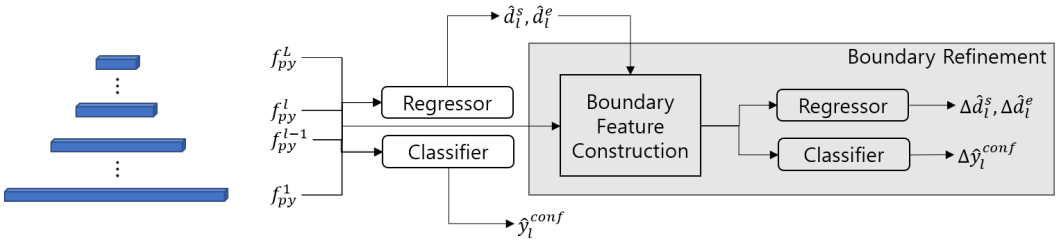


Figure 4.12. Residual offset prediction for boundary refinement.

The refined prediction follows a similar process to the coarse prediction. However, instead of directly predicting the left and right offsets at each time location, residual offsets are predicted based on the coarse prediction results to adjust their boundaries. As depicted in Fig-

ure 4.12, the refined prediction also employs a regressor and a classifier with the same structure as those used in coarse detection. The classifier outputs the confidence score ($\Delta\hat{y}^{conf}$) for the new proposals, while the regressor outputs $\Delta\hat{d}^{s}$ and $\Delta\hat{d}^{e}$, which are the offsets to be additionally adjusted for the boundaries of the proposals detected in the previous step. After this process, the final predicted boundary can be obtained by combining the coarse and refined prediction results as follows.

$$\begin{aligned} \hat{t}_l^s &= t^l - o_{left} \\ \hat{t}_l^e &= t^l + o_{right} \\ o_{left} &= \hat{d}_l^s \Delta\hat{d}_l^s = exp(s_l^c o_l^s + s_l^r \Delta o_l^s) \\ o_{right} &= \hat{d}_l^e \Delta\hat{d}_l^e = exp(s_l^c o_l^e + s_l^r \Delta o_l^e), \end{aligned} \tag{4.8}$$

where $\hat{t}_l^s, \hat{t}_l^e$ are the final predicted starting and ending points time, o_{left}, o_{right} are the left and right offsets, s_l^c, s_l^f are the scale factors of coarse regression branch and refine regression branch, o_l^s, o_l^e are the output of coarse regression, and $\Delta o_l^s, \Delta o_l^e$ are the output of fine regression branch.

4.4.5 Compensation in Anchor-based Method

As mentioned in Section 4.2, the feature vectors included in the feature pyramid may contain incomplete action information. To compensate for these disadvantages of the anchor-free method, a proposal evaluation module (PEM) and a temporal evaluation module (TEM) used in the boundary-based method are introduced. The PEM can confirm the reliability of the generated proposals by evaluating the actionness of the action proposals generated by the anchor-free method. The PEM of BMN [68] was adopted. The PEM of BMN evaluates all possible temporal pairs, which is dense evaluation, while that of the proposed model only evaluates predicted proposals, which is sparse evaluation. The TEM is a module that calculates the probability of the start and end of an action at each temporal location. Tthe idea suggested by BSN++ [69] was adopted to the proposed model.

Proposal Evaluation Module

The purpose of the PEM is to generate confidence scores using complete information of action proposals generated by the anchor-free method. Given the proposals $\omega = \{\omega_c, \omega_r\}_{i=1}^{N_p}$ as shown in Figure 4.13, proposal-level feature vectors f_p^{pem} are created by referring to the predicted proposals. The input for this module is the I3D feature vectors $\hat{F}$ extracted from the video before creating the feature pyramid. N_s vectors are uniformly sampled for each proposal via linear interpolation in the extended proposal area that includes starting area, internal area, and ending area to get intermediate proposal features $\hat{f}_p^{pem} \in R^{C\times N_p\times N_s}$. Then, a 2D convolution layer with a kernel size of $(N_s, 1)$ is applied to the intermediate features to obtain

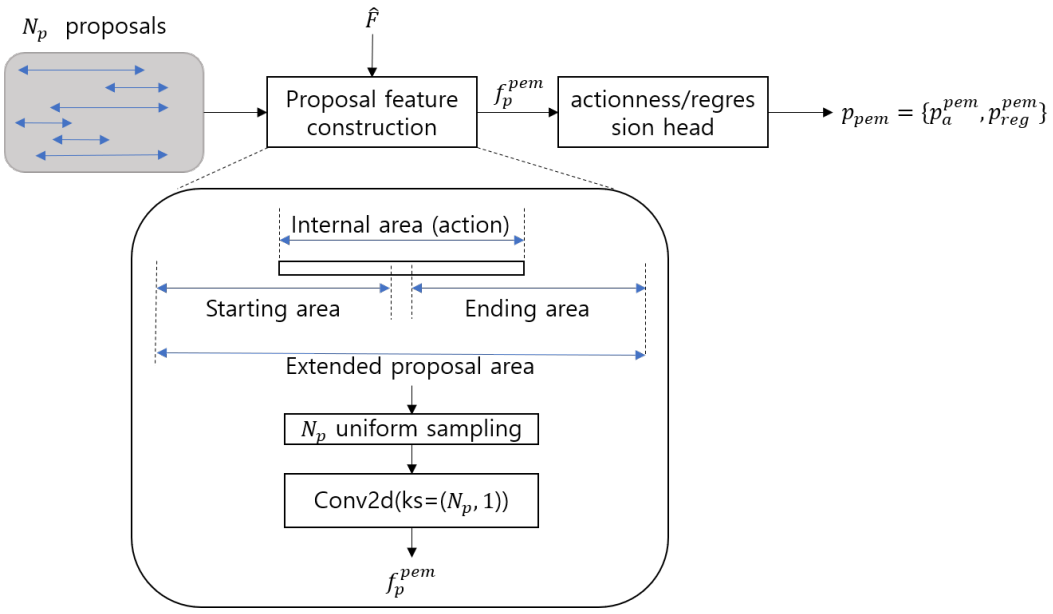


Figure 4.13. Diagram of proposal evaluation module

the proposal-level feature vectors $f_p^{pem} \in R^{C \times N_p}$. Finally, a head consisting of three convolutional layers is applied to f_p^{pem} to get the actionness score p_a^{pem} and the regression score p_{reg}^{pem}. In the coarse prediction and the refined prediction, only confidence scores for classification are obtained, but without scores for regression. Therefore, the regression scores obtained from PEM can be used as regression confidence scores for generated proposals.

Temporal Evaluation Module

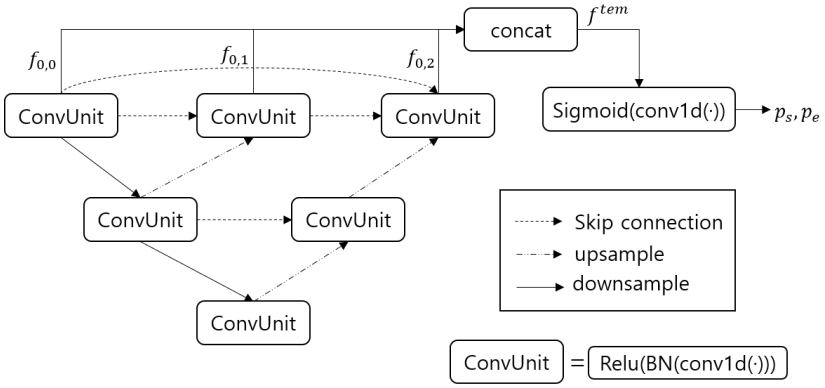


Figure 4.14. Diagram of temporal evaluation module

TEM is designed to evaluate the probabilities of the start and end of an action at all temporal locations in an untrimmed video. The Complementary Boundary Generator introduced in BSN++ [69] was adopted, which is based on UNet [114] and can capture both global context and local details. As shown in Figure 4.14, TEM is constructed with six ConvUnits consisting of a 1D convolution layer with batch normalisation and ReLU activation function. These modules have a hierarchical structure. The receptive field is widened through downsampling operations, enabling the learning of surrounding context information, while features that include both local detail and global context are created by fusing these features and high-resolution features through upsampling operations. Skip connections are used to

reduce the semantic gap between feature maps. The feature vectors f^{tem} for temporal evaluation are obtained by concatenating the three feature vectors obtained from the top three ConvUnits.

$$f^{tem} = concatenate(f_{0,0}, f_{0,1}, f_{0,2}), \tag{4.9}$$

where $f_{0,0}, f_{0,1}, f_{0,2}$ are obtained from the first, second, and third ConvUnits in the top layer, respectively. These feature vectors are concatenated to create the feature vectors f^{tem} for temporal evaluation. A 1D convolution layer followed by a sigmoid activation function is applied to f_{tem} to obtain the starting scores (p_s) and ending scores (p_e). As mentioned in BSN++, the ending boundary can be considered as a starting boundary when the video is played backwards. Hence, the reversed starting scores (p_s^r) and ending scores (p_e^r) are obtained by reversing the feature sequence, and these scores are used for temporal evaluation.

4.5 Model Training

In the proposed framework, both anchor-free and anchor-based methods are used complementarily. Action proposals are generated using both coarse prediction and refined prediction in an anchor-free manner and verify them using the boundary-based method. In the anchor-free part, the model is trained to learn regression offsets and action confidence scores. In the boundary-based part, it is trained to learn actionness scores and local boundary confidence scores of detected proposals. To jointly train models in both parts, a unified multi-task framework is used for optimisation.

The training details of the proposed framework are described in this section. After handling how to make training data, label assignment and loss functions are explained for the four modules of the proposed framework: coarse prediction module, refined prediction module, proposal evaluation module, and temporal evaluation module.

4.5.1 Training Data Construction

Given an untrimmed video V, feature F with a length of l_f can be extracted. The features F can be segmented into observation windows of length l_w with 50% overlap. The training set $\Phi = \{\phi_n\}_{n=1}^{N_w}$ is constructed using N_w observation windows that contains at least one ground truth.

4.5.2 Label Assignment

Given the ground-truth annotations, labels need to be created for training the model. Ground-truth annotations for TAL consists of $\Psi_g = \{t_g^s, t_g^e, l_g^c\}$, where t_g^s and t_g^e are the starting and ending boundaries of action instances, and l_g^c represents action labels. This information must be processed for each module: coarse prediction module, refined prediction module, proposal

evaluation module, and temporal evaluation module. The proposed framework calculates classification loss and regression loss for coarse prediction, as well as classification loss, regression loss, and boundary mean loss for refined prediction. Furthermore, actionness loss and regression loss are computed in the PEM, while boundary confidence loss is determined in the TEM. In this section, how to assign appropriate labels to each module using the ground-truth annotations when calculating each loss is explained.

Coarse Prediction Module: Labels must be assigned to each time location of the feature pyramid. Each layer is designed to detect targets of different scales. Let $\Gamma = \{\gamma_l = (s_{lo}^l, s_{hi}^l)\}_{l=0}^{N_l}$ be the target scales for each layer. A time location t_i^l is assigned to a ground truth ϕ_j when $t_{g,j}^s <= t_i^l <= t_{g,j}^e, s_{lo}^l <= len(\phi_j) <= s_{hi}^l)$ are satisfied, where $len(\phi_j)$ is the length of the action instance. In this way, labels for both regression and classification are assigned to the ground-truth instances at each location t_i^l. The classification label y_{cls}^C assigns 1 as a positive sample if there is a ground-truth instance assigned to t_i^l, and assigns 0 as a negative sample otherwise. For regression, offset labels are all positive numbers and can be assigned as follows.

$$d_{offset}^c = \{t_i^l - t_{g,j}^s, t_{g,j}^e - t_i^l\}. \tag{4.10}$$

Refined Prediction Module: Refined prediction also assigns a ground-truth instance to each time location t_i^l, similarly to coarse prediction. However, what sets refined prediction apart is that the regression branch of this module aims to predict the residual offsets, which are the differences between ground truths and the predicted proposals generated in coarse prediction. As a result, it is necessary to calculate based on offsets predicted in coarse prediction. three types of labels were created: the residual offset label, boundary mean label, and refined offset label. The residual offset labels represent the amount of change from the proposal boundaries detected in coarse prediction to ground truths. An exponential function is used at the end to ensure that the boundary offset always has a positive value. Therefore, if the same method is applied to the residual offsets, the residual offset labels are represented as a ratio to the offsets $\hat{d}_l^s, \hat{d}_l^e$ obtained from coarse prediction and it is expressed as follows:

$$\begin{aligned} \Delta d_g^s &= (t^l - t_g^s)/\hat{d}_l^s \\ \Delta d_g^e &= (t^l - t_g^e)/\hat{d}_l^e. \end{aligned} \tag{4.11}$$

The boundary mean labels are used for aligning the predicted boundaries with the real boundaries. This is based on the assumption that the real boundary will exist near the ground truth, even if the real boundary is misrepresented due to human subjective error. Therefore, the boundary mean labels, m_g^s and m_g^e, are assigned as follows:

$$\begin{aligned} m_g^s &= t_g^s - (t^l - \hat{d}_l^s) \\ m_g^e &= t_g^e - (t^l + \hat{d}_l^e). \end{aligned} \tag{4.12}$$

In other words, the difference between the boundaries obtained from coarse prediction and the ground truth is used. The refined offset labels are intended to constrain refined offsets to ground truths. Although this is an auxiliary means, and boundary refinement can be performed without it, prediction through learning does not output ground-truth values with 100% probability. Therefore, there is a possibility that the result after the refinement process will be worse than the result of coarse prediction. To prevent this, the refined offset labels were introduced, using the same values as the offset labels employed for the coarse prediction regression.

Proposal Evaluation Module (PEM): In the PEM, proposals are evaluated in an anchor-free manner, considering actionness and regression. For this purpose, tIoU was used. The tIoUs are calculated between all ground-truth instances ϕ_i in Φ and each proposal ω_i, with the maximum tIoU g_{tiou} assigned as a label for the proposals.

$$g_{tiou}(\omega_i) = max(\frac{\phi_i \cap \omega_i}{\phi_i \cup \omega_i}). \tag{4.13}$$

For classification, samples satisfying $g_{tiou} >= 0.9$ are assigned as positive samples. Regarding regression, the positive samples are divided into three types according to the tIoU value: $g_{tiou} > 0.7, 0.7 \geq g_{tiou} > 0.3$, and $0.3 \geq g_{tiou} > 0$. These cases are strong positive, intermediate positive, and weak positive. To learn with various positive samples, a 1:1:1 ratio is maintained.

Temporal Evaluation Module (TEM): In the TEM, boundary labels are needed to predict boundary probabilities. A boundary area is set as follows:

$$r_g^{bd} = (t_i - \alpha\eta/2, t_i + \alpha\eta/2), \tag{4.14}$$

where η is the gap between consecutive time locations in the base feature sequence, and α is the width of the boundary area. Then, the overlap is computed between the boundary area (r_g^{bd}) and the area covered by each feature vector (r_f). Overlap is calculated as:

$$Overlap(o_g) = \frac{r_g^{bd} \cap r_f}{r_f}, \tag{4.15}$$

where r_f is the area covered by the feature vector. This overlap is used as labels for TEM.

4.5.3 Loss Functions

As explained earlier, the framework consists of four modules: coarse prediction module, refined prediction module, proposal evaluation module, and temporal evaluation module. The corresponding multi-task objective function can be defined as follows:

$$\mathcal{L} = \lambda_1\mathcal{L}_{coarse} + \lambda_2\mathcal{L}_{refine} + \lambda_3\mathcal{L}_{pem} + \lambda_4\mathcal{L}_{tem} + \lambda_5\mathcal{L}_2(\Theta), \tag{4.16}$$

where $\mathcal{L}_{coarse}$ and $\mathcal{L}_{refine}$ are the objective functions of the coarse prediction and refined prediction, and $\mathcal{L}_{pem}$ and $\mathcal{L}_{tem}$ are those of the PEM and TEM, while $\mathcal{L}_2(\Theta)$ is a regularisation term. $\lambda_i, i = 1...4$, are weight terms to ensure different modules are trained appropriately. The weight terms $\lambda_i, i = 1...5$, are set to $10, 10, 1, 1$, and 10^{-4}, respectively, which are determined empirically.

Coarse Prediction: The objective function of the coarse prediction is composed of the classification loss l^C_{cls} and the regression loss l^C_{loc}.

$$\mathcal{L}_{coarse} = l^C_{cls} + \lambda^C_{loc} l^C_{loc}, \tag{4.17}$$

where λ^C_{loc} is a weight term. l^C_{cls} is a binary focal loss between classification prediction $\hat{y}^C_{cls}$ and ground-truth label y^C_{cls}:

$$l^C_{cls}(\hat{y}^C_{cls,i}) = \frac{1}{N_C} \sum_i l_{focal}(\hat{y}^C_{cls,i}, y^C_{cls,i}), \tag{4.18}$$

where N_C represents the number of features in the feature pyramid. l^C_{loc} is tIoU loss between predicted offsets $\hat{d}_i = (\hat{d}_s, \hat{d}_e)$ and ground-truth offset $d_g = (d_s, d_e)$:

$$l^C_{loc}(\hat{d}_i) = \frac{1}{N^C_{pos}} \sum_i \delta(y_i == 1)(1 - \frac{\hat{d}_i \cap d_g}{\hat{d}_i \cup d^C}), \tag{4.19}$$

where N^C_{pos} is the number of positive samples, and $\delta()$ is the Kronecker delta function.

Refined Prediction: The objective function of refined prediction consists of three types of loss functions. These functions can be represented as follows:

$$\mathcal{L}_{refine} = l^R_{cls} + \lambda^R_{loc} l^R_{loc} + \lambda_\Delta l_\Delta + \lambda_\mu l_\mu, \tag{4.20}$$

where λ^R_{loc} and λ_μ are weight terms, and $l^R_{cls}, l^R_{loc}, l_\Delta$, and l_μ are classification loss, refined regression loss, residual offset loss, and mean loss, respectively. l^R_{cls} is a binary focal loss and l^R_{reg} is tIoU loss in the same way as that of the coarse prediction.

$$\begin{aligned} l^R_{cls}(\hat{y}^R_{cls,i}) &= \frac{1}{N_R} \sum_i l_{focal}(\hat{y}^R_{cls,i}, y^R_{cls,i}) \\ l^R_{reg}(\hat{d}^R) &= \frac{1}{N^R_{pos}} \sum_i \delta(y_i == 1)(1 - \frac{\hat{d}^R_i \cap d_g}{\hat{d}_i \cup d^C}), \end{aligned} \tag{4.21}$$

where N_R and N^R_{pos} represent the number of samples in the feature pyramid and positive samples, respectively. l_Δ, and l_μ are L2 loss for the residual offsets and boundary mean.

Proposal Evaluation Module (PEM): The objective function of PEM consists of classification loss and regression loss, represented as follows:

$$\mathcal{L}_{pem} = l^{pem}_{cls} + \lambda^{pem}_{reg} l^{pem}_{reg}. \tag{4.22}$$

Similar to coarse prediction and refined prediction, l_{cls}^{pem} is the binary focal loss between predicted proposal (ω_C, ω_R) and ground-truth Φ, while l_{reg}^{pem} is the L2 loss. For the training phase, the outputs from both coarse prediction and refined prediction were used for data augmentation purposes.

Temporal Evaluation Module (TEM): The objective function of TEM consists of five terms:

$$L_{tem} = \underbrace{\overrightarrow{l^s} + \overrightarrow{l^e}}_{\text{forward}} + \underbrace{\overleftarrow{l^s} + \overleftarrow{l^e}}_{\text{backward}} + \| f_f^{tem} - f_b^{tem} \|, \tag{4.23}$$

where l_s and l_e are a weighted binary cross-entropy losses, and f_f^{tem} and f_b^{tem} are intermediate features from TEM. Following the approach of BSN++[69], the losses in both forward and backward passes is calculated, and mean-squared loss is applied to the intermediate features.

4.6 Inference

During inference, the framework generates classification scores $\hat{y}_{cls}^C$ and boundary offsets $\hat{d}^C = (\hat{d}_s^C, \hat{d}_e^C)$ in the coarse prediction, as well as classification scores $\hat{y}_{cls}^R$ and residual offsets $\Delta\hat{d}^R = (\Delta\hat{d}_s^R, \Delta\hat{d}_e^R)$ in the refined prediction. PEM generates classification scores p_{cls}^{pem} and regression scores p_{reg}^{pem}, while TEM generates starting scores p_s^{tem} and ending scores p_e^{tem} at each time location t_i. After generating proposals, redundant proposals need to be eliminated or reduced.

4.6.1 Proposal Generation and Score Fusion

Using the scores and offsets obtained from four modules (coarse prediction, refined prediction, proposal evaluation and temporal evaluation), action boundaries $\hat{d}$ are generated as described in Eq.4.24, and ranking scores ξ for each proposal are calculated as in Eq.4.25. Finally, a set of predicted proposals $\hat{\omega}' = \{\hat{t}_i^s, \hat{t}_e, \hat{\xi}\}_{i=1}^{N_p}$ are obtained, where N_p is the number of predicted proposals.

$$\begin{aligned} \hat{d} &= (\hat{t}_s, \hat{t}_e) \\ \hat{t}_s &= t_i - \hat{d}_s^C \Delta\hat{d}_s^R \\ \hat{t}_e &= t_i + \hat{d}_e^C \Delta\hat{d}_e^R. \end{aligned} \tag{4.24}$$

$$
\begin{aligned}
\hat{\xi} &= \hat{y}_{cls}\sqrt{p^{pem}_{cls}\, p^{pem}_{reg}}\sqrt{\hat{p}^{tem}_{s}\hat{p}^{tem}_{e}} \\
\hat{y}_{cls} &= \frac{1}{2}(\hat{y}^{C}_{cls} + \hat{y}^{R}_{cls}) \\
\hat{p}^{tem}_{s} &= (1-\hat{t}_s)p^{tem}_{s}(\hat{t}^{down}_{e}) + \hat{t}_s p^{tem}_{s}(\hat{t}^{up}_{s}) \\
\hat{p}^{tem}_{e} &= (1-\hat{t}_e)p^{tem}_{e}(\hat{t}^{down}_{e}) + \hat{t}_e p^{tem}_{e}(\hat{t}^{up}_{e}) \\
\hat{t}^{down}_{k} &= \lfloor \hat{t}_k \rfloor, k \in \{s, e\} \\
\hat{t}^{up}_{k} &= \lceil \hat{t}_k \rceil, k \in \{s, e\},
\end{aligned}
\tag{4.25}
$$

where $\hat{y}_{cls}$ represent the final classification confidence scores, and $\hat{p}^{tem}_{s}$ and $\hat{p}^{tem}_{e}$ are starting and ending scores at the predicted boundaries. $\hat{p}^{tem}_{s}$ and $\hat{p}^{tem}_{e}$ are obtained by linear interpolation because the predicted boundaries are not an integer, so p^{tem}_{s} and p^{tem}_{e} can not be accessed directly.

4.6.2 Post-processing

Predicted proposal $\hat{\omega}'$ may contain many redundant proposals. Redundant proposals refer to highly overlapped proposals that have very similar results. It is necessary to keep only the one with the highest score and suppress the rest. soft-NMS was applied, which is a method to suppress redundant proposals by decaying its scores. Then, the final proposal set $\hat{\omega} = \{\hat{t}^{s}_{i}, \hat{t}_e, \xi_i\}^{N_p}_{i=1}$ is obtained, where ξ_i is the decayed score of proposal $\hat{\omega}_i$.

4.7 Experimental Settings

In this section, details are provided for experiments using the proposed framework. The datasets, video representation, and training parameters will be stated.

4.7.1 Datasets

To verify the effectiveness of the proposed model, it was tested on two challenging datasets: THUMOS14 [11] and ActivityNet v1.3 [2]. The **THUMOS14** dataset consists of a validation set of 200 untrimmed videos and a testing set of 213 untrimmed videos, and provides temporal annotations for 20 action categories for these videos. These actions are taken from sports events. Since the officially provided training set is composed of trimmed video clips, previous studies trained their models on the validation set and tested them on the testing set. Following this protocol, the model was trained on the validation set and evaluated on the testing set. The **ActivityNet v1.3** dataset contains 10,024 untrimmed videos in the training set and 4,926 untrimmed videos in the validation set. Each set provides 15,410 action instances and 7,654 action instances as temporal annotations. Although 5,044 untrimmed videos are provided as a testing set, temporal annotations are not provided. As a result, previous studies trained

their models on the training set and tested them on the validation set. Experiments were also conducted following this protocol.

4.7.2 Video Representation

On THUMOS14, for feature extraction, the frame interval was set to 5 and feature vectors were extracted every 5 frames. Since 96 feature vectors were needed to create a feature pyramid, the sliding window size l_w was set to 96. On ActivityNet v1.3, as in ASDF [85], each video was sampled using different fps. In this way, each video was created as a video clip with only 768 frames.

For video representation, a two-stream network called the inflated 3D network (I3D) was used, which is pre-trained on a Kinetics dataset. PWC-Net [115] was used to obtain optical flow images. Random cropping and horizontal flipping were used for data augmentation in both datasets, and each image was cropped to 96×96. The I3D network takes 768 images as input and output 96 feature vectors at once.

4.7.3 Implementation Details

To train the proposed model, the Adam optimisation method was used. The batch size was set to 16, and the number of training epochs was 10. The learning rate was set to 10^{-3} for the first seven epochs and it was decayed to 10^{-4} for another three epochs. For soft-NMS, the tIoU threshold was set to 0.75 for both THUMOS14 and ActivityNet v1.3.

4.8 Result on Temporal Action Proposal Generation

The purpose of action proposal generation is to generate high-quality proposals with a high overlap with ground-truth instances. To evaluate this, AR was used, following the convention in the literature. Additionally, the effectiveness of the proposed model was evaluated through comparison with state-of-the-art methods. The model consists of four modules. Ablation studies were conducted to investigate the effectiveness of each module. By comparing the performance when each module was applied and when it was not, whether each module contributed to improved performance was accessed.

4.8.1 Evaluation Metric

In the literature, the performance of TAPG is evaluated by AR. For THUMOS14, AR given 50, 100, 200, and 1000 proposals was calculated. For ActivityNet v1.3, two kinds of evaluation metrics were used. First, 100 proposals were used for calculating AR. Second, the area under the AR vs. AN curve (AUC) was also used as an evaluation metric. AR was calculated under multiple tIoU thresholds. The tIoU thresholds were set to [0.5:0.05:1.0] for

THUMOS14 and [0.5:0.05:0.95] for ActivityNet v1.3, separately. Using these values, AUC was calculated.

4.8.2 Performance Comparison

The model was compared with others on the test set of THUMOS14 and the validation set of ActivityNet v1.3. The two datasets have different characteristics. THUMOS14 contains many action segments in a video clip, and the length of the action segments accounts for a relatively low ratio compared to the length of the video clip. On the other hand, ActivityNet includes long action instances that occupy a large percentage of video clips. Therefore, different tIoU criteria were applied. Each model has reported performance using different feature representation. This is because performance varies based on the type of feature representation, so the authors used different feature representation method to show the best performance. Therefore, in Table 4.1 and Table 4.2, results were collected from original papers because it would be the best.

Table 4.1 lists the results of proposal generation methods on the test set of THUMOS14 dataset, including 13 anchor-based (boundary-based) methods and one anchor-free method, along with the result of the proposed model (CAFN). In TAPG, only one proposal generation method in an anchor-free way has been found. The proposed model achieved comparable performance with the state-of-the-art. In general, boundary-based methods are superior to anchor-free methods in TAP. Currently, AOE [116] is the best method, which makes use of additional linguistic information. Although the proposed model did not outperform this method, it is meaningful that a comparable result was achieved in an anchor-free way.

Table 4.1. Comparison with other state-of-the-art proposal generation methods on THUMOS14 in terms of AR@AN. (*) means the anchor-free method. (unit: %)

Method	Feature	@50	@100	@200	@500	@1000	Average
SCNN[78]	C3D	17.22	26.17	37.01	51.57	58.20	38.03
SST[59]	C3D	19.90	28.36	37.90	51.58	60.27	39.60
TURN[60]	C3D	19.63	27.96	38.34	53.52	60.75	40.04
MGG[64]	C3D	29.11	36.31	44.32	54.95	60.98	53.55
AOE[116]	C3D	**44.56**	50.26	**57.30**	**64.32**	**68.19**	**56.93**
TAG[55]	2-Stream (TSN)	18.55	29.00	39.61	-	-	-
BSN[67]	2-Stream (TSN)	37.46	46.06	53.21	60.64	64.52	52.38
BMN[68]	2-Stream (TSN)	39.36	47.72	54.70	62.07	65.49	53.87
DBG[77]	2-Stream (TSN)	37.32	46.67	54.50	62.21	66.40	53.42
SRG[84]	2-Stream (TSN)	42.19	49.72	56.71	63.78	-	-
BSN++[69]	2-Stream (TSN)	42.44	49.84	57.61	65.17	66.83	56.38
TCANet[88]	2-Stream (TSN)	42.05	**50.48**	57.13	63.61	66.88	56.03
MR[117]	2-Stream (I3D)	44.23	50.67	55.74	-	-	-
RTD-Net[66](*)	2-Stream (TSN+I3D)	41.52	49.32	56.41	62.91	-	-
CAFN (*)	2-Stream (I3D)	41.61	49.96	56.57	63.59	67.03	55.75

Table 4.2 lists the results obtained from the validation set of the ActivityNet v1.3 dataset. In this dataset, it can be clearly seen that the boundary-based methods show superior perfor-

mance. The performance of RTD-Net and the proposed model (CAFN), both performed in the anchor-free method, did not outperform other methods performed in the boundary-based method.

Table 4.2. Comparison with other state-of-the-art proposal generation methods on ActivityNet v1.3 in terms of AR@AN. (*) means the anchor-free method. (unit: %)

Method	Feature	AR@100(val)	AUC(val)
BSN[67]	2-Stream (TSN)	74.16	66.17
SRG[84]	2-Stream (TSN)	74.65	66.06
BMN[68]	2-Stream (TSN)	75.01	67.10
DBG[77]	2-Stream (TSN)	76.65	68.23
BSN++[69]	2-Stream (TSN)	76.52	68.26
MGG[64]	2-Stream (I3D)	74.54	66.43
RTD-Net[66](*)	2-Stream (I3D)	73.21	65.78
AOE[116]	C3D	**77.67**	**69.71**
CAFN (*)	2-Stream (I3D)	70.95	64.28

Table 4.2 indeed shows no advantage of the anchor-free method compared to the boundary-based method in terms of performance. However, by considering the GPU during training, as shown in Table 4.3, a clear advantage for anchor-free methods can be seen. Boundary-based methods require a significant amount of memory because they evaluate all possible temporal pairs for dense search, whereas anchor-free methods require less memory due to sparse search. Therefore, there is a trade-off between performance and memory efficiency, and research using this point like the proposed model will be justified.

Table 4.3. GPU memory usage (unit: MiB).

	Boundary-based			Anchor-free	
Method	BMN	DBG	BSN++	RTD-net	CAFN
GPU Memory	13,055	8,299	36,695	1,716	4,549

4.8.3 Ablation Study

In this section, the outcomes from ablation studies on various components are presented so as to comprehensively evaluate the proposed model. Table 4.4 presents six versions of the model that are tested to assess the effectiveness of each module. Model A represents the case where only coarse prediction is performed. Models B, C, and D are designed to evaluate refined prediction, TEM, and PEM, respectively. Model E tests the effectiveness of boundary refinement when both PEM and TEM are employed. Model F, on the other hand, incorporates all modules. By examining the AR, it can be observed that when all modules are utilised, the performance surpasses that of the other variant models. However, it is worth noting that the performance difference between models D, E, and F gradually increases, but the difference is not significant. This observation suggests that further research is necessary to better understand the implications of these findings.

In the refined prediction stage, three loss functions are used for regression, namely boundary mean loss, refined offset loss, and residual offset loss. The boundary mean loss is used

Table 4.4. Ablation study on different versions of the proposed model (CAFN).

Model	@50	@100	@200	@500	@1000	Average
A. coarse	36.54	43.55	49.71	57.22	62.22	49.85
B. coarse + refine	38.00	45.71	52.44	59.99	64.59	52.15
C. coarse + refine + PEM	41.31	49.17	55.36	61.52	65.08	54.49
D. coarse + refine + TEM	**42.22**	49.84	56.13	62.18	65.99	55.27
E. coarse + PEM + TEM	41.47	**50.12**	56.53	63.12	66.48	55.54
F. All	41.61	49.96	**56.57**	**63.59**	**67.03**	**55.75**

to learn the relationship between the coarse predicted proposals and the actual boundaries, while the refined offset loss is for the refined boundaries. The residual offset loss is for learning the residual offsets. It may seem unnecessary to use all three types of loss functions, so an ablation study is conducted to investigate their effectiveness. The results are shown in Table 4.5. It can be concluded that the refined offset loss directly contributes to the performance improvement. On the other hand, the boundary mean loss and residual offset loss do not show a significant contribution to the performance improvement in terms of AR. However, when all loss functions are used, AR@50 and AR@100 are the largest, indicating that the two loss functions contribute to proposal ranking.

Table 4.5. Ablation study on boundary mean loss, refined offset loss, and residual offset loss.

Model	@50	@100	@200	@500	@1000	Average
w/o boundary mean loss	41.19	49.39	55.97	63.04	66.81	55.28
w/o refined offset loss	41.04	48.68	55.66	62.32	66.18	54.78
w/o residual offset loss	41.53	49.77	**56.70**	**63.65**	**67.16**	**55.76**
All	**41.61**	**49.96**	56.57	63.59	67.03	55.75

4.9 Result on Temporal Action Localisation

For completing the process of TAL, both proposal generation and action classification are required. Therefore, an action classification step is necessary to complete TAL. In many studies on TAPG for pipeline TAL (especially in two-stage TAL), TAL results are reported using an external classifier. Following the convention in the literature, an external classifier was used.

UNet, SCNN-cls, and P-GCN are widely used as external classifiers. The P-GCN classifier was used for action classification. P-GCN is a proposal-level classifier that recognises actions by taking generated proposals as input. Experiment were conducted on the THUMOS14 dataset.

4.9.1 Evaluation Metric

The THUMOS14 dataset is derived from the dataset used in the THUMOS Challenge, which provides a conventional metric for performance evaluation. Previous studies using this dataset

have reported performance using this convention. This convention was adopted and performance using mAP at different tIoU thresholds was reported. Specifically, AP was calculated to report performance per action category, and AP values at tIoU thresholds of 0.3, 0.4, 0.5, 0.6, and 0.7 were used to report overall performance on TAL by calculating mAP.

4.9.2 Performance Comparison

The results of the proposed model (CAFN) was compared with several state-of-the-art methods, including not only the pipeline methods (multi-stage methods) but also end-to-end methods (one-stage methods). Table 4.6 shows the best scores, with the highest scores shown in bold. Among the pipeline methods, when comparing the results of CAFN with ContextLoc, which shows the best performance (mAP: 50.9), the proposed model (CAFN) showed comparable performance with mAP of 49.41. In particular, in terms of the anchor-free method, the proposed model outperformed RTD-Net, which is the only method performed in an anchor-free way among the pipeline methods.

Including the end-to-end methods in the comparison, it can be seen that pipeline methods do not perform as well as the latest end-to-end methods. In particular, recent studies tend to focus on end-to-end models, and significant advances have been made using this methodology. Additionally, considering the results in Section 3.2, the use of class labels may not be helpful when only performing proposal generation, but it is beneficial when detecting and recognising actions simultaneously.

4.10 Investigation on Challenges – Comparative Study

This section reports on a number of comparative studies, the same as those conducted in Chapter 3. However, since the relationship between detection and recognition required two different sets with the same number of proposals, and since TAPG was performed for the pipeline method, experiment using class labels was not able to be performed. Therefore, experiments for the remaining three cases out of the five investigations were conducted.

4.10.1 Experiment on Multi-modal Data

To perform a multi-modal test with the proposed model, an experiment was conducted using two types of data, RGB and Flow. Table 4.7 shows the results of the experiment. Similar to the results obtained in Chapter 3, the results obtained when using only the Flow channel are better than when using only the RGB channel. The best results are obtained when both channels are used. The RGB channel is suitable for expressing the appearance of an action, while the Flow channel is suitable for expressing motion information. According to the results, it seems that motion information indicating how the motion has changed is more important than the appearance of the action. However, since each channel has missing information, the best results are obtained when both channels are used, indicating that they are complementary.

Table 4.6. Performance comparison on TAL. (*) takes the anchor-free approach in the pipeline method.

Type	Model	Feature	Classifier	0.3	0.4	0.5	0.6	0.7	mAP
End-to-End method (One stage)	A^2Net[118]	I3D	-	58.6	54.1	45.5	32.5	17.2	41.6
	GTAN[119]	P3D	-	57.8	47.2	38.8	—	—	—
	PBRNet[86]	I3D	-	58.5	54.6	51.3	41.8	29.5	—
	AFSD[85]	I3D	-	67.3	62.4	55.5	43.7	31.1	52.0
	TadTR[108]	I3D	-	74.8	69.1	60.1	46.6	32.8	56.7
	ActionFormer[109]	I3D	-	82.1	77.8	71.0	59.4	43.9	66.8
	TriDet[120]	I3D	-	**83.6**	**80.1**	**72.9**	**62.4**	**47.4**	**69.3**
Pipeline method (Multi-stage)	BSN[67]	TSN	UNet	53.5	45.0	36.9	28.4	20.0	36.76
	BMN[68]	TSN	UNet	56.0	47.4	38.8	29.7	20.5	38.5
	MGG[64]	TSN	UNet	53.9	46.8	37.4	29.5	21.3	37.78
	G-TAD[71]	TSN	UNet	54.5	47.6	40.3	30.8	23.4	39.3
	DBG[77]	TSN	UNet	57.8	49.4	39.8	30.2	21.7	39.8
	RTD-Net[66]	I3D	UNet	58.5	53.1	45.1	36.4	25.0	43.62
	ContextLoc[121]	I3D	-	**68.3**	**63.8**	**54.3**	**41.8**	**26.2**	**50.9**
	BSN[67]	I3D	P-GCN	63.6	57.8	49.1	—	—	—
	MR[117]	TSN	P-GCN	66.29	60.99	50.10	—	—	—
	G-TAD[71]	TSN	P-GCN	66.4	60.4	51.6	37.6	22.9	47.78
	AOE-Net[116]	C3D	P-GCN	67.1	60.6	50.9	37.4	23.5	47.89
	RTD-Net[66]*	I3D	P-GCN	**68.3**	62.3	51.9	38.8	23.7	49.0
	Ours	I3D	P-GCN	68.24	63.10	53.81	38.56	23.32	49.41

One thing that is worth noting is that the results of the proposed model (CAFN) outperformed those of BSN, BMN, and BSN++ in Table 3.6. When not only using both modalities but also using either RGB or Flow independently, the proposed model outperformed the state-of-the-art boundary-based methods.

Table 4.7. Multi-modal test on THUMOS14.

Channel	@50	@100	@200	@500	@1000	Average
RGB	38.73	46.58	52.90	60.24	64.00	52.49
Flow	40.02	48.02	55.28	62.45	66.24	54.40
Both	**41.61**	**49.96**	**56.57**	**63.59**	**67.03**	**55.75**

4.10.2 Experiment on Discontinuous Actions

As mentioned in Chapter 3, it is challenging to create large-scale datasets manually. Therefore, videos are collected from the internet, movies, and TV programmes to create a video dataset. However, in these videos, the continuity of motion may be disrupted, or unrelated scenes may be inserted due to post-production work such as editing. This sub-section reports on experiments that were conducted using manually selected discontinuous motions in the THUMOS14 dataset. Table 4.8 shows the results. In contrast to the general multi-modal test results, the average performance of using only the RGB channel (54.60) is better than that of using only the Flow channel (52.45). This indicates that the Flow channel is more sensitive to disrupted motion continuity. Nevertheless, using both channels can improve performance even in discontinuous actions.

Compared to Table 3.7, the proposed model, CAFN, outperformed other models (BSN, BMN, BSN++) when using either RGB or both modalities. However, when using only Flow data, some other models outperformed the proposed one. It is worth noting that the Flow data might have been affected by the discontinuity of actions, potentially making it unreliable. Therefore, it can be concluded that the proposed model is more robust to discontinuous actions than the other models.

Table 4.8. Performance of discontinuous actions on THUMOS14.

Channel	@50	@100	@200	@500	@1000	Average
RGB	39.84	48.68	57.06	62.87	64.57	54.60
Flow	36.86	46.16	53.83	61.39	64.02	52.45
Both	**41.81**	**50.76**	**57.72**	**64.73**	**67.52**	**56.51**

4.10.3 Cross-corpora Test

In this sub-section, the results from a cross-corpora test where the proposed framework was used to train a model on one dataset and apply it on the other. Specifically, a model was trained using the training set of ActivityNet v1.3 and tested on the testing set of THUMOS14, and vice versa. The results are shown in Tables 4.9 and 4.10. Similar to the results obtained in

Chapter 3, it can be observed that the performance of cross-corpora testing is worse than that of normal testing using the same dataset. This indicates that the existing benchmark datasets are biased towards specific purposes. Therefore, it is important to consider the distribution of data in the dataset to ensure generalisation when constructing training data.

Table 4.9. Performance of Cross-corpora Test on THUMOS14.

Type	@50	@100	@200	@500	@1000	Average
Normal	**41.81**	**50.76**	**57.72**	**64.73**	**67.52**	**56.51**
Cross	6.26	13.01	24.49	43.04	53.88	28.14

Table 4.10. Performance of Cross-corpora Test on ActivityNet v1.3.

Type	@1	@5	@10	@100	AR@AN
Normal	**33.84**	**47.23**	**54.89**	**70.95**	**64.28**
Cross	9.94	29.21	37.51	56.05	48.63

4.11 Summary

In this chapter, the proposed TAPG model (CAFN) and boundary refinement module have been explained. TAL was performed using a pipeline method. The pipeline method (especially, two-stage approach) consists of two steps: TAPG and action classification. To address TAPG, the complementary anchor-free network (CAFN) was proposed. This model takes advantage of the strengths and compensates for the weaknesses of anchor-based and anchor-free methods. Anchor-based methods rely on detecting actions using a predefined set of anchors. The performance of this method depends on the anchor configuration, and increasing the number of anchors to improve performance leads to inefficient computation and requires a large amount of memory. Additionally, since predefined anchors are used, the boundary of the detected action is not accurate, but dense evaluation is possible by increasing the number of anchors, making it less likely to miss the target. On the other hand, the anchor-free method requires less memory as it detects targets without anchors, resulting in good computational efficiency. However, the feature information used for prediction is incomplete. To address this, a complementary method was proposed so as to leverage the strengths of each approach while compensating for their weaknesses in a mutually complementary way. First, action proposals were generated using an anchor-free method and it was used as dynamically generated anchors. This reduced the number of repetitive computations and verified the predicted proposals using complete information.

This chapter mainly addressed the second and third research question in Section 1.2. To answer the second research question, "How can various length of human actions be dealt with?", this model was built upon the Feature Pyramid Network (FPN). Each layer of FPN is designed to detect actions at different scales. In this way, multi-scale actions were modelled and detected. To enable feature vectors to exchange information across different layers, a pyramidal attention mechanism was applied. Action instances were detected by predicting the left and right offsets. These instances were then converted into a single vector with the

same shape through sparse sampling, which was subsequently used for classification.

In addition, as the answer of the third research question, " How can the boundary of human actions be detected?", a boundary refinement module was proposed to improve the accuracy of the proposals generated by the anchor-free method. The boundaries of actions were further refined by constructing boundary features and predicting residual offsets. These boundary features were constructed using Gaussian sampling, and their parameters were predicted through mean error learning. As a result, the proposed model can predict more accurate boundaries, particularly useful for handling inconsistent annotations.

The validity of the proposed model was verified by comparing its performance with state-of-the-art methods and each module included in the model was evaluated through ablation studies. Additionally, the performance evaluation of the proposed model was done on the challenging cases introduced in Chapter 3. Most of the models that demonstrate excellent performance in TAPG rely on dense search, such as the boundary-based method. However, obtaining comparative results using an anchor-free approach is significant.

Chapter 5

Conclusions and Future Work

In this thesis, TAL has been addressed, including the handling of various action durations, multi-scale representation, boundary refinement, post-processing, and more. Mainly TAPG, the first step in the two-stage approach, has been addressed using the pipeline approach. To begin, challenging factors have been set in TAL and experiments have been conducted to investigate how these factors affect the results, providing insights for future research directions. Additionally, a complementary anchor-free network (CAFN) and a boundary refinement module have been proposed. CAFN utilises both anchor-based and anchor-free methods and the boundary refinement module improves the accuracy of boundaries through Gaussian sampling and mean error learning. These provide a reasonable answer to the research questions posed at the beginning of the thesis. Overall, this work has allowed to draw conclusions about pipeline TAL and suggest potential research directions for future work.

5.1 Summary

In the following subsections, the work presented in Chapters 3 and 4 will be summarised.

5.1.1 Investigation on Challenges in Pipeline TAL - Comparative Study

In Chapter 3, comparative studies were conducted based on five investigations. While TAL is a promising field with new papers being published every year, it still faces many challenges and problems compared to similar fields such as object detection.

First, many researchers who adopt the pipeline strategy have made efforts to improve the performance of TAPG. They have done so because the detection-then-recognition approach has shown excellent performance in object detection, and they assumed the same would hold true for TAL. An experiment was conducted to test if this assumption holds true. In general, the assumption holds true in TAL, but some exceptional cases were found and to know why it happened, the results were investigated. It was found that TAL performance in a pipeline approach is affected not only by the proposal generation result but also by the number of true positive samples in the proposal sets.

Secondly, an experiment was conducted to investigate the effect of using action class labels, which is the main difference between the proposal generation of two-stage methods

(pipeline) and one-stage methods (end-to-end). Existing one-stage methods were modified to set up two different experimental environment: proposal generation with binary classification, and one-stage method with multi-label classification. Through experiments conducted, it was found that action class labels are useful information for TAL performance improvement because this helps classify actions to multiple labels while boundary detection performance was not as good as proposal generation performance. It may be because of the complexity of whole TAL task. Proposal generation task classifies temporal segments to two classes (action and background) while one-stage method classifies the segments to multiple classes (ActivityNet: 200, THUMOS14: 20).

Thirdly, as the size of the dataset grows, a large amount of video clips collected from the internet, films, and TV programmes are included in the dataset, and the continuity of action in these data is damaged due to editing or occurrence of unexpected situations. In this respect, an experiment was conducted to determine whether the anchor-based method and anchor-free method used in TAL can operate robustly in this situation. Discontinuous actions were collected manually from THUMOS14 dataset, and a trained model was evaluated using the discontinuous actions. From the experiment, it is found that anchor-free methods are more robust to discontinuous actions than anchor-based (boundary-based) methods.

Fourth, in many studies of TAL, RGB channels and Flow channels are used as multi-modal data. The effect of these two types of data on performance was investigated. Experiments were conducted in three ways: using only RGB, using only Flow, and using both data. It was found that the Flow channel is more effective in detecting actions than the RGB channel. However, since each channel includes different information, using both types of channels is superior to using only one type of information. An experiment was also conducted following the third investigation with multi-modal data. Unlike using all action instances, using only the RGB channel was better than using only the Flow channel in the case of discontinuous actions. This may be because the Flow channel is better suited to representing action continuity, which is disrupted in discontinuous actions. Nevertheless, using both types of channels is still beneficial for improving performance in this situation.

Lastly, a cross-corpora test was conducted from a generalisation point of view. A model trained on different datasets was used. Through this experiment, it was found that the datasets used for benchmarking are biased. This bias may be related to difficulties in collecting data of desired classes or labelling collected videos. Thus, it is essential to collect unbiased data to improve the general performance of TAL models. While it is possible to target only actions in specific fields, such as sports, appropriate data collection is crucial to creating an algorithm that is helpful in daily life.

5.1.2 Anchor-free Pipeline Temporal Action Localisation

A Complementary Anchor-Free Network (CAFN) and a boundary refinement module were proposed. The proposed approach was complementary in nature, combining the strengths of anchor-based and anchor-free methods to compensate for incomplete information. A feature

pyramid structure was utilised to handle multi-scale issues, incorporating pyramidal attention to establish lateral connections between features of varying scales. This structure compensated for the disadvantage of using incomplete information in the anchor-free method. Boundary refinement was also performed to improve the accuracy of coarse predictions for more precise boundary prediction. This involved predicting a refined boundary by learning the error between the coarse prediction and the ground-truth labels. It was possible that the boundary obtained in the coarse prediction step might differ from the actual boundary, which could adversely affect the results when using these boundary features for further work. Thus, creating boundary features at locations close to the actual boundary by correcting errors was proposed to train the model with the correct boundary information, which was verified through experiments and ablation studies.

It is possible to perform TAPG using only the anchor-free method, but there are drawbacks. First, the feature vector used for prediction may not align well with the ground-truth action instances, leading to incomplete or background-heavy information. Additionally, the regression branch in the anchor-free method only outputs left and right offsets without confidence scores, making it difficult to measure the reliability of the results. To compensate for these shortcomings, a complementary approach was proposed to incorporates an anchor-based method. Rather than predefining anchors, proposals generated by the anchor-free method were used to reduce repetitive calculations in the anchor-based method. A PEM enabled proposal-level evaluation, and the resulting predictions were verified by a TEM that outputs starting and ending probabilities. These scores served as confidence scores for the proposal boundaries. By combining the strengths of both methods, their individual disadvantages was overcome, as demonstrated by the ablation study results.

The anchor-free method is superior to the anchor-based method in terms of both space complexity and time complexity. Additionally, the proposed model is meaningful as it achieves comparable results to the anchor-based method using the anchor-free method. However, the approach has limitations as a pipeline strategy was used. Recent research in TAL tends to use one-stage methods (end-to-end methods), and results within the past two years have shown that one-stage methods outperform multi-stage methods. This may be because multi-stage methods cannot utilise label information from previous steps, even though TAPG performs better without labels. Nonetheless, there is value in TAPG itself, as there are situations where actions must be detected without labels, such as in the detection of unspecified actions. In such cases, class-independent detection may be more useful than classifying actions into predefined classes.

5.2 Conclusions

In Section 1.2, three research questions and the corresponding subsidiary questions have been asked, these have been answered through this thesis. This section provides overall answer of the questions.

5.2.1 Investigation on Challenges in Pipeline TAL

The first question was about the factors affecting TAL performance, and five investigations have been done.

The first investigation was about the relationship between detection and recognition performance, and from which, two subsidiary questions were arisen: "When the performance of proposal generation improves, will the performance of recognition also improve?" and "Is there any relationship between the performance of proposal generation and that of recognition?". As shown in Tables 3.1 and 3.2, better recognition results tend to be obtained from better proposal generation models in general. However, some exceptional cases were noticed such as DBG(TSN) and BSN++(TSN). In TAL, the false positive rate is relatively high due to the ambiguity of action boundary, which can have a significant impact on the action classification results. As demonstrated in Section 3.1, it can be concluded that a higher true positive rate in TAPG leads to better results in subsequent action classification.

The second investigation was to know whether there is any advantages of using action labels in terms of boundary detection. Based on the results presented in Section 3.2, it was observed that proposal generation performance is better when actions are detected in a class-independent way without using class labels. However, most of the recent research in the TAL field has been conducted using one-stage methods, which have been shown to outperform two-stage methods. Therefore, it is concluded that labels play a more critical role in action classification than boundary detection.

The third investigation was concerning the robustness to discontinuous actions between anchor-based (boundary-base) methods and anchor-free methods and its reason. Based on the results obtained in Section 3.3, it can be said that the anchor-free method is more robust in this situation. There could be several reasons for this. For instance, the anchor-based method uses proposal-level representation to evaluate the units of temporal segments. Therefore, when undesirable scenes such as abrupt viewpoint changes are included in the segments, this method may not avoid such situations. On the other hand, in the anchor-free method, the representation used for offset prediction is likely to include only partial information of the action. Consequently, there is a possibility that unwanted scenes can be avoided. For this reason, it can be concluded that the anchor-free method is more advantageous for discontinuous actions.

The fourth investigation was to know whether Flow data always have a positive effect on performance. As shown Table 3.6, Flow data is helpful in improving TAPG performance in general. However, from Tables 3.7 and 4.7, it can negatively affect performance when the continuity of actions is disrupted.

The fifth investigation was a cross-corpora test using two different datasets (THUMOS14 and ActivityNet) to know how models trained on different datasets perform on a dataset that was not used for training. As shown in Tables 3.8 and 3.9, the performance was deteriorated on a dataset that was not used for training. The performance was varied according to the size and classes of the dataset. ActivityNet dtaset is large and the samples were collected from

200 actions while THUMOS14 dataset is relatively small and it has only 20 classes. It can be concluded that dataset should be large and unbiased for a general use.

5.2.2 Anchor-free Pipeline Temporal Action Localisation

In this sub-section, the answer of the research questions 2 and 3 are summarised with conclusion. The questions were as follows:

- How can various length of human actions be dealt with?
    1. How to model multi-scale actions?
    2. How to convert actions of different lengths into input vectors to represent them with the same shape?
- How can the boundary of human actions be detected?
    1. How to model the boundaries of actions?
    2. How to deal with the inconsistent boundaries of action determined by people's subjective judgements?

In the proposed model (CAFN), the Feature Pyramid Network (FPN) was used to handle various lengths of actions. Instead of using the pure FPN, a pyramidal attention module was applied, allowing features to exchange information. Each layer of FPN was designed to detect a different scale of targets, enabling the detection of multi-scale actions. Generated proposals were converted into vectors with the same shape using sparse sampling. This was done in three areas: the starting area, internal area, and ending area. A predefined number of vectors were sampled in each area, and they were then concatenated to construct proposal-level feature vectors.

Action boundaries were first generated via offset prediction. Then, these boundaries were verified by the Temporal Evaluation Module (TEM), which predicts boundary scores. The feature vectors of the boundaries were constructed, including nearby information, as the boundaries were ambiguous and of very short duration. Therefore, investigating the neighbouring context was necessary. To give more weight to the area close to boundaries, Gaussian sampling was used. Additionally, to improve robustness against inconsistent annotations, the mean of the Gaussian function was determined through learning, ensuring that the sampling was done near generally accepted boundaries, regardless of inconsistent annotations.

5.3 Future Work

Despite the achievements that have been made, the work of this thesis has several limitations. In this regard, potential research directions for future work can be considered.

5.3.1 Fine-tuned Backbone

In Chapter 3 and Section 4.2, videos were converted to feature vectors using a pre-trained I3D model from the Kinetics dataset for video encoding. While state-of-the-art methods have demonstrated sufficient generalisation ability using pre-trained models, as seen in the cross-corpora test, models trained on datasets other than the target dataset may not have the best performance. Therefore, fine-tuning the model used for feature extraction on the target dataset can improve performance. Additionally, the features were primarily extracted using models designed for action recognition tasks. As these models use trimmed video clips as training data, they do not consider the temporal information about the boundary or context. As pointed out in TSP [54], further performance improvement can be achieved by fine-tuning the backbone model for feature extraction to include temporal information on the boundary.

5.3.2 Alternative Video Representation

Experiments were conducted using two-stream networks, specifically the Temporally Segmented Network (TSN) and the inflated 3D network (I3D). However, there are other backbone networks that can be utilised for various feature extractions. For example, the SlowFast network [6] is a two-stream network with different frame rates, and the Pseudo-3D Residual Network (P3D) [42] is a spatio-temporal representation that improves the computational efficiency of 3D CNNs. Furthermore, many models have been studied and published for action recognition tasks, which can be used as substitutes for TSN and I3D. Since different models contain different information, they can provide inspiration for a video representation that is advantageous to TAL. Additionally, recently proposed models have shown improved recognition performance, which will ultimately contribute to improving the performance of TAL.

5.3.3 More Effective Feature Manipulation for Anchor-free Method

Anchor-free methods utilise a pyramidal structure to process input features and handle multi-scale issues. To improve performance, FPN [82] inserts upward/downward lateral connections, while pyramidal attention proposed by Pyraformer [111] was used for the same purpose in the proposed model. Several other methods have been proposed to enhance the feature pyramid. DRFPN [122] highlights the issue of lateral connections composed of upsampling and convolution layers and aims to improve it, while ImFPN [123] fuses features of different scales using similarity. These various methods of feature fusion can improve performance in object detection, and similar approaches may also be useful in detecting actions.

5.3.4 One-stage Temporal Action Localisation

The pipeline approach was chosen because it is more scalable from an application perspective, as TAPG can be performed in a class-agnostic way. However, as confirmed through the

comparative study, this method may not be as effective in recognising the detected proposals, as class labels of actions are not considered. Recently, TAL tends to be studied as a one-stage method, which is simpler as the final result can be obtained with one model, and action detection and recognition are performed simultaneously. This allows for the creation of feature embeddings for detection and recognition by reflecting the labels of each class. This point is evident from the results of recently published one-stage methods. This study can be improved by adopting a one-stage method that can reflect the characteristics of each action in the detection of actions.

5.3.5 Applications Beyond Temporal Action Localisation

TAL plays a crucial role in developing application services that require video understanding. Although it is just one of the many tasks related to video understanding, detecting actions in a video has various applications in everyday life, especially concerning people. For instance, TAL can be used to identify abnormal behaviours in recorded video footage such as CCTV, or to search for a video that contains a specific behaviour, thereby improving productivity compared to manual inspection. Additionally, from a research perspective, TAL can be extended to semi-supervised learning, which utilises both labelled and unlabelled data, or few-shot TAL, which involves training with a limited number of samples.

References

[1] T.-Y. Lin *et al.*, *Microsoft coco: Common objects in context*, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312 (cited on p. 17).

[2] B. G. Fabian Caba Heilbron Victor Escorcia and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 961–970 (cited on pp. 17, 26, 53, 104).

[3] S. Vantigodi and R. Venkatesh Babu, "Real-time human action recognition from motion capture data," in *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 2013, pp. 1–4. DOI: 10.1109/NCVPRIPG.2013.6776204 (cited on p. 25).

[4] A. Shahroudy, J. Liu, T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," Jun. 2016. DOI: 10.1109/CVPR.2016.115 (cited on p. 25).

[5] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 568–576 (cited on p. 25).

[6] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019 (cited on pp. 25, 33, 119).

[7] A. Piergiovanni and M. Ryoo, "Temporal Gaussian mixture layer for videos," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 5152–5161. [Online]. Available: https://proceedings.mlr.press/v97/piergiovanni19a.html (cited on p. 25).

[8] S. N. Gowda, M. Rohrbach, and L. Sevilla-Lara, "Smart frame selection for action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1451–1459, May 2021. DOI: 10.1609/aaai.v35i2.16235. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16235 (cited on p. 25).

[9] Y. Wang *et al.*, "Internvideo: General video foundation models via generative and discriminative learning," *arXiv preprint arXiv:2212.03191*, 2022 (cited on p. 25).

[10] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action tubelet detector for spatio-temporal action localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4415–4423. DOI: 10.1109/ICCV.2017.472 (cited on p. 25).

[11] Y.-G. Jiang *et al.*, *THUMOS challenge: Action recognition with a large number of classes*, http://crcv.ucf.edu/THUMOS14/, 2014 (cited on pp. 26, 53, 104).

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324 (cited on p. 27).

[13] T. Li, Z. Sun, and X. Chen, "Group-skeleton-based human action recognition in complex events," *CoRR*, vol. abs/2011.13273, 2020. arXiv: 2011.13273. [Online]. Available: https://arxiv.org/abs/2011.13273 (cited on p. 27).

[14] Y. Zhang, P. Tokmakov, M. Hebert, and C. Schmid, "A study on action detection in the wild," *CoRR*, vol. abs/1904.12993, 2019. arXiv: 1904.12993. [Online]. Available: http://arxiv.org/abs/1904.12993 (cited on p. 27).

[15] K. Sozykin, S. Protasov, A. Khan, R. Hussain, and J. Lee, "Multi-label class-imbalanced action recognition in hockey videos via 3d convolutional neural networks," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2018, pp. 146–151. DOI: 10.1109/SNPD.2018.8441034 (cited on p. 27).

[16] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, Dec. 2007 (cited on p. 28).

[17] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011 (cited on p. 28).

[18] L. Smaira, J. Carreira, E. Noland, E. Clancy, A. Wu, and A. Zisserman, "A short note on the kinetics-700-2020 human action dataset," *CoRR*, vol. abs/2010.10864, 2020. arXiv: 2010.10864. [Online]. Available: https://arxiv.org/abs/2010.10864 (cited on pp. 28, 59).

[19] S. Abu-El-Haija *et al.*, "Youtube-8m: A large-scale video classification benchmark," *CoRR*, vol. abs/1609.08675, 2016. arXiv: 1609.08675. [Online]. Available: http://arxiv.org/abs/1609.08675 (cited on p. 28).

[20] C. Liu and P. C. Yuen, "Human action recognition using boosted eigenactions," *Image and Vision Computing*, vol. 28, no. 5, pp. 825–835, 2010, ISSN: 0262-8856 (cited on p. 29).

[21] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (cited on p. 29).

[22] H. Dong *et al.*, "Fashion editing with adversarial parsing learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8117–8125. DOI: 10.1109/CVPR42600.2020.00814 (cited on p. 29).

[23] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," ser. AAAI'18/IAAI'18/EAAI'18, New Orleans, Louisiana, USA: AAAI Press, 2018, ISBN: 978-1-57735-800-8 (cited on p. 29).

[24] D. Kotovenko, M. Wright, A. Heimbrecht, and B. Ommer, "Rethinking style transfer: From pixels to parameterized brushstrokes," *CVPR*, 2021 (cited on p. 29).

[25] Y. Zhang, G. Jia, L. Chen, M. Zhang, and J. Yong, "Self-paced video data augmentation by generative adversarial networks with insufficient samples," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20, Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 1652–1660, ISBN: 9781450379885. DOI: 10.1145/3394171.3414003. [Online]. Available: https://doi.org/10.1145/3394171.3414003 (cited on p. 29).

[26] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001. DOI: 10.1109/34.910878 (cited on p. 31).

[27] M.-C. Roh, H.-K. Shin, and S.-W. Lee, "View-independent human action recognition with volume motion template on single stereo camera," *Pattern Recognition Letters*, vol. 31, no. 7, pp. 639–647, 2010, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2009.11.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865509003286 (cited on p. 31).

[28] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558. DOI: 10.1109/ICCV.2013.441 (cited on p. 31).

[29] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417 (cited on p. 31).

[30] B. B. Amor, J. Su, and A. Srivastava, "Action recognition using rate-invariant analysis of skeletal shape trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 1–13, 2016. DOI: 10.1109/TPAMI.2015.2439257 (cited on p. 31).

[31] D. Vishwakarma and R. Kapoor, "Hybrid classifier based human activity recognition using the silhouette and cells," *Expert Systems with Applications*, vol. 42, no. 20, pp. 6957–6965, 2015, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.04.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741500281X (cited on p. 31).

[32] A. A. Chaaraoui, P. Climent-Pérez, and F. Flórez-Revuelta, "Silhouette-based human action recognition using sequences of key poses," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1799–1807, 2013, Smart Approaches for Human Action Recognition, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2013.01.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865513000342 (cited on p. 31).

[33] T. Guha and R. K. Ward, "Learning sparse representations for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1576–1588, 2012. DOI: 10.1109/TPAMI.2011.253 (cited on p. 31).

[34] H. Wang, C. Yuan, W. Hu, and C. Sun, "Supervised class-specific dictionary learning for sparse modeling in action recognition," *Pattern Recognition*, vol. 45, no. 11, pp. 3902–3911, 2012, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.04.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031200204X (cited on p. 31).

[35] J. Zheng, Z. Jiang, and R. Chellappa, "Cross-view action recognition via transferable dictionary learning," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2542–2556, 2016. DOI: 10.1109/TIP.2016.2548242 (cited on p. 31).

[36] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 158–170, 2016. DOI: 10.1109/TCYB.2015.2399172 (cited on p. 31).

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition.," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#SimonyanZ14a (cited on p. 32).

[38] C. "Szegedy *et al.*, "Going deeper with convolutions," 2015 (cited on p. 32).

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cited on p. 32).

[40] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15, USA: IEEE Computer Society, 2015, pp. 4489–4497, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.510. [Online]. Available: https://doi.org/10.1109/ICCV.2015.510 (cited on p. 32).

[41] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, "Mict: Mixed 3d/2d convolutional tube for human action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 449–458. DOI: 10.1109/CVPR.2018.00054 (cited on p. 32).

[42] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE, 2017, pp. 5534–5542, ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.590. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICCV.2017.590 (cited on pp. 32, 119).

[43] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459. DOI: 10.1109/CVPR.2018.00675 (cited on p. 32).

[44] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3034–3042. DOI: 10.1109/CVPR.2016.331 (cited on p. 32).

[45] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/

paper/2014/file/00ec53c4682d36f5c4359f4ae7bd7ba1-Paper.pdf (cited on p. 33).

[46] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 4724–4733 (cited on pp. 33, 58, 86).

[47] J. Jonides, "Further toward a model of the mind's eye's movement," *Bulletin of the Psychonomic Society*, vol. 21, no. 4, pp. 247–250, 1983. DOI: 10.3758/BF03334699 (cited on p. 33).

[48] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17, San Francisco, California, USA: AAAI Press, 2017, pp. 4263–4270 (cited on p. 33).

[49] L. Zhonghong, Y. Yang, S. Ying, S. Jialun, and W. Yukun, "Aarm: Action attention recalibration module for action recognition," in *Proceedings of The 12th Asian Conference on Machine Learning*, S. J. Pan and M. Sugiyama, Eds., ser. Proceedings of Machine Learning Research, vol. 129, PMLR, 18–20 Nov 2020, pp. 97–112. [Online]. Available: https://proceedings.mlr.press/v129/zhonghong20a.html (cited on p. 33).

[50] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803. DOI: 10.1109/CVPR.2018.00813 (cited on p. 34).

[51] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cited on p. 34).

[52] R. Girdhar, J. João Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 244–253. DOI: 10.1109/CVPR.2019.00033 (cited on p. 34).

[53] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 6836–6846 (cited on p. 34).

[54] H. Alwassel, S. Giancola, and B. Ghanem, "Tsp: Temporally-sensitive pretraining of video encoders for localization tasks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021 (cited on pp. 34, 119).

[55] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *ICCV*, 2017 (cited on pp. 35, 40, 46, 87, 106).

[56] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundam. Inf.*, vol. 41, no. 1–2, pp. 187–228, Jan. 2000, ISSN: 0169-2968 (cited on p. 35).

[57] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," in *IEEE Conference on Computer Vision and Pattern Recognition 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 344–353. DOI: 10.1109/CVPR.2019.00043 (cited on pp. 35, 39).

[58] J. Gao, K. Chen, and R. Nevatia, "Ctap: Complementary temporal action proposal generation," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 70–85, ISBN: 978-3-030-01216-8 (cited on pp. 36, 40, 41, 46).

[59] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "Sst: Single-stream temporal action proposals," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6373–6382. DOI: 10.1109/CVPR.2017.675 (cited on pp. 36, 87, 106).

[60] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017 (cited on pp. 36, 39, 41, 46, 87, 106).

[61] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster R-CNN architecture for temporal action localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018 (cited on p. 36).

[62] C. Liu, X. Xu, and Y. Zhang, "Temporal attention network for action proposal," *Proceedings - International Conference on Image Processing, ICIP*, pp. 2281–2285, 2018, ISSN: 15224880. DOI: 10.1109/ICIP.2018.8451429 (cited on pp. 36, 47).

[63] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91 (cited on p. 36).

[64] Y. Liu, L. Ma, Y. Zhang, W. Liu, and S.-F. Chang, "Multi-granularity generator for temporal action proposal," in *Proceedings of the IEEE Conference on Computer Vi-*

sion and Pattern Recognition, 2019, pp. 3604–3613 (cited on pp. 37, 42, 106, 107, 110).

[65] L. Li, T. Kong, F. Sun, and H. Liu, "Deep point-wise prediction for action temporal proposal," in *Neural Information Processing*, Cham: Springer International Publishing, 2019, pp. 475–487, ISBN: 978-3-030-36718-3 (cited on pp. 37, 42).

[66] J. Tan, J. Tang, L. Wang, and G. Wu, "Relaxed transformer decoders for direct action proposal generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 13 526–13 535 (cited on pp. 37, 45, 47, 68, 106, 107, 110).

[67] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *European Conference on Computer Vision*, 2018 (cited on pp. 37, 40, 41, 47, 58, 59, 87, 106, 107, 110).

[68] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen, "Bmn: Boundary-matching network for temporal action proposal generation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3889–3898 (cited on pp. 37, 40, 41, 47, 58, 59, 84, 85, 97, 106, 107, 110).

[69] H. Su, W. Gan, W. Wu, J. Yan, and Y. Qiao, "Bsn++: Complementary boundary regressor with scale-balanced relation modeling for temporal action proposal generation," in *AAAI Conference on Artificial Intelligence*, 2020 (cited on pp. 37, 41, 58, 59, 84, 85, 97, 98, 103, 106, 107).

[70] L. Wang, H. Yang, W. Wu, H. Yao, and H. Huang, "Temporal action proposal generation with transformers," *CoRR*, vol. abs/2105.12043, 2021. [Online]. Available: `https://arxiv.org/abs/2105.12043` (cited on pp. 37, 45, 47).

[71] M. Xu, C. Zhao, D. S. Rojas, A. Thabet, and B. Ghanem, "G-tad: Sub-graph localization for temporal action detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020 (cited on pp. 38, 110).

[72] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, (ICLR), ser. ICLR '17, 2017. [Online]. Available: `https://openreview.net/forum?id=SJU4ayYgl` (cited on p. 38).

[73] Y. Bai, Y. Wang, Y. Tong, Y. Yang, Q. Liu, and J. Liu, "Boundary content graph neural network for temporal action proposal generation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12373 LNCS, pp. 121–137, 2020, ISSN: 16113349. DOI: `10.1007/978-3-030-58604-1_8` (cited on p. 38).

[74] J. Gao, Z. Yang, and R. Nevatia, "Cascaded boundary regression for temporal action detection," in *BMVC*, 2017 (cited on pp. 39, 43, 87).

[75] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Computer Vision – ECCV 2016*, Cham: Springer International Publishing, 2016, pp. 20–36, ISBN: 978-3-319-46484-8 (cited on p. 40).

[76] V. Escorcia, F. Caba Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 768–784, ISBN: 978-3-319-46487-9 (cited on pp. 41, 87).

[77] C. Lin *et al.*, "Fast learning of temporal action proposal via dense boundary generator," in *AAAI Conference on Artificial Intelligence*, 2020 (cited on pp. 41, 47, 58, 59, 84, 85, 106, 107, 110).

[78] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016 (cited on pp. 41, 106).

[79] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "1984, Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984 (cited on p. 41).

[80] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010. DOI: 10.1109/TPAMI.2009.167 (cited on p. 41).

[81] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94 (cited on p. 41).

[82] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106 (cited on pp. 42, 88, 89, 119).

[83] J. Gao *et al.*, "Accurate temporal action proposal generation with relation-aware pyramid network," in *AAAI Conference on Artificial Intelligence*, 2020 (cited on p. 42).

[84] H. Eun, S. Lee, J. Moon, J. Park, C. Jung, and C. Kim, "Srg: Snippet relatedness-based temporal action proposal generator," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 4232–4244, 2020. DOI: 10.1109/TCSVT.2019.2953187 (cited on pp. 42, 47, 106, 107).

[85] C. Lin *et al.*, "Learning salient boundary feature for anchor-free temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 3320–3329 (cited on pp. 43, 63, 65, 84, 85, 93, 105, 110, 137).

[86] Q. Liu and Z. Wang, "Progressive boundary refinement network for temporal action detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020 (cited on pp. 43, 110).

[87] X. Wang *et al.*, "Cbr-net: Cascade boundary refinement network for action detection: Submission to activitynet challenge 2020 (task 1)," Jun. 2020. [Online]. Available: http://arxiv.org/abs/2006.07526 (cited on p. 43).

[88] Z. Qing *et al.*, "Temporal context aggregation network for temporal action proposal refinement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 485–494 (cited on pp. 44, 106).

[89] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986. DOI: 10.1109/TPAMI.1986.4767851 (cited on p. 44).

[90] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, *Soft-nms – improving object detection with one line of code*, cite arxiv:1704.04503Comment: ICCV 2017 camera ready version, 2017. [Online]. Available: http://arxiv.org/abs/1704.04503 (cited on p. 45).

[91] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 6469–6477. DOI: 10.1109/CVPR.2017.685. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.685 (cited on p. 45).

[92] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3588–3597. DOI: 10.1109/CVPR.2018.00378 (cited on p. 45).

[93] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020*, 2020, pp. 213–229, ISBN: 978-3-030-58452-8 (cited on p. 45).

[94] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955. DOI: 10.1002/nav.3800020109 (cited on p. 45).

[95] L. Wang, Y. Xiong, D. Lin, and L. Van Gool, "Untrimmednets for weakly supervised action recognition and detection," in *CVPR*, 2017, pp. 4325–4334 (cited on p. 48).

[96] Z. Shou, D. Wang, and S. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 1049–1058. DOI: 10.1109/CVPR.2016.119 (cited on p. 48).

[97] S. Nag, X. Zhu, Y.-Z. Song, and T. Xiang, "Temporal action detection with global segmentation mask learning," *arXiv preprint arXiv:2207.06580*, 2022 (cited on p. 49).

[98] P. Lee and H. Byun, "Learning action completeness from points for weakly-supervised temporal action localization," in *IEEE/CVF International Conference on Computer Vision*, 2021 (cited on p. 49).

[99] B. Fernando, S. A. Shirazi, and S. Gould, "Unsupervised human action detection by action matching," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1604–1612, 2016 (cited on p. 49).

[100] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012 (cited on p. 53).

[101] H. Zhao, A. Torralba, L. Torresani, and Z. Yan, "Hacs: Human action clips and segments dataset for recognition and temporal localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019 (cited on p. 53).

[102] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *International Journal of Computer Vision*, 2017 (cited on p. 53).

[103] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 510–526, ISBN: 978-3-319-46448-0 (cited on p. 53).

[104] R. Zeng *et al.*, "Graph convolutional networks for temporal action localization," in *ICCV*, 2019 (cited on p. 58).

[105] Y. Xiong *et al.*, "CUHK & ETHZ & SIAT Submission to ActivityNet Challenge 2016," *arXiv e-prints*, arXiv:1608.00797, arXiv:1608.00797, Aug. 2016. DOI: 10.48550/arXiv.1608.00797. arXiv: 1608.00797 [cs.CV] (cited on p. 58).

[106] B. G. Shyamal Buch Victor Escorcia and J. C. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proceedings of the British Machine Vision Conference (BMVC)*, G. B. Tae-Kyun Kim Stefanos Zafeiriou and K.

Mikolajczyk, Eds., BMVA Press, Sep. 2017, pp. 93.1–93.12, ISBN: 1-901725-60-X. DOI: 10.5244/C.31.93. [Online]. Available: https://dx.doi.org/10.5244/C.31.93 (cited on p. 63).

[107] T. Lin, X. Zhao, and Z. Shou, "Single shot temporal action detection," ser. MM '17, Mountain View, California, USA: Association for Computing Machinery, 2017, pp. 988–996, ISBN: 9781450349062. DOI: 10.1145/3123266.3123343. [Online]. Available: https://doi.org/10.1145/3123266.3123343 (cited on p. 63).

[108] X. Liu *et al.*, "End-to-end temporal action detection with transformer," *IEEE Transactions on Image Processing (TIP)*, 2022 (cited on pp. 63, 110).

[109] C.-L. Zhang, J. Wu, and Y. Li, "Actionformer: Localizing moments of actions with transformers," in *European Conference on Computer Vision*, ser. LNCS, vol. 13664, 2022, pp. 492–510 (cited on pp. 65, 110).

[110] T. Lin, X. Zhao, and Z. Shou, "Single shot temporal action detection," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17, Mountain View, California, USA: Association for Computing Machinery, 2017, pp. 988–996, ISBN: 9781450349062. DOI: 10.1145/3123266.3123343. [Online]. Available: https://doi.org/10.1145/3123266.3123343 (cited on p. 87).

[111] S. Liu *et al.*, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022 (cited on pp. 90, 119).

[112] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *CoRR*, vol. abs/2009.06732, 2020 (cited on p. 91).

[113] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: A simple and strong anchor-free object detector," 2021 (cited on p. 92).

[114] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597 (cited on p. 98).

[115] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," 2018 (cited on p. 105).

[116] V. Vo-Ho, S. Truong, K. Yamazaki, B. Raj, M. Tran, and N. Le, "Aoe-net: Entities interactions modeling with adaptive attention mechanism for temporal action proposals generation," *Int. J. Comput. Vis.*, vol. 131, no. 1, pp. 302–323, 2023 (cited on pp. 106, 107, 110).

[117] P. Zhao, L. Xie, C. Ju, Y. Zhang, Y. Wang, and Q. Tian, "Bottom-up temporal action localization with mutual regularization," in *European Conference on Computer Vision*, Springer, 2020, pp. 539–555 (cited on pp. 106, 110).

[118] L. Yang, H. Peng, D. Zhang, J. Fu, and J. Han, "Revisiting anchor mechanisms for temporal action localization," *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, Aug. 2020. DOI: 10.1109/TIP.2020.3016486 (cited on p. 110).

[119] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," Jun. 2019, pp. 344–353. DOI: 10.1109/CVPR.2019.00043 (cited on p. 110).

[120] D. Shi, Y. Zhong, Q. Cao, L. Ma, J. Li, and D. Tao, "Tridet: Temporal action detection with relative boundary modeling," *arXiv preprint arXiv:2303.07347*, 2023 (cited on p. 110).

[121] Z. Zhu, W. Tang, L. Wang, N. Zheng, and G. Hua, "Enriching local and global contexts for temporal action localization," in *ICCV*, 2021 (cited on p. 110).

[122] J. Ma and B. Chen, "Dual refinement feature pyramid networks for object detection," *ArXiv*, vol. abs/2012.01733, 2020 (cited on p. 119).

[123] L. Zhu, F. Lee, J. Cai, H. Yu, and Q. Chen, "An improved feature pyramid network for object detection," *Neurocomputing*, vol. 483, pp. 127–139, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.02.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222001588 (cited on p. 119).

Appendices

Appendix A

Datasets

A.1 Action Classes

A.1.1 THUMOS14

Table A.1. Class labels in THUMOS14 dataset. Actions that exist in ActivityNet v1.3 in a similar form are in bold.

Billiards	**CleanAndJerk**	**CliffDiving**	**Diving**	CricketBowling
CricketShot	FrisbeeCatch	BaseballPitch	GolfSwing	**HammerThrow**
HighJump	**JavelinThrow**	**LongJump**	**PoleVault**	**Shotput**
ThrowDiscus	SoccerPenalty	BasketballDunk	TennisSwing	**VolleyballSpiking**

A.1.2 ActivityNet v1.3

Table A.2. Class labels in ActivityNet v1.3 dataset. Actions that exist in THUMOS14 in a similar form are in bold.

Applying sunscreen	Arm wrestling	Assembling bicycle
BMX	Baking cookies	Baton twirling
Beach soccer	Beer pong	Blow-drying hair
Blowing leaves	Playing ten pins	Braiding hair
Building sandcastles	Bullfighting	Calf roping
Camel ride	Canoeing	Capoeira
Carving jack-o-lanterns	Changing car wheel	Cleaning sink
Clipping cat claws	Croquet	Curling
Cutting the grass	Decorating the Christmas tree	Disc dog
Doing a powerbomb	Doing crunches	Drum corps
Elliptical trainer	Doing fencing	Fixing the roof
Fun sliding down	Futsal	Gargling mouthwash
Grooming dog	Hand car wash	Hanging wallpaper
Having an ice cream	Hitting a pinata	Hula hoop
Hurling	Ice fishing	Installing carpet
Kite flying	Kneeling	Knitting
Laying tile	Longboarding	Making a cake
Making a lemonade	Making an omelette	Mooping floor
Painting fence	Painting furniture	Peeling potatoes
Plastering	**Playing beach volleyball**	Playing blackjack

Continued on next page

Table A.2 – continued from previous page

Playing congas	Playing drums	Playing ice hockey
Playing pool	Playing rubik cube	Powerbocking
Putting in contact lenses	Putting on shoes	Rafting
Raking leaves	Removing ice from car	Riding bumper cars
River tubing	Rock-paper-scissors	Rollerblading
Roof shingle removal	Rope skipping	Running a marathon
Scuba diving	Sharpening knives	Shuffleboard
Skiing	Slacklining	Snow tubing
Snowboarding	Spread mulch	Sumo
Surfing	Swimming	Swinging at the playground
Table soccer	Throwing darts	Trimming branches or hedges
Tug of war	Using the monkey bar	Using the rowing machine
Wakeboarding	Waterskiing	Waxing skis
Welding	Drinking coffee	Zumba
Doing kickboxing	Doing karate	Tango
Putting on makeup	**High jump**	Playing bagpipes
Cheerleading	Wrapping presents	**Cricket**
Clean and jerk	Preparing pasta	Bathing dog
Discus throw	Playing field hockey	Grooming horse
Preparing salad	Playing harmonica	Playing saxophone
Chopping wood	Washing face	Using the pommel horse
Javelin throw	Spinning	Ping-pong
Making a sandwich	Brushing hair	Playing guitarra
Doing step aerobics	Drinking beer	Playing polo
Snatch	Paintball	**Long jump**
Cleaning windows	Brushing teeth	Playing flauta
Tennis serve with ball bouncing	**Bungee jumping**	Triple jump
Horseback riding	Layup drill in basketball	Vacuuming floor
Cleaning shoes	Doing nails	**Shot put**
Fixing bicycle	Washing hands	Ironing clothes
Using the balance beam	Shoveling snow	Tumbling
Using parallel bars	Getting a tattoo	Rock climbing
Smoking hookah	Shaving	Getting a piercing
Springboard diving	Playing squash	Playing piano
Dodgeball	Smoking a cigarette	Sailing
Getting a haircut	Playing lacrosse	Cumbia
Tai chi	Painting	Mowing the lawn
Shaving legs	Walking the dog	**Hammer throw**
Skateboarding	Polishing shoes	Ballet
Hand washing clothes	**Plataform diving**	Playing violin
Breakdancing	Windsurfing	Hopscotch
Doing motocross	Mixing drinks	Starting a campfire
Belly dance	Removing curlers	Archery
Volleyball	Playing water polo	Playing racquetball
Kayaking	Polishing forniture	Playing kickball
Using uneven bars	Washing dishes	**Pole vault**
Playing accordion	Playing badminton	

Appendix B

Feature Extraction

For THUMOS14 dataset, feature vectors were extracted every 5 frames and each vector was made from 5 image frames as in other work.

For ActivityNet v1.3 dataset, to handle all videos efficiently, all boundary-based methods resized videos to get a fixed number of feature vectors per video. This was done using linear interpolation. However, as pointed out in ASDF[85], sampling image frames from video directly is more useful, so by following their methods, images were sampled. Each video has different length and different frame per second (FPS). The goal was making video have the same number of feature vectors. First, image frames were sampled at different frame rates per video, so each video consisted of 768 image frames. This is because the feature pyramid needs 96 feature vectors, and the I3D network needs 768 image frames to generate 96 feature vectors. In this way, all videos were represented by 96 feature vectors, which is the same effect as linear interpolation. Then, all videos can be processed at once.

Appendix C

Feature Pyramid Construction

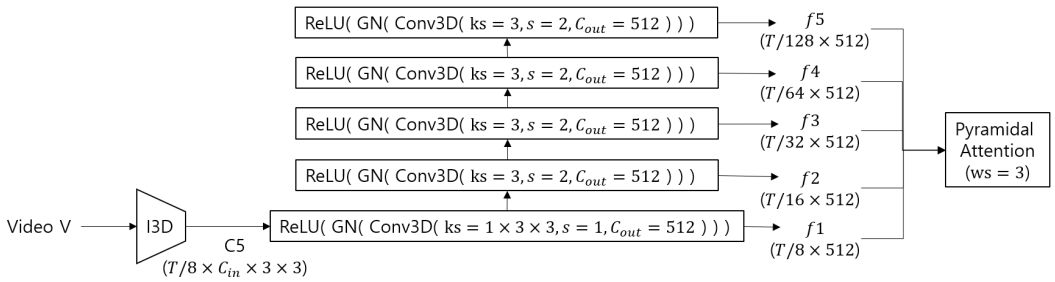


Figure C.1. Details of Feature Pyramid Construction. T is the number of images which is 768, C_{in} is the dimension of input channels which is either 1024 or 2048 depending on which type of feature used ($RGB, Flow \in \mathbb{R}^{1024}, Both \in \mathbb{R}^{2048}$). ks is kernel size and s is stride. In pyramidal attention, ws is set to 3, which is the window size including attentive sibling nodes and the updated node itself.

Figure C.1 shows the details of the feature pyramid construction. The I3D network consumes 768 image frames to generate 96 feature vectors. Feature maps were collected from the C5 layer. To make the first layer of the feature pyramid, a 3D convolution layer was applied to reduce the dimensions of width and height followed by group normalisation and ReLU. For the rest of the layers, the same operations consisting of 1D convolution, group normalisation and ReLU are applied repeatedly. The difference with the first layer is the stride s is set to 2 to increase the receptive field gradually. Then, the feature pyramid is updated through the pyramidal attention whose window size ws is set to 3 indicating the node itself and two sibling nodes are attending.