Multi-Modal Livestream Highlight Detection from Audio, Visual, and Language Data

Charles Ringer

Doctor of Philosophy

University of York

Computer Science

August 2022

Any sufficiently advanced technology is indistinguishable from magic.

- Arthur C. Clarke

Abstract

Livestreaming is the act of live broadcasting via the internet and allows viewer-host interaction via a text-based chat system. In particular, video game livestreaming is prevalent, where streamers host individual play sessions or present esports competitions. While livestreaming is an emerging entertainment medium, it is popular. For example, every minute, about 1900 hours of footage is livestreamed on Twitch.tv, currently the most popular video game livestreaming platform.

It can be challenging for viewers to access the content they are most likely to enjoy. One solution is 'highlight videos', which can entertain users who did not watch a broadcast, e.g. due to a lack of awareness, availability, or willingness. Furthermore, livestream content creators can grow their audiences by using highlights to advertise their streams and engage casual followers. However, hand-generating these videos is laborious. Thus there is great value in developing automatic highlight detection methods.

Video game streaming provides the viewer with a rich set of audio-visual data, conveying information about the game through game footage and the streamer's emotional state via webcam footage. Analysing both the game and the behaviour of broadcast personnel is crucial for modelling the exciting aspects of livestreams. Furthermore, livestreaming offers a unique opportunity to understand the viewing experience through the text-based chat system. However, livestream data has a significant set of challenges, e.g. how to fuse multimodal data captured by different sources in uncontrolled, noisy conditions. Thus deep learning models able to leverage complex data are appealing for highlight detection methods.

This thesis explores the application of deep learning highlight detection models to the domain of livestreaming. Multimodal highlight detection methods are developed for personality-driven livestreams and esports broadcasts. The unique nature of livestream audience chat language is explored, and audience-based highlight methods are proposed. Finally, a model capable of modelling all these modalities in one system is presented.

Contents

List of Tables

List of Figures

Acknowledgements

Firstly, I must thank my two supervisors, Dr James Alfred Walker and Dr Mihalis A. Nicolaou. This research would not have been possible without your constant guidance. Your reassurance that I was headed in the right direction, even when the path felt overgrown and difficult to traverse, helped me immensely.

Secondly, my dearest Heather. You have been my rock throughout these five years. I cannot thank you enough for your endless love and support. Embarking on the journey that is a PhD can be scary and perhaps foolish, but you always spurred me on. Not many people would move around the country because of their partner's folly. Thank you for everything.

Thirdly, Amelie. You are the most wonderful, friendliest, cuddliest, loving dog a man could ask for. Although you do not realise it, you provided me with comfort and companionship on those days when I worked at home alone, sometimes going all day without speaking to another soul.

Fourthly, my parents. From a young age, you made me believe that anything was possible if I put my mind to it and worked hard enough. In that respect, this thesis is as much your work as it is mine. You have also supported me, Heather, and Amelie through the more challenging parts of this PhD, and I am incredibly grateful for that. Mother, you require a special mention for reading this entire thesis and giving me feedback, despite not having a clue about the topic.

Finally, my friends and fellow IGGI students. The IGGI community is a diverse, friendly, and warm group of people. You acted as a sounding board when times were challenging and celebrated the good times. I am glad to say that I was part of it.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

The list of works published as part of the completion of this thesis is as follows:

- C. Ringer and M. A. Nicolaou. 2018, "Deep unsupervised multi-view detection of video game stream highlights", *Proceedings of the 13th International Conference on the Foundations of Digital Games (FDG '18)*, 2018 pp. 1–6. [166]

- C. Ringer, J. A. Walker and M. A. Nicolaou, "Multimodal Joint Emotion and Game Context Recognition in League of Legends Livestreams", *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1-8 [169]

- C. Ringer and M. A. Nicolaou, "Streaming Behaviour: Livestreaming as a Paradigm for Analysis of Emotional and Social Signals", 2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), 2019, pp. 182-185 [167]

- C. Ringer, M. A. Nicolaou and J. A. Walker, "TwitchChat: A Dataset for Exploring Livestream Chat", Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2020 [165]

- C. Ringer, M. A. Nicolaou and J. A. Walker, "Autohighlight: Highlight Detection in *League of Legends* Esports Broadcasts via Crowd-Sourced Data" Machine Learning with Applications, 2022, [168]

Chapter 1

Introduction

This thesis details research into applying artificial intelligence techniques, namely deep learning, to the challenge of detecting the most interesting and exciting 'highlight' moments during a video game livestream. Video game livestream broadcasts are live-broadcast entertainment containing a wide range of video game based content. Livestream broadcasts are primarily video data, i.e. audio-visual data, although livestreaming is unique in that viewers can also interact with the stream hosts and other viewers through a textual 'chat' system. Therefore, to achieve high-quality modelling in this domain, a wide range of deep learning techniques are required from various fields, e.g. Computer Vision, Digital Signal Processing, Affective Computing, and Natural Language Processing.

1.1 An Introduction to Livestreaming

Livestreaming is a broadcast format which has become popular as stable access to high-speed internet has become widespread, at least in the western world [189]. It focuses solely on live broadcasts rather than pre-recorded, edited, media. These livestream broadcasts are distributed through online platforms, e.g. Twitch.tv[1], Youtube Gaming[2], Facebook Gaming[3] and Huya Live[4]. Of these, Twitch.tv is the most popular with an average of 26.5 million daily viewers [208], and will be the focus of this thesis. Livestreaming, like video sharing platforms, e.g. YouTube, has a low barrier to entry for content creators compared to tra-

[1] www.twitch.tv
[2] gaming.youtube.com
[3] www.facebook.com/gaming/
[4] www.huya.com

Table 1.1: A comparison of different broadcast formats

Broadcast Format	Live/Prerecorded	Interactive	Barrier to Entry	Distribution Channel	Example Platforms
Livestreaming	Live	Yes	Low	Internet	Twitch.Tv
Video Sharing Platforms	Prerecorded	Delayed	Low	Internet	Youtube
Video Streaming Platforms	Prerecorded	No	High	Internet	Netflix, Disney+
Television Broadcasts	Both	No	High	Television	Fox (US), BBC (UK)
Radio Broadcasts	Both	Occasionally	High	Radio/Internet	Sirius XM (US), BBC (UK)

ditional broadcast mediums such as television and radio. Livestreaming requires affordable, easy-to-attain broadcast equipment, e.g. a computer and internet connection, although external equipment such as cameras and microphones can be useful [207]. This accessibility is part of the reason that livestreaming has seen such a meteoric growth over the last few years [158]. Table 1.1 details the key differences between livestreaming and other forms of broadcast media.

Livestreaming has become so popular in recent years that it is now a leading form of video game based entertainment, with emerging communities of millions of broadcasters and viewers [93]. Video game playing is the most prevalent genre on livestreaming platforms that 'livestreaming' is often synonymous with 'video game livestreaming'; indeed, this thesis focuses only on such streams. Livestream viewership is large enough that some video game streamers can treat livestreaming as a full-time occupation [87]. Initially, Twitch.tv began as a category on the now-defunct general-purpose livestreaming platform Justin.tv. The video gaming category became so popular that Justin.tv transformed it into a dedicated service and ultimately shut down Justin.tv. Of course, as Twitch.tv grew, the number of non-gaming broadcasts have grown. Therefore, while Twitch.tv remains primarily focused on video games related content; there are other categories, e.g. the 'Just Chatting' category, a catch-all for various types of broadcasts, which is often the most popular category on the platform [206].

Visitors to Twitch.tv are presented with an interface where select streams are promoted and otherwise different categories, i.e. games, are shown. Visitors can then use this interface to select a game and then subsequently select a channel streaming that game. Additionally, channel recommendations and search tools aid visitors in finding a channel that they wish to watch. This interface is demonstrated in Figure 1.1. Once visitors have selected a stream to watch, they are presented with the interface in Figure 1.2. In this interface, visitors are shown the stream in the main window. The interface also provides a chat message box to the right, where users can type messages, and a chat window where they can read messages

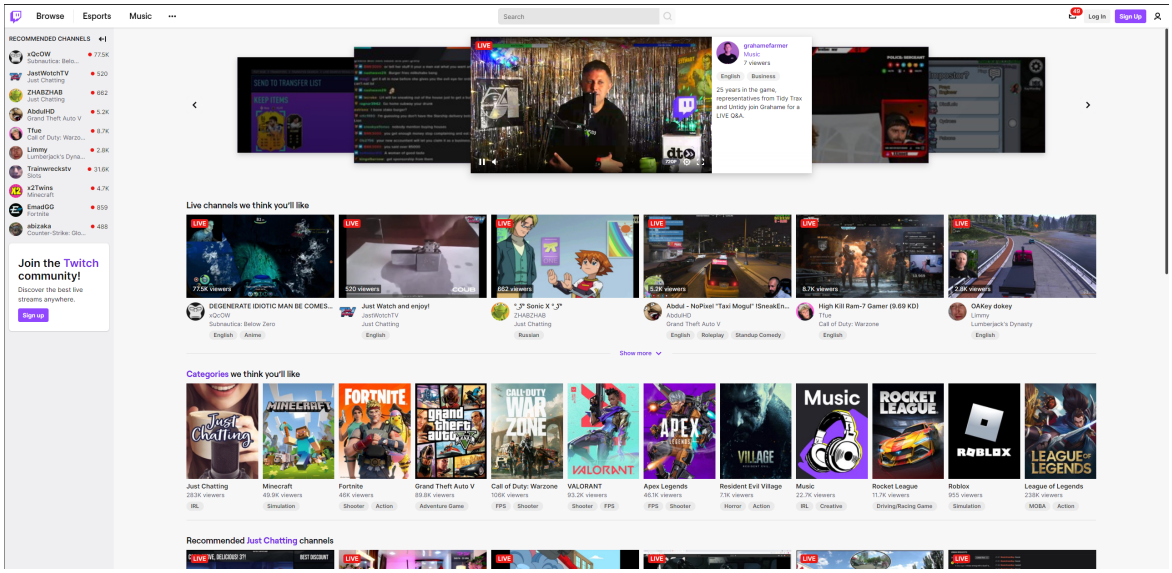


Figure 1.1: The interface for Twitch.tv. Viewers are shown promoted streams as well as recommend popular streams and categories.

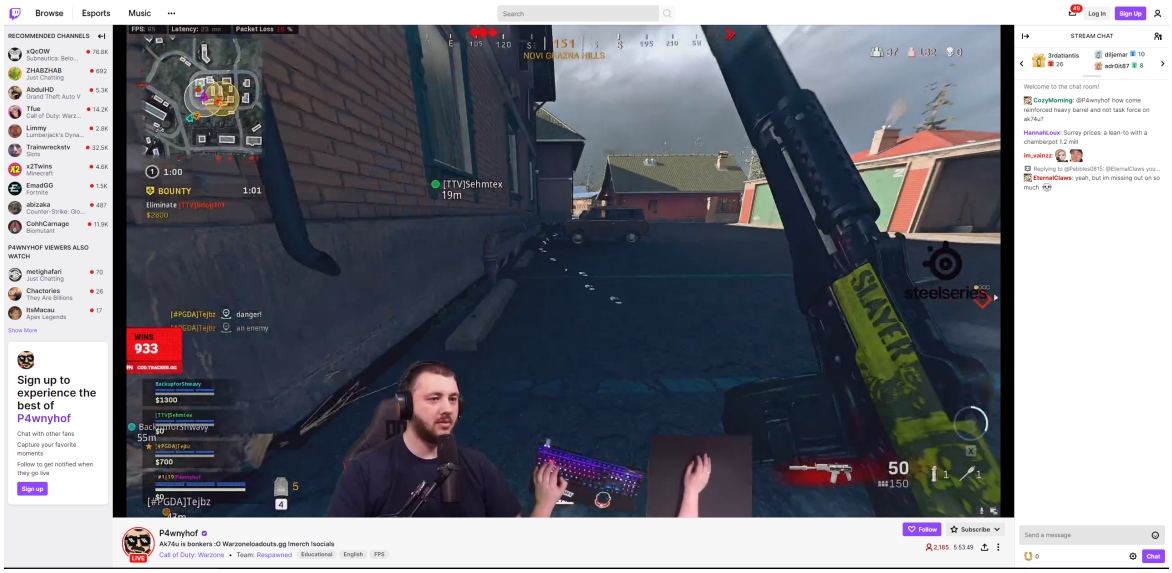


Figure 1.2: A typical stream as it appears to viewers in a 'personality-driven' broadcast

already sent. These messages can be used to interact with other viewers, the streamer(s), and to act as a form of cheering, as is common in traditional sports. There are additional ways in which viewers can interact, e.g. they can choose to follow a stream and receive notifications or subscribe to a stream, a paid option that often has additional in-stream benefits like having their name read by the streamer or a special set of images (emotes) usable in chat messages. Finally, the visitor can use the left-most pane to browse for a new stream to watch from a promoted selection.

1.1.1 Types of Video Game Livestream Broadcasts

How channels present themselves on livestreaming platforms varies, although there are some commonalities. This thesis distinguishes between 'Personality-Driven' and 'Game-Driven' broadcasts. While both are video game livestreams, they have different broadcast styles and attract viewers in different manners.

1.1.1.1 'Personality-Driven' Broadcasts

'Personality-driven broadcasts' describes a single streamer broadcasting and presenting to the viewers. During a typical personality-driven stream, the streamer broadcasts both game footage and video of their face via a webcam while communicating with viewers via audio and text chat. Personality-driven streams are enriched with social signals conveyed between streamers and viewers via facial expressions, body movements, vocal cues, and written language. Furthermore, the various data sources in a broadcast, e.g., camera stream, game footage, audio, text chat, can be considered data views that carry inherent correlations since they all describe events occurring during a stream. This unique interactive setting facilitates the joint, multi-view analysis of data sources in a coherent and self-contained manner. Figure 1.4 details the large number of possible interactions. Personality-driven streams are typified by a high amount of interaction between the streamer, and the audience, e.g. the streamer will often ask and answer questions as well and thank viewers. These streams tend to have a loose structure, with no set itinerary. Some streamers focus on a particular game while others play a variety. The audience expects the streamer to play games while providing entertainment, e.g. through chatting and making jokes. The stream presented in Figure 1.2 is a prime example of a personality-driven broadcast. This paradigm has challenges for modelling the audio-visual elements of the stream, i.e. as required for highlight detection models. This is especially so when considering occlusions in the streamer's webcam, e.g. Figure 1.3. While this broadcast type is referred to as 'personality-driven' broadcasts, they are not restricted to only having one personality presenting content, although most streams do.

1.1.1.2 'Game-Driven' Broadcasts

The other broadcasts studied in the thesis are 'game-driven' esports broadcasts. These are presented very differently from personality-driven broadcasts. While single streamer broad-

Figure 1.3: Common occlusions present in a 'personality-driven' livestream webcams.

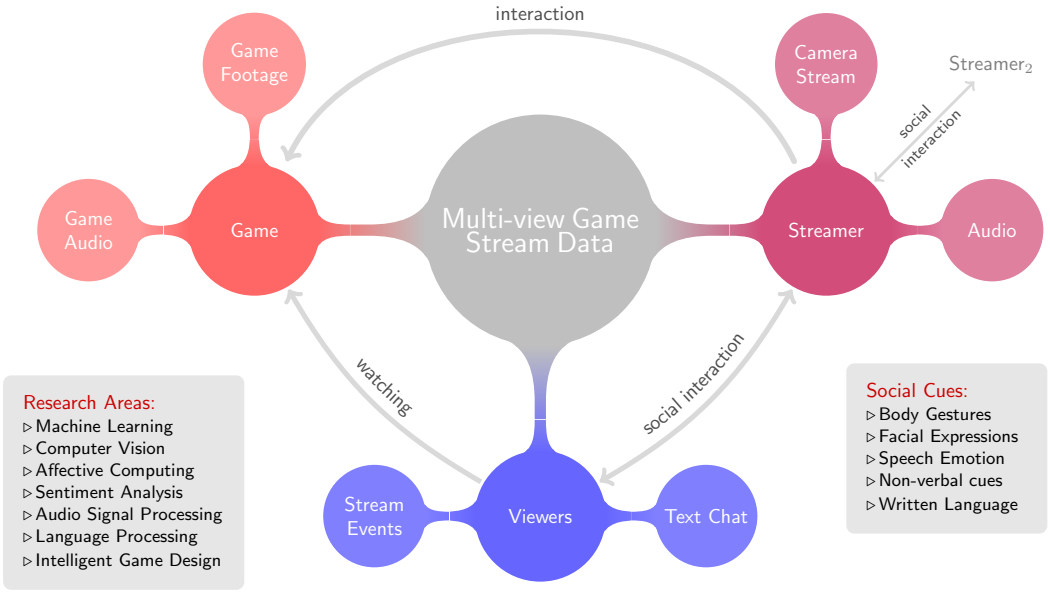
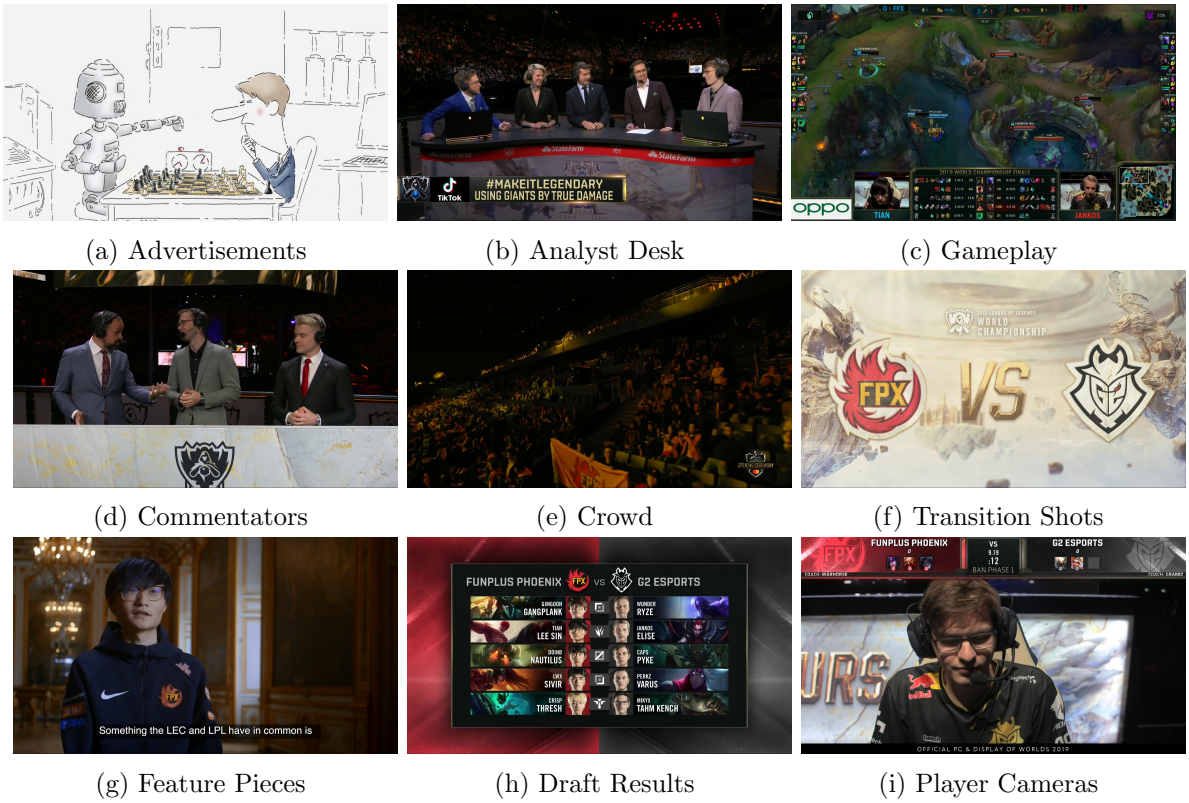


Figure 1.4: Illustration of main opportunities presented by personality-driven livestream data and related research areas. The broadcast includes a video stream capturing the streamer's face and upper body and an audio recording from the streamer's microphone. This is accompanied by the footage of the game, including in-game audio. The streamer interacts with viewers by video, voice, and text. The viewers interact with streamers and other viewers via text and stream-specific events. A frequent setting includes the streamer interacting with other streamers via webcam video and voice (e.g., during a co-op game). [167]



(a) Advertisements (b) Analyst Desk (c) Gameplay

(d) Commentators (e) Crowd (f) Transition Shots

(g) Feature Pieces (h) Draft Results (i) Player Cameras

Figure 1.5: The many different constituent elements which make up a *League of Legends* esports broadcast.

casts are informal, esports broadcasts are generally professionally produced and presented in a style similar to their traditional sports cousins. Esports are a growing multi-billion dollar industry. Rather than the intimate, casual and highly interactive nature of single streamer broadcasts, esports broadcasts generally function as organised, non-interactive shows with a ridged structure. Esports are often presented similarly to their traditional sports cousins, e.g. between games an 'analyst desk' discusses the results of previous matches and upcoming games. During a match, a set of commentators commentate on the gameplay video footage. The visual feed shows various scenes, e.g. gameplay, the analyst desk, and advertisements. Elements of this broadcast structure is shown in Figure 1.5. The critical difference compared to traditional sports broadcasts is the format of the games. There are no constraints to the content of esports, bar the need for it to be digital, whereas sports are constrained by physics and human performance. This variance results in significant differences in the visual appearance of different esports.

1.2 The Case for Automatic Highlight Detection

This thesis explores applying deep learning methods to this rich, emerging livestream data paradigm. In particular, it will focus on how it is possible to model, detect and extract the most salient moments in a stream under the moniker 'highlight detection'. Highlight detection is a broad term for a range of techniques which aim to identify the most 'important' moments from an input video. It is impossible to strictly define a highlight because it is subjective, and thus it becomes essential to rely on proxy highlight signals or crowd-sourced data.

Despite these challenges, highlight detection has significant potential applications. Livestreaming is not only popular with viewers but also with those wishing to make content. For instance, Twitch.tv has an average of $95,839$ concurrent channels, with a maximum peak of $233,935$ during December 2020 [206]. Illustrated another way, every minute, 500 hours of footage is uploaded to Youtube[5] and about 1900 hours of footage is livestreamed on Twitch[6]. Most streams, especially those with smaller audiences, are personality-driven broadcasts. The quantity of these broadcasts can make it difficult for less popular streamers to stand out and grow their audience. It is equally challenging for viewers to find the content they are most likely to enjoy. Advertising longer-form livestreams via shorter edited videos, i.e. highlight videos, distributed via video-sharing platforms and social media is a potential solution to this problem.

This work is motivated by the observation that automatic highlight detection offers additional avenues for streamers/broadcasters to reach their audience and for viewers to find content that suits them. Highlight videos allow streamers to reach viewers who would not otherwise watch a livestream, e.g. because they are unable or unwilling to commit to watching a specific broadcast at a particular time. It is no surprise that streamers utilise video-sharing platforms to distribute highlight videos[7]. However, manual generation of these highlights is time-consuming. Because of the time implication of editing videos, many successful streamers choose to employ editing teams, which are costly. Therefore, producing these highlight videos costs time, money, or both. These are resources that less popular streamers may not have. For example, their income from streaming is likely to be far lower than the most

[5] https://www.statista.com/
[6] https://twitchtracker.com/statistics
[7] One concrete example is the streamer Dr DisRespect, although there are numerous

subscribed channels; therefore, they are likely to work full-time jobs alongside building their streaming business. Access to low-cost, fast, highlight video editing solutions is likely attractive for these streamers. Hence, there is a natural opportunity to utilise deep learning models of highlight moments to aid these streamers.

Highlight videos are also helpful for those livestreaming game-driven broadcasts, i.e. esports organisations. Firstly, many motivations for personality-driven broadcast highlights also apply to esports broadcasts. Esports can be seen as analogous to traditional sports broadcasts, and it is prevalent for old media, especially television, to broadcast sports highlights rather than, or as well as, the entire event. There is no reason to believe that esports would not also be popular when presented in a similar format. Furthermore, automatic highlight detection offers additional benefits. For example, such techniques can likely run in, or near, real-time and thus, it would be possible to retrieve highlights while the event is occurring and then upload them immediately after a match has finished or even post partial highlights, e.g. a small clip, to social media during an event. Such functionality would allow the event organisers to advertise to potential viewers while the event is occurring and improve viewer numbers.

There is also the potential for personalised highlights, especially for larger channels with a diverse audience. These highlights could be tuned to the viewer's preferences, e.g. by biasing a detection system for the kind of events a particular viewer enjoys. This type of content would be infeasible without automatic systems, as it would require a human editor to learn the preferences of many viewers and then edit a unique video for each one. Therefore, research into automatic highlight detection for livestreamed esports has practical, real-world applications that benefit many stakeholders.

1.2.1 Defining Highlights

Until this point, the term 'highlight' has been used without clarity or precision. This is by design. Highlights are nebulous and hard to define. Formal definitions, e.g. from dictionaries, describe highlights in vague terms, e.g. as the most interesting, entertaining, or exciting moments in 'something'[8]. This is not particularly useful for developing a computational highlight model because such models require a binary definition of which moments are highlights and which are not. While the 'something' in formal definitions is relatively

[8]www.merriam-webster.com/, https://dictionary.cambridge.org/, https://www.collinsdictionary.com/

straightforward in this case, i.e. a livestream video game broadcast, determining a ground truth value for which moments are interesting, entertaining, or exciting is a challenge in its own right. Furthermore, consider that for each viewer, there are likely to be different moments that they consider to be the most interesting, entertaining, or exciting.

This problem is further exacerbated by the realisation that, while livestream data is abundant, formalised datasets with ground truth annotations do not exist in an easily accessible way, in contrast to similar challenges such as image recognition, where the ImageNet dataset [41] is widely used by the computer vision community, e.g. it has over $32,000$ citations. Developing such a dataset for livestream highlight detection is challenging and extremely time-consuming, especially if a group of human annotators decides on ground truth labels. Furthermore, as deep learning models get ever more complex, they require training on more extensive datasets. This makes training data which relies on a strict input to ground truth label mapping through manual annotation even less attractive. Scaling the annotation process is laborious.

In this thesis, several approaches to tackling this problem are explored. In parts, a supervised dataset is gathered and annotated, although small. In other sections, proxy signals are used, e.g. information more easily gathered, which may align closely with when highlight moments occur. Finally, and perhaps most successfully, crowd-sourced data is utilised, where large datasets of edited broadcasts posted to video sharing sites, e.g. YouTube, are leveraged. Each approach has unique advantages and disadvantages, which will be discussed in detail in the relevant chapters. Some models presented in this thesis also use unlabelled data to aid pre-training. Again this is detailed more thoroughly where appropriate. Therefore, this thesis details research which explores Supervised, Unsupervised, Semi-Supervised and Self-Supervised approaches to highlight detection.

1.2.2 The Suitability of Deep Learning for Highlight Detection

The above discussion motivates the value of developing techniques for performing highlight detection on livestream data to various stakeholders. Next, it is essential to discuss why Deep Learning is the most appropriate method for developing such techniques given this data domain. Deep Learning is a subset of Machine Learning, which can be considered a subset of Artificial Intelligence (AI). If AI is the field of algorithms which can make decisions, then Machine Learning is the field of algorithms which can learn to make decisions. Rather

than prescribe a behaviour pattern for an algorithm, e.g., in Expert Systems or Finite State Machines, Machine Learning prescribes both a behaviour pattern and a method of improving that behaviour through minimising error. Machine Learning algorithms are varied. For instance, decision trees, k-nearest neighbours, and perceptrons all approach learning in very different ways. The last approach, the perceptron, forms the foundation of the neural network, which is at the heart of deep leaning. In general, the difference between 'machine learning' and 'deep learning' is often the complexity of the model used; deep learning models often involve millions of weights, compared to some machine learning techniques, which require very few.

Deep Learning is primarily driven by two observations. Firstly, that accessible computational power is growing near exponential, as predicted by Moore's law. Secondly, as internet-enabled devices become ubiquitous, at least in the western world, the amount of data that can be gathered about how humans live is becoming immeasurable. Deep learning, as a field, focuses on developing algorithms which can assimilate a vast volume of complex data, e.g. images, audio, and text, and model them in valuable ways, e.g. spatially or temporally. Figure 1.6 details the relationship between AI, Machine Learning, and Deep Learning, as well as the role that adjacent fields such as data science (and subsequently big data) play. Livestreaming data is incredibly complex. A stream contains at least one video feed, consisting of a series of images, an audio feed, and a chat log. It would be impossible or very problematic to model this data satisfactorily using more straightforward machine learning or data science techniques. However, deep learning techniques have been developed precisely for complex data. Furthermore, as discussed above, available livestream data are abundant. Therefore, Deep Learning is the natural approach; it benefits from large-scale data and can model highly complex inputs.

1.2.3 Selecting Modalities

Data presented to a viewer during a livestream is rich and complex. Unlike traditional broadcast media, e.g. television, alongside the audio-visual video broadcast, livestreams display a text chat where the viewers can interact with each other and the host. This thesis is interested in audio, visual and textual data. Most games have some form of 'telemetry' data, moment-to-moment tabular data, which describes the state of the game associated with them. This data should, in theory, provide strong cues to highlight moments. For example,

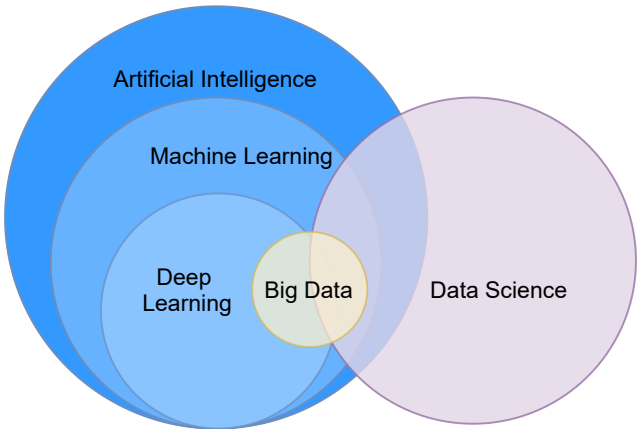


Figure 1.6: A venn diagram of the family of Artificial Intelligence techniques, along side related fields.

a model could easily determine if the player had accomplished a particular task. However, such data is often propriety and thus is not readily available, especially in conjunction with a live video feed. Therefore, while the potential utility of such data is evident, this thesis only examines data accessible to the viewer, i.e. audio, visual and textual data. Throughout the thesis, the term 'modality' will describe a single data stream, e.g. audio data. However, in parts, input data from a given data stream will be handled differently, e.g. video data may be segmented into a streamer feed and a video game feed. In these cases, for simplicity of discussion, the term modality is extended to include different data stream handling.

1.3 Research Goals

There are several core challenges to this research. Firstly, the livestream data format has not been widely explored, especially for highlight detection techniques. Therefore, in order to satisfactorily model highlight moments first the data itself must be understood. Next, both audio-visual and textual models must be developed. Finally, unification of these models to predict highlights in a holistic manner is likely to prove fruitful and thus warrants exploration. As such, the core goals of this thesis can be summarised by the following:

- Explore dataset properties, preprocessing, and labelling required to successfully model the most exciting moments in livestreamed esports broadcasts.
- Develop methods for utilising video data to perform audio-visual highlight detection suitable for both personality-driven livestreams and game-driven livestreams.

- Develop methods for utilising chat to perform Natural Language based highlight detection.
- Explore the potential for multimodal models using Audio, Visual and Language data.

1.4 Contributions

The key contributions of this thesis can be summarised by the following:

- The first publicly available dataset of livestream chat data, alongside an analysis of the dataset [165].
- The first unsupervised architecture for multimodal audio-visual highlight detection for personality-driven broadcasts.
- State-of-the-art multimodal audio-visual highlight detection for game-driven broadcasts.
- State-of-the-art textual highlight detection for game-driven broadcasts.
- A unified audio-visual-textual highlight detection model performs exceptionally well compared to just broadcast (video) or audience (textural) data.

1.5 Thesis Structure

The following chapter, Chapter 2, contains a review of existing literature, covering deep learning methods, deep learning applications, and the livestreaming phenomenon in detail. Next, Chapter 3, presents potential highlight detection methodologies for personality-driven broadcasts. Subsequently, Chapter 4 explores crowd-sourced highlight detection for game-driven broadcasts. Chapter 5 explores livestream chat and its suitability for highlight detection. The final research chapter, Chapter 6, proposes a unified audio-visual-textual model and proposes future work. Finally, Chapter 7 concludes the thesis.

Chapter 2

Literature Review

The work presented in the thesis spans many research communities. While livestreaming is a relatively new broadcast media, there is a wealth of related literature. This chapter introduces an overview of this literature in three sections. Firstly, appropriate deep learning methods are introduced and explained. Next, the application of these methods in related domains is discussed. Finally, moving away from deep learning, the existing literature around livestreaming is presented.

2.1 Deep Learning Methods

Many Deep Learning models are constructed from several core 'building block' techniques, e.g. those discussed in Section 2.2. In the case of this thesis, which deals with audio, visual, and language data, these building blocks are techniques which can accommodate spatial data, e.g. images, temporal data, e.g. data over time, or textual data, e.g. converting text to numeric values. This section will describe the fundamentals of how these techniques function.

2.1.1 Multilayer Perceptrons

The most basic neural network building block is the Perceptron [173]. The Perceptron is a 'universal function approximator' that, in theory, can approximate any mathematical function. To achieve this, the Perceptron is provided with a rank one tensor of scalar inputs, which are then multiplied by a similarly sized tensor of weights. The resultant values are summed and then 'activated' through a non-linear function. This activated value is multiplied

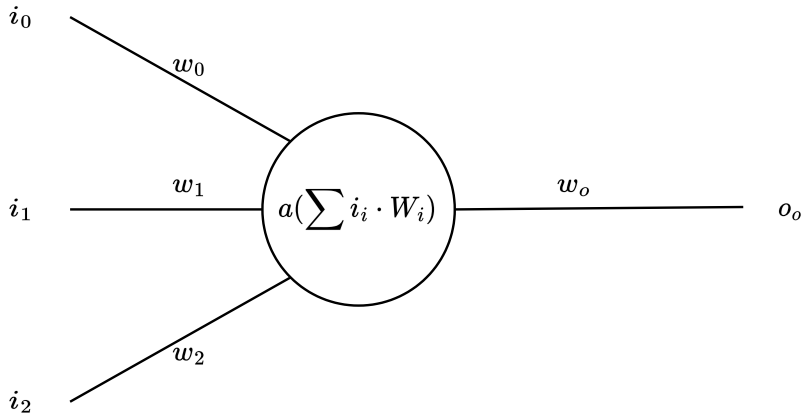


Figure 2.1: A simple perceptron with three inputs and one output. i values are inputs, w values are weights, and o values are outputs. $a()$ is the activation function.

by a final weight to determine the output value of the Perceptron. A mechanism, alongside training data formed from pairs of inputs and expected outputs, is then applied to modify the weights such that the model outputs match the expected outputs. The architecture of a Perceptron with three inputs and one output is shown in Figure 2.1 and is defined as $f(x) = w_o \cdot a(\sum i_i \cdot w_i)$ where w is a learnt weight, i is an input value and a is an activation function. The summing and activation operation is referred to as a neuron and is inspired by the human brain. The activation function can be any non-linear function. Sigmoidal functions,e.g. $\sigma(x) = \frac{1}{1+e^{-x}}$, are historically popular [137], although the Rectified Linear Unit (ReLU) [2], which replaces negative input values with 0, i.e. $\text{ReLU}(x) = max(0, x)$, has become more common in modern deep learning applications.

A common extension to the Perceptron, especially as computing power has increased, is the Multilayer Perceptron (MLP). MLPs have the same basic approach as the Perceptron; however they allow for multiple neurons both in terms of in a 'layer', e.g. inputs are connected to many neurons rather than just one, and multiple layers of neurons, i.e. the output from the first set of neurons is feed into a second layer and so on. MLPs work well for data that exists as a vector of distinct features with no spatial relationship between features. This thesis does not particular use of data in this form as audio-visual and language livestream data is too complex to be formatted in such a manner. However, MLPs are also extremely useful in more complex networks where data is first processed by one of the techniques. For example, often, a model, e.g. a highlight classification system, will first process complex data such that a single vector of latent features is recovered. Then an MLP can be applied

to these latent features to decide whether this data is a highlight.

2.1.2 Convolutional Neural Networks

While MLPs work well for tabular and vectorised data, they do not work well in situations where the input is not a rank one tensor or where there is a spatial relationship between features. Digital photographs are one example of input data with a tensor rank greater than one which contain a spatial relationship between input features, i.e. pixels. This is because a digital photograph is represented as a tensor with shape $h \cdot w \cdot d$, where h is the number of pixels an image has in the y dimension, w is the number of pixels an image is in the x dimension, and d is a depth value relating to the number of colour channels an image has. Usually, for a colour image, d is three because computer graphics are based on a red, green, and blue (RGB) colour channel system. Likewise, there is only a single d dimension for a greyscale image.

Image data has a complex three-dimensional shape and a spatial relationship between features that is not contingent on their global location within an image. Imagine a machine learning model trained to detect if an image contains a human face. Intuitively, it is more important to identify if an image contains groups of features that resemble an eye or a mouth and that these groups are the correct distance apart than it does to train a model to recognise that certain features are located in exact locations, given that images are often cropped differently or taken from different angles. Without this additional characteristic, it would have been possible to flatten an image into a single rank one tensor and then treat it as tabular data, e.g. by modelling via an MLP. Figure 2.2 demonstrates this issue, using synthetic images of digits.

Convolutional Neural Networks (CNNs) [3] are a solution to this problem because they can be formatted for input data tensors with a rank greater than one and also allow for spatial modelling between features. They achieve this by replacing the neurons in a typical neural network with convolution kernels, the weights of which are learnt during training.

Convolution kernels are an established technique in wider image and signal processing applications. For image processing, each kernel is a small matrix of weights, typically 3×3 or 5×5 which are applied in a sliding window fashion to each pixel in an image to produce a new value for that pixel based on surrounding pixel values. Equation 2.1 demonstrates a generalised application of a convolution kernel w with dimensions $2a + 1, 2b + 1$ to an input

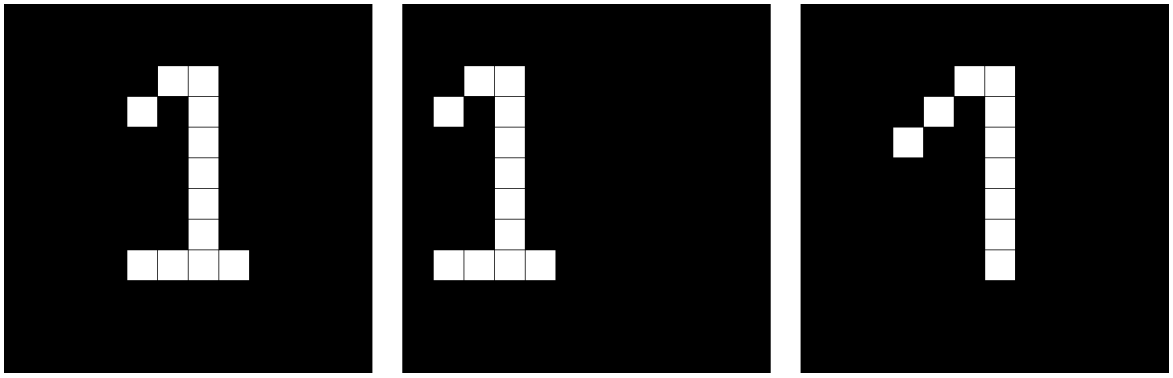


Figure 2.2: Synthetic images demonstrating the need for spatial modelling for images. Left, the digit '1'. Centre, the digit '1' is offset to the left. Right, the digit 7. Note that the images left and centre should be classified as the same, but if the input data is flattened and the values are compared, the right image is closer to the left image than the centre image.

image $f(x, y)$.

$$g(x, y) = \sum_{d_x=-a}^{a} \sum_{d_x=-b}^{b} w(dx, dy) f(x - dx, y - dy) \tag{2.1}$$

When these kernels are determined by humans, i.e. during traditional image processing, they are often designed for a single task, e.g. ridge detection or image blurring, examples of which are demonstrated in Figure 2.3. However, inside a CNN the kernel weights are learnt during the training process. Thus they do not necessarily map to human-understandable concepts such as blurring. Further, there can often be many kernels per layer, sometimes hundreds, allowing a single CNN to learn to identify a wide range of visually important features.

CNNs are not just constructed of these convolutional layers. There are several other aspects. Firstly, pooling operations are ubiquitous [61]. A pooling layer takes the input from one convolutional layer and reduces its size prior to the next convolutional layer. There are many mechanics for this, although the most common is Max Pooling. Max Pooling works by sliding a filter across the output of a layer and, for each position, selecting the max value to carry through to the next layer, illustrated in Figure 2.4. This has the effect that the input to the next layer shrinks for each pooling operation. CNNs tend to have many convolutional and pooling layers such that while the input size may be significant, e.g. $224 \times 224 \times 3$ is a typical size [187], after processing, the output can be very small. This output is often flattened into a single vector to be used as input to an MLP, which then performs classification. The CNN

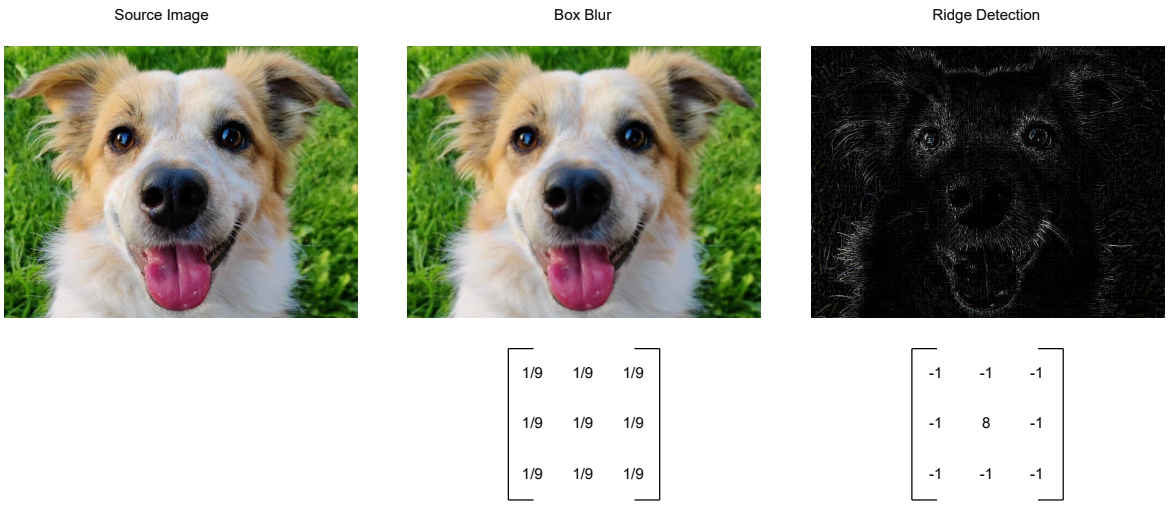


Figure 2.3: Examples of hand crafted convolution kernels used in image processing.

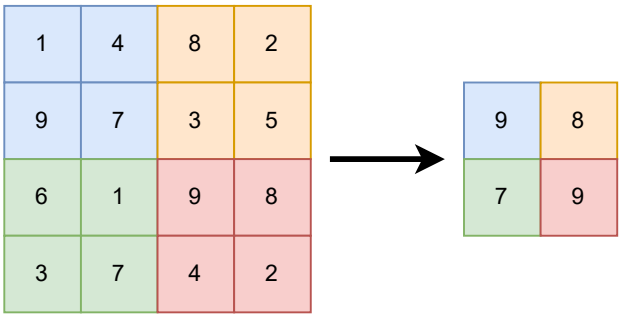


Figure 2.4: Example of 2×2 Max Pooling with a stride of 2

segment of the network is referred to as a 'feature extractor'.

So far, the application of CNNs to image processing has been discussed. However, the convolutions can operate across an arbitrary number of dimensions. While 2D convolution is very popular for image processing, 1D convolutional neural networks are commonly used for audio processing, e.g. [72, 136]. Additionally, recent works in video processing have begun to apply 3D convolution to visual video data, e.g. [84], an approach which is explored in Chapter 4.

2.1.3 Recurrent Neural Networks

The previous section explored the use of CNNs for spatial modelling. While CNNs can also be used for temporal modelling, e.g. 1D CNNs for audio processing or 3D CNNs for video processing, it is more common for a specific layer to be utilised to model temporality, namely a Recurrent Neural Network (RNN). An RNN is constructed of a series of recurrent layers

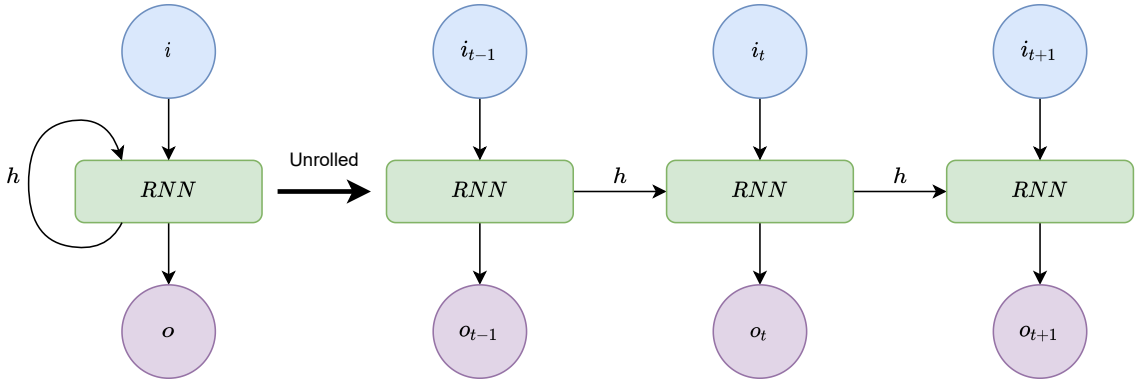


Figure 2.5: Basic RNN neuron architecture. Left shows a single pass through the network, right unrolls this across several time stamps.

containing one or more recurrent neurons. A recurrent neuron works by retaining a hidden state h between input samples. This hidden state is then added to the next input state such that the output for that sample is biased by previous samples, $f(x_t) = a(\sum xi_t \cdot w_i + h_t)$. h for the next sample is then calculated based on a learnt weight, $h_{t+1} = h_w \cdot a(\sum xi_t \cdot w_i + h_t)$. Figure 2.5 demonstrates this architecture.

Basic RNN neurons can, in theory, model relationships along an arbitrarily long sequence. However, in practice, the long-term gradients tend to either 'vanish', i.e. tend towards zero, or 'explode', i.e. tend towards infinity. This can make training RNNs difficult, especially for long sequences. One of the most popular variants of the recurrent neuron is the Long Short-Term Memory (LSTM) neuron [73]. The LSTM aims to enable long-term modelling. This is achieved by adding an internal state cell for each neuron. The state consists of three gates. The 'forget gate' is responsible for learning how much information from previous samples to retain and how much to 'forget'. The 'input gate' is responsible for learning how much of the current input to use. Finally, the 'output gate' is responsible for learning how much to output. The forget gate is calculated via $f_t = \sigma(x_t \cdot Uf + h_{t-1} \cdot Wf)$, where Uf is the input weight and Wf is the hidden state weight matrix; in Figure 2.6 this is represented by the left-most σ operation. The input gate is calculated via $i_t = \sigma(x_t \cdot Ui + h_{t-1} \cdot Wi)$ where Ui is an input weight and Wi is the hidden state weight matrix associated with the input; in Figure 2.6 this is represented by the middle σ operation. Finally, the output gate is calculated via $o_t = \sigma(x_t \cdot Uo + h_{t-1} \cdot Wo)$, the right-most σ operation. LSTMs have two hidden state values. The hidden state h is responsible for short-term memory, and the cell state c is responsible for long-term memory. The hidden state for a time step is calculated as

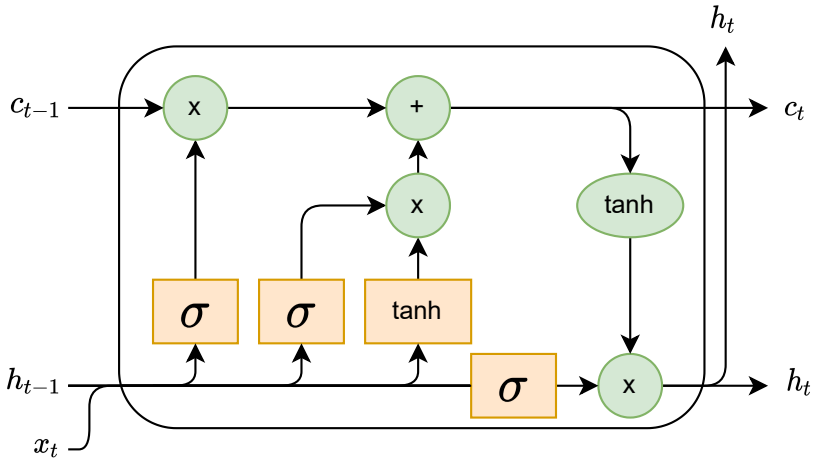


Figure 2.6: Long Short-Term Memory (LSTM) Neuron Architecture.

$h_t = o + t \cdot \tanh(Ct)$ where tanh is the hyperbolic tangent function, $\tanh = \frac{e^x - e^{-x}}{e_x + e^{-x}}$. The cell state is calculated as $c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(x_t \cdot Uc + H_{t-1} \cdot Wc)$ where Ux and Wc are both learnt weights. This ability for LSTMs to model long-term memory without vanishing or exploding gradients means that it performs very well across many tasks compared to basic RNNs. Therefore LSTMs have become ubiquitous as the go-to time series modelling neuron.

The biggest weakness of the LSTM neuron is that the additional weights required to model the three gates and two states results in slower training and inference compared to a basic RNN. However, the forget gate approach is very valuable. Therefore, more recently, Han et al. [34] proposed the Gated Recurrent Unit (GRU), a neuron which retains a forget gate, thus being able to model long-term relationships, but is more straightforward than an LSTM because it omits the output gate. GRU layers have shown good performance in many applications when used as a replacement to LSTMs, e.g. [39], although there can sometimes be variations in performance metrics, e.g. recall, between models trained with the two approaches despite overall similar performance [65]. The GRU architecture is displayed in Figure 2.7.

All of the RNNs discussed so far have been unidirectional. Each sample in a sequence is biased only by those samples that occur before it. However, it is possible that for certain problems, the predictions for a given timestep should be biased on samples before and after the input. For these problems, often a Bi-Directional RNN is used. Here each layer in the RNN consists of two sets of neurons. One set is fed an input sequence in the standard order, i.e. $[i_0, i_1, i_2 ... i_t]$, and one set is fed the reversed sequence, i.e. $[i_t, i_{t-1}, i_{t-2} ... i_0]$. This

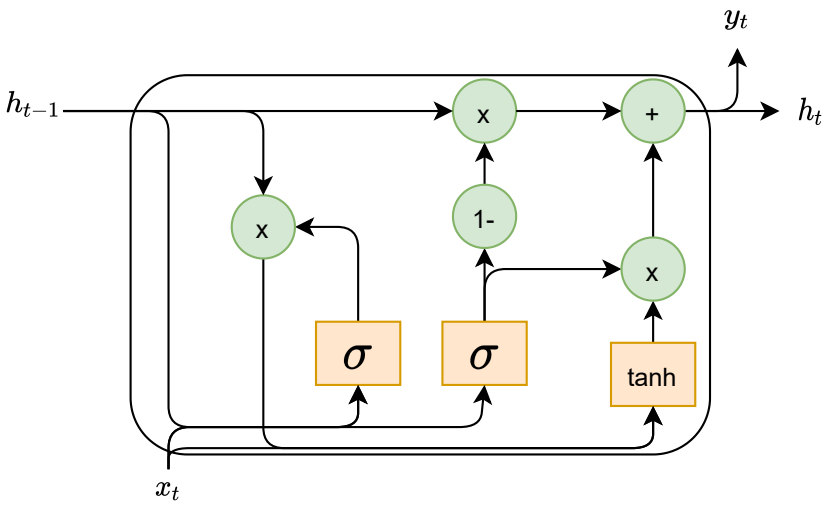


Figure 2.7: Gated Recurrent Unit (GRU) Neuron Architecture.

doubles the layer weights. However, it is necessary if a training target for a time step relies on information from samples that appear before and after it. This approach has been used in certain livestream highlight detection models, e.g. [69].

2.1.4 Word Vectorisation

MLPs, CNNs and RNNs all rely on inputs formatted as a tensor of scalar values. This is because, as shown in Figure 2.1, the heart of neural networks is the multiplication of input values by weights, which are then improved throughout training. This poses a serious problem for any neural network approach to modelling language because there is no simple way to convert words to scalar values. As a result, considerable research has been carried out into the process of 'word vectorisation'. That is, finding some way to convert a word into a meaningful vector of scalars. This processing is foundational in studying neural network-based natural language processing models [126, 205]. These word vectors are then usable in downstream tasks, e.g. an MLP or LSTM, [12]. An advantage of many word vector training models for this domain is that they do not require prior knowledge of the semantic meaning of tokens, which is useful when dealing with livestream-specific words and emotes whose meanings are uncertain.

There are multiple methods for generating these word vectors, and each has similar components. First, they take some input data sampled from a corpus of text. Next, they project the input into a latent vector space. Finally, an output is generated and compared

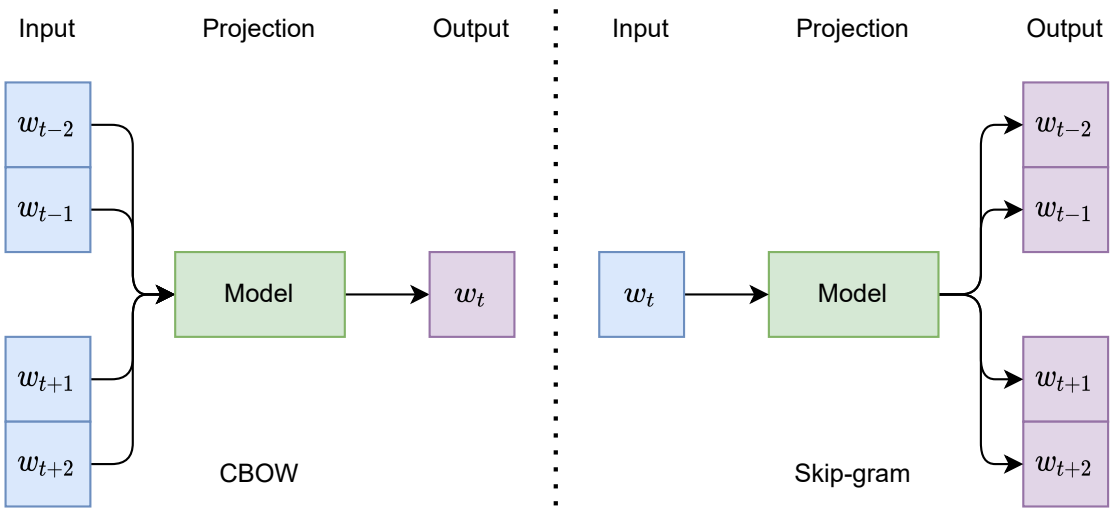


Figure 2.8: Continuous Bag-of-Words (CBOW) and Skip-Gram training schemes.

to the training target. This process aims to learn a vector space where tokens with similar semantic meanings are co-located. In theory, this vector space is such that meaningful vector operations can be applied to words. For example, the result for 'king' minus 'male' may result in a vector co-located near words such as 'queen'. Two popular training methods are Continuous Bag-of-Words (CBOW), and Skip-Grams [126]. Both techniques train a model by sampling a sequence of text. The Skip-Gram model selects the middle word from a sequence as input and expects the model to output the other words in the sequence. CBOW is the opposite, where the central word is predicted from its surrounding context. Figure 2.8 demonstrates these schemes. While these techniques are powerful, they can both be computationally expensive during training given they output, for each word, a distribution across the entire vocabulary, i.e. while CBOW is trained against a single token, it must output a value for each word and thus the final layer has many weights.

Of particular interest to the research presented in the first study of Chapter 5 are Skip-Gram Negative Sampling (SGNS) methods [128], which are particularly attractive for two reasons. Firstly, they are weight-efficient compared to CBOW and Skip-Grams because they reframe the problem into a classification problem where two input samples are provided, and the model must classify if they are co-located. Secondly, they require data in the format of two tokens and a label. This label is usually a $[0, 1]$ binary value, describing if the samples are close together in the corpus, a formulation which can be modified to accommodate the temporal aspect of livestream chat. Analysis of SGNS models has shown that they gener-

ate unusually shaped vector spaces [130] due to the negative sampling objective, although they still retain the ability to encode semantics. A detailed discussion of SGNS methods is presented in Chapter 5 and thus is omitted from this literature review.

2.1.5 Transformers

A significant update to word vectorisation approaches is recent work into using Transformer architectures to model natural language, e.g. [212]. Transformers have become a go-to method for 'Language Models' (LM), models designed for general-purpose language understanding. These models can then be fine-tuned for a range of downstream tasks, e.g. as discussed in [46]. The term transformer is used both to describe a transformer model, e.g. an encoder-decoder machine translation system as shown in Figure 2.9 and a component in the system, the 'transformer block', a set of layers which lend the model its name. Standard transformers have two parts, the encoder and the decoder, both of which consist of a set of transformer blocks [212]. The encoder takes a text segment and converts it into a vector of latent features, much like word vectorisation techniques but applied at the sequence, e.g. sentence, level. The decoder then decodes this latent vector back into text. However, the target text the decoder is trained to predict is often different from the input data. For example, the transformer may be trained for neural machine translation and thus expected to output translated language, or trained in natural language generation where the transformer outputs the next word in a sequence. Encoder-only models are often used for tasks where decoding is not required, e.g. text classification models.

Before processing by the transformer blocks, the first stage in a transformer model is to find a vector for each word in the input. These vectors are conceptually identical to the word vectors discussed above, although they are trained end-to-end as part of the transformer model rather than learnt separately through co-location schemes such as CBOW or Skip-Grams. Next, each word vector is combined with a positional encoding vector representing that token's position in the input sequence. There are multiple ways to generate this positional encoding. For instance, initially, a sine encoding was proposed [212] where the encoding was generated as a series of sine values uniquely representing positions in a sequence. Alternatively, other works, e.g. [46], have moved away from the hard-coded sine encoding and instead implement a learnt position vector, which is learnt in the same manner as the word vector. The choice of positional encoding technique is important. For instance,

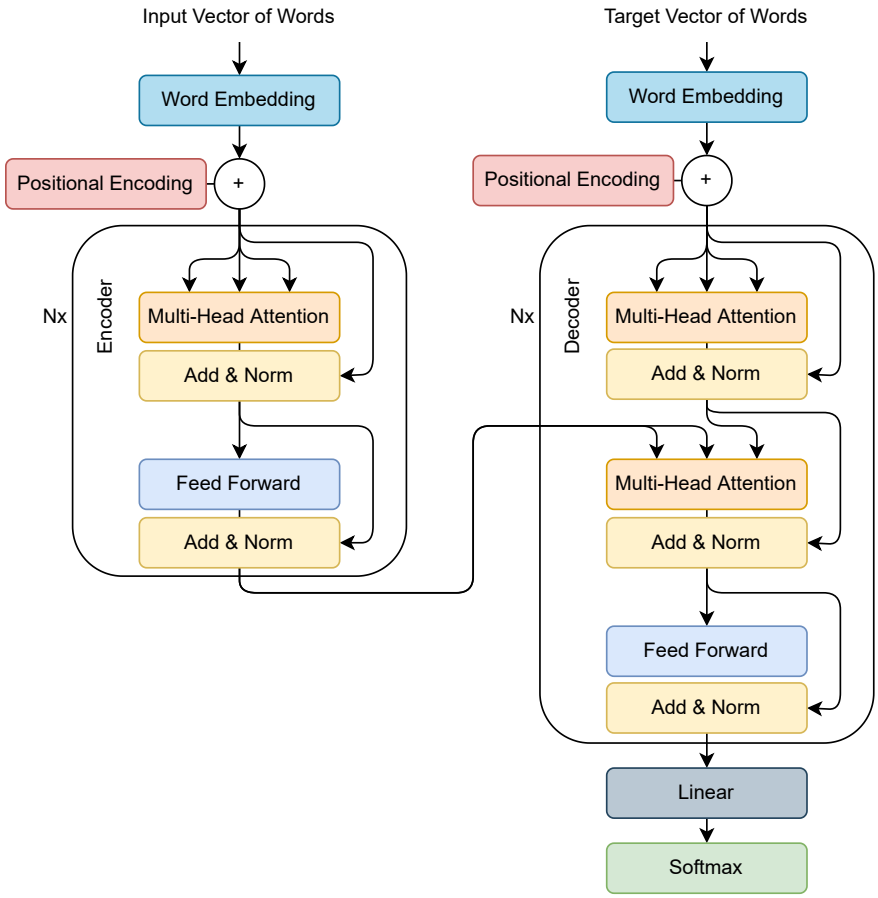


Figure 2.9: A generic transformer architecture, e.g. for neural translation or language generation. Adapted from [212].

the sine approach offers relative positional encoding, i.e. comparing two encodings reveals how far apart the source words were, and absolute encoding, i.e. where the word appears in the sequence. In contrast, a learnt vector encoding is only capable of absolute encoding, but because the vector space is learnt during training, in theory, the quality of the absolute encoding should be better. An, albeit more complex, alternative which retains both the quality of learnt vector encoding and the absolute/relative position properties of sine encoding is to generate a pairwise encoding for each pair of tokens in the input sequence [75, 185].

Once a vectorised representation of the input sequence has been generated, the next step is to pass this data into a transformer block. This block consists of several layers which can be repeated an arbitrary number of times, e.g. as with CNN or RNN layers. The key feature of the transformer block is the multi-head attention layer (MHA). MHA allows each word in the input to 'attend' to other input words, which allows the embeddings for each

word to become conditioned on the other words in the input. For each attention 'head', this is carried out by first calculating Key, Query, and Value embedding tensors for the input embeddings. The dot product for the Key tensor for an input and all Query values is then calculated. This, combined with the Value tensors, allows each input to understand which other inputs are important to them.

MHA uses multiple attention heads, which allows the model to attend to tokens in different ways, e.g. learning different types of relationships. In some ways, this is similar to the multiple filters used in CNNs. Using an MHA layer on its own is equivalent to modelling the input data as a bag-of-words, i.e. there is no understanding of the order of tokens. This is why the positional encoding discussed above is vital for a transformer, especially in a language modelling context where the order of words is critical. The final step of the transformer block is to process the output from the MHA layer using a fully connected feed-forward layer, i.e. one layer from an MLP. This processes the MHA output ready for the next stage, be it another attention layer or the downstream task. Figure 2.9 details an encoder-decoder architecture. The encoder follows the mechanisms outlined above. During training, the decoder is passed the ground truth target sequence and includes a layer where this sequence, after encoding and an MHA layer, is combined with the output from the encoder.

2.1.6 Multimodal Machine Learning

Livestream data is inherently multimodal, i.e. an event, in this case, a livestream, is described by multiple 'views', e.g. visual data, audio data and textual data. The problem of analysing multiple views is an open challenge and an active area of research. Generally speaking, attempts to tackle this problem have focused on the question of fusion, i.e. how can multiple views be joined, or fused, into a single model. There is no consensus on the most appropriate fusion technique, with studies performing fusion in different ways [90, 159]. Some of the most popular techniques for fusion in 'end-to-end' systems, where the inputs are raw data extracted from the video and the outputs are the classifications, include early (feature) fusion and late (decision-level) fusion. While early fusion refers to concatenating raw features or representations, e.g. [141, 217], late fusion is usually performed on bottleneck features, e.g. [183]. Other fusion approaches exist, e.g. model fusion, where separate models are utilised for each view and subsequently aggregated, e.g. EmoNets [88]. Such approaches are

not naturally suited to end-to-end systems, e.g. as presented in the second study of Chapter 3, as it is often not possible to perform the aggregation during training. However, when modalities are trained separately, model fusion becomes useful, especially if fusing features or raw inputs from modalities is complex. Chapters 4 and 6 discusses examples where model fusion is helpful because of challenges in other fusion approaches.

A significant advantage of utilising multimodal data, as Ngaim et al. [141] note, is that exploiting information from multiple views can significantly aid learning from noisy and imperfect data, for example, when audio is noisy, or there are occlusions in the visual data. This suggests that multimodal fusion may be well suited to modelling livestream data, where noise is introduced due to uncontrolled conditions. Multimodal learning is also crucial for modelling behaviour and affect, discussed in more detail in Section 2.2.3 and Chapter 3 because it is well known that audio information is more suited to predicting emotional arousal and visual data is more suited for modelling valence [209]. Therefore, fusing these views allows for holistic modelling of affect.

2.1.7 Training Methods

There are many ways to formulate input and target data when training a neural network. Perhaps the most common is supervised learning, the paradigm under which the techniques above have been introduced. In a supervised setting a known target label exists for each input sample. Therefore training a model is relatively straightforward. The model is tasked with learning the mapping between inputs and labels. These labels can be derived in many ways. For instance, when training using tabular data, selecting one variable as the training target is typical. To illustrate this, imagine training a model to predict how much a house is worth. There may be historical information about house sales regarding the number of bedrooms, the house's location, if it has a garage and so on, alongside its value. In this case, the model is tasked with learning a regression to map the variables describing a house to its selling price. Then, when the model is deployed, information about a new house can be input, and the model will predict its selling price based on this learnt mapping. It is also possible to generate these ground truth labels by querying humans through a process known as annotation. This is particularly common in emotion detection literature, discussed in Section 2.2.3, where subjective judgements about a person's emotional state need to be made.

While supervised learning is the traditional method for training a predictive model, it has several drawbacks. Namely, it is a requirement that every data sample has a corresponding label. While this restriction may seem trivial for a small, tabular dataset, it becomes an issue for large-scale deep models. In part, the power of deep learning is due to the ability of these complex networks and neuron architectures to leverage massive datasets. Learning on massive datasets not only allows the models to learn very complex mappings, but it is also necessary to prevent common training issues, e.g. overfitting, exacerbated by a deep models' complicated nature. Furthermore, the richer the training data, the more likely that determining a training label is non-trivial, e.g. livestream highlight detection would likely rely on the time-consuming subjective annotation process if supervised training schemes were always used.

Therefore, in recent years other training methods have become more popular. These aim to mitigate the penalty for gathering human-annotated supervised data. Firstly, unsupervised learning is the process of learning from the data itself. Often this is in the form of clustering where the space of data points is examined to find patterns, e.g. traditional machine learning methods such as K-Means [114] or topic modelling [4]. Auto-encoders are a popular approach for unsupervised deep models, utilised in Chapter 3. Here a model, the 'encoder', which can be constructed from any of the building blocks discussed above, is used to generate latent features from an input sample. Then a set of layers, the 'decoder', mirroring the initial layers, is used to reconstruct the original input. Once trained, this autoencoder can be used in a variety of applications. For example, the encoder can act as a feature extractor in a system which requires one but has no way of pre-training on structured, e.g. labelled data. Additionally, auto-encoders can be used in novelty detection systems through reconstruction error based novelty detection, as reviewed by Pimentel et al. [155] and used for highlight detection in Chapter 3. While not directly explored in this thesis, an extension of unsupervised learning is self-supervised learning [48]. Here proxy labels from the training data are used to pre-train a general feature extraction model, which is then fine-tuned on a specific task or used as a feature extractor in conjunction with a predictive model. An example of a self-supervised pre-training task is the transformer pre-training utilised in Chapter 5.

Finally, training methods exist that are neither completely supervised nor completely unsupervised. These are referred to by several terms, such as weakly-supervised learning

or semi-supervised learning. These methods use datasets where some samples have known ground truth labels and some samples have unknown or uncertain labels. These methods can be beneficial for leveraging massive data without the human effort cost required to label and clean the data and where unsupervised or self-supervised learning is not suitable. Semi-supervision relies on the assumption that data samples with similar ground truth values are co-located in latent space and that it is possible to cluster these co-located samples [30]. This is a requirement because there only exists ground truth labels for certain parts of the input data distribution. In particular, Chapter 4 takes a positive-unlabelled approach to training data, where there are concrete labels for only positive training samples. This is a special case of standard semi-supervised learning because only labels relating to one class are known. These techniques can result in noisier training than traditional supervised learning, so research studies how best to compensate for this noisy data. Chapter 4 implements a noise aware ranking model proposed by [218] which itself was inspired by [79]. This approach can mitigate noise by utilising a bagging approach alongside a heuristic noise prior. Additionally, for the binary classifier comparison model, the weighting methodology proposed by [52] is implemented to weight the output by the performance on positive data. Other attempts at learning from very noisy data include [139] which focused on noisy labels in both positive and negative data and proposed probabilistic 'flipping' of labels to tackle this problem. Similarly, Sukhbaatar et al. [196] proposed label flipping for computer vision tasks with convolutional neural networks. Further work into noise label vision tasks includes [108] where 'side' information, e.g. clean data, is used to improve learning under noisy conditions. Reed and Lee [163] applied bootstrapping to this problem, finding that it creates a robust classifier in multi-class scenarios. Additionally, [112] applied Importance Reweighting to noisy label classification. Finally, $\propto$-SVM [224] is a Support Vector Machine model which models the latent labels for unknown data alongside the proportions of known labels. This concept has been applied to event detection in video data, e.g. [104].

2.1.8 Tensor Decompositions

The final deep learning concept discussed in this section is tensor decomposition methods. These are not nearly as common as the methods discussed above but are nevertheless essential to discuss given the application of Tensor-Train decomposition [146, 148] to the fusion challenge in the second study of Chapter 3. The Tensor-Train layer approximately

decomposes a large tensor into a set of smaller, simpler tensors in a weight-efficient manner. Importantly, it can do this without constructing the tensor. Therefore it is possible to utilise tensor decompositions as a fusion mechanism to model high-order relationships between multiple modalities, which are very large tensors, in a weight-efficient manner. This approach is similar to Zadeh et al. [225], where Tensor Fusion was employed to fuse multiple modalities in a sentiment analysis task. Other researchers have used similar approaches to aid different tasks, e.g. Yang et al. [222] utilises a (recurrent) Tensor Train layer directly on the pixels of a video, replacing the convolutional layers. Kossaifi et al. [101] used similar tensor methods on the unflattened output of a Convolutional Neural Network and showed that these types of decomposition could be trained in an end-to-end model. Anandkumar et al. [7] provide a comprehensive overview of tensor decompositions for learning latent variable models and Panagakis et al. [149] provide a discussion of tensor methods for computer vision applications.

2.2 Deep Learning Applications

The previous section briefly introduced this thesis's key deep learning methods. However, deep learning, by its nature, is general-purpose and thus, it makes sense to introduce research where these methods have been applied to problems either similar to livestream highlight detection or related to video game playing in general.

2.2.1 Detecting Interesting Moments in Videos

2.2.1.1 Livestream Highlight Detection

Naturally, the most pertinent prior work is the small body of work examining livestream highlight detection. Livestreaming is relatively new and therefore there is not a huge amount of research on this task, most of which being published during this research. Chu et al. [37,38] examined various facets of *League of Legends* esports broadcasts, building models of highlight detection by focusing on modelling hand-crafted features such as the number of players on screen. They also experimented with event detection by utilising text recognition on in-game messages, e.g. when the first kill occurs. While Chu et al. was an excellent first work in this area, the reliance on hand-crafted features results in techniques that require significant human processing to generate training data and may not generalise well to other games,

e.g. because these features are game-specific. Song [191] examined single frame-based visual highlight detection, i.e. omitting temporality, for esports streamed on Yahoo. The lack of temporal consideration was a conscious decision to enable the system to operate fast enough for deployment within Yahoo Esports. More recently, Wang et al. [213] experimented with rank-based highlight detection for the mobile title *Honor of Kings*. This particular work takes a multimodal ranking approach similar to the proposed approach in Chapter 4. Finally, Fu et al. [58] examined audio-visual highlight detection through a CNN-LSTM architecture, and chat data through a character-level LSTM model discussed in Chapter 5.

The works discussed above focus on audio-visual data (i.e. videos), except for [58]. However, esports are nearly always livestreamed in settings that allow for viewer interaction via text chat. There is some emerging research into how cues from the chat domain can indicate when a highlight occurs in the game. Han et al. [69] utilised a bidirectional GRU in a supervised setting to learn a highlight detection model from text chat only, with promising results which outperformed baseline models. Additionally, Liaw and Dai [109] proposed an attention-based model for highlight detection, utilising the text in crowd-sourced highlight clips as highlight cues. Ping and Chen [156,157] explored vectorised chat messages, focusing on time-lag alignment due to the audience members having to react to events. Finally, Jiang et al. [86] used chat messages to broadly identify the parts of a stream where a highlight is likely to have occurred. Highlight segments are then recovered from these parts by modelling how a user interacts with the video, e.g. pausing and skipping.

Tangentially related, Ishii et al. looked at gameplay generation through highlight cues [80,81]. Hand-crafted highlight features optimised an AI Agent to create exciting gameplay. This work aimed to create entertaining agents for a livestream setting. Similarly, Yang et al. [221] used computer vision methods, e.g. optical flow, alongside a human-determined notion of what entertaining gameplay is, to generate *Angry Birds* agents who played the game in an entertaining manner. These works do not formally model 'highlights' but instead use some notion of which type of events may be considered highlights to a viewer to guide AI game-playing agents.

2.2.1.2 Sports Highlight Detection

Professional sports are a popular domain for event detection research and have natural parallels with this thesis. Ren et al. [164] studied highlight detection in soccer games, studying

four matches with good results, especially when detecting goals scored. In this work, they used a set of hand-crafted features based on audio and visual data, using these features as input to a multi-resolution auto-regressive model that simulates temporal attention to focus on exciting moments in the match. Xu and Chua [220] used not just audio-visual features but also external, text-based information in their work towards the detection of highlights in both soccer and American football. A similar approach, applied to baseball games, is proposed by Chiu et al. [33] where webcast text was aligned with video content detection to determine salient moments. The use of external text logs detailing the game in these works is particularly interesting as, in some ways, it mirrors the chat logs present in livestreams, although it is safe to assume that text logs used in [33] are likely to provide a less noisy highlight signal than chat logs because in a livestream participants are free to type what they wish when they wish, rather than solely focusing on the stream.

Sun et al. in [198] analysed the excitement level of sports commentators using audio features, mainly Mel Frequency Cepstrum Coefficients (MFCCs) and pitch data, to detect highlights. This approach demonstrates the link between affect, as detected in the commentator's voice, and salient moments in the stimulus, i.e. the game highlights. There are clear parallels to this research, especially in personality-driven streams where the streamer is not just the game player but also the commentator. Nguyen and Yoshitaka [142] adopted a cinematography and motion-based approach, whereby they analysed the type of camera shots used to detect highlights, particularly emotional events. Whilst this is not necessarily directly applicable to this research as often the webcam angle is fixed, and players rarely change the in-game camera angle to create drama, video games sometimes employ these techniques, especially when presenting the gamer with gameplay elements such as quick-time events.

Spijkerman and van der Haar [192] utilised convolutional neural networks trained on footage directly obtained from the Formula 1 video game to detect highlights in broadcasts from physical Formula 1 races. This approach is made possible because the video game is designed to look as realistic as possible. In this work, the models were trained to detect cars within a scene as the authors suggested that such signals are indicators for highlights. Additionally, Moodley and van der Haar [133] also used CNN models for detecting events in Cricket matches. Specifically, an AlexNet was trained to detect when a 'stroke' occurred, i.e. the batsman hit the ball. Finally, approaches focused on Optical Flow calculations, the change in a scene between frames, show promise when summarising sports videos, e.g. [125].

Zhu et al. [230] utilised behaviour analysis alongside optical flow to detect the behaviour of racket sport players and thus classify in-game events. While some of these sports highlight detection methodologies are not suitable for livestreaming, e.g. they fuse techniques that have since been superseded or use data unavailable in a livestream, they do serve as inspiration for the types of techniques that are useful, e.g. convolutional networks and the use of optical flow vectors.

2.2.1.3 Video Highlight and Event Detection

As well as detecting interesting moments in sports, it is also important to consider the literature on the topic of detection events in general videos. Simonyan and Zisserman [186] and Feichtenhofer et al. [55] utilised optical flow combined with object detection to detect actions performed by humans. This two-stream network can detect a single action from a pool of 101 actions, such as an archer drawing a bow. These works present promising results with the highest reported accuracy of 90.62% in [55]. This is interesting for this research as the types of actions presented in the dataset are also present in some popular video games, especially those that employ a 3D 1^{st} or 3^{rd} person viewpoint. More recently there has been work on state-of-the-art inflated 3D convolutional networks from [28] pre-trained for action recognition on kinetics tasks, [91], which are utilised in Chapter 4. There are other examples of using motion features to understand scenes. For example, Giannakopoulos et al. [63] considered motion, audio data, and text data gathered from video comments in their work on detecting violence in videos shared on video-sharing platforms. When all three modalities were considered, the proposed system performed well at 82% accuracy. However, using visual data only was not particularly successful, with 59% accuracy, although the authors offer no suggestions as to why this is except that the extracted feature set could be improved. While applied to a domain completely dissimilar to this research, Xu et al. [219] used unsupervised learning to detect events partly based on motion when analysing scenes of pedestrians walking. This work was focused on detecting anomalies that, when reframed into a video game setting, may constitute rare in-game events such as winning and losing.

In terms of rank-based highlight detection for videos, a technique utilisedm in Chapter 4, there have been several recent works. For example, Sridevi and Kharde [193] discussed a pairwise approach to ranking for video summarisation. Hong et al. [74] proposed a multiple instance ranking architecture for this problem. Additionally, [218] is a crucial piece of work in

video highlight detection. The authors utilised crowd-sourced videos assuming that a short clip would contain a highlight. It used ranking models for state-of-the-art highlight detection in a range of kinetic tasks, e.g. surfing. Sun et al. [197] took a ranking approach to detect highlights in human action videos utilising crowd-sourced data. Additionally, Video2Gif [67] is a rank-based detection system trained to generate gifs from longer videos using human-generated gifs as training data. Finally, Yao et al. [223] also utilised ranking for video summarisation in first-person videos.

2.2.2 Deep Learning in Video Games

Some work has examined how deep learning can be applied to video games. These span an extremely wide range of applications. For example, Procedural Content Generation, e.g. [111, 171], game-playing agents, e.g. [132], and player behaviour modelling, e.g. [20] are all popular research topics. Two particular areas are most relevant to this thesis; analysing gameplay content and analysing player experience. These will be explored below.

2.2.2.1 Game Content Analysis

Game content analysis describes a broad range of methods that attempt to learn meaningful information about the game content, e.g. events in a gameplay session, through techniques such as analysing the game screen. Guzdial and Riedl [66] developed unsupervised techniques for building complete game levels from observing gameplay videos. This system first determines sections of game levels by comparing the content in subsequent frames. Once the level has been built, a probabilistic graph model is derived, representing how components appear in levels. Finally, new content is generated from this model. Lewis et al. [107] used Starcraft replays to discover strategies. This was achieved using reasonably simple statistical models after first extracting features from the reply, such as actions per minute and the ratio between macro and micro actions. This was achievable because Starcraft replays are not merely videos. They also contain other extractable metadata. The authors learned key strategy points, such as distributing units around the map and keeping track of them, which leads to more success than grouping units. Similarly, Rioult et al. [170] used the player's in-game positions to predict winners in Defence of the Ancients, arguing that topological clues can help in analysing performance and team balance in MOBA games.

Trivedi et al. [203] proposed contrastive learning for learning generalised game repre-

sentations based on the game screen, finding that it appears to be better than supervised learning. The authors suggest that these representations can be used in various tasks, for example, gameplaying agents, game world models, procedural content generation and player modelling. Similarly, [132], while known as a critical piece of reinforcement learning literature, used a CNN approach to learn a representation of the game state from which an agent was trained. More generally, general video game playing has moved to provide a pixel-based representation of the game state for agent learning, [103]. Finally, particularly interesting to the first study in this thesis, Alvernaz and Togelius [5] used an autoencoder to take a video game scene, in this case, the Visual Doom environment, and learn a latent space representation of the game scene, transforming the 120x160x3 pixel image into a 128 feature representation. These 128 features were then used in a behaviour network, trained to play the game using neuroevolution, expressly Covariance Matrix Adaptation Evolution Strategy. While, unfortunately, the results were not as good as competing algorithms such as Deep Q-Learning, this work does show that a lower latent space representation can still be used as input to an AI agent with some success.

2.2.2.2 Player Experience Analysis

Most prior work into audio-visual player experience models do not use livestreaming platforms as their data source. For example, the Platform Experience Dataset [89] is a data set of players playing the platformer game *Infinite Mario Bros*. This data set has been utilised in several studies. For example, Asteriadis et al. [9] used this data set to cluster player types. Additionally, Shaker et al.'s [184] work focused on modelling player experience using visual and behavioural cues. This work explored fusing visual cues from captured webcam data and behaviour data from in-game play traces. A model of neuroevolutionary preference learning and automatic feature selection was proposed, which provided very accurate models of engagement (91%), frustration (92%) and challenge (88%). This is perhaps the most exciting work into visual affect detection for video game players because it uses real-world video game data and appears to perform well in various affective modelling tasks important to video games, such as engagement, frustration, and challenge. However, it relies on in-game data to train the model. This makes it very difficult to apply to a video game streaming setting as game data is challenging to gather.

Additionally, 'off-the-shelf' affect models, i.e. software-as-a-service solutions, have been

used to study player experience. For example, Blom et al. [19] explored how these models could be used to personalise game content, and Tan et al. [200] performed a feasibility study exploring player experience modelling via visual cues, concluding that facial analysis is a rich data source which warrants more exploration. Another work exploring visual affect detection applied to games is Cao et al.'s [25] work which proposed a purposefully lightweight computational model that can be deployed in a video game setting with little overhead. The model, an improvement on and modification of Local Binary Pattern, performed very well; however, the test set was small and contained participants performing posed emotions. Thus it is unclear how well this approach would translate to an 'in-the-wild' affect detection scenario such as real-world livestream video game players.

Makantasis et al. [117,118] used supervised learning to learn player affect not from images of the player's face but from the gameplay itself, both in terms of visual data, using CNNs, and audio data, using MFCCs. This is particularly relevant to this thesis, considering that player affect is likely to be a strong indicator of highlight moments. The challenge with such an approach is that it requires a dataset of affect annotations to be gathered, which, as discussed in Chapter 3, is a challenging task. Mavromoustakos-Blom et al. [120] explored if in-game affect cues correlated with post-game experience reports for *Hearthstone* players. This work is especially interesting for modelling important emotional events for the player, rather than those important moments for the viewer, as with highlight detection. Research has also explored the impact that a player's emotions have on other players in multiplayer games, for instance, [182] which showed that manifestations of player emotion impact other players.

Such is the interest in affect modelling for game players that several datasets of annotated gameplay have been released, although none are focused on the livestream paradigm. The MUMBAI dataset [49] focused on board games rather than video games but is interesting because video footage of the players is gathered in a naturalistic setting, similar to a livestream setting. The AGAIN dataset [124] covered nine games and, like the works mentioned above, contained gameplay and annotation without the player's voice or facial features.

2.2.3 Affective Computing

2.2.3.1 Developing Computational Models of Affect

Previously the small body of work into games as a medium for affect elicitation has been discussed. More generally, there is a wealth of research in affective computing, which has influenced the various computational models used in this thesis, especially when considering personality-driven livestreams. These models are typically developed using data dissimilar to a livestream setting, for example, using posed actors, e.g. the Extended Cohn-Kanade Dataset [116] or synthetic interactions, e.g. the SEMAINE dataset [123]. In contrast, streaming platforms offer a more naturalistic setting where affect and behaviour are spontaneous. Computational models of affect have been built using a variety of data sources. Perhaps most common are visual and audio affect models, discussed in detail below. Textual models have been developed from written text through a process often referred to as sentiment analysis, [17]. It is also possible to develop models based on the physiological signals released by the body, such as Galvanic Skin Response data, e.g. [10]. Other techniques for modelling affect exist but are significantly less common. Examples include DeepMood [26] which uses the way participants input data into their mobile phones to model affect. Additionally, these modalities can be combined into multimodal, or multi-view, models of affect. The models are often trained on a classification task, in the case of predicting categorical emotions such as Ekman's '7 basic emotions' model [50, 51] or a regression task, if using a continuous model of emotion such as the circumplex model [174]. The key difference between these tasks is that 'basic emotion' classification aims to detect particular emotions, in this case, fear, anger, joy, sadness, contempt, disgust, and surprise, whereas the circumplex model splits affect into two axes. 'Valence' describes whether the person's emotional state is positive or negative, and 'Arousal' describes how strongly the person feels this emotion.

2.2.3.2 Speech Emotion Recognition

Audio models of affect often focus on using extractable features such as Mel-frequency Cepstral Coefficients (MFCCS), spectral features such as a Fourier transform or other values, e.g. fundamental frequency, to determine the affect of the speaker. There are various methods for doing this with no fixed consensus best approach. For a holistic overview of speech emotion recognition, the overview by Schuller [179] is very informative.

MFCCs are extremely popular for affect detection and have been used in multiple studies. An early approach to using these hand-crafted features was Busso et al. [24] who used the features derived from the pitch of the fundamental frequency. The study discovered that features such as the average and range of pitch over time are important for determining emotional speech. The approach, while simple, showed reasonable success, 77% accuracy, when performing binary classification between neutral and emotional speech. Kishore et al. [97] used MFCCs and a Subband based Cepstral parameter as input to a Gaussian Mixture Model in a classification task, using classes based on the 'basic emotions' model. Another work using MFCCs to tackle classification is Lalitha et al. [105] who trained a neural network as a classifier, again using the 'basic emotions' model. This work's interesting insight is that sad audio samples were often misclassified as happy samples. This demonstrates the underlying difficulty that affect modelling from audio data has when detecting valence, compared to arousal, where it excels. Aounai and Ayed [8] also used MFCCs and the 'basic emotion' model. This work aimed to improve on Kishore et al. [97], finding that a stacked Support Vector Machine autoencoder architecture performed best, with a classification accuracy of 73.01%. Ghosh et al. [62] used Fourier Coefficients and MFCCs fed to a variety of autoencoders for learning affect from speech using the same database as [24], further reinforcing the value of MFCCs for emotion classification tasks.

Some approaches use unsupervised learning for feature extraction rather than predetermined features or spectral features (e.g. MFCCs). One example of this is Huang, and Mao [76] who used methods such as K-means clustering, sparse auto-encoders and sparse restricted Boltzmann machines to learn feature representation of affect and suggest that this feature learning improves results over using raw data when tested on a variety of speech corpora. Unsupervised learning also appears popular in domain adaptation of pre-trained affective models. Sun et al. in [198] analysed the excitement level of sports commentators using audio features, mainly MFCCs and pitch data, to detect highlights. In doing so, they took a trained Gaussian Mixture Model model and updated it through unsupervised learning to account for differences between the training and test data. Additionally, Wang et al. [54] used a deep autoencoder for feature extraction. Zhang et al. [228] trained across six corpora of affective speech data and argued that whilst unsupervised learning does not outperform supervised learning, adding unlabelled data to the dataset can improve results. Furthermore, Zhang et al. pointed out that a plethora of unlabelled data exists, whereas labelled data

is more challenging to produce. This is true for livestream data. Deng et al. [44] identified similar issues to [198] where variance in the training data and test data considerably impacts the quality of the affect detection. They present an Adaptive Denoising Autoencoder capable of affect modelling across corpora, improving performance against other baseline approaches to overcome this. Lastly, Chang and Scherer [29] used a deep convolutional generative adversarial network and unlabelled data to learn representations of emotion in speech, allowing their system to outperform supervised only approaches on the IEMOCAP dataset [22].

2.2.3.3 Visual Affect Modelling

Visual models are also a popular approach for modelling affect [227]. Often they consist of taking images of participants' faces, either captured as stills or extracted from a video, annotating them and then using a supervised learning algorithm to train a model, often constructed at least in part from a convolutional neural network. As briefly mentioned in Section 2.2.2.2, there are several commercial solutions for visual affect modelling, e.g. by Microsoft, Affectiva, nViso, and Kairos [60]. Numerous solutions to visual modelling exist because visual data can more easily model basic emotions, given that these categories tend to relate more to valence than arousal.

Kollias et al. [99] explored visual affect detection using retrainable deep networks for online learning. While online learning is not directly explored in the thesis, there are obvious benefits to such an approach when deploying a highlight detection model, especially considering how much data is generated on Twitch.tv. Jain et al. [82] used a CNN-RNN architecture for emotion classification on video data, similar to the approach in the second study of Chapter 3. Their approach achieved state-of-the-art results on the dataset. Deep models are complex by nature, which can be a problem when modelling visual affect because images are feature-rich, resulting in potentially slow inference. This has resulted in research focused on deep, but fast models. For example, Schwan et al. [180] applied fine-tuning to a modified VGG-16 network [187] with good results. Additionally, in [77] the authors used a VGG-16 network for real-time facial and emotional recognition. Finally, Salunke and Patil [175] used a small, lightweight CNN architecture for training categorical emotions, with good results.

There has been some work investigating visual affect modelling but not using RGB images and facial/body gesture recognition for affect detection. For example, Szwoch and Pawel

[199] investigated detecting affect using a Microsoft Kinect sensor to gather depth data. Whilst this work is not hugely relevant to the study of game streamers, it uses technology developed for games, the Kinect, and hints at potential affect detection applications within video games. Recently, DeepPhys [31] used visual data through a webcam, to measure affect by detecting physiological data through visible changes in participants. DeepPhys can determine heart and breathing rates data using convolutional attention networks on the webcam data. This approach to recovering physiological data is exciting for livestream affect modelling because heart rate and breathing rate are likely indicators of player experience, especially during particularly tense or exciting moments in a game. Soleymani et al. [190] examined continuous valance detection using EEG signals and facial expressions. This work proposed affect detection for video highlight detection, which is very fitting for this thesis and is similar in intent to the system proposed in the second study of Chapter 3.

Finally, hand-crafted visual features have also been used, rather than feature extraction from raw images. For example, optical flow of key facial points was used in [188] to predict happy, sad or angry faces. Similarly, Strupp et al. [195] used point-based features to detect basic emotions with applications in robotics research. Support Vector Machines were used in [161] on hand-crafted features, although the model could only predict categorical basic emotions. Of potential interest to livestreamed affect detection is the work of Miyakoshi and Kato [131] who used Bayesian Networks applied to key facial points to model affect in partially occluded faces. This approach seems to show promise, although the use of the 'basic emotions' model makes it of questionable worth compared to approaches trained on circumplex style annotations.

2.2.3.4 Multimodal Affect Models

Generally speaking, the visual models outlined perform well at detecting. On the other hand, audio models are generally good at detecting arousal because vocal features reveal how strongly people feel rather than what they are feeling. Therefore, given the presence of audio and visual data in most videos, it makes sense to combine these modelling techniques to develop robust ways to model both arousal and valence.

Fusion is the biggest challenge for multimodal affect models [159]. Busso et al. [23] proposed a method for emotion recognition using facial data, expressly expressions, and speech data, examining a selection of fusion techniques. Firstly they used a decision level fusion

technique where the decision, in this case, an emotion classification, is made by giving the decisions made by each model a weighted score and classifying by the sum of these weights. They also explored feature-level fusion, where latent space features from all modalities are combined as inputs into the classification decision process. They found that, as expected, performance using multimodal data outperformed unimodal data. Additionally, they found that decision fusion and feature fusion had similar overall results, but there were variances for certain emotions. Similarly, Kessous et al. [94] conducted a study into emotion recognition for speech-based interaction, e.g. conversations, using facial data, body features and speech. A collection of Bayesian classifiers was employed where one classifier took input from all input views and performed feature-level fusion. In contrast, outputs from the other models, each trained on a single, different, view, were used for decision fusion by combining the posterior class probabilities. These models were trained to predict affect using the 'basic emotions' model, with the authors finding that both decision level and feature level fusion significantly improved results over unimodal modes.

Approaching similar data, albeit using a valence-arousal emotional model, is Nicolaou et al. [143] who sought emotion detection improvements, especially in naturalistic settings, by utilising a more complex model of emotion. Two predictive models were discussed, bidirectional LSTM networks and Support Vector Machines for Regression, along with a method for fusion which accounts for correlations and covariances in views. A further work [144] examined measuring 'interest', a particularly interesting task for this thesis as measuring interest within a game streaming context provides insights into modelling player experience. This work used the SEMAINE dataset, with the addition of interest annotations, and employed Robust Canonical Correlation Analysis to fuse audio and visual views, showing a strong correlation between interest and other emotion dimensions.

Lin et al. [110] took a probabilistic approach to audio-visual fusion when applying Semi-Coupled Hidden Markov Models to artificially noisy data. In this work, the authors argue that given noisy data, the Semi-Coupled Hidden Markov Models approach showed promise compared to feature-level fusion at preventing over-fitting. This model-level fusion is a departure from more common feature-level or decision-level fusion. Kahou et al. [88] used an 'in-the-wild' style database gathered from clips of Hollywood films which, whilst containing acted affect, was designed to mimic natural emotions. It also contained a variety of camera angles, lighting, and poses which added an extra challenge to the dataset. The task was to

classify this data using the 'basic emotions' model, which was achieved using a convolutional neural network to detect visual affect and a deep belief network for audio feature extraction combined with temporal pooling for audio classification. These models appear to have success given the difficulty of the dataset. Gan et al. [59] used a multimodal deep regression Bayesian network for affect detection in videos. Two evaluation methods were used; a classification task based on MediaEval 2015 [106] classes with three possible classifications, low, medium, and high, for both arousal and valence, and a regression task based on MediaEval 2016 [64]. The extracted features from the modalities were whichever features were proposed in the original dataset rather than hand-crafted by the authors or learnt as part of the model. Comparisons were made to various other fusion techniques, such as Canonical Correlation Analysis, with the method appearing to outperform other fusion methods.

With deep learning, it is possible to construct end-to-end models, i.e. from raw video input to predictions, with state-of-the-art results. Tzirakis et al. [210] proposed an architecture where two convolutional networks are used to extract features. An 2D image processing CNN was used for visual feature extraction. The audio network used temporal convolution to extract features from the raw audio. Features from both models were then concatenated into a single feature vector which was used as input to an LSTM for feature fusion and valence/arousal regression. This architecture is somewhat similar in layout to the architecture used during the first study of Chapter 3, although a different set of features are used as input into the LSTM layers. In the work, the authors compared this approach to existing state-of-the-art models, with the proposed model significantly outperforming these comparisons on the RECOLA dataset used in the AVEC competitions [177,178]. End-to-end models are very attractive because they remove the requirement for human-designed features and instead allow the network to learn which features are essential for the given task.

Many works in this section have discussed multimodal fusion as the key challenge in audio-visual modelling. However, another issue with audio-visual data is that it is both multimodal and temporal. Martinez and Yannakakis [119] have investigated ways in which temporal features can be extracted from multimodal data in an affect detection for video game settings. They argue that sequence mining has some advantages over traditional approaches to multimodal feature extraction, stating that this approach, on the given data, outperforms statistical feature extraction techniques for affect detection. One key takeaway from this work for this thesis is the discovery that changes in player arousal, measured

through physiological measures and self-reporting, have a relationship with critical game events, reinforcing the belief that there is a link between game content/events and player affect.

2.3 Livestreaming

As an emerging entertainment medium, livestreaming is a fertile field for research on a range of topics, e.g. from talent scouting [16] to emotional labour [215], and by utilising a range of disciplines, e.g. from sports science [83] to deep learning [213]. Primarily, research has focused on the sociological aspects of streaming and why watching somebody else play is compelling. That said, some artificial intelligence techniques have been explored, mainly applied to livestream chat. These research interests are discussed in detail below.

2.3.1 Analysis of Game Streams

Prior research has often focused on viewer communities [201]. For example, Hamilton et al. [68] performed an interview-based study of streamers and viewers, finding that livestreaming platforms serve as 'third places', i.e. a place which hosts gathering of individuals but which is not work or home. Further, they describe livestream audiences as having both cool and hot moments, i.e. there are moments within the stream that elicit a strong reaction in the crowd, and this reaction then allows participants to share in the emotion. Flores-Saviaga et al. [56] also examined streamer communities, finding that there are several types of livestreams, from very popular streams hosting esports tournaments and featuring celebrities down to 'clique' streamers broadcasting to only a handful of viewers. Interestingly, despite the opportunity to build intimate relationships with viewers in these clique streams, there seem to be issues with viewer retention in these streams.

Diwanji et al. [47] performed a small-scale study of three streaming sessions to examine audience behaviour. They found that the most common type of behaviour was reaction, i.e. the audience reacting to an in-stream event. This is particularly useful for this thesis as these types of interactions are precisely what the highlight detection system in Chapter 5 aims to model. Continuing the trend of using streams to study interactions between the streamer and the viewer Kaytoue et al. [92] examined the viewership of streams. They discovered that the number of viewers a stream has and how this changes over time is predictable. This prediction

is based on correlating popularity at the start of a stream and after time has passed. They also observed that peaks in audience size are explainable by factors such as the day of the week and time of day. For instance, viewership increases towards the end of the week and is affected by the streamer's local time. Finally, they presented a methodology for ranking the popularity of streamers based on a Condorcet method. Wulf et al. [216] studied livestream viewers, discovering that the sorts of things viewers enjoy in mainstream broadcasts, e.g. sports, translate to Twitch.tv, for example, suspense is linked with enjoyment. Deng et al. [40, 43] found that several factors differentiate Twitch.tv from other streaming platforms which do not feature live data, e.g. YouTube. For instance, the popularity trends seem to be more predictable, i.e. popular games remain popular, and flash crowds are event-driven, indicating the value of studying esport tournaments. Some work has also been undertaken on modelling streamers and viewers through graphs and finite state machines. For example, Nascimento et al. [138] found that spectators show clear behavioural patterns such as channel surfing, and most viewers tend to watch a small fraction of streamers.

Smith et al. [189] discussed many aspects of live-streaming, focusing on 'Let's Play' streams. These are streams where the streamer is playing through a particular video game whilst interacting with the audience. Let's Play streams are an environment very similar to what this thesis studies, albeit with the additional restriction that the games are generally single-player or co-operative and have some story or campaign progression to work through. In contrast, many of the most popular streams are of players playing competitive online multiplayer games, e.g. *Fortnite*, *League of Legends*, and *Hearthstone*. The authors discuss how streaming is a naturally interactive domain because the viewers can directly interact with the streamer via text chat. Therefore, they argue, it is impossible to fully disentangle the streamer's experience and viewers because viewers can 'co-labour' play with the streamer. Another study into the streamer-spectator relationship is 'All the Feels' by Robinson et al. [172]. 'All the Feels' is a visual overlay that presents biometric and emotional information, such as heart rate, to the viewers. This lets viewers feel more in touch with the streamer's emotional state and gaming experience. This work contained a small study where streamers used this overlay and showed that the availability of this data made spectators feel more connected to the streamer. This study did not develop modelling techniques for understanding emotion and biometrics. Instead, it used an off-the-shelf product, Affdex[1],

[1]Developed by Affectiva.

which is suitable for a design-oriented task such as 'All the Feels' but does not offer the quality or fidelity required for this thesis. Off-the-shelf modelling, especially Affdex, has also been used by other parties interested in emotions in games. For example, the game Nevermind[2] uses affect detection to change the game's content based on biometrics and affect in a psychological horror game setting.

2.3.2 Livestream Chat Analysis

Text-based chat interaction is one of the defining features of livestreaming compared to other broadcast media. Such is the importance of chat, Chapter 5 focuses entirely on modelling this data. Prior work into understanding chat data has primarily been disparate with little in the way of shared data or goals, hence the desire to publish the TwitchChat dataset as part of this thesis to build a shared foundation for livestream chat study.

Nematzadeh et al. [140] studied the large-scale nature of livesteam chats and, in particular, focused on the difficulties for meaningful communication when there are many participants. In particular, they found that as the number of viewers in a stream increases, there is a transition between conversational chat environments to 'cacophonous' large-scale chats where there is lower per capita participation, more repetition and less information per message. This cacophony is analogous to the cheering effect observed in traditional sports settings. An important observation is that while there are often many individual users sending messages, thus leading to the cacophony, there tends to be a set of coherent 'voices' or personas. For instance, Ford et al. [57] found that when the number of chat participants increases, the number of unique voices remains relatively stable. Despite this, Musabirov et al. [134] used Topic Modelling, a form of unsupervised clustering which generates a set of 'topics' from a corpus to explore equality between voices, finding that as the number of chat participants increased, the more unequal the topics present were. This 'crowd speak', especially understanding that massive chats do not result in an explosion of topics, results in a surprisingly coherent chat stream, even if the messages come from many authors.

One area of research which has received attention is modelling toxicity from chat participants. In particular, Poyane [160] studied *Dota2*, a Multi-Player Online Battle Area game similar to the *League of Legends* game studied in parts of this thesis. The study showed a statistically significant increase in toxic communication as chat size increases. Kim et al. [95]

[2]Released in 2015 by Flying Mollusk

focused on identifying toxic emotes. In particular, they manually labelled a small dataset of 29,721 samples which was used to train a classifier to identify toxic chat messages which were previously undetected due to their use of Twitch.tv specific emotes. The work from Seering et al. [181] examined how livestream chat moderators could promote pro-social behaviour and reduce toxic messages, e.g. by banning toxic members.

Sentiment analysis of the chat logs has the potential to provide insights about viewers. For instance, Kobs et al. [98] found that emotes are sentimentally valuable and thus can aid in sentiment detection. They also found that sentiment values from chat correlate to in-stream events. Similar to the finding in Chapter 5, Kobs et al. found that Twitch chat is so unique that existing sentiment analysis techniques are unsuitable. Similarly, Topic Modelling was used by Konstantinova et al. [100] to discover that chat messages were more likely to be short but contain emotionally charged content during an esports match, compared to messages sent outside of the match, where messages tend to be social or analytical. Carnein et al. [27] explored clustering for livestream chats using a modified DBSTREAM clustering algorithm which allows for new data points to be added to a clustering. Carnein et al. found that this approach seemed to work well at clustering the data, although it was trained on a small dataset of about one million messages, and no insights about the nature of livestream chats were gained.

Outside of natural language processing, there has been a linguistic study of livestream chat that observed significant variations compared to other internet communities [147], and that chat content is heavily related to stream content [162]. While these prior works begin to uncover the properties of livestream chats, they do not offer a statistical or machine learning approach to understanding the meaning of tokens through analysis of how the chat is constructed. For instance, some prior works attempt to uncover the meaning of stream-specific tokens, e.g. [15], and generally, these works assume that tokens, such as 'Kappa', have a particular meaning. Finally, Nakandala et al. [135] explored the use of gendered conversation in the livestream context by utilising vectorisation techniques.

Emoji and other ideograms are popular within livestream chats, so a recent analysis of how emojis are used is relevant. For instance, [214] shows that ideograms often have a complicated and highly personal meaning. Therefore, it can reasonably be assumed that particular emojis and emotes evolve to have a rich Twitch.tv specific meaning, although the meaning of ideograms within livestream chats is not the direct subject of this thesis.

More generally, work has been carried out on how ideograms are used in conversant text. For instance, research shows that norms surrounding the meaning of ideograms propagate through social networks [151] and that geographic location, in the case of livestreaming, perhaps tied to the nationality of the streamer, can affect ideogram usage [113].

Chapter 3

Personality-Driven Livestream Highlight Detection

3.1 Introduction

Highlight detection for personality-driven livestreams is a challenging task due to the lack of available labelled data to train a model with. While there is abundant livestream data, evinced by the massive number of concurrent streams, edited highlight videos for personality-driven livestreams are not nearly as accessible. Twitch.tv has a 'Clips' feature where viewers can select entertaining moments and clip them from a stream. Users of the site can then browse and watch these clips. However, this feature is not popular enough to build a high-quality supervised learning dataset.

Furthermore, edited broadcasts are often posted on separate video-sharing platforms. However, tracing the original livestream from the edited broadcast is challenging. This is due to the observation that for personality-driven broadcasts, highlight videos are often titled to entice a potential viewer rather than factual information, such as when the original stream occurred. Thus, building a suitable supervised training set from this data for personality-driven broadcasts is also complicated. Therefore, establishing a strong indicator for highlight moments is very difficult. Because highlights are subjective, there are not necessarily clear indicators from audio-visual data. Furthermore, manual annotation is extremely expensive and time-consuming. As a result it is reasonable to approach highlight detection through the detection of signals which can be considered proxies for highlights, which is the focus of this chapter.

While it may be possible to construct a semi-supervised dataset with the data available, it is expected that such an approach would be less successful for personality-driven broadcasts due to the heterogeneity of stream presentation. Stream features such as webcam placements and lighting are often highly personal to individual streamers. Therefore semi-supervised is likely to be much more suited to game-driven broadcasts where there is more homogeneity between broadcasts and, as such, is explored in Chapter 4. This chapter will explore two alternative methods for explicitly training a highlight detection model on personality-driven streams. Firstly, an unsupervised learning approach is discussed where 'novelty' is considered a proxy measure for highlights, assuming that highlights must be novel to differentiate themselves from non-highlight moments. Secondly, a supervised model is trained on hand-labelled data. Here, the labels are not strictly binary highlight labels but describe both the streamer's emotion and the game context. Utilising the streamer's emotions is one way to leverage the unique features of personality-driven livestreams because they focus on a presentation and performance.

This chapter focuses on two studies. The first, detailed in Section 3.2, explores novelty as a proxy for highlighting via unsupervised learning. The second, detailed in Section 3.3, utilises more objective supervised labels, e.g. game events and streamer emotion, to build a supervised model which can predict proxy signals for highlights. Finally, Section 3.4 concludes these two studies' findings.

3.2 Novelty as a Highlight Signal

Novelty detection algorithms are a suite of techniques which aim to detect samples in a dataset which are outliers and, thus, unlike the others. The vast majority of stream segments can not be considered highlights. Otherwise, the act of highlight detection lacks purpose. Because highlights are expected to be rare, one may reasonably assume that the outlier moments, i.e. those most unlike others, are likely to constitute a highlight. Thus novelty detection aims to find the moments in a stream most unlike the others under the assumption that these novel moments are likely to be attractive to the viewer. This is achieved by analysing both audio and visual feeds for a stream. The audio data from a stream represents a mixture of the in-game audio, the streamer's voice, and other audio artifacts, e.g. stream notifications. This mirrors the audio feed that is presented to a viewer. The visual feed,

i.e. video, is split into two streams, one for the streamer's webcam and one for the in-game footage. This is achieved by manually retrieving the coordinates of the streamer's webcam and then extracting that portion of the frame as a separate image while replacing the webcam with a black patch in the game feed.

Novelty detection as a proxy for highlight detection has the added benefit that novelty detection methods can act in unsupervised settings. Unsupervised models, assuming sufficiently cost-effective processing, would allow for the complete democratisation of highlight generation. Streamers would only need to upload their stream without requiring prior information about what sorts of moments are highlights or not, e.g. as required in a supervised learning setting. While it is undoubtedly a difficult challenge to develop a highlight model from only unsupervised data, the benefits of such a system, should novelty be a suitable proxy, are evident. The system described in this section was the first highlight detection system for audio-visual livestream data proposed using entirely unlabelled data. Other systems have either used labelled data or taken a supervised approach. e.g. [169, 219, 220], or inferred labels through the crowd-sourced video editing, e.g. [168, 213].

Further to neatly sidestepping the challenge of labelling data, unsupervised learning modelling has the potential to work well considering the inherently 'in-the-wild' paradigm. Personality-driven livestreams generate organic, real-world data [145, 194] and are subject to many complicating factors such as visual and audio occlusions. Such occlusions can either be temporary, e.g. the streamer looking away from the camera, or permanent, e.g. overlays placed over the game scene or the music that the streamer is listening to. Other difficulties include streamers having their webcams at various angles, using varying lighting levels, and choosing different volume levels between their voice, music and the game audio. Unsupervised learning in a novelty detection setting may mitigate the variance resulting from the heterogeneity of presentation in personality-driven broadcasts.

The rest of this section is organised in the following manner; in Section 3.2.1, the unsupervised system is described, in Section 3.2.2 the dataset used is presented, in Section 3.2.3 the experiment is described, with the results presented in Section 3.2.4, finally discussion and conclusions are provided in Sections 3.2.5 and 3.2.6 respectively.

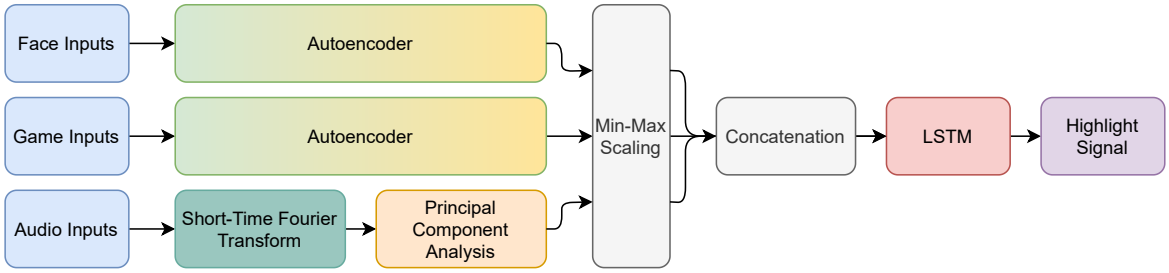


Figure 3.1: Overview of the system

3.2.1 Methodology

The unsupervised highlight detection architecture proposed is a multi-stage system. Firstly, 'novelty' features are extracted from each modality, i.e. webcam, game footage, and audio. These novelty features describe when a particular input, i.e. moment in the stream, is novel compared to other moments in the stream. Next, these features are scaled and fused using a recurrent network. The time series output from the recurrent network is then used to assess which stream segments are highlights. Each part of this system is described in detail below. The network architecture is illustrated in Figure 3.1. The architecture is similar to the one utilised by Malhotra et al. [150].

3.2.1.1 Face and Game Scene Analysis

For novelty detection, convolutional autoencoders are utilised to analyse the player's webcam and in-game footage. The networks are composed of two stacked VGG16-like networks [187], omitting the fully-connected layers. Given a video frame, the first network, the encoder, produces a 512-filter encoding. The second network, the decoder, is the reverse of the encoder, employing up-sampling rather than max-pooling. A final convolutional layer then takes the 64 filter output and reconstructs the input image. This architecture is shown in Figure 3.2. Each convolutional layer has a 3×3 filter window, and each Max Pooling and Up-Sampling layer uses a 2×2 window with a stride of two, following works such as Deep Convolutional Auto-Encoder with Pooling - Unpooling layers [204], and Stacked What-Where Auto-Encoders [229]. The reconstruction errors from both networks are then utilised as indicators of novelty.

For the face autoencoder, the weights from the VGG Face Descriptor model [26, 152] are used for the encoder, something made possible because the VGG Face Descriptor model is trained on photographs of people, a setting very similar to the extracted webcam footage.

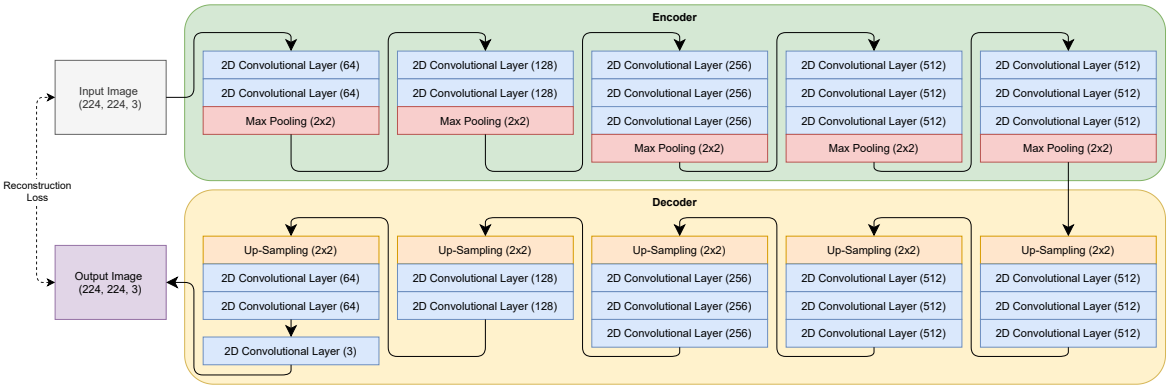


Figure 3.2: Autoencoder architecture

These were frozen during training as no noticeable improvement was observed from fine-tuning the encoder or training both end-to-end. Utilising pretrained weights reduced the amount of computation needed at training time. However, no such pretrained network exists for game footage; so both the game footage encoder and decoder were trained together. All models were trained until convergence, generally about 50 epochs. Once trained on frames from a video, the reconstruction error can be used as an indicator for novel frames in a video. This novelty is then used in the downstream system. This downstream system takes novelty measures from all modalities to determine which segments are likely highlights. Using reconstruction error as an indicator of novelty is an established technique. More details on reconstruction-based novelty detection can be found in Primentel et al.'s recent survey [155].

3.2.1.2 Audio Stream Analysis

Considering the arousal and valence axis of affect models, intuitively, arousal would seem like the most important element to model, given that it describes how strongly a person feels an emotion. Moments where the streamer is very excited or angry may be entertaining, but moments with low arousal, regardless of valence, e.g. misery and contentment, are less likely to be highlights. Since detecting arousal is most likely to yield results, this section's approaches focus on processing and modelling critical audio frequencies since these relate most closely to arousal.

The audio stream is split into segments via a sliding window. Each window is 400ms long, and there is a 300ms overlap between sequential windows. This yields a sampling rate of 10 samples per second. A large window was utilised to alleviate potential issues arising from asynchrony between audio and video modalities, e.g., the preservatory and anticipatory co-

articulation phenomena often causing audio cues to be delayed by around 0.12 seconds [90]. For each window, the Short-Term Fourier Transform is computed, and the magnitudes are retained since they are deemed good indicators for arousal [11]. Additionally, uncommon frequencies in human speech are discarded to isolate the streamer's voice from the game sounds. Frequencies between 300 - 3400 hertz were retained. This range was chosen because it is the standard frequency range for telephone calls and thus is likely to represent the streamer's voice. Principal Component Analysis (PCA) is applied to the entire broadcast, with the k first principal components retained. This is aimed at capturing variation in the broadcast, and thus discovering novelty, while reducing dimensionality and noise.

3.2.1.3 Recurrent Layer for Late Fusion

A recurrent layer is utilised for indicating reconstruction-based novelty across all modalities. The recurrent layer is fed with a time series consisting of reconstruction errors from face and game scene autoencoders and the extracted audio features, in-effect performing a multi-dimensional smoothing operation. Values are normalised between $[0, 1]$ to avoid biasing modalities that cover different ranges, e.g. video compared to audio. Training entails forecasting reconstruction error values corresponding to the next frame, $t+1$ at time t, thus effectively incorporating information from all modalities. Two Long Short-Term Memory (LSTM) [73] layers are utilised, each with three neurons and a fully connected layer with a single output neuron using a sigmoid activation. Each neuron in the LSTM layer retains a latent state that captures helpful information from previous time frames and generates the output values along with the current input.

3.2.1.4 Highlight Detection as Reconstruction-based Novelty

The prediction error of the recurrent layer on a given stream is used as a highlight indicator. A threshold is applied to the error from the recurrent layers to select highlight segments, empirically determined as 0.01%. The frames with the highest prediction error are then selected as highlight frames. These highlight frames are then used to generate the full highlight clips. Each detected frame is treated as an 'apex frame' of a highlight event. Like the Twitch.tv clip system, a highlight clip is considered the stream segment 10 seconds before the apex frame and 5 seconds after. Proximal apex frames are linked together if not doing so would cause an overlap between clips. This approach ensures that the appropriate context

is included in the highlight clip and that the clip is self-contained, e.g. a reaction from the streamer can be detected as a highlight frame, with a preceding game or stream event causing the reaction being included in the clip.

3.2.2 Data

Player Unknown's Battlegrounds (PUBG) is a multiplayer online battle royale game. A number of players are spawned simultaneously to explore an island, collect weapons and other items, kill other players and ultimately be the last player alive. The playable area of the island shrinks over time, and so, while the players start fairly spaced out, as the game continues, players are forced closer together, increasing the likelihood of interesting moments later in the game. While ultimately, PUBG is a game of being the last player alive and therefore necessitating killing other players, it is largely a game of avoiding other players, at least until the player can gather enough resources or position themselves well enough to make encounters favourable. PUBG is a particular applicable game for novelty-based unsupervised highlight detection because it often has long periods of low-intensity gameplay and short bursts of concentrated action. Thus it can be expected that these short bursts would be detected as novel moments. Furthermore, the low and high-intensity gameplay ratio makes highlight detection a worthwhile task. Finally, there is an abundance of high-quality streams available due to the game's popularity.

A dataset was gathered by downloading livestreams from Twitch.tv as they were being played. Each recorded stream was segmented into videos spanning a single game, and downtime between games was removed. It makes little sense to look for highlights when the game is not being played, as streamers often take short breaks in-between games where they will leave the stream or browse social media. The dataset consists of videos from two male streamers, one American and one German, streaming in English. There is a total of 11 videos, and each video is between 19 minutes 30 seconds and 30 minutes 40 seconds long. Each video was preprocessed as follows. Firstly, the video data is down-sampled to a sampling rate of 10 frames per second, improving training speed while retaining granularity. Secondly, the player's face is extracted from the game footage stream, and replaced with a mask, resulting in 2 images per source frame, one from the game and one from the webcam. Finally, each frame, webcam and game footage is resized to $224 \cdot 224 \cdot 3$ to match the VGG16 input dimensions.

3.2.3 Experiment

To explore the applicability of the proposed novelty approach, it is essential to apply the model and then perform a post hoc classification of the generated highlights to explore the kinds of highlights the model generates. Each clip is annotated by the researchers into one of four categories: 'funny', 'action', 'social-interaction', and finally, 'no highlight'. Funny videos are streamer-focused events where the streamer makes a joke, laughs, or is, in other ways, amused. Action highlights stem from game events, e.g. the streamer is engaged in a firefight. Highlights tagged as 'social-interaction' include community-led events, where the streamer interacts with viewers meaningfully, e.g., thanking subscribers, answering questions, or reacting during a similar interaction. 'Social-interaction' highlights are important for a compelling personality-driven stream and are often found in streamer highlight clips that are manually segmented. Finally, clips containing no noteworthy events are labelled 'no highlight'. The model is trained individually on each video in the data, following a one-shot approach. The resultant model is used to generate highlights. The model was implemented using Tensorflow [1] and Keras [36], using the ADADELTA [226] optimiser for both autoencoders and the recurrent network, employing a mean-squared error loss function for all models.

3.2.4 Results

In total, 98 segmented highlight clips were generated by applying the method to the dataset. Table 3.1 shows the results by using all available modalities, i.e. the streamer's face, the game, and the audio stream, where out of 75 clips with interesting content, 51 are tagged as 'funny' or 'action', with the remaining 24 labelled as 'social-interaction'. Figure 3.3 shows visual examples of the types of highlights generated with this approach.

3.2.4.1 Modalities and Highlight Detection

Evaluating the proposed architecture when observing different combinations of modalities is valuable to examine which impacts positive prediction most. The results of this ablation study are summarised In Table 3.2. The complete model was compared to individual modalities, Face, Game, and Audio, and the Face and Audio domains combined. Overall, the model fusing all modalities performs the best. This is an expected result since utilising

Video	Funny	Action	Interaction	Highlight Total	No Highlight
S1.1	1	2	4	7	0
S1.2	0	1	1	2	4
S1.3	0	3	2	5	3
S1.4	2	2	4	8	2
S1.5	0	3	4	7	4
S1.6	0	1	1	2	2
S2.1	5	2	0	7	1
S2.2	3	1	2	6	1
S2.3	6	4	2	12	2
S2.4	3	1	3	7	1
S2.5	6	5	1	12	3
Total	26	25	24	75	23

Table 3.1: Generated highlight clips by category using all modalities.

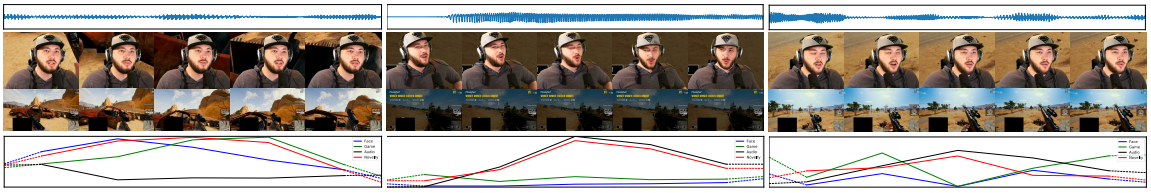


Figure 3.3: Example highlights discovered by the proposed method. Left: The streamer begins aiming their rifle, which changes the game scene enough to trigger a highlight, agreeing with a change in the facial expression. Center: The streamer wins a game and shouts in celebration; detected by indicating novelty in the audio features. Right: During a firefight, gunfire causes a spike in the audio, triggering a highlight.

audio-visual information from streamer behaviour and the game provides a more informed approach.

Interestingly the number of generated highlights segments across all modality combinations is similar, except for the audio-only model, which produced considerably more. This is likely due to the impact of in-game audio (e.g. gunfire) on detecting novelty. These moments are novel compared to the usual sounds in a stream, e.g. the player talking and in-game ambient noises. The observation further supports that 29% of highlights selected were action highlights, containing only a few funny or social interaction clips. The face modality is better at determining funny and social-interaction highlights, although it has a worse precision in action clips, which is expected given that using only data from the streamer's face does not consider any context regarding the game. The percentage of No Highlight clips is compa-

Modalities	No. Videos	Funny %	Action%	Interaction%	Total Highlight%	No Highlight%
Face, Game, Audio	98	27	26	24	77	23
Face, Audio	95	22	23	28	74	26
Face Only	96	14	14	24	52	48
Game Only	94	4	18	7	29	70
Audio Only	126	8	29	18	56	44

Table 3.2: Summary comparison of highlight-detection over multiple modalities

rable for the audio-only and face-only models and is often due to unusual gestures that the streamer might perform that can potentially occlude the face. This is expected since 'face novelty' itself is not necessarily sufficient to guarantee a highlight, although it may indicate it.

The game footage model is the worst highlight predictor, given the results. Whilst this poor performance might appear counter-intuitive, the results show that using game footage is mainly helpful at generating action highlights[1]. This is likely due to the lack of social cues from the streamer's face and audio, as omitting such modalities makes the problem of detecting when the game scene is interesting to the viewer rather than merely anomalous more challenging. Finally, using a face and audio model provides better results than all single-modality models and slightly worse results than all available modalities. This finding suggests that the game alone is a poor indicator of highlights, other than action clips however it can be helpful to corroborate information extracted from other modalities and improve performance.

3.2.4.2 Highlights over Time

There are fewer 'no highlight' segments detected towards the end of a video than at the start, as shown in Figure 3.4. This indicates that detected clips later in the broadcast are more likely to be interesting. In more detail, 61% of 'no highlight' results occur in the first 30% of a video, as opposed to 19% of funny clips, 4% off action clips and 41% of interaction clips. Furthermore, most action clips, 92%, and funny clips, 69%, occur in the last 50% of the video, as opposed to only 22% of 'no highlight' clips. 60% of action clips occur in the last 20% of the video duration. By considering only segments generated from the last half of each video, 91% of clips are interesting in some way. Several likely explanations for this

[1] Although on-screen events may be indicators of other highlight types, for example, new subscribers pop-ups or humorous on-screen events.

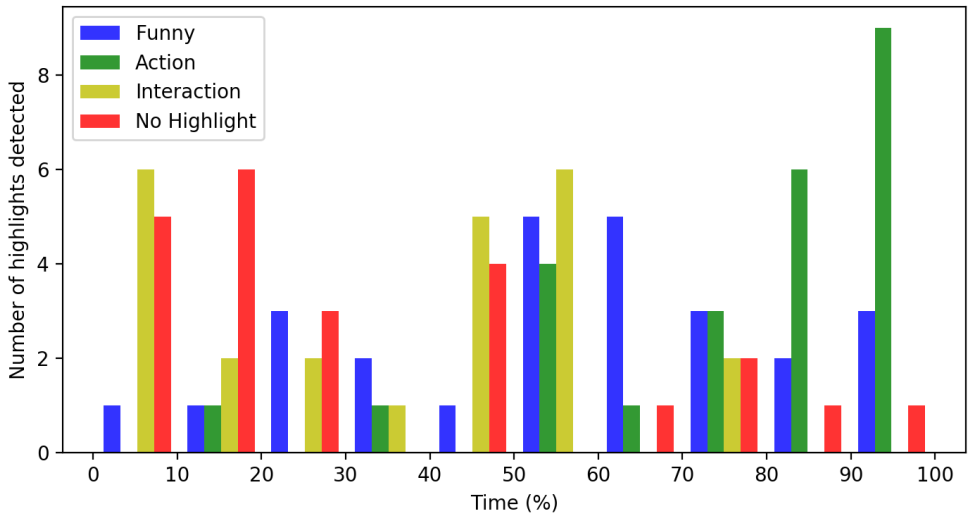
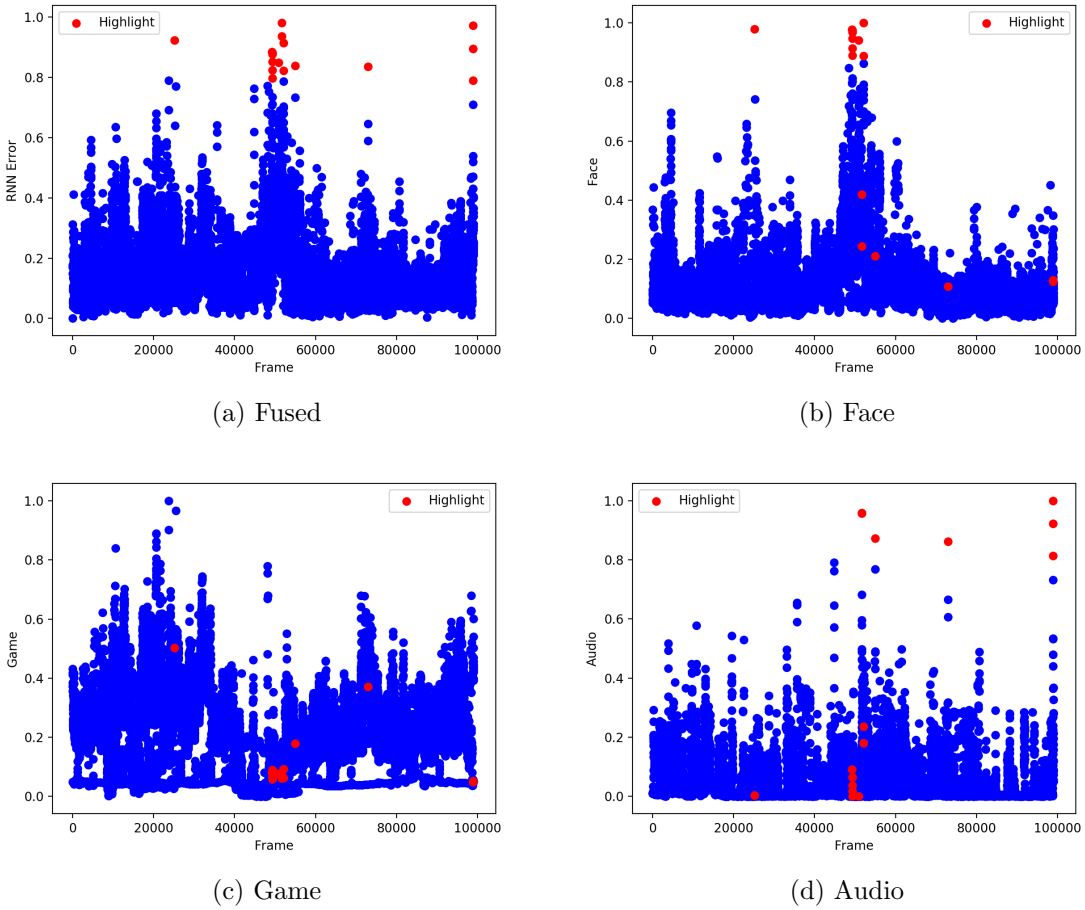


Figure 3.4: Highlights by type over time

are mostly related to the game's design. Firstly, the game is designed so that the play area shrinks over time, forcing interaction between players towards the end of the game. Hence the more considerable amount of action highlights towards the end of the video. Secondly, there are fewer viewer interactions as the game progresses since the game intensity increases, and players must focus more on the game. Finally, the number of funny highlight segments initially increases due to more in-game events occurring and, therefore, more events for the streamer to react to, but then at the end of a game, the number of these moments reduces, likely because the streamer needs to focus on gameplay and has less capacity to be humorous.

3.2.4.3 Novelty Across Modalities

Figure 3.5 shows (a) the RNN prediction error fusing all modalities, (b) the output from the face autoencoder, (c) the output from the game autoencoder, and (d) the first principal component of the Fourier coefficients of the audio channel. Errors over time are plotted for a particular video, `S1.1`, colouring samples selected as highlight apexes in red. In general, for the face and game modalities, sharp 'spikes' indicating highlights can be observed in the distribution, with errors on game footage being less clear because data in this domain results in higher average output with a wider spread. For this particular stream, the game footage appears to have less impact on the final highlight frame detection than the face and audio modalities, where clear spikes are transferred to the fused results. Finally, spikes which



(a) Fused

(b) Face

(c) Game

(d) Audio

Figure 3.5: Errors over time indicating novel events for a particular video (S1.1). (a) Fused prediction error. (b) Face video reconstruction error. (c) Game footage reconstruction error. (d) Audio features over time.

only occur in one modality are often smoothed out in the final fused model, while spikes appearing in more than one modality are accentuated. This further affirms a multimodal model's benefits compared to just a single modality.

3.2.5 Discussion

There are several key observations. Firstly, it is clear how critical multimodal approaches are for highlight detection. Secondly, analysis of the types of highlights generated suggests that the streamer's emotions, modelled through webcam and audio data, significantly impact the generation of highlights. This is evidenced both in Figure 3.6, which shows examples of the kinds of social cues present in generated clips, and Figure 3.7, where you can see the

Figure 3.6: Streamer webcam Sequences from detected highlight segments.

impact that the audio waveform of the streamer laughing has on prediction. Thirdly, this work shows that highlight moments are triggered by the streamer, the gameplay, and the audience. Therefore, it is evident that models can consider cues for all domains that are likely to perform well.

While it is clear that the face and audio domain are beneficial, it is surprising that the game modality struggles so much, given that it offers much visual information about the game. This is likely due to the lack of a 'baseline' state for game footage. The audio and face modalities have a baseline state, e.g. the streamers at rest and speaking typically, which allows the novelty detection algorithm to accurately model which moments are genuinely novel. However, the game footage has no baseline state as the environment is complex and regularly changing. This likely makes it much more difficult for the game autoencoder to learn a robust model, and thus the novelty detection is negatively impacted.

Finally, there is a limitation with this work in that it is unlikely to perform well in all games. *Player Unknown's Battlegrounds* was chosen because it is a Multiplayer Battle Royale game. This genre is known for its generally high downtime, containing only a handful of short, high-intensity gameplay segments. This method was trialled on several other settings, including strategy games such as *Hearthstone*, where it was observed that nuanced gameplay, with a more even mixture of low and high-intensity moments, led to poor performance. There is no obvious way this model can be improved for these games inside the novelty-based approach. While the assumption that novelty is a reasonable indicator of a highlight moment for high-downtime, low-action games like *Player Unknown's Battlegrounds*, it is not the case for other, more complex games; thus, a different approach is likely warranted.

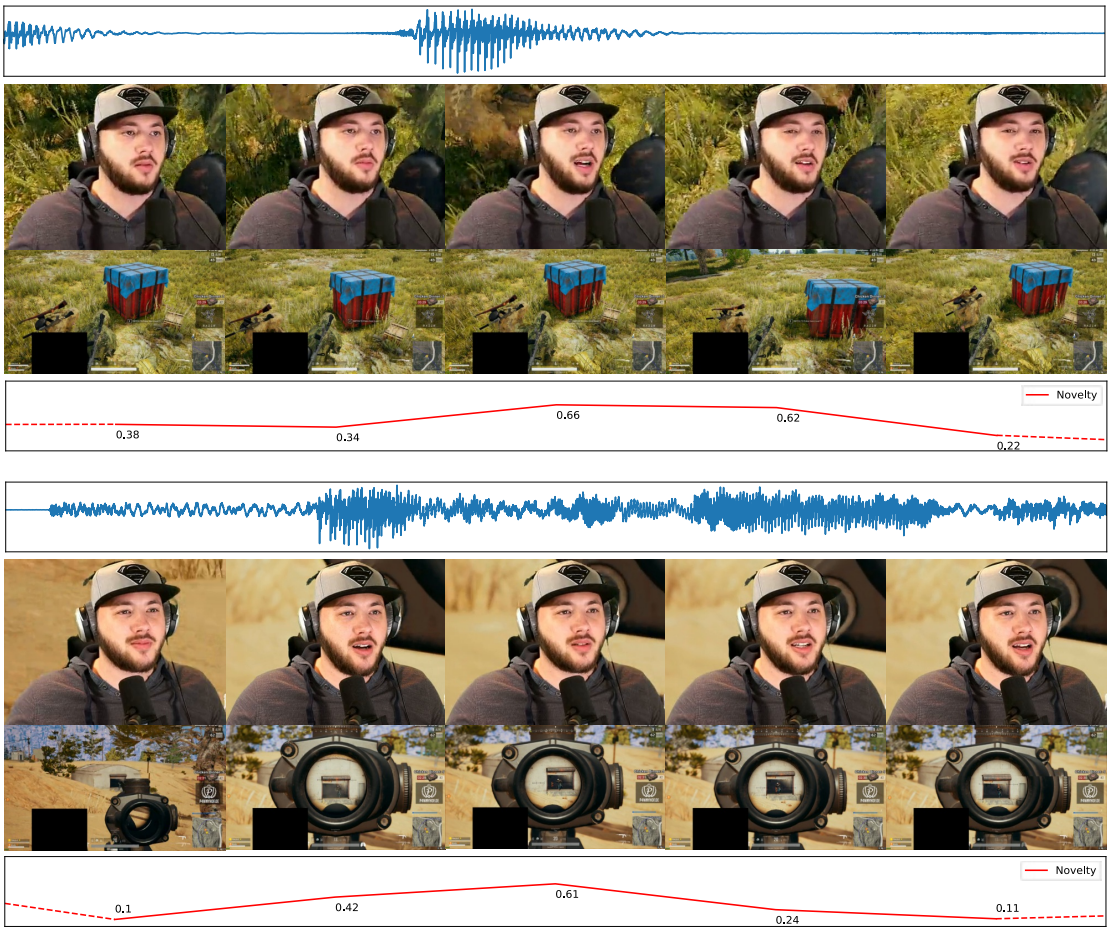


Figure 3.7: Example highlights. Audio waveform, face frames, game frames, and the prediction error are shown. Above: The streamer makes a joke after picking up a good item at a crate. Below: The streamer initiates a firefight with another player. In both cases, the highlight apex is the centre frame.

3.2.6 Conclusions

This section presents an unsupervised deep learning architecture for highlight detection based on audio-visual data broadcast during a typical game stream and represents the first such approach for this problem. The model considers a measure of reconstruction-based novelty as a proxy for indicating highlights while jointly analysing the player's facial footage, streamed game footage, and audio. It shows that multimodal models are much more likely to perform well than single modality models. In particular, this work shows the significance of considering social signals from the streamer when detecting interesting highlights in personality-driven streams. Furthermore, it shows that an unsupervised approach, if sufficiently performant, would likely be beneficial in a real-world setting. However, this chapter also shows there are limitations to this approach, and as such, for specific genres of games, a supervised or semi-supervised approach may be more applicable.

3.3 Streamer Emotion & Game Context as a Highlight Signal

Section 3.2 explored highlight detection through unsupervised learning. In contrast, this section will study highlight detection in personality-driven streams via supervised learning. In particular, Section 3.2 indicated that streamer affect is likely to have a large impact on if a moment is a highlight. Thus the models in this section explore joint streamer emotion and game event detection under the assumption that detecting both in-game events and moments of high arousal are likely to aid in highlight detection. Therefore, this study presents a series of models trained to identify various game and affect-based labels.

As well as examining the applicability of supervised learning to predicting these moments, and thus the suitability of such a proxy-based approach to highlight detection, this section explores two additional concepts. Firstly the effect that jointly learning emotion and game context has on model performance is tested. Because the dataset utilised in this study, detailed in Section 3.3.2, is relatively small compared to the vast datasets used to train some modern deep learning systems, e.g. the datasets used in Chapters 4 and 5, the models may overfit when trained on a single task. Thus training on all tasks simultaneously may aid prediction performance by minimising overfitting. Secondly, this section explores fusion mechanisms. Several novel architectures are presented to achieve this, focusing on multimodal approaches. In particular, a novel method for fusing multiple modalities by modelling

Figure 3.8: Example screenshot from a *League of Legends* livestream.

high-order interactions using a 'Tensor Train layer' [222] is presented and evaluated.

This section, and many of the following chapters, focuses on the game *League of Legends*. *League of Legends* is a 'Multiplayer Online Battle Arena' game where two teams of five players compete to reach and destroy the opposing team's base. To achieve this goal, players will have to fight computer-controlled monsters, fight other human players, and destroy static buildings on the map. Simultaneously teams must protect their base from the other team. Unlike traditional sports where the rules and game mechanics tend to be simple, *League of Legends* is highly complex, e.g. players can choose from over 150 characters to play, each with unique abilities and gameplay. It is one of the most popular esports, with at least five professional leagues and numerous global competitions[2]. In addition, it is an incredibly popular game for streamers and is regularly in the top 3 most popular games being streamed on Twitch.tv[3]. Figure 3.8 gives an example of gameplay for a personality-driven stream featuring *League of Legends*.

The study presented in this section is motivated by the observation that unsupervised learning, while successful when applied to *Player Unknown's Battleground*, has limited applicability when generalising to other games. Therefore, exploring supervised learning may

[2]https://eu.lolesports.com/
[3]https://twitchstats.net/

aid in developing generalisable detection techniques across broadcast settings. As discussed previously in this chapter, labelling highlight moments through annotation is infeasible. However, it is simpler to annotate video segments based on more objective measures such as the in-game content. These labels can then be used as a guide for a highlight detection system. For example, a system could select the moments where the player was engaged in combat or those where the player demonstrates high levels of arousal.

The rest of this section is organised in the following manner; in Section 3.3.1, the multimodal supervised system is described, in Section 3.3.2 the dataset used is presented, in Section 3.3.3 the experiment is described, with the results presented in Section 3.3.4. Finally, discussion and conclusions are provided in Sections 3.3.5 and 3.3.6 respectively.

3.3.1 Methodology

Three models with similar underlying structures are presented to compare fusion techniques. Like the previous section, three modalities are used, the streamer's webcam and game footage alongside the audio stream. For each modality, a set of latent features are extracted from each frame using a set of CNNs. When considering modalities containing image data, a 2D CNN is used, whereas a 1D CNN is applied to the audio stream due to the one-dimensional shape of the audio data. These features are then modelled temporally using several Long Short-Term Memory (LSTM) [73] recurrent layers. Finally, several fully connected layers are used to extract a set of classifications, depending on the task presented to the network. The difference between models, discussed in Section 3.3.1.2, lies in how the multimodal latent representations after the feature extraction stage are fused to build a shared representation of the input. The high-level architecture for each model is demonstrated in Fig. 3.9.

3.3.1.1 Feature Extraction

All models use the same set of CNN architectures for extracting a vector of latent features from each modality. A visual layout of these feature extraction architectures and the structure of the residual and convolutional building blocks can be found in Fig. 3.10. Each feature extraction model generates 512 features per frame. These features are then fused and used in the downstream classification task.

The two image networks use a 2D CNN with a series of residual blocks for streamer and game data. These residual blocks aid in back-propagating gradient by utilising connections

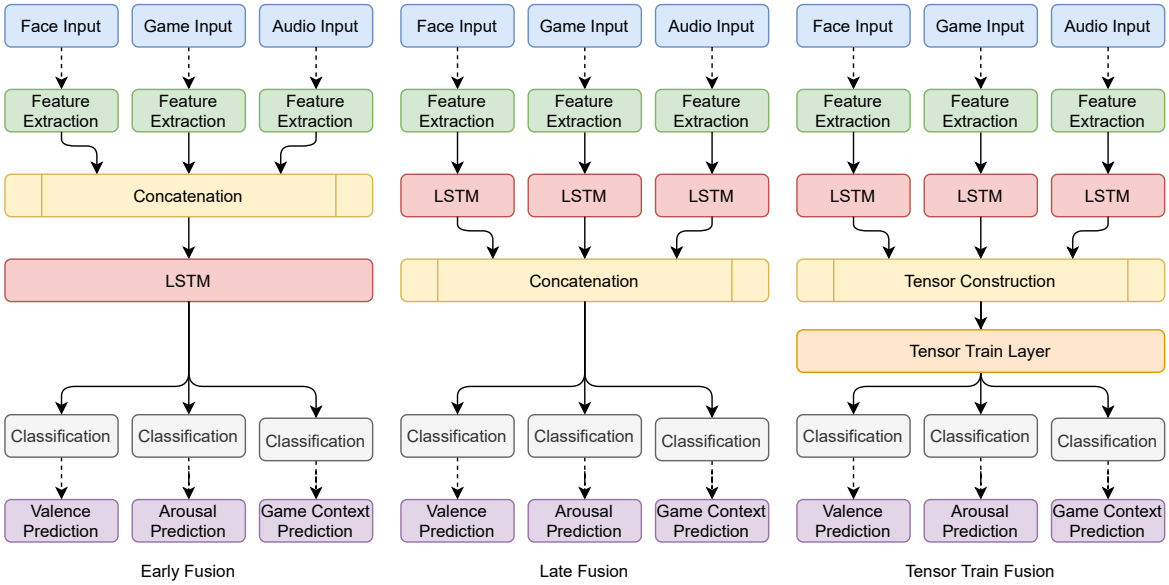
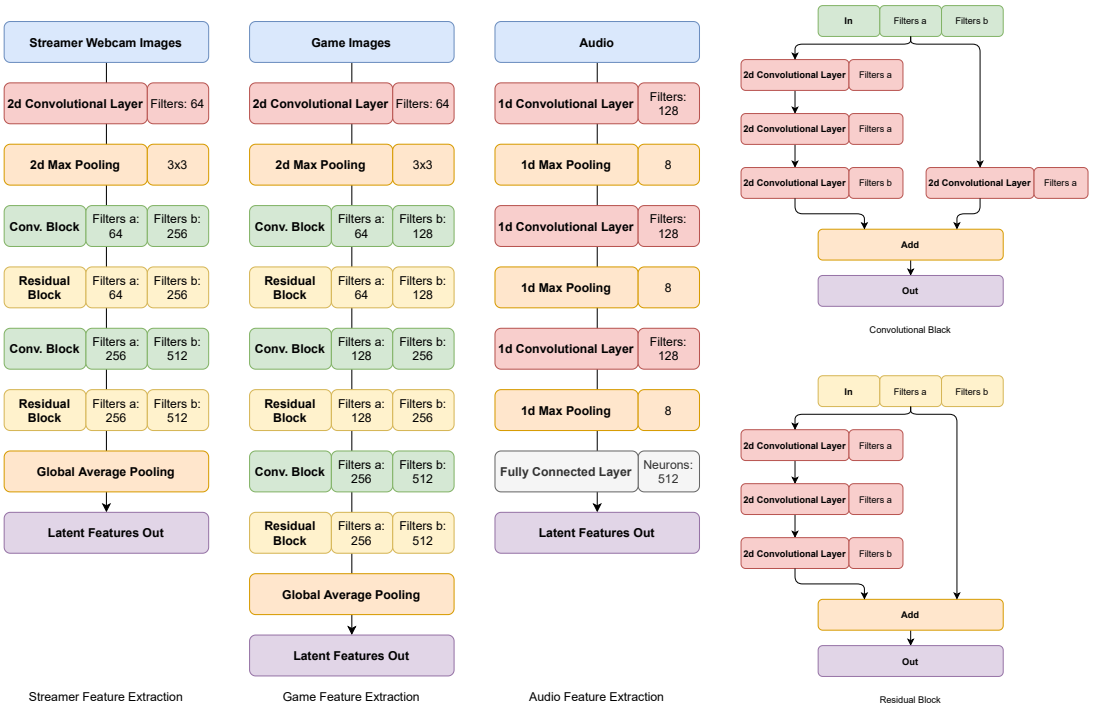


Figure 3.9: Comparison between the high level architectures of the three fusion techniques presented in this paper. All modes use the same feature extraction and fully connected classification layer shapes but differ in how the input modalities are fused.

which skip several layers, allowing the gradient to flow more easily to layers earlier in the model. This technique works well for deep CNN models and is well established in the literature, further discussion of this concept can be found in [71]. The main difference between the two image networks is that the input for the streamer network is a smaller input image, and as a result, this network requires fewer layers and weights to model the features required satisfactorily.

Rather than using a Principal Component Analysis approach to audio modelling, this section adopts a CNN. This is because it was observed during the previous study that training a CNN autoencoder was extremely tricky and often led to mode collapse. Thus, given the unsupervised setting, a Principal Component Analysis approach was necessary to avoid this issue. However, this chapter focuses on supervised learning; thus, mode collapse is less of a concern. Therefore this study is applied to take a more sophisticated CNN approach. A feature extractor with three 1D convolutional layers followed by a dense layer was used for modelling audio. After each convolutional layer across all feature extractors, ReLU activation and Batch Normalisation are applied. This audio feature extractor is trained to operate on the raw audio signal extracted from video data.



(a) Feature Extraction

(b) Convolutional and Residual Blocks.

Figure 3.10: Architectures of feature extraction modules. Each 'Convolutional Block' and 'Residual Block' is a set of multiple layers, shown right.

3.3.1.2 Multimodal Fusion and Temporal Modelling

Once latent features have been extracted from the various modalities, the next step is to fuse the modalities in addition to temporally modelling per-frame features. This process aims to extract a single feature vector corresponding to a joint representation across both modalities and time. Traditionally, concatenating input or representation space vectors has been one of the most popular approaches [90]. Nevertheless, this method can fail in terms of modelling interactions between modalities. The fusion approaches detailed in this work include both early and late fusion and a novel method based on tensor decompositions to facilitate efficiently modelling high-order interactions between modalities. The methods are detailed in the following sections.

3.3.1.3 Early Fusion

In the early fusion model, fusion is performed by taking the 512 features extracted for each modality per frame and fusing them into a single vector of $l = 1536$. The resulting vector

represents the fused representation of an input clip across modalities and time. A stack of 20 feature vectors, one for each frame in the segment, is then fed into 2 LSTM layers, each with 384 neurons. Before and after the LSTM layers, batch normalisation and 20% drop out are applied to guard against overfitting and aid generalisation.

3.3.1.4 Late Fusion

Late fusion is implemented with separate LSTM sections, each with 128 neurons per layer, for each modality. Batch Normalisation and Dropout layers are applied before and after the layers. Concatenation of the various modalities occurs after the LSTM step and right before classification. As such, a single feature vector with 128 features for each modality, 384 features total, represents the latent representation of the entire clip.

3.3.1.5 TensorTrain Fusion

Both early and late fusion models concatenate feature vectors from each modality into a single vector. However, this fusion approach has no explicit representations which capture the relationship between variables in different modalities. To better capture interactions between different modalities, a tensor is constructed that models up to third order interactions between modalities. That is, given feature vectors v_x, v_y, v_z each corresponding to separate modalities, the cross product is calculated via Equations 3.1 and 3.2. As such the fused feature tensor $z \in \mathbb{R}^{129\times129\times129}$ is created which contains v_x, v_y, v_z, $v_x \otimes v_y$, $v_x \otimes v_z$, $v_y \otimes v_z$, and $v_x \otimes v_y \otimes v_z$. The resulting tensor is shown in Fig. 3.11. .

$$z = \begin{bmatrix} 1 \\ v_x \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_y \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_z \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} 1 \\ v_x \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & v_x \\ v_y & v_x \otimes v_y \end{bmatrix} \tag{3.2}$$

At this point, it would be possible to flatten this feature tensor and then pass it through a series of dense classification layers. However, due to the size of the tensor, $2,146,689$ elements, this is computationally infeasible. Connecting this tensor to a fully connected layer with 384 neurons would result in $384 \cdot 2,146,689 = 824,328,576$ weights for this layer alone. Therefore, a Tensor Train layer [222] is used to connect the latent tensor with the

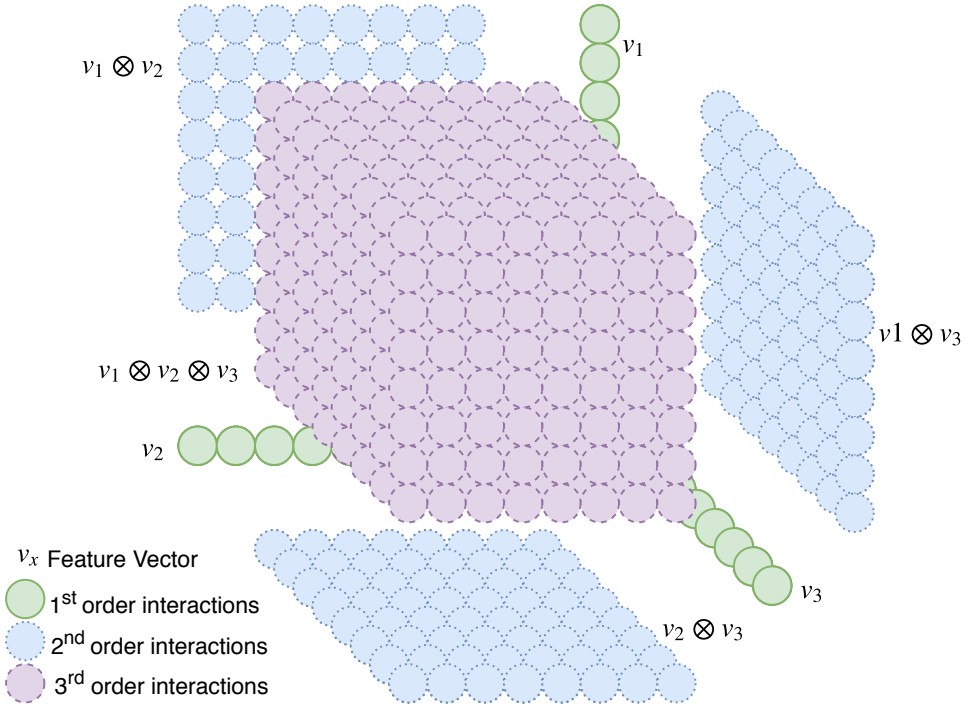


Figure 3.11: Exploded structure of the 1st, 2nd and 3rd order interaction tensor constructed before the Tensor Train layer. Note: Each feature here represents 16 features in the model. Best viewed on a computer.

classification layers. This layer replaces a dense layer and represents its weight matrix as a series of smaller tensors, the tensor cores. A tensor train with ranks $(1, 2, 4, 4, 2, 1)$ is used in this case. Each element in the weight tensor is then approximately represented as a product of these tensors, thus allowing the model to learn the weighted mapping between the input tensor z and the output vector with far fewer parameters, around $11,000$, resulting in a space-saving of 4×10^{-5}. The Tensor Train layer extracts 384 features, the same number of features concatenating the original feature vectors, which are then passed to the classification layers. This approach thus incurs only a small increase in weights whilst, in theory, modelling these important higher-order interactions.

3.3.1.6 Classification

Regardless of the fusion mechanic, the fused feature vector is then passed into several dense layers to perform classification. These layers act as separate task-specific 'heads' for each task. Three heads are used, one for game context classification and two for emotion detection, as each sample has two emotion labels, one for arousal and one for valence. For each of these

Table 3.3: Distribution of annotations in the raw data set.

Valence			Arousal		Game Context								
Neg	Neut	Pos	Neut	Pos	In Lane	Shopping	Ret. to Lane	Roaming	Fighting	Pushing	Defending	Dead	Misc.
246	6,227	727	6,755	445	2,418	294	591	1,422	892	213	233	831	308

heads, the feature vector is passed through a 128 neuron dense layer with ReLU activation before a final classification layer, which applies a softmax across n neurons where $n =$ the number of classes (e.g. for valence $n = 3$).

3.3.2 Data

Video data was gathered from streamers playing *League of Legends* on Twitch.tv. The data set consists of 10 streamers, five male and five female, streaming in English. 20 minutes of footage was gathered from 3 games for each streamer for a total of 10 hours of footage. This data was then segmented into five-second long non-overlapping segments for 7200 video clips. Each clip was manually annotated by the researchers across three labels, two related to the streamer's affect and one related to what the streamer was doing in the game. While the number of streamers is limited by the effort required to manually annotated data, the size of the data set is in keeping with other works, e.g. [37, 38].

3.3.2.1 Affect Annotation

Each clip was annotated for affect, using the streamer's facial, bodily and vocal cues to judge their emotional state. Affect was annotated across two dimensions, valence and arousal. 'Valence' relates to the positive/negative axis of emotion, whereas 'arousal' relates to how strongly someone is feeling/displaying emotion. For valence, each clip was rated on a three-point scale, positive, neutral or negative. For arousal, a two-point scale, neutral or positive, was used because video games, especially *League of Legends*, are not generally designed to elicit negative arousal, and thus, negative arousal samples were not present in the data set. Therefore each clip receives a valance and arousal classification according to visual displays of the following:

Negative Valence Negative feeling, e.g. sadness.

Neutral Valence A lack of discernible valence.

Positive Valence Positive feeling, e.g. happiness.

Neutral Arousal A lack of discernible arousal.

Positive Arousal Strong emotional response, e.g. anger or excitement.

As seen from Table 3.3 there is a considerable imbalance between classes with a skew towards neutral affect in both dimensions. This is to be expected; often, gamers are engrossed in gameplay and, as a result, do not show outward emotion. This adds another complicating factor to the difficulty of learning affect in a livestream setting.

3.3.2.2 Game Annotation

Each clip was also annotated for game context, relating to what the streamer was doing during the clip. This behaviour is not always represented on screen for the duration of the clip, in most cases, due to players switching their camera in-game to observe what others are doing. Still, the annotation represents the majority of frames in the video. Eight categories were chosen, which represent the majority of gameplay events. A ninth 'miscellaneous' category was also used. This category was for occasions where the player was not doing one of the other categories.

In Lane The player is farming, i.e. killing for in-game resources, 'creeps' (game-controlled enemies) in one of the three main lanes. Often the default action, especially during the early parts of a game.

Shopping The player is spending in-game resources on items to power up the player's hero. This is done by selecting items from a tabular menu.

Returning to Lane The player is walking back to the lane after respawning, shopping or returning to base for health.

Roaming The player is roaming the 'jungle' area, the space between lanes. Potentially the player is also fighting jungle-specific creeps.

Fighting The player is engaged in player vs player combat with the enemy team. This can be one vs one or as part of group combat.

Pushing The player is pushing into and attacking the enemy base. This often occurs after the player has spent sufficient time in the lane.

Defending The player is defending their base. This can be seen as the opposite of Pushing, i.e. when one team is pushing, the other is often defending.

Dead The player has been killed and is awaiting respawn.

Miscellaneous Something not covered above, for example, the streamer has briefly left the game and is browsing the internet (e.g. if they are dead and waiting for a respawn).

Similarly to the affect annotations, there is an imbalance between game event classes, although less pronounced. 'In Lane' and 'Roaming' are the most popular activities, representing 33.58% and 19.75% of the data, respectively, shown in Table 3.3. Likewise, Pushing and Defending are the least common events, likely due to the 'snowball' effect present in *League of Legends* games, where once a team has established a dominant position, the game can end very quickly.

3.3.2.3 Data Preprocessing and Over-Sampling

Because there are large sample imbalances across all classification tasks, over-sampling is applied to aid in balancing the samples and thus improving training performance. Over-sampling our multi-label data is complex because it is crucial to ensure that over-sampling a minority class in one task does not increase the majority class in a different task. Therefore, the least and most represented classes across all annotations are calculated, with a weighting applied to account for the varying number of classes between outputs, $w_c = T_c/(T_d/(1/N_c))$ where w_c is the representation weight for a class c, T_c is the total for this class, T_d is the total data points in the data set, and N_c is the number of classes for this output. Next, a data point is selected randomly, which is in the least represented class and not in the most represented class. The selected sample is then over-sampled in the data set. This process is repeated until either a predefined threshold is reached, the chosen threshold for this work was the size of the initial dataset, or no data satisfies the selection requirement. The result of this oversampling is shown in Table 3.4.

The data is preprocessed by taking each five-second clip and extracting visual and audio frames at a rate of four frames per second. Game images are $128 \times 128 \times 3$ down-sampled images taken from the frame. They represent the game context on the screen and, similar to the previous section, have a black patch placed over the streamer's webcam. The streamer's webcam images are $64 \times 64 \times 3$ down-sampled images taken from the frame represent what is

Table 3.4: Impact of the Oversampling Technique on the Training Dataset. Values listed as a percentage of the dataset with this label.

	Valence			Arousal		Game Context							
	Neg	Neut	Pos	Neut	Pos	In Lane	Shopping	Ret. to Lane	Roaming	Fighting	Pushing	Defending	Dead
Before	0.033	0.862	0.104	0.937	0.063	0.35	0.043	0.084	0.206	0.13	0.03	0.033	0.123
After	0.259	0.483	0.257	0.725	0.275	0.181	0.097	0.097	0.146	0.103	0.181	0.097	0.181

present in the streamer's webcam and contain no gameplay data. The audio represents a joint stream containing the streamer's voice, the game audio, and occlusions such as any music the streamer is listening to and represents the raw audio waveform as a single vector of 5512. Therefore, the input data for each clip consists of 20 frames (4 fps $\times$ 5 seconds) represented as two image tensors and one audio vector. The temporal and spatial down-sampling was chosen empirically to provide a reasonable middle ground between representing the original clip and reducing the data passed into the network for performance reasons.

3.3.3 Experiment

Each model presented in Section 3.3.1 was trained on three tasks. Firstly, to learn a joint representation of both game context and streamer affect, and thus classify valence, arousal and game context simultaneously. Secondly, the task of only learning the game context. Finally, the task was learning just the streamer's affect.

Keras [36] with the Tensorflow backend is used for implementing each model. Models were optimized with ADAM [96] with $\alpha = 0.0005$. Each network was trained for 100 epochs for each learning task using an NVIDIA GTX 1080 GPU. The number of weights per model can be found in Table 3.5. For each output, a set of class weights were implemented as an additional measure to tackle the bias in the data set. To calculate class weights for each class x, an initial weight i_x is calculated as $i_x = T_d/(T_x \times N_c)$ then a scaled weight w_x is calculated so that all weights for an output sum to one via Equation 3.3. For these equations T_d is the total data points in the data set, T_x is the total data points for class x and N_c is the number of classes for this output, e.g. for a valence class $N_c = 3$ as there are 3 possible valence classifications.

$$w_x, w_y, ... = \frac{i_x}{sum(i_x, i_y, ...)}, \frac{i_y}{sum(i_x, i_y, ...)}, ... \tag{3.3}$$

Table 3.5: Comparison of Trainable Weights Across Models.

Task	Early Fusion	Late Fusion	Tensor Train Fusion
Affect + Game	9,382,029	6,629,517	6,640,939
Affect	9,331,717	6,579,205	6,590,627
Game	9,282,824	6,530,312	6,541,734

Table 3.6: F1 Scores for each affect label across all models. For each model F1 scores for each task and each label are reported. Best result for each class in **bold**.

Fusion	Task	Neg V	Neut V	Pos V	Neut A	Pos A
Early	Single	0.206	0.905	0.345	0.966	**0.540**
Late	Single	0.088	0.918	0.340	0.968	0.491
TT	Single	0.135	**0.928**	0.315	0.969	0.537
Early	Joint	0.194	0.911	**0.362**	0.969	0.509
Late	Joint	**0.286**	0.925	0.297	0.964	0.465
TT	Joint	0.102	0.926	0.325	**0.971**	0.476

3.3.4 Results

3.3.4.1 Affect and Game Context Classification

Model and label values for Precision, Recall, and F1 Score are calculated for each task. These metrics were used because accuracy, e.g. the average correct classifications across a whole data set, is not necessarily the best measure of success when testing on unbalanced data. In these cases, high accuracy can be achieved by simply always classifying the majority class, e.g. for arousal outputting only Neutral Arousal classifications would yield an accuracy of 0.94. As such, the discussion of the results will focus on the individual class F1 Score values because it represents the harmonic average of Precision and Recall. F1 scores for all models

Table 3.7: F1 Scores for each game context across all models. For each model F1 scores for each task and each label are reported. Best result for each class in **bold**.

Fusion	Task	In Lane	Shopping	Returning	Roaming	Fighting	Pushing	Defending	Dead
Early	Single	0.842	0.724	0.591	0.794	0.565	0.610	**0.667**	0.924
Late	Single	0.791	0.776	0.513	0.774	0.581	0.582	0.452	0.937
TT	Single	0.837	0.777	0.518	**0.819**	0.557	0.405	0.473	**0.948**
Early	Joint	0.778	0.797	0.496	0.667	0.515	0.568	0.544	0.899
Late	Joint	0.840	**0.828**	**0.615**	0.805	**0.635**	**0.652**	0.557	0.906
TT	Joint	**0.848**	0.765	0.580	0.791	0.630	0.635	0.574	0.930

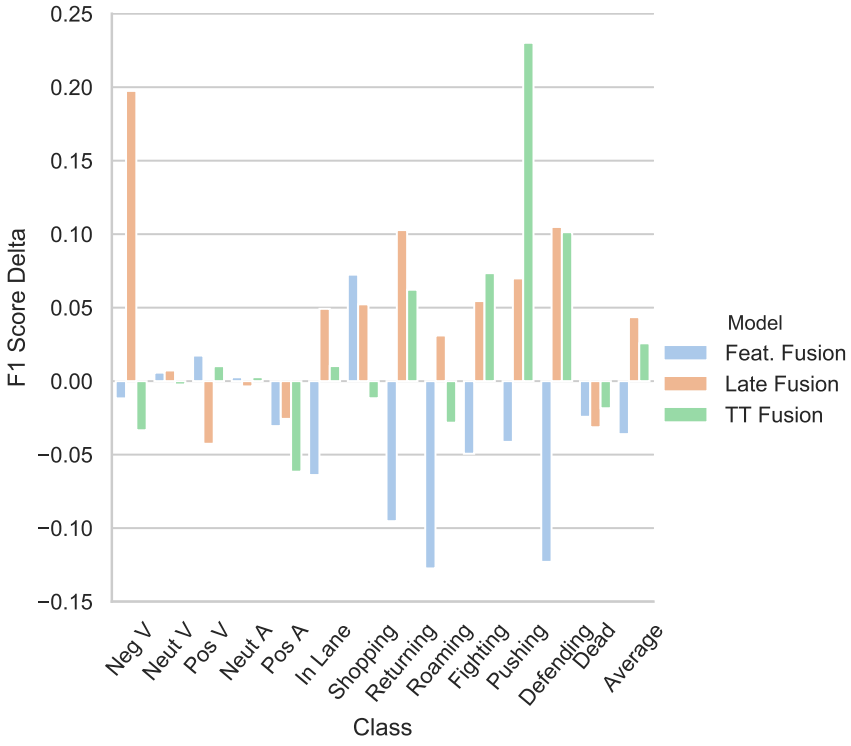


Figure 3.12: Delta change in F1 Score performance. Positive values occur when joint task learning outperforms single task learning. Negative values occur when single task learning outperforms joint task learning.

across all classes in both single and joint task learning are shown in Tables 3.6 and 3.7.

3.3.4.2 Joint vs Single Task Learning

One of the aims of this study is to explore if learning both game context and affect classifications simultaneously would improve results. While learning these tasks simultaneously has the drawback that the model has fewer variables to dedicate to each task, jointly learning the task may also lead to learning more generalisable and thus robust representations. Additionally, joint task learning requires only a single model to perform recognition across all tasks, thus resulting in faster training and inference as only one model needs to be trained/queried.

Fig. 3.12 shows the delta in performance between single and joint task learning, showing that whilst there is a large degree of between-class variance, the models which perform fusion after the LSTM step see an improvement when learning jointly, whereas the early fusion model performs worse.

3.3.5 Discussion

3.3.5.1 Affect and Game Context Classification

Tables 3.6 and 3.7 shows that affect classification is a much harder task than game context classification, further reinforcing the discussion regarding the difficulty of in-the-wild affect detection. Importantly, while results such as a high of 0.362 for F1 Score for positive valence and a high of 0.286 for negative valence may seem poor, they are significantly higher than a random baseline, which has expected F1 scores of 0.044 (negative) and 0.106 (positive), because of the hugely imbalanced data set. These affect detection results are in keeping with similar works, for instance, [202]. There is still interesting research to be done to accurately model streamer emotions. However, a deep analysis of suitable techniques is outside this thesis's scope. It is clear from these studies that affect detection is an important but challenging task. However, Section 3.2 indicates that modelling highlight moments may inherently model some elements of affect.

Regarding game context in general, all models perform better when classifying examples of 'In Lane', 'Shopping', 'Roaming' and 'Dead', where F1 scores range from 0.667 (Early Fusion, 'Roaming', joint task learning) to 0.948 (TT Fusion, 'Dead', single-task learning), compared to other context classes. It is difficult to know precisely why the models perform better in these categories, but it is possible confusion arises for classes such as 'Pushing' and 'Defending', which can be visually similar. Consider two players on opposite teams are very close to each other and at the edge of a base. Only players' positions and prior knowledge about which side they are on provide clues as to if the streamer is pushing or defending. Contrast this with the 'Dead' class, where the screen is mainly greyscaled with a fixed message, resulting in a more straightforward classification.

3.3.5.2 Joint vs Single Task Learning

As Figure 3.12 shows, for late fusion and TT Fusion, there is a general if inconsistent improvement when learning a joint representation with average F1 Score improvements of 0.044 and 0.026. However, early fusion degrades F1 Score performance with an average change of -0.036. Furthermore, a Wilcoxon Signed-Rank Test shows that both the change in early fusion and late fusion are statistically significant at $P < 0.05$, $P = 0.047$ and $P = 0.03$ respectively. A key takeaway from these results is that seemingly fusion techniques occurring

after temporal modelling see improved results from learning both tasks jointly, but early fusion, which occurs before the LSTM, sees a performance reduction.

3.3.5.3 Determining the 'Best' Model

It is difficult to ascertain from these results which model performs the best. No model is a 'winner', i.e., outperforming the other approaches across all categories. Each model appears to perform better on some tasks and worse on others. Early fusion outperforms TT fusion and late fusion for emotion prediction tasks, although TT fusion appears best at classifying the neutral, majority, classes. This is possibly due to the LSTM layers applied to the fused representation in the early fusion systems. This format may facilitate a temporal, multimodal representation that can account for cross-modality cues manifesting at different points in time, e.g. anticipatory co-articulation. In general, post-LSTM fusion (late, TT-fusion) provides better performance at game context recognition. Furthermore, TT fusion seems to slightly outperform late fusion in emotion recognition, perhaps due to TT fusion better modelling the interactions between modalities.

The early fusion network has significantly more weights than the other two models, Table 3.5. Additionally, it is the only model that sees performance degradation when joint task learning. Therefore whilst its performance is comparable to other models, it uses approximately 3 million more weights and therefore has a much higher computational cost. Late and TT fusion models are roughly only $2/3^{\text{rds}}$ of the size of the early fusion network.

3.3.5.4 Applying Joint Task Learning to Highlight Detection

This thesis is interested in highlight detection. Until this point, this study has not directly considered this application, although the value in understanding affect and game context is clear. The classification models presented in this chapter can be used for highlight detection by segmenting a video across time and then processing each video segment by calculating the predictions for each class across all tasks. These signals can then be used, alongside prescriptive measures of highlight, e.g. the player is in a fight and has high arousal, to select moments.

As discussed previously, determining ground truth highlight labels is exceptionally challenging. The downside of this approach is that initially, it appears a human editor is required to define which combination of signals represents a highlight. However, it may be possible

to mitigate this by employing a second model which can learn the mapping between signals from the models presented in this work and highlights. This model would require only a few weights, given that the inputs would be at maximum 13 scalar $[0, 1]$ values. However, this lightweight model, alongside initial human-sculpted highlight definitions, could potentially be deployed and fine-tuned over time using crowd-sourcing labels, e.g. via social media feedback. Furthermore, this approach has the advantage that because the definition of a highlight can be easily tweaked, it is possible to develop personalised highlight detection models.

3.3.6 Conclusions

In this section, the problem of modelling streamer affect jointly with game context is posed within the framework of a deep learning architecture that fuses audio-visual stream data. Furthermore, an annotated dataset of emotion and game context for video game livestreams is discussed, the first of its kind at the time of research. The benefits of such data were discussed. For example, the ease of training supervised learning models. The pitfalls, for example, the difficulty in annotating such data, are also discussed.

Furthermore, results comparing early and late fusion approaches are presented. Results show that the, in general, post-temporal modelling fusion generally outperforms early fusion across tasks. Additionally, the difficulty of affect classification in this environment is shown. Jointly modelling game context and streamer affect in a multimodal setting constitutes an extremely challenging problem. That said, the performance of the supervised approach suggests that should a suitable game context and emotion to highlight mapping be developed, either manually or through crowd-sourced fine-tuning, this approach may perform and generalise better than the unsupervised approach.

3.4 Personality-Driven Livestream Highlight Detection Conclusions

This chapter has explored two techniques for highlight detection in personality-driven livestreams. These techniques were both, in part, successful. However they were also very different and thus spotlight different challenges. For example, it is clear from the novelty study that unsupervised learning is helpful because it does not require human data curation. This is

extremely valuable in livestream highlight detection, given the vast quantity of data across hundreds of different games. However, the model has several limitations. Firstly, it is unlikely to generalise well to certain games as it requires games with large amounts of low-intensity gameplay and a few short moments of high-intensity gameplay. Secondly, it approaches modelling through a one-shot style approach where the model is fit for each stream which, while effective, has the downside that processing is time-consuming. However, it did indicate how strong the streamer's emotional reaction was in determining highlights.

These observations drove the development of the second study discussed in this chapter. While the joint emotion and in-game event modelling showed promise, especially in classifying in-game events, this study highlighted how challenging hand annotation could be when training machine learning models which require massive datasets. For instance, the 10 hours of data took several days to annotate. This challenge would be even more significant if the model were instead trained to predict highlights, given that highlight annotation requires subjective judgement and an understanding of the broader context, which would increase the annotation effort. This study also showed that various fusion mechanisms perform better under different circumstances. For instance, early fusion is generally better for affect modelling, but late/TT fusion is generally better for game context modelling.

Chapter 4

Game-Driven Livestream Highlight Detection

4.1 Introduction

While Chapter 3 dealt with audio-visual highlight detection for personality-driven livestreams, this chapter examines game-driven livestreams. These game-driven livestreams are fundamentally different to personality-driven livestreams. Certain features of personality-driven streams are not present in game-driven streams, for example, the streamer's webcam overlay as well as their focus on interacting with viewers. However, because game-driven streams are popular, it affords additional opportunities for highlight detection, especially considering determining highlight labels.

Automatic sports highlight detection has been a popular area of research for some time. In recent years the rise of esports, competitive video games often broadcast on livestreaming services, has generated a demand for esports specific highlight detection tools. This chapter, as with the second study in Chapter 3, focuses on *League of Legends*[1]. However, this chapter exclusively studies professional esports broadcasts, unlike the previous chapter where personality-driven *League of Legends* livestreams were used. Due to their video game-based nature, esports have the advantage that those who play them professionally often also livestream out-of-competition play sessions as an additional revenue source. A typical content cycle would be for a live broadcast to occur, either as part of an esports competition or from an individual streamer, and then a highlight video would be manually created by a

[1] https://www.leagueoflegends.com/

video editor. This highlight video shows the most exciting moments of the broadcast and can be shared through video-sharing platforms, e.g. YouTube. This dual content system allows fans to catch up on a broadcast if it was missed and mirrors traditional sports highlights broadcasts, although for esports the two streams are often on different platforms rather than both being televised. This chapter contains a single study into esports highlight detection, presented in Section 4.2, while Section 4.3 concludes this chapter.

4.2 Crowd-Sourced Highlight Detection

This section tackles audio-visual highlight detection through weakly supervised learning with positive-unlabelled data. The use of positive-unlabelled data is motivated by the challenges in labelling data discussed in Chapter 3. Weakly supervised learning has many apparent advantages over traditional supervised learning. Namely, it is possible to leverage massive amounts of data due to the minimised human cost of data gathering. For this study, this is achieved by gathering two datasets. One is of raw, unedited broadcasts, and the second is a dataset of edited highlight videos. Human editors have made these highlight videos and released them on video-sharing platforms like YouTube. Moments are considered a highlight if these video editors select them, and thus this work takes a crowd-sourced approach to highlight definition, similar to [218]. By selecting a range of highlight videos from multiple highlight editors, this study aims to generate highlight detection methods that can broadly recreate the style of already popular videos. This data formulation, therefore, results in one dataset, the broadcasts, with unknown labels, and the other with only positive labels, hence the application of positive-unlabelled weak supervision.

In particular, this study focuses on using data in its raw format, i.e. unedited broadcasts and highlight videos, as these most closely mimic a real-world use case for such a technology. Therefore, videos often contain adverts, analyst desk segments, and other non-game footage. This decision is motivated by the desire to evaluate operationalisable techniques and the observation that human data processing is a bottleneck for such a system. Additionally, several state-of-the-art highlight detection methods are implemented to provide comparisons for the 'Auto-Highlight' models introduced in this study.

4.2.1 Methodology

Existing state-of-the-art weakly-supervised highlight detection models primarily have similar components to those discussed in Chapter 3. The raw data is generally passed through feature extraction systems, e.g. a Convolutional Neural Network (CNN) or audio frequency analysis techniques. These features are then used in the downstream systems. Towards the end of the pipeline, there must be a decision-making mechanism that determines if the incoming features form part of a highlight or not. The output from this decision-making mechanism is usually the final output of the system and consists of a time-series signal over the length of a video, where each point in the signal relates to the likelihood that a single video segment is a highlight. For these weakly-supervised systems ranking networks have become popular for this task. Finally, at some point the data from each input modality, e.g. visual and audio data, must be fused. Once again, exactly how this fusion mechanism is implemented can vary between models. Two methods are trialled in this study, feature fusion, i.e. fusion of the latent features, via concatenation and model fusion, i.e. aggregation across a selection of models. For a particular penitent example of model fusion, see [213].

For this study, these three processes; feature extraction, fusion, and decision making, are deliberately separated rather than trained end-to-end as with the emotion and game event work in Chapter 3. This is done to explore the utility of using pretrained feature extraction networks. The system architecture proposed in this work is discussed in detail below and is additionally presented in Figure 4.1.

Very few state-of-the-art models utilise data in a format similar to the proposed labelled-unlabelled data. The only existing model which approaches audio-visual highlight detection for esports titles using this approach is [213], which is implemented as a state-of-the-art comparison model. Additionally, the architecture from [218] is implemented as it inspired the Auto-Highlight approach, although it cannot utilise the benefits of the multimodal nature of broadcast data because it omits a fusion step.

4.2.1.1 Feature Extraction

Different pretrained feature extraction models are required for each modality. While there are only two data sources, video frames and audio, the data is treated as four separate modalities. Firstly, a stack of RGB video frames. Secondly, RGB frames are converted into optical flow frames to represent motion within a video segment. Thirdly, features are

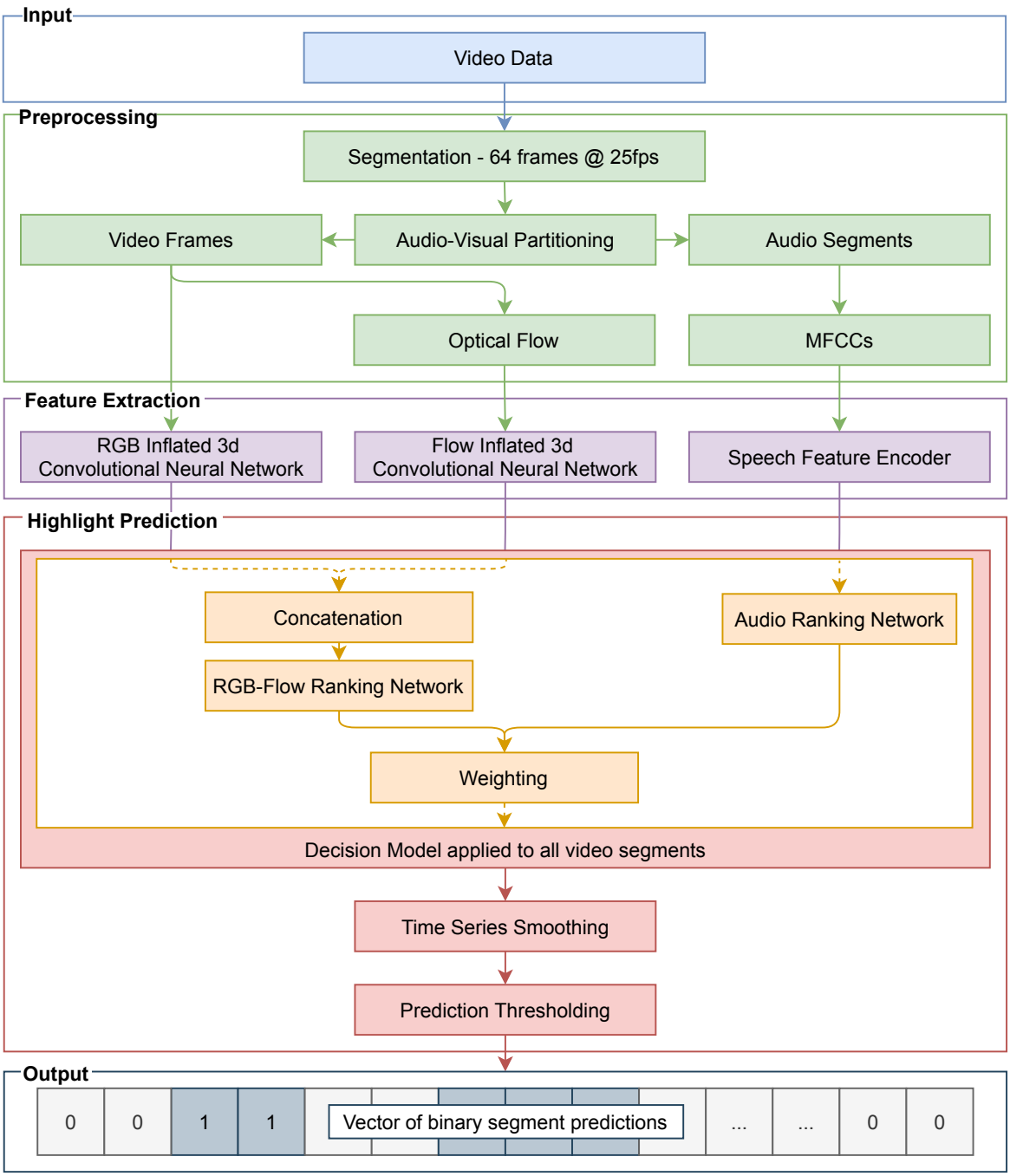


Figure 4.1: Full system architecture for Auto-Highlight model using multimodal hybrid fusion and smoothing. This figure details the end-to-end system, from input data, i.e. video data, through preprocessing, feature extraction, segment prediction and finally, smoothing and thresholding operations. The output is a binary vector where each element relates to if a video segment is predicted as a highlight or not.

extracted from only a single frame. Finally, features from audio samples. Note that there is no need to extract webcam frames from the game scene for game-driven streams because this paradigm does not feature a single personality and therefore lacks the need for a webcam overlay. Precisely how these features are extracted is detailed below.

Video Feature Extraction There is evidence that pretraining a network on a large, generic dataset outperforms training a network to generate features for a particular task, e.g. [6]. To this end, this study utilises a state-of-the-art inflated 3D convolutional network from [28] which has been pretrained for action recognition on the kinetics dataset [91]. This two-stream model considers RGB video frames (RGB) and optical flow frames (Flow). It processes both modalities separately but using the same convolutional architecture. A 3D CNN is similar to a 2D CNN, e.g. as used in Chapter 3, but with an additional convolution dimension that convolves over time, represented as a stack of frames. A 3D CNN is used rather than a CNN-RNN style architecture like in Chapter 3. This is motivated by recent works which have shown that, at least for short video clips, a 3D CNN is generally more performant [28]. Using a pretrained inflated 3D convolutional network has the limitation of requiring a fixed input dimension equal to the pretrained model. This means that the input time dimension is a constraint, unlike a CNN-RNN architecture. Therefore visual models use frame sizes of $224 \cdot 224\cdot$ pixels and a stack of 64 frames total. Video data is sampled at 25 frames per second, and therefore 64 frames equate to 2.56 seconds of video per training sample. For both data streams, RGB and Flow, 1024 features are generated for each 64 frame video segment. The RGB network receives raw frames taken directly from the video. This study uses the optical flow method proposed in [53] on greyscale frame data for the optical flow network.

Audio Feature Extraction Audio data is represented as a time-series vector of data sampled at $192,000$ hertz. To keep the length of the audio samples consistent with the video data, $491,520$ audio samples, i.e. 2.56 seconds, are grouped into a single audio training example. These samples are then processed by calculating the mel-frequency spectrograms and passing these into a state-of-the-art pretrained speech feature encoder [85]. The encoder consists of several LSTM layers and generates 256 audio features for use in the downstream decision-making model.

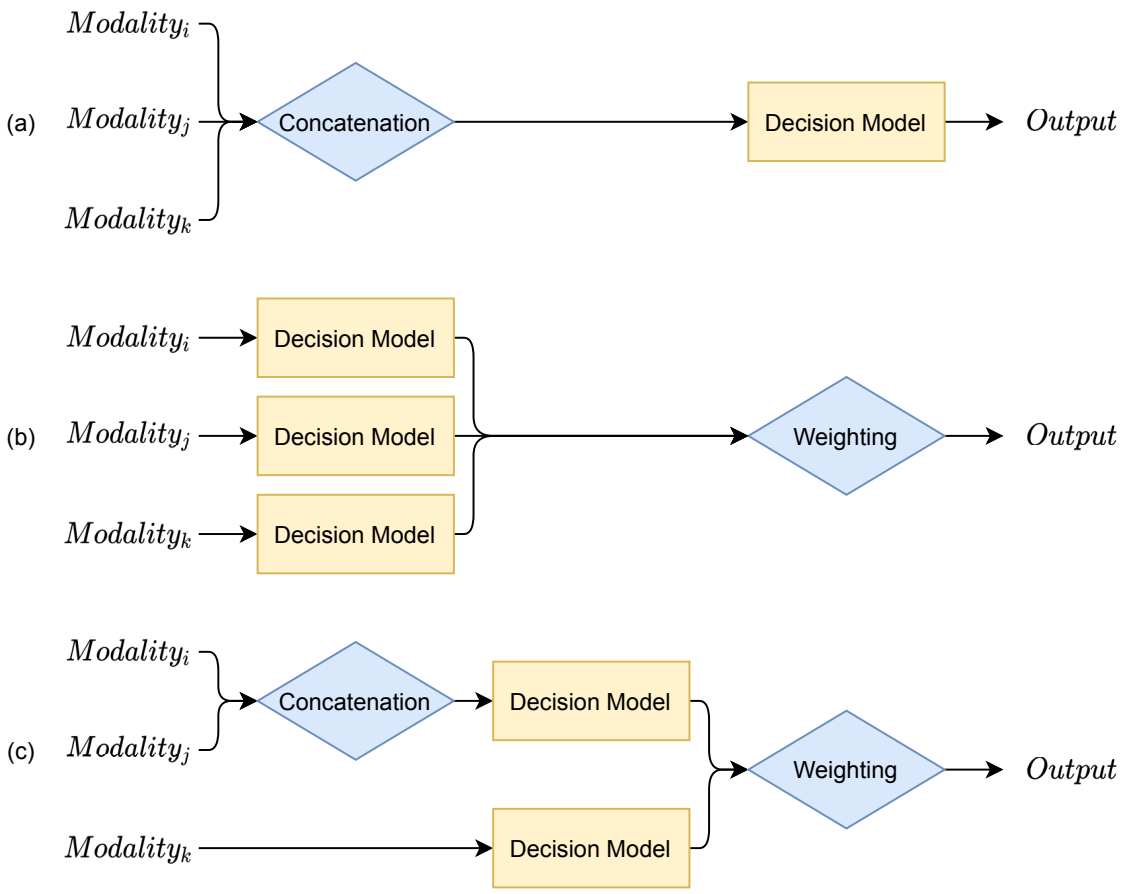


Figure 4.2: Comparison of Feature (a), Model (b), and Hybrid (c) fusion architectures. For simplicity three modalities are shown but in practice any number of modalities can be used.

Single Frame Extraction One of the existing state-of-the-art approaches implemented, [213], uses features from a 2D CNN that extracts features from only a single frame in addition to the video features detailed above. An Alexnet [102] network trained in object recognition on the ImageNet database [42] is utilised as a frame-based feature extractor. Features are extracted from the final bottleneck layer before classification is performed, resulting in a feature vector of 9216 features for a single frame.

4.2.1.2 Fusion

Three fusion methods, detailed below, are presented in this work. For a visual overview of how these fusion methods differ, see Figure 4.2.

Feature Fusion Feature fusion is perhaps the most intuitive method for fusion. After feature extraction, the modalities are fused, e.g. via concatenation, resulting in a single

decision model provided with all modalities. However, multimodal feature fusion may not work well in certain situations. For example, if there is a significant disparity between the number of features from the modalities, the modality with more features will likely be over-represented in the decision-making model output. This is especially worth considering when comparing audio feature extraction, which has 256 features, and single-frame feature extraction, which has 9216 features. Therefore, this approach is reserved for the modalities which produce the same number of features, i.e. RGB and Flow. Furthermore, feature fusion is performed by concatenation and not Tensor-Train fusion because it is only used for fusing RGB and Flow inputs, which are unlikely to benefit from the more complex fusion method given that they come from the same data source.

Model Fusion Model fusion utilises a decision-making model for each modality and then fuses them via weighted aggregation of the outputs. For a given video segment, a highlight prediction is generated for each modality via separately trained decision-making models, p_m. The predictions for each modality are then weighted by w_m. This weighting allows the mechanism to be fine-tuned based on the performance of each modality if desired, equal weighting is equivalent to aggregation. The weighted predictions are then summed to produce a single segment prediction p, i.e. $p = \sum_{m \in modalities} p_m w_m$. Past studies, for example, the second study in Chapter 3 have shown that sometimes fusing later in the system can improve performance. That said, model fusion is performed on decisions rather than features and thus loses fidelity in the fusion step, although it is unclear how much this impacts performance. In a positive-unlabelled setting such as this study, the weights for each modality need to be determined heuristically. This is because it is impossible to evaluate the model performance on training data that mimics the test data, unlike in a supervised setting. For all models, an equal weighting for each modality is used. However, this is not a requirement, e.g. [213] used uneven weighting, which put more focus on the video model. For their work, they selected the weights $\{0.7, 0.15, 0.15\}$ for RGB, Frame, and Audio outputs, respectively.

Hybrid Fusion Finally, a mixed 'hybrid fusion' approach is proposed. The RGB and Flow feature extraction networks are applied to the same input data domain and have the same number of features. Therefore it makes sense to perform feature fusion because it is performed on the rich domain features, thus providing more information to the decision model. However, feature fusion is not appropriate for cross-domain modalities, i.e. visual

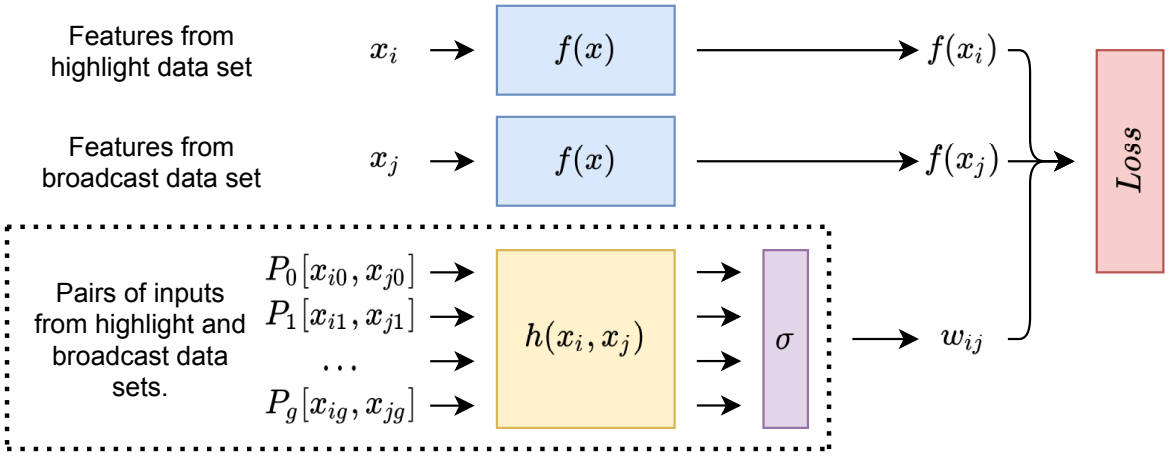


Figure 4.3: The rank network architecture. $f(x)$ and $h(x)$ are both multi-layer perceptrons.

and audio data, due to the disparity in feature vector size. Therefore model fusion needs to be employed to fuse audio and visual modalities. Feature fusion is applied to the two visual modalities to maximise performance, and model fusion is applied to fuse the visual decision-making model and the audio decision-making model. Hybrid fusion has the added benefit of equally weighing visual and audio decision-making. This hybrid fusion technique allows the system to utilise rich features from the same domain, e.g. RGB and Flow data, while maintaining the ability for the model to work with other modalities, e.g. audio, without having a significant imbalance in terms of the number of features per feature set. Thus, hybrid fusion provides the best characteristics of feature fusion of the visual modalities and RGB, Flow, and Audio model fusion.

4.2.1.3 Decision Making

Ranking Models At some point, a decision needs to be made about if the extracted features constitute a highlight or not. In Chapter 3 this was achieved as part of the LSTM layers in the first study or fully connected layers in the second study. However, this is a trickier task for this study because the training labels are known to be extremely noisy. Data within the 'highlight' dataset are established as positive labels, as segments in this dataset are selected by a video editor to be a highlight. However, it is known with certainty that the broadcast dataset is noisy. It must contain a mixture of highlight and non-highlight moments because the highlight moments are sampled from the original broadcast data. Therefore, a noisy-label mitigation mechanism is likely worthwhile regardless of the decision-making

mechanism.

Ranking networks are popular in highlight detection literature; thus, this study employs one for decision-making. A ranking network is a multilayer perceptron with a single output. The model is provided with pairs of samples during training, one from each dataset. The expected output is that the sample from the highlight dataset is 'ranked' higher than the sample from the broadcast dataset, i.e. the output for the highlight sample is greater than the broadcast sample. Each sample uses the same set of layers such that weight updates affect the ranking of samples from both datasets. Often for such models, a Hinge loss is used for weight adjustment.

The Auto-Highlight models follow the modifications proposed by [218]. Namely, the addition of a secondary network responsible for determining if an input pairing is 'valid' or not. This is a noisy-data mitigation technique aimed at alleviating issues when a particular input pairing contains a sample from the broadcast dataset which should be considered a highlight. In this case neither sample should be ranked higher than the other. Therefore this secondary network takes a group of samples and, using a heuristic noise prior, applies a weighting to all losses in that batch. This modification performed well in [218] and given that their noisy data is similar to this study, such a modification is likely to yield good results.

For this study, the assumption is that one in eight samples is invalid. This heuristic was determined from the observation that matches are often around 40 minutes to one hour long and highlight videos tend to be between five and eight minutes. Therefore eight pairs of samples are grouped per training input. A softmax is applied to the output from the 'validity' network for these eight samples and then used to weight the ranking loss function, detailed in Equation 4.1 where f is the ranking network, h is the validity network, s denotes a sample, and Pg denotes a group of samples. All layers use a ReLU activation function; thus, the output from the model is positively unbounded, which is necessary to rank samples correctly. For a further detailed discussion regarding this modification and its motivation, the interested reader is referred to [218]. Figure 4.3 details the architecture used.

$$
\begin{aligned}
Loss = \sum_{g=1}^{m} \sum_{(s_i,s_j)\in P_g} \tilde{w}_{ij} max(0, 1 - f(x_i) + f(x_j)) \\
s.t. \sum_{(s_i,s_j)\in P_g} \tilde{w}_{ij} = \sum_{(s_i,s_j)\in P_g} \sigma_g(h(x_i, x_j)) = 1 \\
\tilde{w}_{ij} \in [0, 1]
\end{aligned}
\tag{4.1}
$$

Table 4.1: Architectures for each ranking model. The architecture is detailed as a list of feedforward neuron layers, reading left to right. The weights of the ranking network, $f(x)$, and validity network, $h(x)$, are also detailed. Memory requirements are listed in MB and are an approximation. *Feature fusion, † [213].

Modality	Architecture	$f(x)$ Weights	$h(x)$ Weights	Training Memory	Inference Memory
RGB	[512, 128, 1]	590,593	1,114,883	281.10	0.60
Flow	[512, 128, 1]	590,593	1,114,883	281.10	0.60
RGB + Flow*	[1024, 512, 128, 1]	2,688,769	4,785,923	572.08	2.70
Audio A†	[64, 1]	16,513	32,899	68.68	0.02
Audio B	[512, 128, 1]	197,377	328,451	78.60	0.20
Frame	[4096, 1024, 512, 256, 128, 64, 1]	42,645,507	80,394,243	2,639.04	42.71

Ranking Model Weights Because each modality has a different number of input features, it makes sense to vary the number of layers in the ranking network accordingly. The architectures used are detailed in Table 4.1, alongside details about the number of weights and memory requirements of the models. Two values are given for the memory requirements of each model. Training memory refers to training both $f(x)$ and $h(x)$ whereas inference memory refers to just calculating a single prediction value from $f(x)$. Accurate memory profiling with Keras is currently an open issue, and thus this study presents an approximation[2]. Note that the implementation for [213] uses an architecture of shape $[64, 1]$ for the audio ranking network (Audio A). However, it performed poorly. A $[512, 128, 1]$ architecture (Audio B) is used for all models except the comparison to [213].

4.2.1.4 Binary Classifier

Section 4.2.4.4 compares the ranking approach to a binary classifier. Ranking models are popular in literature, yet comparing ranking with binary classification is still valuable, given that most prior studies do not do this. This comparison examines if the assumed benefits of using a ranking network exist compared to using a more a straightforward binary classifier. To test a binary classification network, the ranking models used in the hybrid fusion model are replaced with binary classifiers. These classifiers are multilayer perceptrons with the same topology as a single instance of the ranking network $f(x)$, without the additional validity network. They are trained on the same data by assuming that all samples in the

[2]https://github.com/tensorflow/tensorflow/issues/36327. The solution proposed by James Mishra is used for this work.

highlight dataset are positive samples and all samples in the broadcast dataset are negative, although it is known that the broadcast dataset contains mixed labels. Additionally, because the classifier network is performing binary classification, the activation for the neuron in the final layer is sigmoidal, not ReLU, and the loss is calculated using binary cross-entropy.

To mitigate the known deficits of this training paradigm, i.e. that broadcast samples are known to be noisy, a positive-unlabelled weighting technique, presented by [52], is implemented. A small portion of the training dataset is reserved, in this case 10%. Then the average output for all positive samples, $\overline{o_p}$ in this reserved data is calculated. This value is then used as a weighting for the output when determining if a sample is a highlight or not, i.e. for a sample x the output $f(x)$ becomes $f(x) \times \overline{o_p}$. Intuitively, the average output for all positive samples can be seen as a confidence score relating to how confident the network is that a known positive sample is positive. Using this confidence as a weighting reduces the model's output and, as [52] shows, shrinks the decision boundary, thus making positive classifications less likely but improving precision. This weighting does not affect average precision calculation, the main metric used in experiments for this study, but has a noticeable effect on classification given a fixed decision threshold, i.e. as in a deployed setting. These weighting constants were calculated as; RGB and Flow: 0.97, Audio: 0.71.

4.2.1.5 Time-Series Smoothing

One apparent weakness of this ranking approach is that the model must decide if a video segment is a highlight based on only that segment. However, in reality, a human video editor would not do this. Instead, they would use the context of surrounding video clips to inform their decision. For instance, an uninteresting segment would likely be included if it links two exciting sections. Likewise, a short but exciting section may be omitted from the highlight video if clips before or after are uninteresting or contribute little to the match's narrative. However, influencing the segment decision-making based on factors outside that segment is impossible during training because positive label samples, i.e. those taken from the highlight dataset, are gathered without non-highlight context. The only way in which it would be possible to intrinsically model context would be if the dataset was fully labelled, e.g. scenarios (a) and (b) in Figure 4.4, although as discussed in Section 4.2.2, this is infeasible for the data set gathered in this study. Therefore, a feature-agnostic context modelling technique may be valuable if applied to decision-making.

A solution to this challenge is to apply a smoothing convolution to the highlight signal across the whole video, which allows for context modelling in a heuristic manner. This smoothing increases the highlight signal for segments in a broadly engaging video portion and reduces the signal for moments in a generally uninteresting segment. Because the smoothing convolution is applied to the highlight signal rather than included as part of the rank model training, it can be applied to positive-unlabelled data. A convolution kernel of length $L = 101$ is used because this roughly equates to 30 seconds of footage surrounding the sample due to the sliding window segmentation approach described in Section 4.2.3. The convolution kernels are all symmetrical, so, for technical reasons, a half-kernel length is used in kernel initialisation of $n = \lceil L/2 \rceil$. It is not clear which values would be most appropriate for this smoothing kernel, so three options are implemented:

- Linear: A uniform smoothing kernel where each kernel element has the same value. The kernel initialisation is defined as $k = 1 \in R^n$.
- Gaussian: A Gaussian with height 1 and the peak at the centre of the kernel. The standard deviation is set to $n/2$. The kernel initialisation is defined as $k = \text{Gaussian}(i, 1, 0, n/2), \forall i \leq n$.
- Power of 2: Each element is calculated as 2^{-x} where x is the distance to the centre of the kernel. The kernel initialisation is defined as $k = 2^{-i}, \forall i \leq n$.

All kernels are then normalised such that the kernel sums to 1 by Equation 4.2. For a given point in the time-series highlight signal x_i and kernel k the smoothed value x'_i is calculated via Equation 4.3. In theoretical terms, position one in a kernel vector represents the kernel's centre, position two represents the two values on either side of the centre and so on. When calculating the resultant value, the absolute index is used to correctly index the right element of the kernel vector when calculating the value for time-series samples before the current time step.

$$k' = \frac{k_i / min(k) \forall K_i \in k}{\sum_{i=0}^{k} k_i} \tag{4.2}$$

$$x'_i = \sum_{j=-n}^{n} x_{i+j} \cdot k_{|j|} \tag{4.3}$$

4.2.2 Data

4.2.2.1 Training Data

This work tackles highlight detection through weakly-supervised learning by utilising two datasets, one of positive samples and one of unlabelled samples. This data-label formulation is similar to traditional semi-supervised learning, but labels only exist for positive samples. There is a considerable wealth of both broadcast data and fan-made highlight videos. Broadcast data contains a mixture of highlight moments, which the model aims to capture, and non-highlight moments, whereas highlight videos contain only highlight moments. In an ideal world, it would be possible to find the highlight moments in the broadcast through manual annotation, for example, by utilising a set of participants who would watch the broadcast and select their favourite moments. However, as discussed throughout this thesis manual annotation is extremely expensive and time-consuming, prohibiting leveraging the benefits of applying deep learning to massive datasets. Another reasonable approach would be to align the highlight video segments with those in the original broadcast, e.g. by detecting which frames from the broadcast are also in the related highlight video, e.g. the text dataset detailed in [58], discussed in Chapter 5. However, while less time-consuming than manual annotation, it still has a significant data gathering penalty as naming conventions for both broadcasts and highlights are inconsistent, requiring human preprocessing. Furthermore, the frame-matching process can be computationally expensive. Such an approach also requires a corresponding highlight video for every broadcast video and a broadcast video for each highlight video in the dataset. These challenges make automated broadcast-highlight matching impractical for large-scale datasets.

Instead, a positive-unlabelled approach is proposed in this study, inspired by [218]. This approach involves collecting two datasets, one of the mixed labels from broadcasts, and one of positive labels, from highlight videos. The key to minimising the cost of gathering data is that there is no connection between the datasets, i.e. the datasets do not necessarily contain matching source broadcasts and highlight videos. The broadcast dataset contains replays of *League of Legends* professional broadcasts from various competitions. The dataset of human-made highlight videos was gathered from multiple highlight makers who post their videos online. Some highlight videos contain footage from a single game. Others show highlights from multiple games, especially if the contest is in a 'best of n' format. Figure 4.4

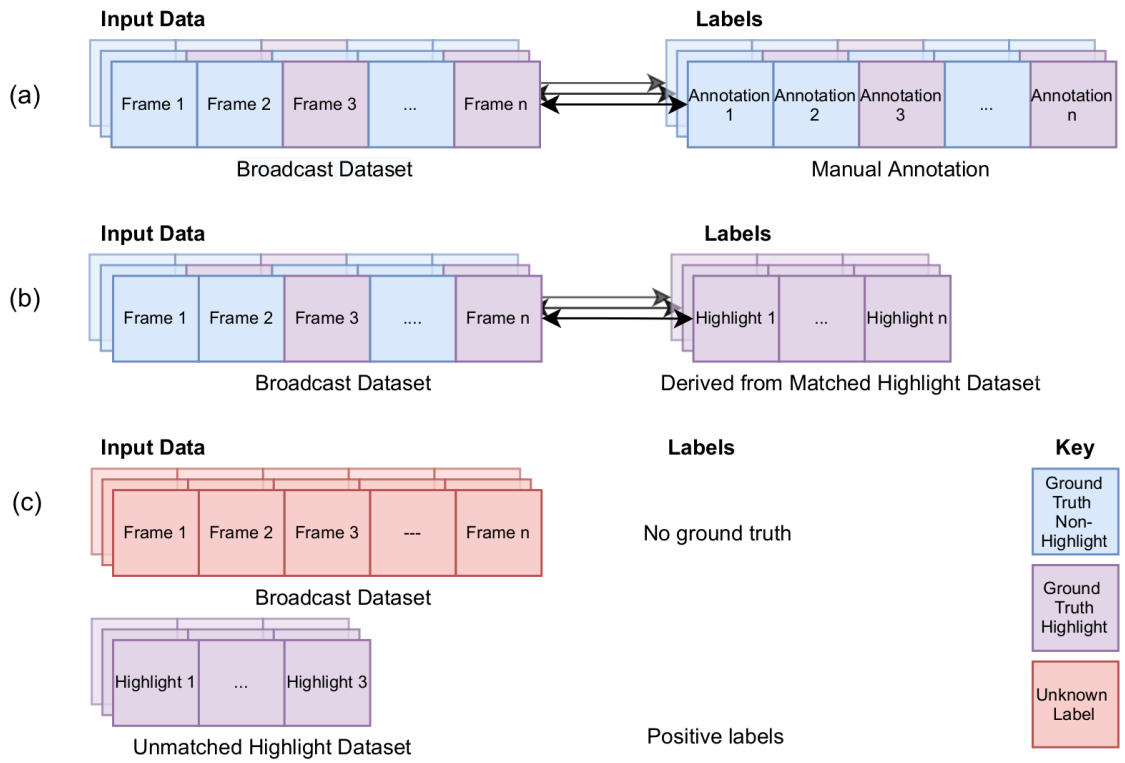


Figure 4.4: Comparison of the different potential dataset formulations. Notice that for traditional manual annotation (a) or frame-matched annotation (b), each video in the broadcast dataset needs a corresponding set of labels, either through expensive manual annotation or a matched dataset of highlights. This is not the case with the positive-unlabelled data (c). The cost to gather training data is therefore minimised, and there is no requirement to have corresponding highlight videos for each broadcast video.

demonstrates how this data differs from the two options for supervised learning and, in doing so, minimises the cost of gathering data. In total, 257 broadcast videos and 676 highlight videos were gathered, resulting in more than 193 hours of broadcast video and more than 178 hours of highlight footage. Minimal preprocessing was applied to most closely mimic a real-world automated system. Each video was processed by converting frame dimensions to $224 \times 224 \times 3$. Additionally, the frame rate was set to 25 frames per second. No audio preprocessing was applied.

4.2.2.2 Test Data

While the training data structure follows the positive-unlabelled paradigm, highlight detection in esports broadcasts is an emerging area of research, so there is currently a lack of test sets with which to evaluate models. Therefore, a test set of data with labels is constructed

and used to evaluate the range of models presented in this study. 20 games are selected, five each for the four major *League of Legends* esports leagues - LCS (North America), LEC (Europe), LPL (China), and LCK (Korea). Following the frame matching annotation technique described above, the human-made highlights from the two most popular highlight channels are manually retrieved, and frame matched to the original broadcast to find which points in the main broadcast the highlights appear. These highlight channels are popular, having $275,000$ and $356,000$ subscribers. In contrast, the official *League of Legends* esports channel hosting full-length games has fewer subscribers with $252,000$, and no other highlight channels have more than $40,000$ subscribers. This process results in a test set of 20 complete games, constituting 18 hours, 6 minutes and 51 seconds of broadcast data. It also contains a broad mix of broadcast settings, players, advertisements, and commentators from across the world, thus mimicking real-world challenges, as shown in Figure 1.5. This size of the test set is in keeping with prior works; [213] used just four videos for evaluation, and [37] used 24 games but from only one tournament, the 2014 World Championship.

Frames from both highlight videos are compared to the associated source broadcast to determine ground truth annotations for a given broadcast video. The broadcast frame that is the closest match is selected, given a distance threshold because some highlight videos contain content absent in the original broadcast, e.g. the content creator's brand. Frames are subsampled by a factor of eight for processing speed, resulting in a highlight annotation fidelity of 0.32 seconds. This process is still computationally expensive; thus, it took several days to calculate labels for the test dataset. This observation further highlights the advantage of the positive-unlabelled approach for training data because it deliberately minimises processing time. Segments are labelled as a highlight if they appear in either of the two highlight videos to capture the breadth of content, which can be considered a highlight. This process results in 203805 individual test set samples, with the shortest video contributing 4840 samples and the longest contributing 14498 samples. Overall each video in the test set contributed a median of 10003 samples.

4.2.3 Experiment

In total, four experiments are presented in this study. Firstly, a comparison of all single modality models, including [218]. Secondly, a comparison of all multimodal models, including the Auto-Highlight Hybrid fusion approach and [213]. Thirdly, experimentation of applying a

time-series smoothing step. Finally, a comparison of ranking models and binary classification, with and without noisy label mitigation. All models are implemented in Python using a range of popular machine learning libraries. PyTorch [153], Keras [35], and Tensorflow [1] were all used to compose and train models, with Numpy [70], Scikit Learn [154], Librosa [121] and OpenCV [21] used for a range of auxiliary functions. There is a highlight label fidelity of 0.32 seconds. However, due to the limitations of the feature extraction models, each clip must be of length 2.56 seconds. Therefore a sliding window system is applied to input samples. For this sliding window, a segment's annotation is calculated from the timestamp in the middle of the segment. All models were trained using 2 PCs. The first has a Ryzen 5 1600 CPU, 16GB of RAM and an Nvidia 2080ti GPU. The second had an Intel i7 9700 CPU, 64GB of RAM, and 2x Nvidia Titan RTX GPUs. These machines were chosen because they represent consumer-grade hardware and thus indicate if the techniques described in this work are likely to be operationalisable.

All experiments compare several models by evaluating their performance on the test set. The principal metric for performance is the mean average precision ($\bar{m}AP$). This metric is popular in rank-based highlight detection literature because it evaluates performance without determining a decision boundary for the model, which is a tricky task given that ranking networks are deliberately trained without one. $\bar{m}AP$ is calculated by taking the mean of the average precision (AP) for all videos in the test set. AP is calculated by averaging the precision achieved at all recall values for all samples in a video. It is unclear if this average precision is expected to be normally distributed, and as such, the median average precision ($\tilde{m}AP$) is also reported. Precision, in general, is a reasonable metric for evaluating the models because the quality of a highlight video suffers if it contains many dull non-highlight moments. A high recall and low precision model would generate a very long video that would be unlikely to function as a 'highlight reel'. Additionally, accuracy is an inferior measurement because the positive class constitutes a small amount of the test data.

4.2.4 Results

4.2.4.1 Experiment One - Modalities Comparison

The first experiment compares the performance of each modality if it is treated as the sole model for highlight detection. Here the four modalities, RGB video frames (RGB), optical flow video frames (Flow), audio (Audio), and single-frame features (Frame), are compared

Table 4.2: Results for Experiment One. Minimum Average Precision, Maximum Average Precision, Mean Average Precision, and Medium Average Precision are shown for four single modality models alongside a random baseline. [1] [218].

Modality	Min. AP	Max. AP	$\bar{m}AP$	$\tilde{m}AP$
Random	0.025	0.394	0.118	0.104
RGB[1]	0.049	0.454	0.328	0.325
Flow	**0.064**	0.513	0.335	0.332
Audio	0.041	**0.574**	**0.366**	**0.343**
Frame	0.031	0.371	0.151	0.147

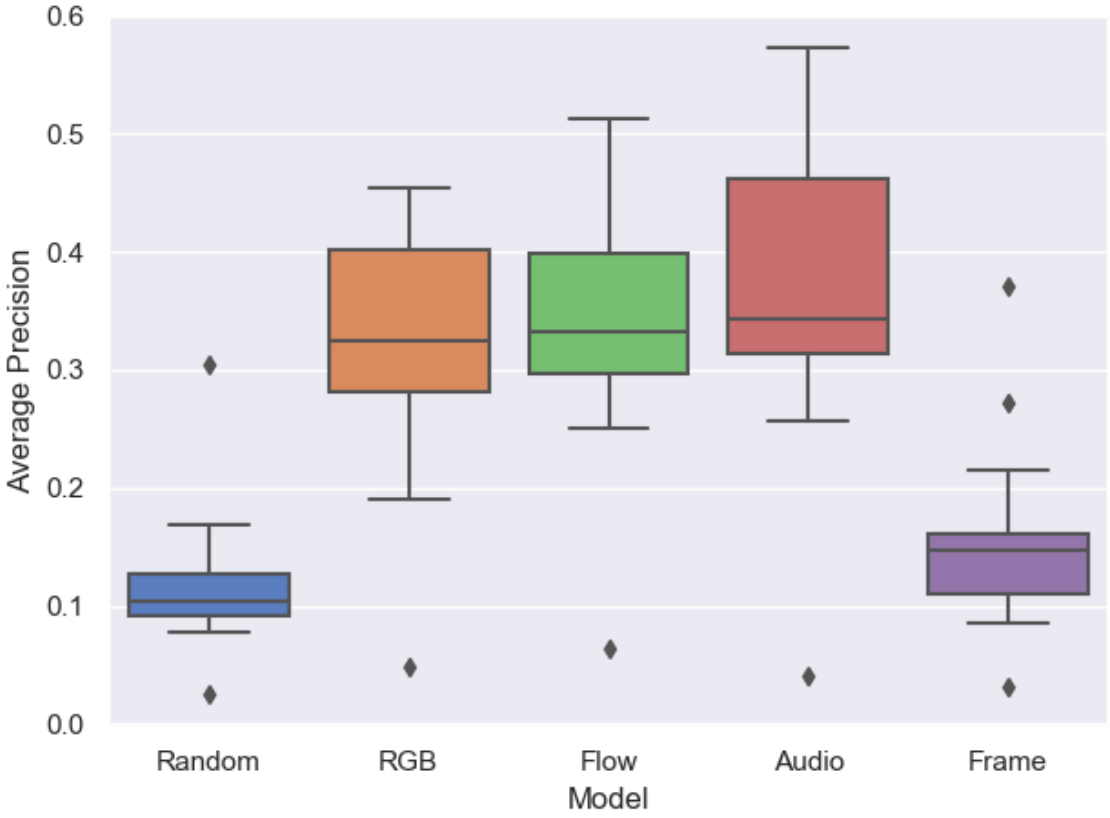


Figure 4.5: Box and Whisker plots for all models evaluated in Experiment One.

alongside a random baseline. The random baseline assigns a random value between $[0-1]$ for each sample for all videos in the test set. It is included to demonstrate the baseline challenge of this task. Note that because [218] uses just RGB frames to determine highlights, the RGB modality here represents the implementation of that approach.

Table 4.2 details the minimum AP (Min AP), maximum AP (Max AP) as well as both

averaged AP measures ($\bar{m}AP$ and $\tilde{m}AP$) across all videos in the test set for the five models tested in this experiment. To accompany this, Figure 4.5 presents box and whisker plots showing the distribution of results for each model across each video in the test set. Understandably, the random baseline performs very poorly. More interestingly is how poorly the Frame model performed. This is likely due to two effects. Firstly, the Frame model is the only one that does not consider temporality and thus has less information about what events occur in a segment. Additionally, the Frame ranking architecture, following [213] is much larger and more complex than other networks and thus may train poorly, e.g. overfitting. The other three models, RGB, Flow and Audio, all perform similarly, with the Audio network slightly outperforming the visual modalities in both $\bar{m}AP$ and $\tilde{m}AP$.

A Wilcoxon signed-rank test is performed to test the statistical significance of these results, which can be found in Table 4.3. Assuming that a p-value of < 0.05 indicates statistical significance, unsurprisingly, the random model is significantly different from all other models. Likewise, the single-frame features are also significantly different from the modalities that capture temporal dynamics in some way (RGB, Flow, and Audio). Perhaps unsurprisingly, given that the feature extraction models are trained similarly following a similar architecture, the RGB and Flow models are not significantly different. The RGB and Flow models are close in performance to the Audio model, indicating the importance of all three modalities. However, the Audio model predictions are significantly different from the visual models, perhaps due to audio cues indicating different types of highlight to visual cues. This difference may further indicate the benefits of a multimodal approach.

Additionally, the Vargha Delaney A measure [211] is also calculated and found in Table 4.3. This is a statistical test for effect size, which describes not just if a pair of models are significantly different, but how much one is statistically 'better' than the other. An A-measure value of $a > 0.5$ indicates that one model is better than another, with values $a > 0.56$ indicating a small effect, $a > 0.64$ indicating a medium effect, and $a > 0.71$ indicating a large effect. The A-measure for a pair of models is symmetric around 0.5 and sums to one. This means that additionally $a > 0.44$, $a > 0.36$, and $a > 0.29$ and indicate small, medium and large effect, albeit negative effect. There is a large effect in the performance improvements of the temporal models, i.e. RGB, Flow, Audio, over the single-frame model. There is also a medium effect size when comparing Audio to RGB and Flow, indicating a genuine performance benefit in using Audio data over Visual data if only one modality is used.

Table 4.3: Statistical testing results for Experiment One. The upper right triangle details Vargha Delaney A measure. Values less than 0.5 indicate column outperforms row and values more than 0.5 indicate row outperforms column. † denotes a small effect size ($a < 0.44 \vee a > 0.56$). ‡ denote a medium effect size ($a < 0.36 \vee a > 0.64$). $\diamond$ and bold denotes a large effect size ($a < 0.29 \vee a > 0.71$). The lower left triangle details p-values calculated via a Wilcoxon signed-rank test. Bold denotes significant values, i.e $p < 0.05$.

	Random	RGB	Flow	Audio	Frame
Random	-	**0.06**$^{\diamond}$	**0.06**$^{\diamond}$	**0.05**$^{\diamond}$	0.32‡
RGB	$\mathbf{2 \times 10^{-6}}$	-	0.48	0.39†	**0.91**$^{\diamond}$
Flow	$\mathbf{2 \times 10^{-6}}$	5×10^{-1}	-	0.42†	**0.91**$^{\diamond}$
Audio	$\mathbf{2 \times 10^{-6}}$	$\mathbf{3 \times 10^{-3}}$	$\mathbf{3 \times 10^{-2}}$	-	**0.92**$^{\diamond}$
Frame	$\mathbf{3 \times 10^{-3}}$	$\mathbf{2 \times 10^{-6}}$	$\mathbf{2 \times 10^{-6}}$	$\mathbf{2 \times 10^{-6}}$	-

Table 4.4: Results for Experiment Two. Minimum Average Precision, Maximum Average Precision, Mean Average Precision, and Medium Average Precision are shown for 6 multimodal models. [2] [213].

Modalities	Fusion	Min. AP	Max. AP	$\bar{m}AP$	$\tilde{m}AP$
RGB+Flow	Model	0.064	0.514	0.375	0.379
RGB+Audio	Model	**0.081**	**0.654**	0.444	**0.456**
RGB+Flow+Audio	Model	0.075	0.616	0.436	0.431
RGB+Audio+Frame[2]	Model	0.025	0.647	0.264	0.280
RGB+Flow	Feature	0.070	0.567	0.418	0.424
RGB+Flow+Audio	Hybrid	0.072	0.642	**0.462**	0.452

4.2.4.2 Experiment Two - Comparison of Multimodal Models

Experiment Two provides a comparison between several multimodal approaches. The system proposed by [213] is presented alongside the hybrid fusion model and a selection of other models indicative of the performance gains from multimodal systems. Three model fusion models are presented, RGB + Flow, RGB + Audio, RGB + Flow + Audio, and one feature fusion model, RGB + Flow.

Similarly to Experiment One, Table 4.4 shows the minimum and maximum AP and both average AP measures for all multimodal models. Likewise, Figure 4.6 contains the corresponding box and whisker plots. The RGB + Audio + Frame architecture presented in [213] is the worst performing model. The hybrid approach is the best performing system in terms of $\bar{m}AP$, although just using RGB + Audio model fusion performs surprisingly

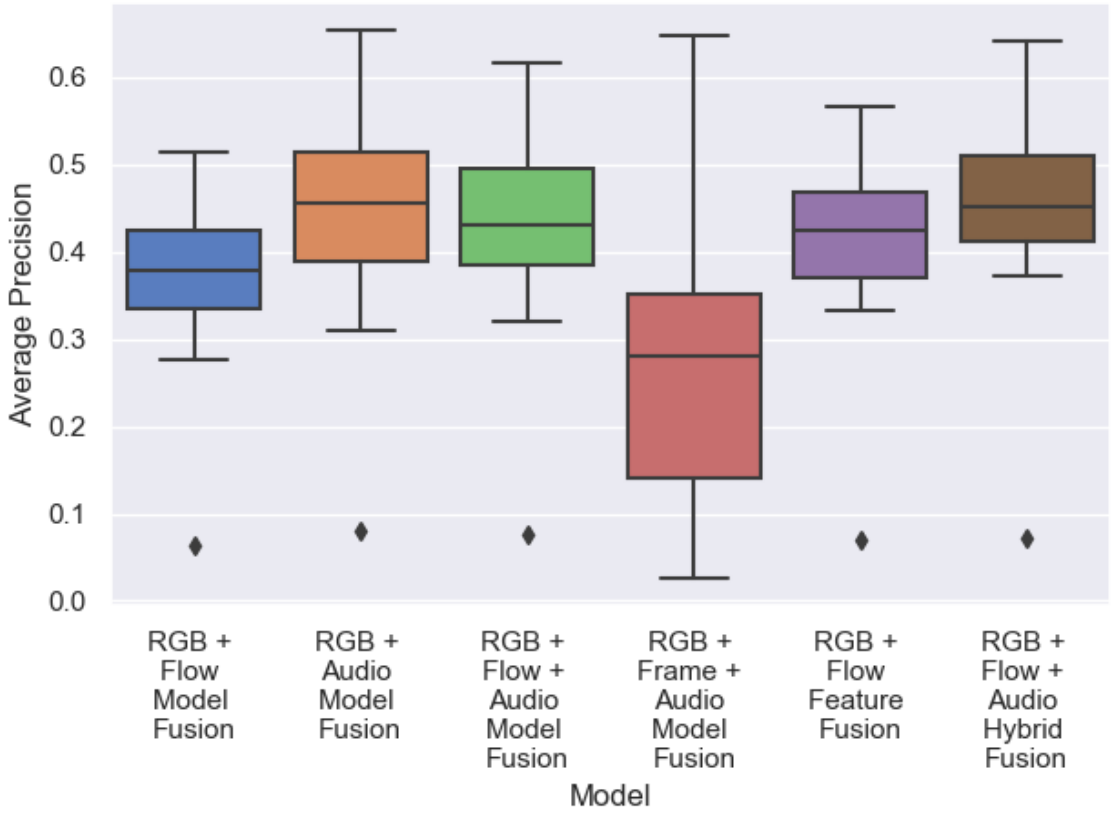


Figure 4.6: Box and Whisker plots for all models evaluated in Experiment Two.

well, perhaps due to the similarities in RGB and Flow data.

The hybrid Auto-Highlight model also produces significantly different predictions for all models except for the RGB + Audio model fusion model, Table 4.5. Again, this may be due to the similarities in RGB and Flow prediction. Table 4.5 also details the A-measure calculations for this experiment. A large effect size is seen when comparing the hybrid approach to the RGB + Flow model fusion model and a medium effect compared to the RGB + Flow feature fusion model. This large effect size indicates the performance benefits of using hybrid fusion across audio and visual modalities compared to visual data.

Interestingly no real effect is demonstrated when comparing RGB + Audio and RGB + Flow + Audio model fusion models, perhaps given that these both include the Audio domain, which appears to be the single strongest indicator of a highlight. The medium effect over the RGB+Flow+Audio model fusion model demonstrates the value of the hybrid fusion Auto-Highlight model over only using model fusion. The RGB + Frame + Audio model., i.e. [213], performs very poorly in this test, likely due to the Frame model being a

Table 4.5: Statistical testing results for Experiment Two. The upper right triangle details Vargha Delaney A measure. Values less than 0.5 indicate column outperforms row and values more than 0.5 indicate row outperforms column. † denotes a small effect size ($a < 0.44 \vee a > 0.56$). ‡ denote a medium effect size ($a < 0.36 \vee a > 0.64$). $\diamond$ and bold denotes a large effect size ($a < 0.29 \vee a > 0.71$). The lower left triangle details p-values calculated via a Wilcoxon signed-rank test. Bold denotes significant values, i.e $p < 0.05$. R - RGB, Fl - Flow, A - Audio, Fr - Frame.

		Model				Feature	Hybrid
		RFl	RA	RFlA	RFlFr	RFl	RFlA
Model	RFl	-	$0.31^{\ddagger}$	$0.32^{\ddagger}$	$\mathbf{0.75}^{\diamond}$	$0.35^{\ddagger}$	$\mathbf{0.24}^{\diamond}$
	RA	$\mathbf{1 \times 10^{-5}}$	-	0.53	$\mathbf{0.84}^{\diamond}$	$0.58^{\dagger}$	0.46
	RFlA	$\mathbf{2 \times 10^{-6}}$	3×10^{-1}	-	$\mathbf{0.84}^{\diamond}$	0.56	$0.40^{\dagger}$
	RFlFr	1×10^{-2}	$\mathbf{8 \times 10^{-5}}$	$\mathbf{3 \times 10^{-4}}$	-	$\mathbf{0.17}^{\diamond}$	$\mathbf{0.12}^{\diamond}$
Feature	RFl	$\mathbf{9 \times 10^{-6}}$	8×10^{-2}	5×10^{-2}	$\mathbf{6 \times 10^{-4}}$	-	$0.37^{\dagger}$
Hybrid	RFlA	$\mathbf{2 \times 10^{-6}}$	9×10^{-2}	$\mathbf{7 \times 10^{-4}}$	$\mathbf{1 \times 10^{-5}}$	$\mathbf{4 \times 10^{-6}}$	-

poor indicator of highlights, thus causing prediction noise.

4.2.4.3 Experiment Three - Time-Series Smoothing Convolutions

For this experiment, smoothing convolutions were applied to the hybrid fusion Auto-Highlight model. All smoothing kernels are experimented with and compared to the model's performance without smoothing. Once again, key results are presented in Table 4.6 and Figure 4.7. While the 'Power of 2' smoothing kernel works poorly, both 'Linear' and 'Gaussian' outperformed the non-smoothing model. Furthermore, the 'Gaussian' smoothing performed better than the 'Linear' model. Additionally, all smoothing kernels produce significantly different predictions except for the Power of 2 model compared to No Smoothing. Gaussian smoothing is the only technique that produced an improvement with a notable effect size compared to not using smoothing. The full Wilcoxon signed-rank test and Vargha Delaney A measure results are presented in Table 4.7.

4.2.4.4 Experiment Four - Comparison of Ranking Network vs Binary Classifier

Throughout the first three experiments, all models use a ranking network to make predictions. However, this fourth experiment compares ranking to binary classification using hybrid fusion and Gaussian smoothing. These are fundamentally different models, one is

Table 4.6: Results for Experiment Three. Minimum Average Precision, Maximum Average Precision, Mean Average Precision, and Medium Average Precision are shown for the hybrid model with various smoothing kernels applied.

Smoothing	Min. AP	Max. AP	$\bar{m}AP$	$\tilde{m}AP$
No Smoothing	0.072	0.642	0.462	0.452
Linear	**0.087**	0.719	0.487	0.484
Gaussian	0.079	**0.726**	**0.511**	**0.503**
Power of 2	0.073	0.642	0.442	0.438

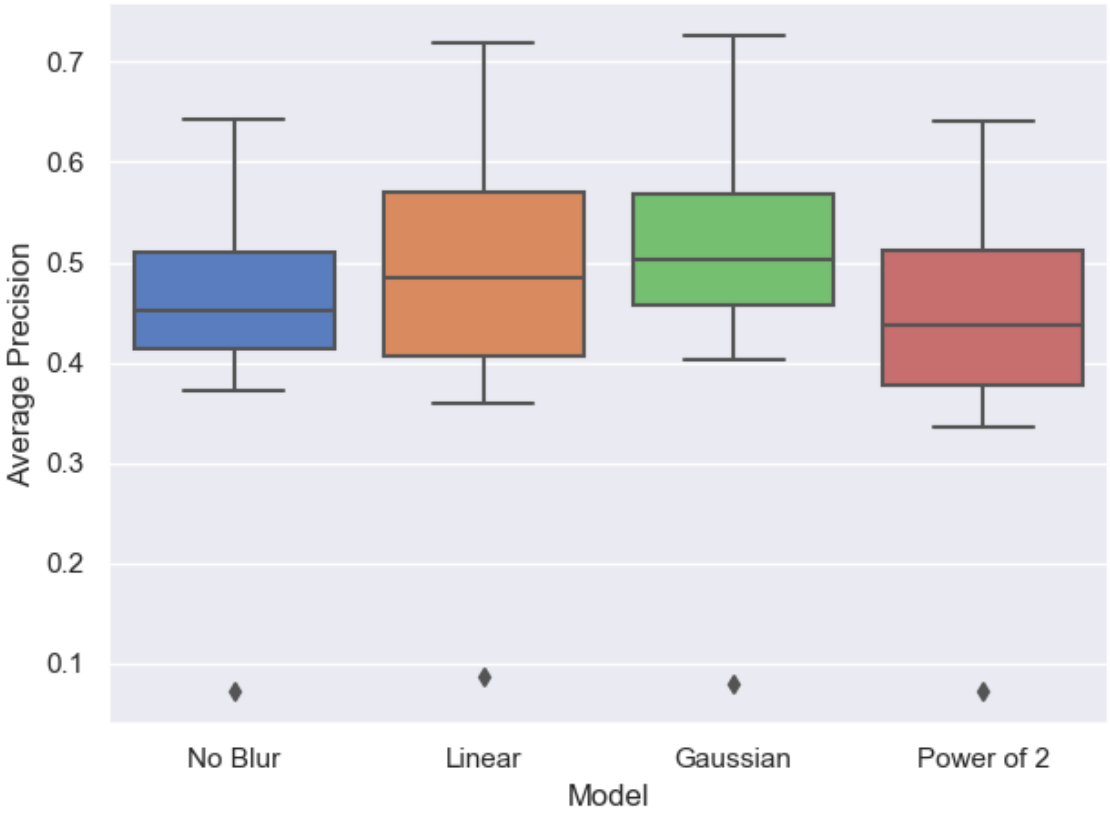


Figure 4.7: Box and Whisker plots for all models evaluated in Experiment Three.

Table 4.7: Statistical testing results for Experiment Three. The upper right triangle details Vargha Delaney A measure. Values less than 0.5 indicate column outperforms row and values more than 0.5 indicate row outperforms column. † denotes a small effect size ($a < 0.44 \vee a > 0.56$). ‡ denote a medium effect size ($a < 0.36 \vee a > 0.64$). $\diamond$ and bold denotes a large effect size ($a < 0.29 \vee a > 0.71$). The lower left triangle details p-values calculated via a Wilcoxon signed-rank test. Bold denotes significant values, i.e $p < 0.05$.

	No Smoothing	Linear	Gaussian	Power of 2
No Smoothing	-	0.44	$0.34^{\ddagger}$	$0.58^{\ddagger}$
Linear	$\mathbf{3 \times 10^{-3}}$	-	$0.42^{\ddagger}$	$0.63^{\ddagger}$
Gaussian	$\mathbf{2 \times 10^{-6}}$	$\mathbf{2 \times 10^{-3}}$	-	$\mathbf{0.71}^{\diamond}$
Power of 2	2×10^{-3}	$\mathbf{2 \times 10^{-6}}$	$\mathbf{2 \times 10^{-6}}$	-

Table 4.8: Results for Experiment Four. Minimum Average Precision, Maximum Average Precision, Mean Average Precision, and Medium Average Precision are shown for the hybrid model alongside a binary classifier.

Decision Model	Min. AP	Max. AP	$\bar{m}AP$	$\tilde{m}AP$
Classifier	0.046	0.719	0.499	**0.542**
Ranker	**0.079**	**0.726**	**0.511**	0.503

binary classification and the other outputs unbounded ranks. However, the post-processing applied to the ranking models transforms the unbounded ranks into binary classification. Thus the two models can be compared like for like. Table 4.8 details the minimum AP, maximum AP, and both average AP measures for the hybrid model and a binary classifier model and Figure 4.8 details the test set box plots. Only one set of scores for the binary classifier is presented in these results because the weighting to account for uncertain labels does not affect average precision calculations.

Initial experiments are also presented to set a detection threshold rather than relying solely on $\bar{m}AP$. Selecting a suitable threshold is vital for deploying Auto-Highlight in a real-world situation because it is necessary to convert the continuous, unbounded highlight

Table 4.9: Accuracy, Precision, Recall and F1 score for Experiment Four.

Decision Model	Accuracy	Precision	Recall	F1
Classifier	0.878	0.411	0.295	0.343
Weighted Classifier	**0.893**	**0.559**	0.074	0.131
Ranker	0.888	0.488	**0.559**	**0.521**

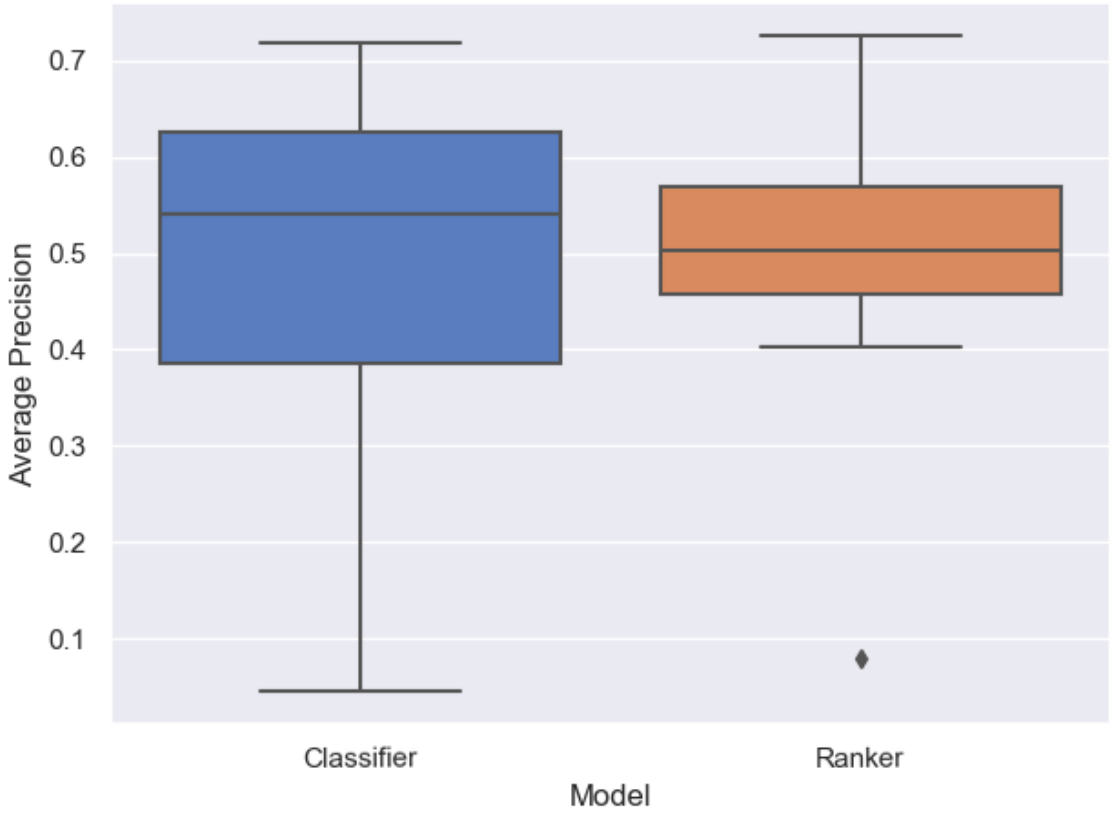


Figure 4.8: Box and Whisker plots for all models evaluated in Experiment Four.

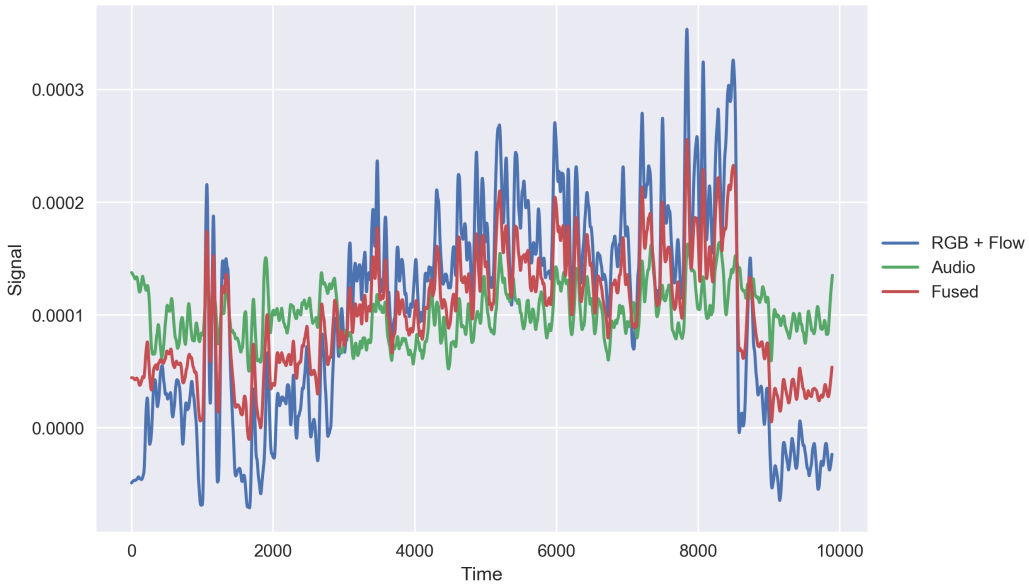
signal into binary segment predictions. This threshold selection is trivial for the classifier because the model is trained using an exact threshold value, in this case, 0.5. The weighting constant requires the threshold to be set at 0.5 to function correctly because it operates by reducing the decision boundary under this assumption. Selecting a threshold for the ranking model is less straightforward and must be determined heuristically. The ranking model has been trained under the assumption that one in eight pairings are invalid, i.e. one in eight samples (12.5%) in the broadcast dataset are highlights. Following this, a dynamic threshold is used, where the 12.5% of samples for a given video with the highest ranks are selected as highlights. This dynamic threshold technique is established in ranking-network literature, for example, [218]. Importantly, this means that models which require a decision boundary to be set heuristically, i.e. ranking models, may appear worse than they are because a poorly chosen threshold may artificially affect results. However, deploying a ranking model in a real-world application would require setting this threshold. Furthermore, the noisy-label mitigation weighting for the classifier model can be applied.

When comparing thresholded models output, calculating the $\bar{m}AP$ becomes impossible, but more traditional metrics such as accuracy, precision, recall and F1 Score are useful. These are shown in Table 4.9. Measuring accuracy is not particularly useful for this task because there is a considerable class imbalance in the data, and accuracy does not weigh one class as more important than the other. In this data, many more video segments are non-highlights, but the task is to detect moments in the minority highlight class. As such, accuracy may lead to models which appear successful but, in practice, rarely detect the exciting moments and instead overclassify the majority non-highlight class. Precision, recall and F1 are concerned only with the positive class, i.e. highlights, and therefore are more suitable. While precision measures are helpful for highlight detection evaluation, as evidenced by their prevalence in literature, recall is also valuable when examining thresholded results. Recall ensures that the model maximises interesting moments. Therefore the F1 Score, the harmonic mean of the precision and recall, is a reasonable metric for evaluating the ultimate performance of a model.

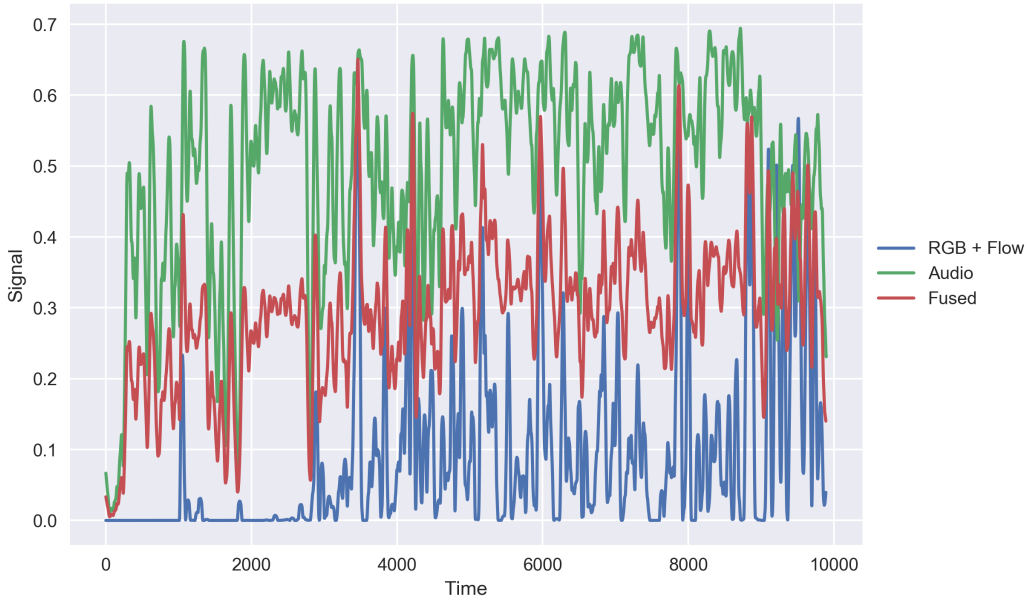
The highlight signal produced by these two approaches is very different. Figure 4.9 shows the output for an indicative game for both the weighted classifier and ranking model. The signals from each half of the model (i.e. RGB + Flow and Audio) alongside the fused weighted signal are shown. Furthermore, the unweighted signals appeared to be more correlated in the ranking model. To test this empirically, the Pearson's correlation coefficient for these signals is calculated for all videos in the test set. On average, the ranking network's decision models are more often correlated. The median correlation was 0.434 for the rank model and 0.291 for the classifier.

4.2.5 Discussion

These results show several important discoveries. Firstly, the proposed Auto-Highlight hybrid fusion model outperforms the current state-of-the-art ranking models. This performance improvement is evidenced both with and without the time-series smoothing. The performance benefit of the smoothing is also an important finding. It shows that modelling video data as part of a time series, rather than merely individual 2.56 second segments, significantly improves performance, even if that modelling is via post hoc heuristic smoothing. This is further evidence that context is essential for modelling the highlight moments in a broadcast.



(a) Ranker Model



(b) Classifier Model

Figure 4.9: Time-series highlight signal for the Ranker (a) and Classifier (b) models from Experiment Four on a single video in the test set.

One somewhat surprising finding is that audio appears to perform the best of all single modalities. This is an exciting result because it would be expected that the visual data, especially in-game, would be more consistent between training examples, and thus, the ranker would be able to better model which moments are interesting. However, sports commentators are trained to make the audience 'feel' when moments are exciting by delivering their commentary with high arousal. The high arousal levels in commentator speech appear to be a powerful indicator of highlight moments, mirroring [198]. That said, there is a broader range of results ($MaxAP - MinAP$) for the audio modality compared to both visual domains, which may indicate that while it performs well in average metrics, it is less consistent. Furthermore, the confidence value determined for the weighted classifier model is lower for the audio model. The findings indicate that audio is a potent indicator but that it also appears in some ways less reliable than visual data, perhaps due to the natural noise present in commentator speech, e.g. they may get animated when arguing about the effectiveness of a particular play, which would falsely appear to be an indicator of a highlight.

The method from [218] performs worse on this domain than in the original paper. However, the method from [213] performs better on this dataset. It is valuable to point out that [218] was evaluated on significantly different data, human action highlight recognition, but [213] was evaluated on similar data, esports coverage of a game similar to *League of Legends*. It is expected that this video game highlight setting would be complex and perhaps more challenging than related tasks, e.g. human action highlight recognition. Firstly because, as discussed above, context is vital and secondly because the visual range of gameplay is small, e.g. the difference between a highlight and non-highlight frame is likely smaller than in human actions. This might also explain the generally poor performance of the single-frame model.

There does appear to be one video in the dataset where all models perform poorly, evidenced by the low-performing outlier in most box plots and Figure 4.10, which shows the precision-recall curves used to calculate the average precision for test set videos using the full Auto-Highlight model, i.e. hybrid fusion with smoothing. Upon manual inspection, this video has much non-game content, especially highlights from previous matches and replays of celebrations with excited commentators providing the voiceover. This extra content is unusual among the test set and explains the poor performance. A human editor would easily understand that these moments are outside of the game being played, but these models

currently cannot. This could be solved using a second model trained only to identify the start and end of a match and then applying the Auto-Highlight model to that section of the broadcast.

Table 4.10 details the average execution time for various parts of the system. These times were calculated using the test set and the Nvidia 2080ti GPU-equipped machine detailed in Section 4.2.3. Feature extraction directly from video data is the most computationally expensive part of the process. This is understandable due to the high complexity of video data. By comparison, calculating the initial segment decisions is very fast, with all models taking around 0.004 seconds. Therefore, even the most complex hybrid fusion model is capable of 'real-time' prediction, i.e. processing faster than data is generated, utilising consumer-grade hardware. However, such prediction is currently infeasible due to the thresholding requirement and smoothing, which are applied to the entire broadcast. These processes are computationally fast, taking approximately 0.5 seconds per video, although this time varies slightly depending on video length. Therefore it would be possible to process segments in real-time and then apply the smoothing and thresholding techniques when the broadcast ends and thus generate and distribute the highlight video extremely quickly.

Calculating the memory usage of the models presented in this work is challenging due to the choice of implementation library for the ranking decision models. Keras has no native method for accurate profiling memory usage. However, approximate solutions exist, e.g. the one used for Table 4.1. Table 4.1 details the number of weights for each ranking model used in this work. From this, it is clear that the Frame only model is huge, ~ 42 million weights, compared to all other models. This is approximately 15 times larger than the next largest ranking model in terms of weights, requiring nearly five times more memory during training and over 15 times more during inference. Thus [213] has a much larger memory footprint than other approaches. Not only does the larger model size increase run-time and memory footprint, but it also likely contributed to the poor performance of the frame model due to over-fitting. Comparing weights also determines the relative complexity of models, if not the total memory cost.

To demonstrate the processing time differences between feature extraction and ranking, the RGB and Flow feature extraction models have a combined ~ 25 million weights, and each require over 1300MB of memory during inference, while the RGB + Flow feature fusion decision model requires just 2.7MB. Therefore it is evident that the decision part of the

Table 4.10: Average Execution Time per sample in seconds for both Feature Extraction (including preprocessing) and the Segment Decision Process.

Component	Average Time (seconds)
RGB Feature Extraction	0.215
Flow Feature Extraction	0.298
RGB + Flow Feature Extraction	0.495
Audio Feature Extraction	0.013
Segment Decision Process	0.004

network is extremely lightweight compared to feature extraction. Thus, while the hybrid fusion Auto-Highlight model has approximately six times the weights and nearly five times the memory requirements compared to [218] it still operates quickly and adds little total operating time to the system. However, this approach provides significant performance benefits, which appear worth the additional time and memory requirements. Additionally, all decision models other than the Frame model, owing to its size, are lightweight enough to train quickly, with each model taking just one to two hours to train on consumer hardware.

Finally, discussing the implications of Experiment Four, specifically Table 4.9 is worthwhile. While the performance between the two models is similar when calculating $\bar{m}AP$ and $\tilde{m}AP$, when a threshold is selected, the ranking model performs very well, despite the threshold being chosen heuristically, compared to the, in theory, 'perfectly' selected threshold for the binary classifiers, i.e. the threshold with which the model was trained. The ranking model has similar precision to the weighted classifier and is better than the unweighted classifier. However, it has far better recall and thus a better F1 score. This suggests that a ranking approach is more suitable for deployment than the corresponding classifier model, even with a heuristically decided threshold. Furthermore, the classification noisy label weighting has a large effect. It is the most precise model. However, it seriously affects the recall of the model. This results in an ineffectual model for this domain. The highlights are intended to entertain but also present an abridged narrative of the match. While precision is arguable the most important single metric, such a low recall score would, in practice, lead to a model which fails in this respect.

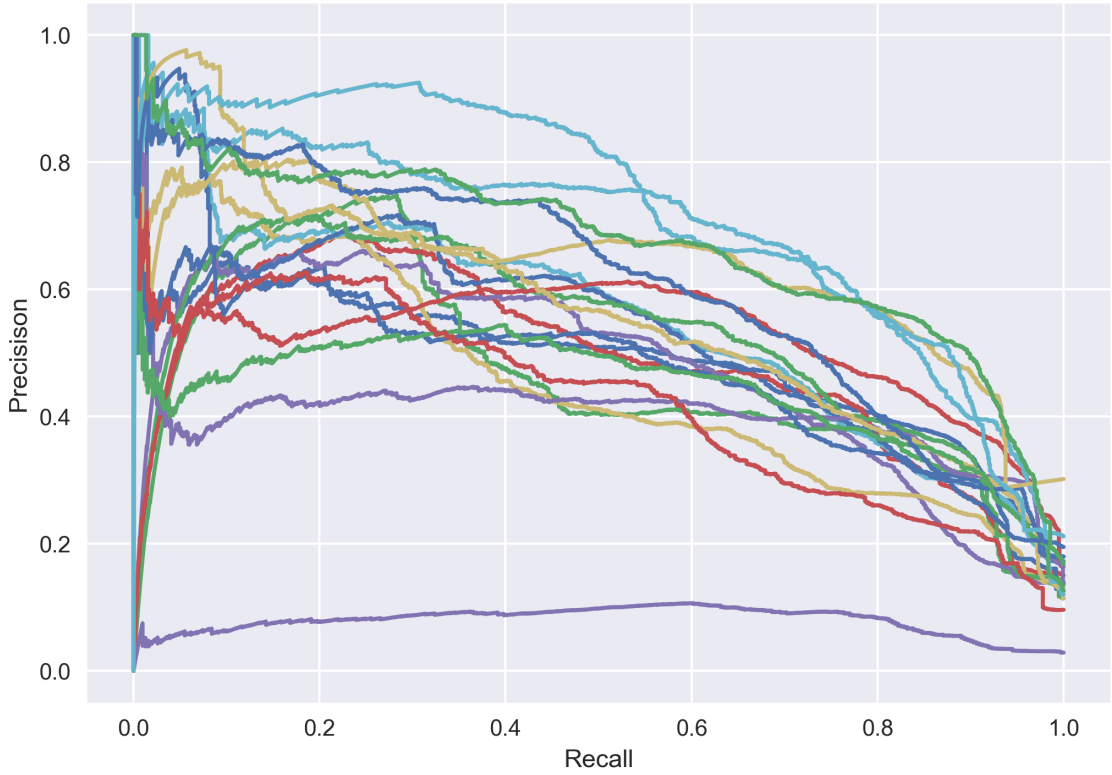


Figure 4.10: Precision Recall curves for all videos in the dataset with the hybrid fusion model with gaussian smoothing.

4.2.6 Conclusion

This study presents a multimodal hybrid fusion model for automatic highlight detection in esports broadcasts. It shows significant improvement over existing techniques, in some cases achieving a $\bar{m}AP$ nearly twice as high as existing state-of-the-art multimodal approaches in this domain. Furthermore, applying a smoothing kernel convolution to the output signal of the model is proposed, which further significantly improves results. Lastly, the assumption that ranking approaches outperform corresponding classification approaches is examined. While classification performance is comparable when averaging across all potential thresholds, when even a heuristically chosen threshold is selected, the ranking model outperforms the classifier model on critical metrics such as F1 Score. While this Auto-Highlight approach has some limitations, e.g. context is only modelled crudely, it provides a solid foundation to operationalise automatic highlight detection for game-driven broadcasts. The results are an improvement over other state-of-the-art models, and additionally, the model can be run quickly, just a few minutes per video, on consumer-grade hardware, which would allow es-

ports broadcasters to quickly edit and release highlight videos after a match has finished.

4.3 Game-Driven Livestream Highlight Detection Conclusions

This chapter has presented a study into highlight detection using game-driven broadcasts. This study is the first in this thesis which has, through positive-unlabelled data, been able to leverage massive datasets of video data. The ability to minimise the cost of data gathering, e.g. in terms of time and human effort, while still being able to train effective detection models is desirable for deployment in a real-world setting. Furthermore, the Auto-Highlight model presented works very well compared to state-of-the-art ranking-based highlight models, and smoothing provides significant benifits as a post-processing technique for context modelling.

The results of this study in some respects a breakthrough concerning livestream highlight detection. By utilising these extremely popular esports broadcasts with an established community of highlight editors, much higher quality highlight detection models are possible compared to Chapter 3. Thus the techniques presented in this chapter are much more likely to be useful within the esports broadcast industry as they are of sufficient quality to be used in a real-world setting and operate quickly enough to allow this. The drawback of the Auto-Highlight model and the positive-unlabelled data is that it is only suitable for game-driven esports broadcasts and not personality-driven broadcasts. This is due to the broadcast-highlight content cycle for esports games. Many personality-driven broadcasts are not edited into highlight videos and those which are often feature extra content, e.g. pop-up images and audio, not present in the source broadcast. However, the esports industry is enormous; thus, while personality streamers cannot benefit from this work, many stakeholders can.

Chapter 5

Highlight Detection from Text Chat Data

5.1 Introduction

So far, this thesis has focused on audio-visual modelling of highlight moments. Audio-visual data goes hand-in-hand because they represent constituent parts of video data. Therefore, it makes sense to develop systems which utilise a combination of these modalities. Furthermore, these modalities have been studied together in detail in wider literature. However, one of the potential benefits of studying highlight detection methods for livestreaming is that text chat data almost always accompanies audio-visual broadcast data. Livestream chat is valuable for this research because chat may be an indicator of events in the stream [86], as well as an interesting language domain to study. Therefore, this chapter investigates livestream chat and its suitability as a signal for highlight detection.

Livestream data is significantly different from typical natural language. These variations, discussed in detail in Section 5.2, make livestream language unique and cause many complicating factors. The way chat is generated is distinctive. For example, it is formatted differently from prosaic text, i.e. there are many authors, sometimes up to thousands. Further, chat is presented as a scrolling window of chat messages, as shown in Figure 5.2. Messages are displayed until enough new messages have been sent that the message leaves the chat window. The higher the volume of messages being sent, the less time a message is visible. Thus, often messages must be sent promptly so that chat participants can respond to in-stream events. Additionally, chat messages make liberal use of slang and often contain

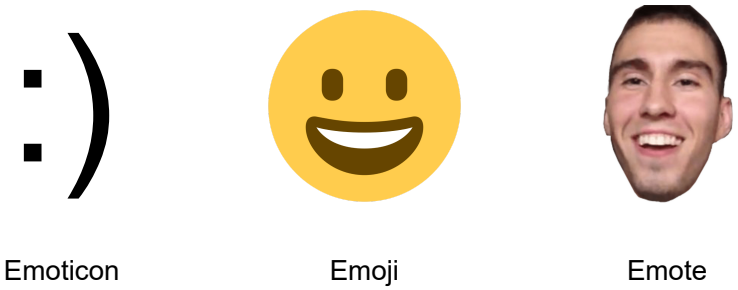


Figure 5.1: Example of the different types of pictographic tokens. Emoticons are created by combining standard characters. emoji are pictograms included in some modern character encodings. Emotes are platform specific pictograms.

Figure 5.2: Example screenshot of livestream chat from a *League of Legends* livestream. Author names have been removed for anonymity.

many misspellings.

This can be due to, for example, viewers speedily typing messages reacting to the stream. These misspellings can also be intentional, e.g. 'leet' speak and other spelling modifications [18]. Additionally, chats heavily use pictographic tokens, for example, emoticons, emoji, and 'emotes' [14], domain-specific emoji with rich, complicated meanings. These pictographic tokens are demonstrated in Figure 5.1. Therefore, carefully analysing livestream data and its properties is vital to developing effective livestream highlight detection models using chat data. Furthermore, a specialised neural network is needed because of the unique qualities of chat data.

This chapter focuses on two studies. The first, detailed in Section 5.2, performs a general exploration of livestream chat and details its idiosyncrasies compared to typical language.

The second, detailed in Section 5.3, details a novel architecture for state-of-the-art highlight detection, alongside comparisons to competing approaches. This approach utilises a custom stacked transformer architecture designed to model the unique features of Livestream chat data, i.e. the temporal nature of chat logs. Finally, Section 5.4 concludes the finding from these two studies.

5.2 Understanding Livestream Chat - The TwitchChat Dataset

There has been increasing research into livestream chat, for example, works discussed in Section 2.3. However, these works tend to be carried out in isolation, and at the time of research there are no publicly available datasets of Twitch chat data. A result of this is that there is no one study which can provide a foundation for livestream natural language processing research. We, therefore, present a large-scale dataset of chat text alongside an analysis of its content. This dataset has been open-sourced as a contribution to the research community[1]. Additionally, this section presents baseline investigations of the dataset, highlighting the uniqueness and challenges of livestream chat analysis. Skip-Gram Negative Sampling (SGNS) word vectors [128] and human evaluation of the learnt space are employed to achieve this. SGNS is known to produce curious vector spaces [130] but, even still, livestream chat vector spaces seem to have a particularly strange shape compared to SGNS models trained on typical English language text.

While livestream chat has some similarities with other non-traditional language data, e.g. social media data, it is strongly evinced in the literature that livestream chat has unique properties. Primarily, chat is strongly linked to streamed content [134,162], e.g., viewers react to events in the stream and is thus fundamental in understanding the context of livestreams. Secondly, due to the vast chat size and time constraints when writing messages, viewers need to respond quickly, resulting in unique properties. Notably, most tokens are entirely unknown to existing social media-focused lexicons, e.g. Vader [78]. Therefore, understanding this domain is both highly challenging and essential as the popularity of livestreaming as entertainment continues to rise.

[1] https://osf.io/39ev7/

Table 5.1: Games chosen for the TwitchChat dataset, alongside their genre and the game ID used in the Twitch API to retrieve chat data.

Game	Genre	Twitch Game ID
League of Legends	Multi-Player Online Battle Arena	21779
Fortnite	Battle Royale	33214
Grand Theft Auto V	Action-Adventure Game	32982
World of Warcraft	Massively Multi-Player Online Role-Playing Game	18122
Dota2	Multi-Player Online Role-Playing Game	29595
CounterStrike: Global Offensive	First-Person Shooter	32399
Hearthstone	Strategy Card Game	138585
Player Unknown's Battlegrounds	Multi-player Online Battle Arena	493057
Overwatch	First-Person Shooter	488552
Rainbow 6 Siege	First-Person Shooter	460630
Apex Legends	Battle Royale	511224
Fifa 19	Sports Game	506103
Minecraft	Sandbox Game	27471
Total War: Three Kingdoms	Grand Strategy Game	502377
Dead by Daylight	Multi-Player Survival Horror Game	491487
Call of Duty: Black Ops 4	First-Person Shooter	504462
Starcraft 2	Real-Time Strategy Game	490422
Magic: the Gathering	Strategy Card Game	2748
Diablo 3	Action Role-Playing Game	313558
Rocket League	Sports Game	30921

5.2.1 The Twitch Chat Dataset

5.2.1.1 Data Collection

The dataset was gathered from Twitch.tv between June and October of 2019 by selecting the most popular channel for a set of games, motivated by [92], and recording all messages being sent in that channel until it went offline, then repeating this process. Twenty different games were selected, representing the most popular games on the platform when data gathering started. These games are listed, alongside their genre, in Table 5.1. Channels were only considered if they were streaming in English, as that is the most popular language on the platform and the common language among the researchers involved in curating this dataset. That said, there is no requirement that chat is communicated in English, only that the streamer has selected English as the channel's language.

The above process resulted in a large dataset of over 60 million tokens from 1,951 documents. 'Token' is a generic term which refers to a single 'unit' of language. In the case of this dataset, a token refers to a single word, emoticon, emoji, or emote. Each 'document' represents text from a single stream session. Data was gathered from 666 different streamers. Summary statistics regarding the distribution of document features, e.g. the distribution of

Table 5.2: Summary statistics describing the distribution of several dataset features. 'Viewers per Stream' was recorded at the start of data gathering.

Feature	Median	Interquartile Range	Min	Max
Stream Documents per Streamer	1	2	1	21
Viewers per Stream	2,211	5,586	0	165,371
Messages per Stream Document	5,196	124,340	2	483,230
Message Length	2	4	1	266

messages per stream document, can be found in Table 5.2. These statistics are not expected to be normally distributed; thus, the Interquartile Range is reported rather than the Standard Deviation. Visualisation of the relationship between these features is omitted here for brevity. However, these figures can be found in Appendix A.

For privacy purposes, all direct references to usernames, both streamers and viewers, in the dataset have been replaced with a random string. This was achieved using a 'salt and hash' anonymisation function where the username was concatenated with a private salt, and the resultant string was then hashed using the SHA1 hashing algorithm. This was done so that despite the anonymisation, it is possible to encode new data using the same process should the dataset be expanded.

5.2.1.2 Data Cleaning Process

The publicly available version of the TwitchChat dataset is published with minimal cleaning and processing. The initial chat logs have been anonymised and formatted into CSV files, alongside a separate metadata file containing features about the stream itself, e.g. the number of viewers. This allows researchers with different goals to access the data in the rawest form and process it in a way that suits their work. During the course of the research presented in this chapter, several additional data processing steps are undertaken to make the dataset suitable for SGNS training.

The data cleaning process's first stage is removing unwanted tokens and messages. For instance, 'stop words', commonly used words, e.g. 'the', 'and', and 'is', are removed because they provide very little information. This is especially important for the SGNS models because, as discussed in detail below, they are trained based on tokens co-located in the text. Retaining stop words would harm training. Additionally, any private messages received are removed as they are not sent to contribute to the general chat but are generally targeted

Table 5.3: Dataset size (in terms of number of tokens and number of unique tokens) before and after data cleaning stages.

Metric	Raw Dataset	After Stage 1	After Stages 2 & 3
Tokens	61,040,692	47,783,915	38,751,630
Unique Tokens	1,658,055	1,405,084	10,011

messages. Further, these often appear when data scraping begins, e.g. welcoming the scraper to the stream. In a similar vein, 'bot' messages are removed. These are messages sent to the general chat by an automatic bot rather than a human and often are used to help moderate the chat and provide broadcast information, e.g. the details of a giveaway if the streamer is running one. Finally, generally malformed and unusual tokens are removed because curating a massive dataset inevitably results in some errant data.

Next, lemmatisation is performed. This takes similar words and processes them to represent them with the same token in the data. For example 'plays', 'playing', and 'played' are lemmatised to their stem work 'play'. This is a typical part of most text preprocessing. This generic lemmatisation is performed using tools provided by the NLTK Wordnet engine [115, 129]. Due to its unique properties, livestream chat data requires several additional lemmatisation steps. Firstly, popular emotes often have many variants. These can be streamer-specific or celebrating some other aspect, e.g. pride. To highlight the breadth of emote variants, a selection of 'Kappas' are shown in Figure 5.3. Emote lemmatisation is applied to 'Kappa', 'lul', 'pog', 'pepe', 'jebait', and 'lmfao'. Secondly, livestream chat contains a large amount of word expansion, e.g. 'goooood' is used rather than 'good', and other common slang spellings, e.g. 'would've' is spelt 'woulda'. Regex pattern matching is used to recognise and lemmatise 38 expansion and slang misspellings. Cleaning the data in this manner resulted in a dataset of approx 47 million tokens.

Stage two of the cleaning process is to select a subset of the cleaned and lemmatised tokens to train the models. Only a subset of unique tokens are used for computational feasibility. This is because a data representation for each token must be stored in the model. Thus the more unique words modelled, the larger the model weight requirements. As Table 5.3 shows, there are still ~ 1.4 million unique tokens after stage one of the cleaning process. Inspired by [127], the 'vocabulary', i.e. the set of all unique tokens, is limited to the $\sim 10,000$ most popular tokens. These tokens are selected by assigning each token a 'document frequency' score, representing the number of documents in which the token appears. All tokens with a

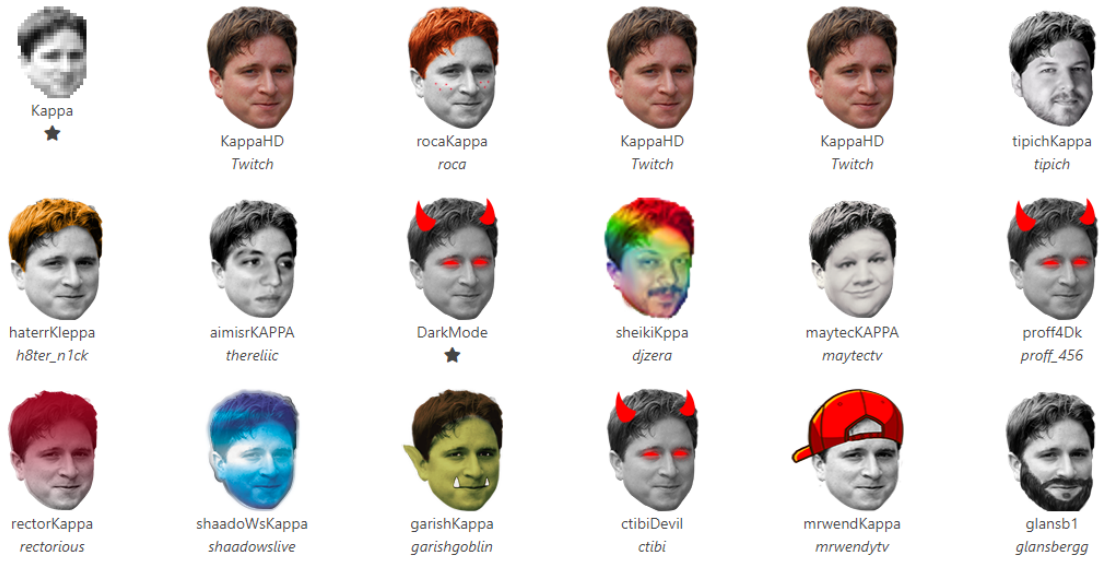


Figure 5.3: Variations of 'Kappa' (top left) collected using the 'View Similar Emotes' feature on https://twitchemotes.com (accessed 08-10-2019).

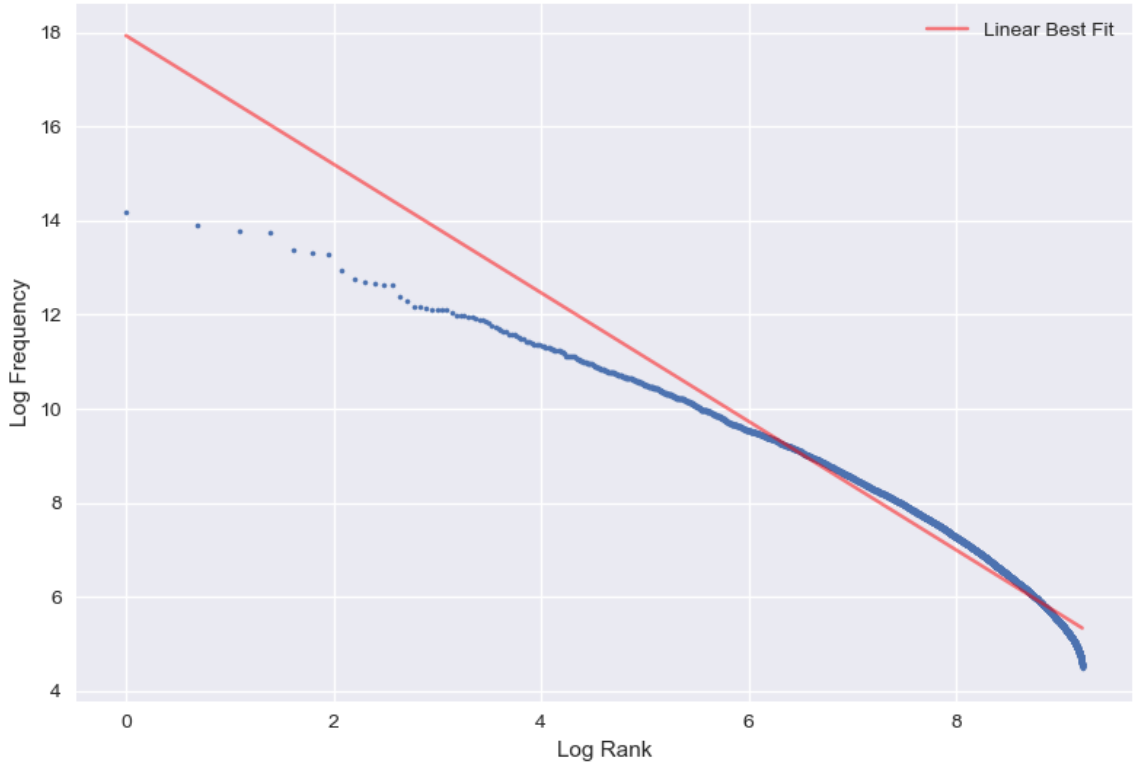


Figure 5.4: Log Frequency/Log Rank for the cleaned dataset. Zipfian distributions are linear when log transformed.

document frequency score greater than or equal to the $10,000^{th}$ most frequent token were selected, resulting in a $10,011$ token vocabulary. Tokens were selected based on document frequency rather than total frequency because certain tokens are heavily used in the streams of certain channels but not elsewhere. This dataset is intended to be general-purpose and describe the overall landscape of livestream chat across many streamers and games. Selecting tokens based on raw frequency would have resulted in certain tokens, mostly emotes, which are streamer specific and thus not used across Twitch in general.

Once the vocabulary has been determined, stage three is to remove all tokens that were not selected. Once completed, the total tokens in the dataset is ~ 39 million, around 63% of the initial dataset, Table 5.3. This is evidence that it is possible to reduce the number of unique tokens and thus model size while retaining a large amount of data. Furthermore, more tokens were removed during phase one of the data cleaning process, where fairly typical processes in natural language work were carried out than were removed to ensure the model was of a computationally feasible size. This cleaned dataset of popular words is used for the remainder of this section, and references to the dataset refer to this cleaned data.

5.2.1.3 Exploring the Distribution of Token Popularity

Zipf's law [231] is an empirical law which describes the relationship between the frequency of a token in data and its 'rank popularity', i.e. the popularity of a word where the most popular word is assigned the rank 1 and so on. This law suggests that the frequency of a token is roughly inversely proportional to its 'rank popularity'. Most languages roughly follow a Zipfian distribution, and thus the second most popular token occurs about half as often as the most popular, the third most popular, about a third as often and so on. Such is the power of this law that it is very surprising to find an example of a language which does not follow it. To test for a 'Zipfian' distribution, the log-rank for each token is plotted against its log frequency. Distributions which follow Zipf's law have a roughly linear relationship between tokens.

Figure 5.4 shows the log-rank log-frequency graph of all $\sim 10,000$ tokens in the TwitchChat dataset. Surprisingly, both the most and least popular tokens in the dataset do not follow Zipf's law. Instead, it appears that the distribution initially becomes Zipfian after the ~ 26 most popular tokens. These 26 most popular tokens are used more often than expected. These 26 tokens include many Twitch specific words and those, e.g. clap, which are likely

to be prevalent in 'cheering' style communications. They are; 'lul*', 'pog*', 'haha*', 'pepe*', 'no*', 'lulw', 'so*', 'go*', 'hehe*', 'lol*', 'kekw', 'way*', 'clap', 'what*', 'kappa*', 'wa', 'chat', 'why*', 'like', 'get', 'trihard', '¡3', 'game', 'na', ':)*', and 'ewww*'. In the above list '*' denotes the stem token for tokens which underwent lemmatisation.

Additionally, the dataset stops becoming Zipfian starting with tokens with rank $\sim 3,000$. After this point, tokens are used less frequently than expected. This interesting finding shows that Twitch chat is very different from ordinary language. In particular, it seems that there are certain tokens which are more or less popular than expected. The overuse of popular tokens is potentially due to the 'cheering' effect observed in certain large streams, where the channel is inundated with a small subset of tokens used to cheer on the streamer or show support. Additionally, it seems that the use of language, in general, is fairly limited on Twitch.tv; only about 3,000 tokens are used as often as expected. This may be due to the fast-paced and time-limited nature of livestreaming - fewer tokens are appropriate for use in the chat context than in traditional natural language.

5.2.2 Case Study: Word Vector Models

To demonstrate the TwitchChat dataset's unique features, an analysis using four SGNS word vector models is carried out. These word vector models are particularly useful for showing the peculiarities of the learnt vector spaces. Additionally, human evaluation of the vector spaces and cluster-based analysis is performed.

5.2.2.1 Skip-gram Negative Sampling Word Vectors

Skip-gram Negative Sampling (SGNS) word vectorisation is a method for taking a set of tokens and finding a semantically meaningful vector space for those tokens. In doing so, each token is assigned a vector, and, ideally, semantically similar tokens have vectors which are similarly located in vector space. Traditionally, SGNS models, much like other forms of word vector models such as Continuous Bag of Words (CBOW) or Skip-Grams [126], use the spatial distance, i.e. the number of tokens in between two selected tokens, as cues for semantic similarity. This spatial distance is key to training an SGNS model and is why including stopwords negatively affects models; many tokens appear spatially close to a stopword token because they are highly prevalent in the text.

The key to training an SGNS model is how samples are selected from the text corpus.

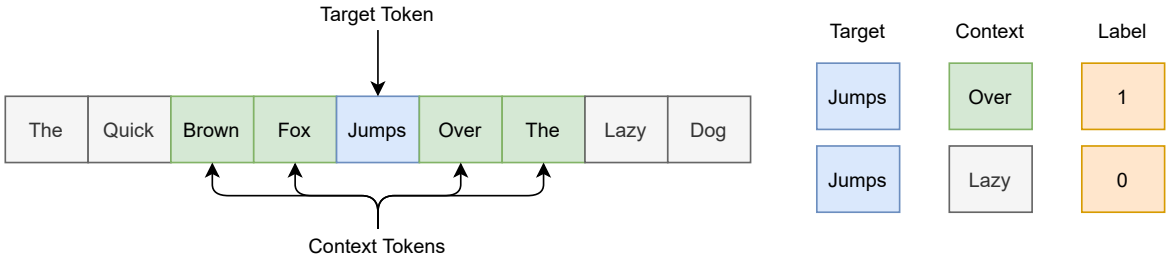


Figure 5.5: Example sampling process for SGNS model. Context window $l = 2$. The positive training sample is sampled from the context window. The negative training sample is sampled from the entire vocabulary.

Unlike CBOW or Skip-Grams, where the architecture is trained to output a token prediction given co-located tokens, SGNS models are trained similarly to binary classification. SGNS input data contains a mixture of 'positive' samples, i.e. co-located words, and 'negative' samples, i.e. non-co-located words. First, a context 'window' hyper-parameter, l, must be selected to achieve this. Given a selected 'target' token, randomly selected from the corpus, all tokens within l tokens are considered potential context tokens. From these, a 'context' token is randomly selected. This token pair is assigned a label of 1 for training. Next, a 'negative' token is selected. This is carried out by randomly sampling a token from the entire vocabulary. This token pair is assigned the label 0 and acts as noise to aid learning semantic vectors [128]. This sampling process is visually displayed in Figure 5.5. There are many ways this negative context word can be sampled from the vocabulary, e.g. by randomly selecting a word from the document, uniformly sampling the vocabulary, or, as in this case, by weighted vocabulary sampling. To carry out this weighted sampling, first, a sampling table is created, where each token value is determined using $v = \min(1, \sqrt{(\frac{f}{s} + 1)} \cdot \frac{f}{s})$, where f is the token's normalised frequency and s is a sampling factor hyper-parameter. For this section $s = 1e^{-5}$. Tokens are then selected as the negative sample based on this sampling table. This results in selection probabilities which are inversely proportional to the popularity of a token. Intuitively, this is because high-frequency tokens are more likely to be selected as 'target' tokens and thus need to be selected less frequently as context tokens to allow the model to learn all tokens in the vocabulary [128] evenly.

Once these positive and negative training samples have been generated, the model, consisting of only a few layers, can be trained. Firstly, the corresponding embedding, i.e. vector, is found for both input tokens. This is done by assigning each token with a unique numeric id, often in the range $[1, vocabularysize]$. This id is then used to index into the embedding

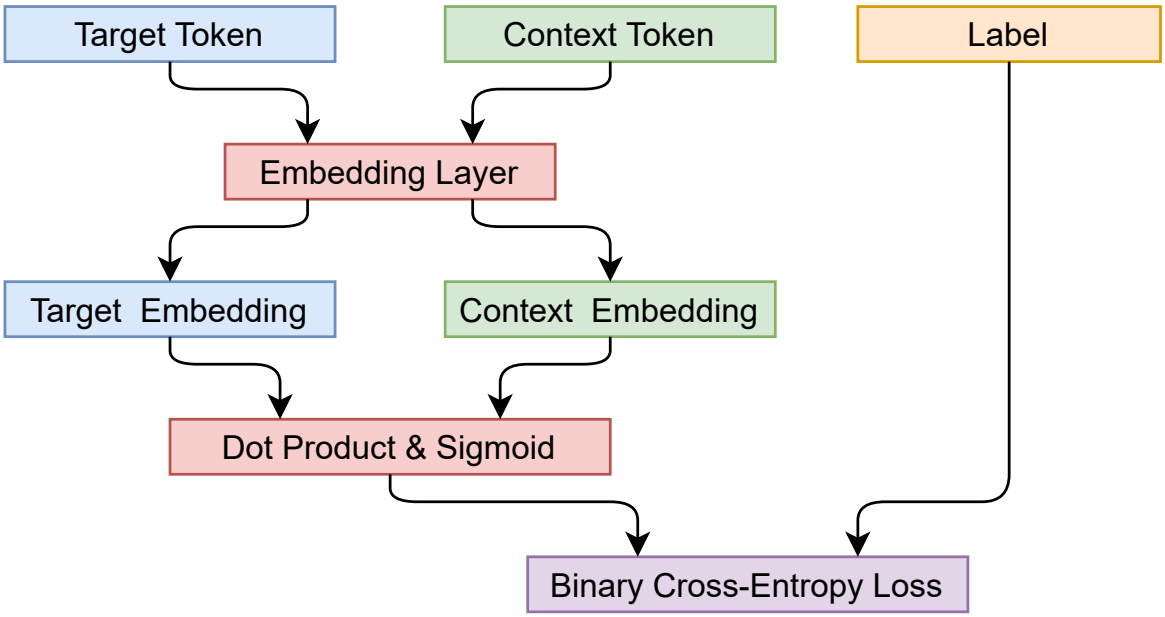


Figure 5.6: The Skip-Gram Negative Sampling (SGNS) training architecture. Once trained the embedding layer can be queried to retrieve token vectors.

layer, which has a weight matrix of size $vocabularysize \cdot embeddingsize$ where the embedding size refers to the length of each vector and thus influences the number of dimensions in the vector space. Once the vectors have been retrieved, the dot product is computed, followed by Sigmoid activation. This results in an output from the network between $[0, 1]$. Output values closer to 1 represent vectors closer in the vector space. This output value, alongside the training label, is then used to calculate Binary Cross-Entropy Loss. This way, token pairs commonly co-located in the corpus are co-located in the vector space. This architecture is visually displayed in Figure 5.6.

5.2.2.2 Dynamic and Temporal SGNS

Developing a livestream vector model requires special attention due to several observations. Firstly, livestream chat is naturally temporal. Typical English language used to train vector models, e.g. prosaic text, is written without time factors, e.g. there is no sense that two sentences which follow each other in prosaic text have a particular time distance between them. Contrast this with livestream chat messages where two messages sent one after another may be sent at similar times, e.g. reacting to a stream event, or sent minutes apart, e.g. during a quiet period in the chat. This fluctuation is often triggered by viewers reacting to stream events, so it seems probable that the temporal distance between messages may

indicate token-relatedness. Intuitively, if two messages are sent at similar times in reaction to an in-game event, their content is likely semantically similar. However, if two messages are sent a long time apart, even if they are co-located in the corpus, they are less likely to be related. Furthermore, this inherent time constraint results in messages that often need to be written promptly to engage with fellow viewers during particular moments. This can have the effect that livestream chat messages often have many repeated tokens and are not formed of complete sentences but instead focus on 'quick fire' reactions. This, especially the repeating tokens issue, suggests that using a fixed window size may not be appropriate for livestream chat. Therefore, this section presents two modifications to the SGNS model, designed to account for the peculiarities of livestream chat compared to typical prosaic text. Firstly, a proposed model dynamically expands the context window to include all tokens within a message that are not the target token. This allows the model to overcome the issues with repeated tokens while neatly handling the challenge of messages, which are often very different lengths, and the concerns over blanketly allowing tokens to be considered part of the context if they are inside the context window but sent by a different viewer.

Secondly, rather than sampling negative samples from the entire vocabulary, using the frequency table method described above, negative samples are sampled from messages sent during the stream. Additionally, if they are sufficiently close temporally, some samples are not assigned a 0 label but are instead assigned some label $0 < l < 1$. In theory, this allows the sampling procedure to have a sampling distribution which approximates the unigram distribution whilst incorporating temporal distances between messages into the model. Furthermore, this allows the models to sample negative training data from real-world examples. Given that the TwitchChat corpus is gathered from many streamers, each of whom may have a particular community who favours particular words, sampling in this manner is expected to create improved vector spaces. This is achieved by reformulating the initial binary classification task into a regression task and then allowing 'negative' training labels to be in the range $[0, 1]$, derived from a transformation of the temporal distance between messages containing the paired tokens.

This transformation is achieved using a Gaussian Radial Basis Function (RBF), $\mathrm{f(x)} = ae^{-\frac{(x-b)^2}{2c^2}}$, where x is the temporal distance between messages in seconds. e, a, b and c are all constants. e is Euler's number. a controls the maximum output value for the function, in this case $a = 1$. b controls the x input value, which results in the peak output. For

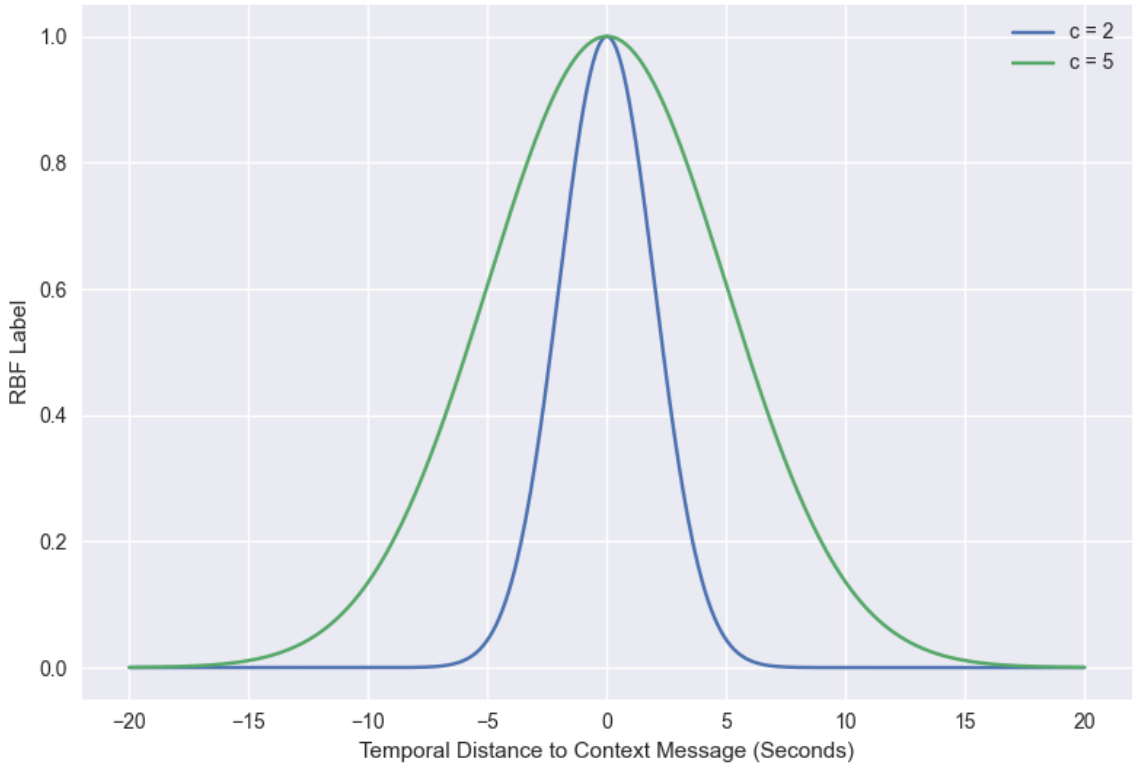


Figure 5.7: Visualisation of RBF transformations for temporal distances between messages.

the purpose of this study, $b = 0$. Therefore the maximum value, 1, occurs when there is 0 temporal distance between the messages. Finally, c is the standard deviation and determines the slope of the function. Two different values for c, $c = 2$, and $c = 5$ are experimented with because it is not initially clear what the most appropriate hyper-parameter is. An RBF is used because it is hypothesised that the curve is similar to the ground-truth value. A visualisation of this function is shown in Figure 5.7.

5.2.3 Experiment

In total, four word-vector models are employed. Firstly, a traditional SGNS algorithm. Secondly, a variant using dynamic windowing with l = message length (DW-SGNS). Both of these treat training as a binary classification problem, and so use Binary Cross-Entropy loss. The key difference is that the DW-SGNS model should, in theory, better sample training examples through the dynamic-windowing and sampling from the same document. Finally, two temporal model SGNS variants, 2T-SGNS ($c = 2$) and 5T-SGNS ($c = 5$) are trained. These models frame the negative sampling challenge as a regression task and thus use Mean

Squared Error loss.

For all models, inspired by [127], an embedding dimension of 300 is used. Each model is trained for a total of $5,000$ epochs, where each epoch consists of 500 mini-batches, and each minibatch contains $16,384$ training samples drawn randomly from the dataset. All models are implemented using Tensorflow's Keras API [1, 36]. Two evaluation techniques are applied. Firstly, a relatedness test, detailed below, is performed, which uses human evaluation of semantic embeddings. Secondly, the vector spaces themselves are explored through a range of techniques.

5.2.3.1 Relatedness Test Details

Evaluating the vector spaces is a difficult task because livestreaming is a unique domain and, as such, traditional techniques are not currently suitable. For example, evaluating vector spaces against a predetermined test set is standard. However, such an approach is inappropriate for livestream data because most tokens popular in the livestream context are unique to livestreams and thus not represented in typical language test sets. For comparison, only 13% of the tokens in the Twitch-Chat vocabulary exist in the Vader sentiment engine [78]. Furthermore, livestream-specific testing sets do not currently exist and curating one constitutes a large body of work and is outside this thesis's scope. Therefore, the evaluation techniques employed cannot require prior knowledge about the meaning of tokens, hence the advantages of a relatedness test over other evaluation methodologies.

The relatedness test employed to evaluate these vector spaces is inspired by [176] and utilises crowd-sourced testing. First, the 100 most popular tokens from the vocabulary are selected, referred to as 'target' tokens. Next, each model is queried, and the 1^{st}, 5^{th} and 50^{th} nearest 'neighbours' for each target token are retrieved. Human participants were shown 20 target tokens alongside their neighbours from all models and were asked to select the neighbour they considered the most related to the target. If two competing models share a neighbour, the neighbour is only presented once. The relatedness test was constructed as an online survey using the Qualtrics platform. Participants were gathered through online advertising on social media sites, such as reddit.com and twitter.com, which deliberately targeted participants who have experience with Twitch.tv due to the unique nature of the language used on the platform.

5.2.4 Results

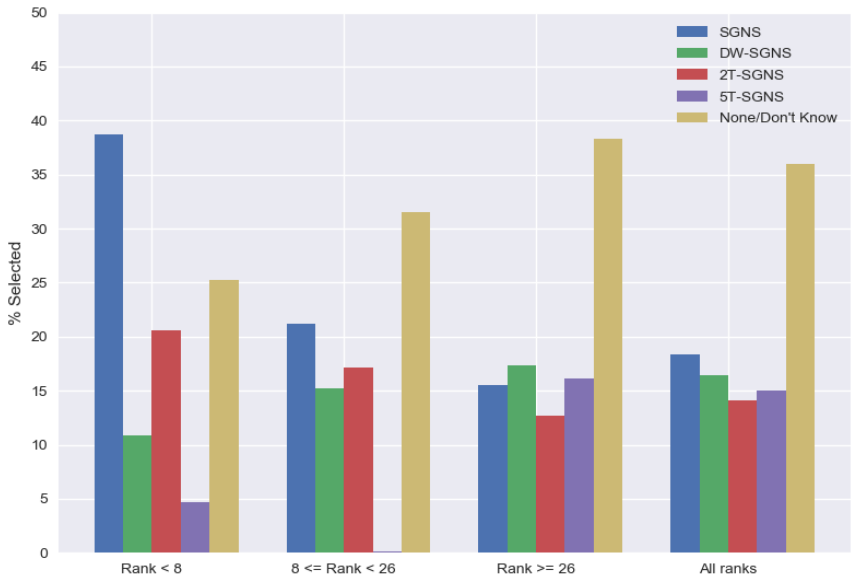
5.2.4.1 Relatedness Test

After cleaning invalid responses, e.g. where respondents who either did not respond or only answered 'None/Don't Know' were removed, there were a total of 154 respondents. Respondents were not forced to answer every question presented, and there was some randomness in the way questions were selected. Therefore there is variance in the number of responses to each question. Each question was answered by a median of 24 respondents, with each target token receiving a minimum of 14 responses and no more than 35 responses.

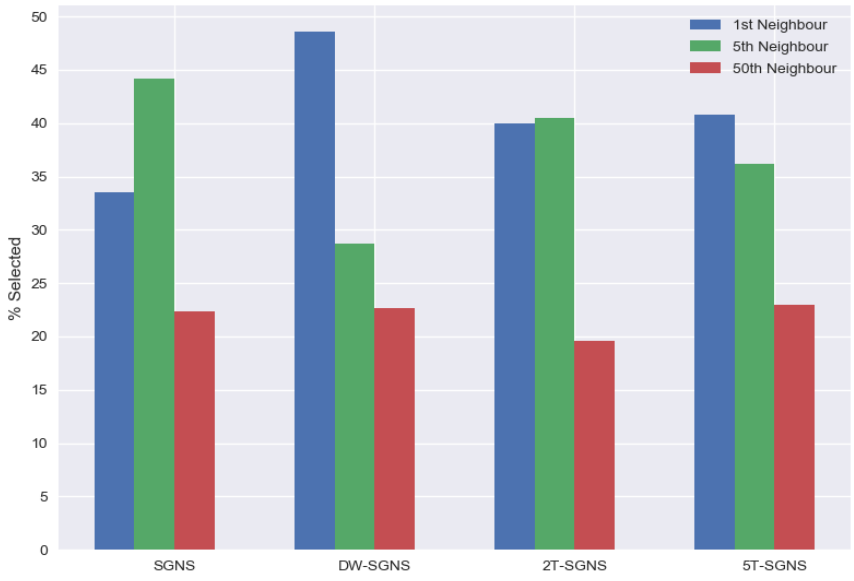
To analyse this study's results, each model's performance is compared. Performance is determined by the percentage of respondents who, for a target token, selected one of the neighbours generated by that model. Because two different models can suggest the same neighbour, summing the percentages across all models may sum to more than 100%. Figure 5.8 (a) shows the 'percentage selected' performance in the relatedness for all models across 3 token rank strata; those with token ranks 1 to 7, i.e. the 7 most popular tokens, those with ranks 8 to 26, and those with rank greater than 26. This stratification is motivated by Figure 5.4 because the Log Frequency - Log Rank graph gradient shifts after the token with rank 7 and rank 26. Figure 5.8 (b) shows the percentage amount that each neighbour, 1^{st}, 5^{th} and 50^{th}, was selected when a model was chosen by the participants.

As with the results in Section 3.3, no model outperforms the others in all situations. SGNS is the most selected model in general and is overwhelmingly the best model with very popular tokens. However, the performance gap is closed for lower-ranked tokens, with DW-SGNS and 5T-SGNS models being selected more often than SGNS for tokens with rank ≥ 26. That said, in general, the models performed poorly. Respondents selected the 'None/Don't Know' answer more than 35% of the time. SGNS for high-ranked tokens is the only instance where a single model was selected more than the 'None/Don't Know' option. In hindsight, it was not ideal to couple 'None', i.e. all models failed to produce a token related to the target, and 'Don't Know', i.e. the respondent cannot answer the question, e.g. because they are not familiar with the target word. This coupling makes it challenging to evaluate if the models performed poorly or if the meaning of tokens in the livestream context is obscure, even to those who frequent such platforms.

Because it is unclear if poor performance on lower-ranked tokens is due to poor quality



(a)

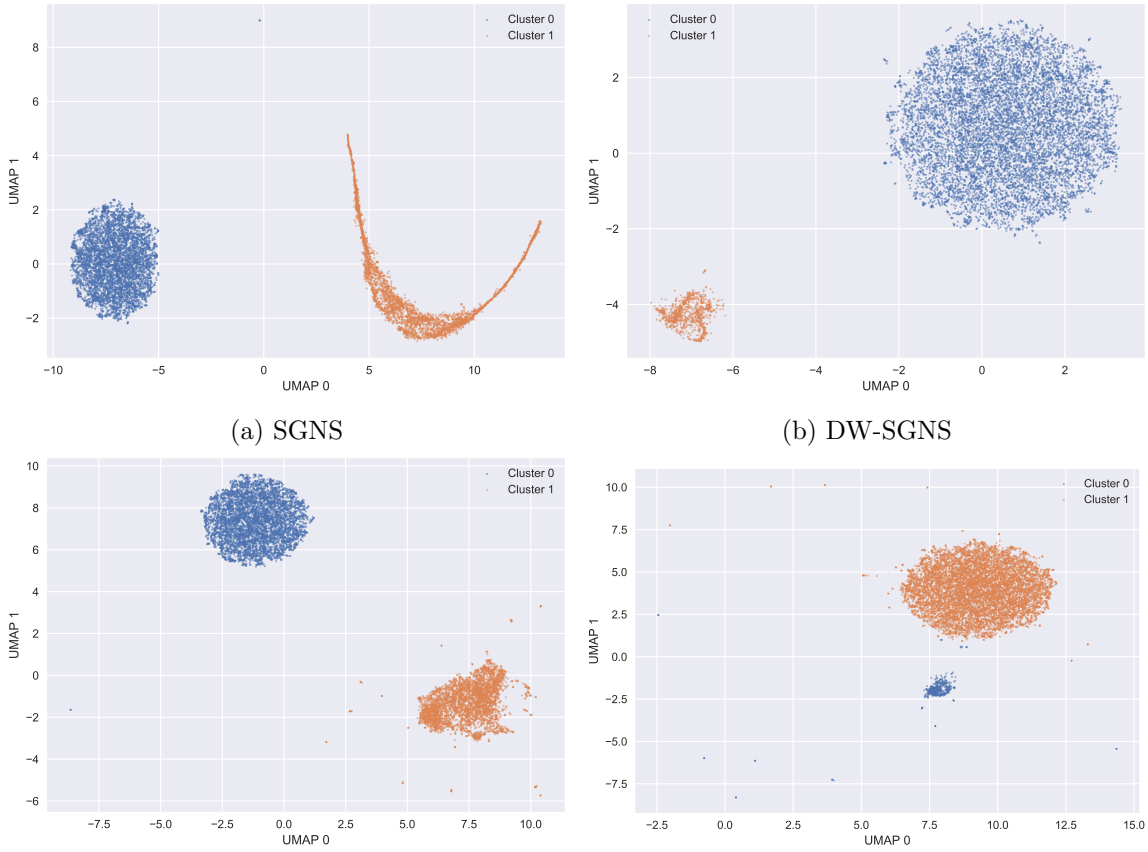


(b)

Figure 5.8: Relatedness survey results. (a) shows the % each model was selected across different frequency rank bands. (b) shows the % a neighbour was selected for each model.

models or because participants understand these tokens less, and thus the quality of response is lower, several tests are performed. Firstly, the Vader [78] dictionary is queried to see which tokens exist in its lexicon and which are Twitch.tv specific. In total, 21 tokens, i.e. 21%, exist in Vader, which is still a relatively small number, despite Vader-known tokens being more represented in the top 100 words compared to the general dataset where only 13% of tokens are in Vader. Because Vader contains words more typically used in common parlance, it can be expected that 'known' tokens are likely to be understood by participants. Next, given that respondents are more likely to respond 'None/Don't Know' for lower-ranked tokens, a Mann-Whitney U test is performed comparing the ranks of 'known' (to Vader) and 'unknown' (to Vader) tokens across the test set. In theory, if the 'known' tokens also tend to have high ranks, this may help explain the increase in 'None/Don't Know' responses for lower-ranked tokens. However, this test shows no statistically significant difference between the ranks of the known and unknown tokens, meaning that the known tokens are evenly distributed. Therefore, it is reasonable to believe that the performance difference is likely due to these models finding worse vectors for less common tokens. It is interesting to observe how the performance of the SGNS and 5T-SGNS changes as the popularity of tokens decreases. SGNS steadily becomes less often selected as token rank decreases, while 5T-SGNS is more often selected for lower-ranked tokens. This may be due to the sampling procedure for 5T-SGNS, where a much more extensive selection of tokens can produce non-0 labels, thus allowing the model to learn vectors for less popular tokens better. Unfortunately, this relatedness test only examined the most popular 100 tokens, so further experimentation would be required to confirm this.

Focusing on Figure 5.8 (b), we would expect to see that neighbours closer in the vector space are selected more often than those further away. This would show that the vector spaces are learning semantic token embedding. In general, this figure confirms this expectation, evidenced by the fact that the 50^{th} neighbour is consistently the least selected answer for all models. This confirms the assumption that the models are learning to co-locate tokens semantically. Interestingly, for both SGNS and 2T-SGNS, the 5^{th} neighbours are selected more than the 1^{st} neighbour, which is an unexpected result. It is not clear exactly why this is. Indeed, for 2T-SGNS, the percentage selected for both the 1^{st} and 5^{th} are near identical and may indicate that the space learnt by the 2T-SGNS model is better at co-locating many tokens which are all semantically similar. In fact, for the 5T-SGNS model, there is only a 5%



(a) SGNS

(b) DW-SGNS

(c) 2T-SGNS

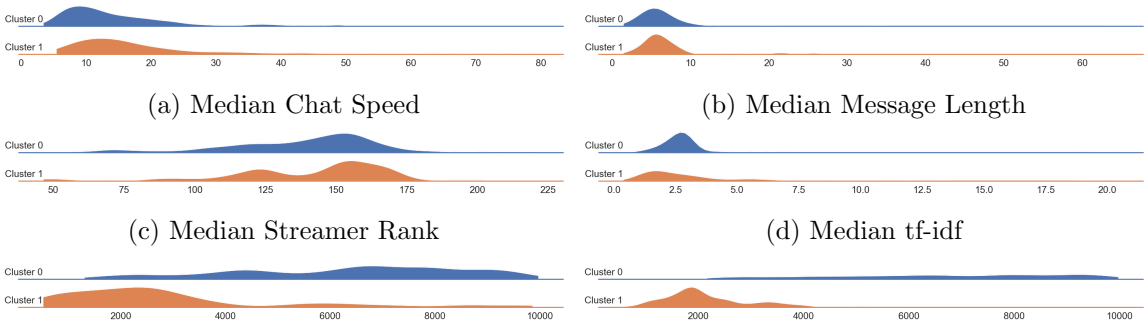
(d) 5T-SGNS

Figure 5.9: UMAP transformation of the vector space for each model (a-d). Each data-point represents a token in the vocabulary. Data-points are coloured based on the which k-means cluster they belong to.

difference between how often the 1^{st} and 5^{th} neighbour are selected. Perhaps this ability to more accurately co-locate many tokens is a benefit of the temporal SGNS models, although future work further examining the intricacies of chat data vector spaces would be required.

5.2.4.2 Visualising the Vector Spaces

The vector spaces have a dimension of 300, so the shape of the space is challenging to visualise. Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [122] is a dimensionality reduction technique which can decompose a high dimensional space into a very low dimension space, e.g. two dimensions. This low dimension space can then be visualised. UMAP is particularly attractive as a decomposition technique because it promises to retain both the global and local geometry of the original vector space in the transformed, lower-dimensional space. Additionally, the space can be clustered, e.g. by using k-means



(a) Median Chat Speed

(b) Median Message Length

(c) Median Streamer Rank

(d) Median tf-idf

(e) Document Frequency Rank

(f) Token Frequency Rank

Figure 5.10: KDE plots comparing the distributions of the 100 tokens closest to the centre of each cluster for SGNS.

clustering, which, combined with the UMAP visualisation, indicates the underlying structure of the high dimensional space. Applying k-means clustering to each model reveals that all models have two core clusters selected through the elbow method. The UMAP decompositions and clustering for all models are shown in Figure 5.9.

It is not initially clear why the models have two clusters. This property is remarkable, given that typical word vectors, e.g. those available in Tensorflow's Projector tool[2] form a single cluster. While all models produce a space which can be split into two clusters, each model generates differently sized, and shaped clusters, e.g. 5T-SGNS has a much larger cluster 0 and data-points, bar several outlier tokens, which are generally closer together. It may be that relaxing the label constraint allows for a less fragmented vector space because some token pairs may never appear in the same message, despite them appearing temporally close together. The vector space learnt by SGNS contains a strange crescent-shaped cluster. This specific shape is likely an artifact of the UMAP process rather than accurately represented in the vector space. However, it does indicate that the SGNS space may have even stranger geometry than the other models.

To further understand cluster make-up, it is possible to examine the properties that tokens in these clusters contain. Figure 5.10 shows kernel density estimation (KDE) plots of features for the 100 words closest to each cluster centroid. This section presents only plots for the SGNS model for brevity. The other models exhibit similar behaviour, and plots for these models can be found in Appendix B. 'Chat speed' refers to the number of messages sent in the 10 seconds surrounding a token's appearance. Cluster distributions which skew

[2]https://projector.tensorflow.org/.

higher in 'Chat speed' indicate tokens used more often when many messages are being sent, e.g. because the streamer has many viewers or because a reaction-worthy event has occurred in the stream. 'Message Length' is the average length of the messages containing the token. 'Streamer Rank' describes the popularity of the streamers whose stream this token appears in. Streamer Rank is calculated by ranking the mean number of views the streamer had for all documents for that streamer. The most popular streamer is assigned the rank of 1. 'Term Frequency-Inverse Document Frequency' (tf-idf) describes how important a certain token is to a given document. It is calculated by multiplying the 'term frequency', the number of times the token appears in a given document, i.e. $\text{tf(t,d)} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$ by the 'inverse document frequency', the log of the number of documents divided by the number of documents the token appears in, i.e. $\text{idf(i,D)} = \log \frac{N}{|\{d \in D : t \in d\}|}$. Thus tf-idf becomes $\text{tf-idf}(t, d, D) = tf(t, d) \cdot idf(i, D)$. High tf-idf indicates tokens often appearing in a given document but not in many other documents. These may be tokens specific to a particular game, streamer, or subculture. 'Document Frequency Rank' is the number of documents a token appears in, and 'Token Frequency Rank' is the total number of appearances for a given token.

These KDE plots show that the most significant difference between the clusters is how popular a token is, both in terms of 'Token Rank', i.e. overall popularity, and 'Document Rank', how many documents each token appears in. Cluster 0 tokens tend to be more critical to the documents they appear in, despite being less used. The message length is reasonably consistent between the two clusters, but cluster 0 tokens appear more often when chat is slower. Overall, it is evident that cluster 0 tokens are less prevalent in general but often have a high tf-idf and a flatter distribution across median stream rank. This can be interpreted to mean that cluster 0 tokens are probably tokens specific to certain streamers or games, possibly indicating game-specific terms or personalised emotes. Cluster 1, on the other hand, appears to be made up of more platform-wide terms, as they are more popular terms that appear in more documents. Further, because Cluster 0 tokens are more likely to appear when chat is slower, it is possible that these tokens are used in genuine discourse, e.g. discussing game strategy, rather than cheering.

5.2.5 Discussion

Livestream data is very different to typical text data. For instance, the popularity of tokens does not follow a Zipfian distribution, and the vast majority of popular tokens are not used outside of the platform. This, alongside the lack of existing datasets, makes studying this domain challenging. The TwitchChat dataset provides a foundation for shared livestream chat research, and the analysis presented in this section highlights the challenges in this domain.

Additionally, the learned vector space appears to cluster tokens into two clusters, rather than the expected once cluster which is observed from vectorising traditional language. Furthermore, minor changes to the model, e.g. varying the 'c' value in temporal models, results in very different spaces. However, all models are separable into two clusters which is an important discovery. The differentiating factor for these clusters seems to be how the tokens are used, measured through metrics such as token rank, document rank and tf-idf.

5.2.6 Conclusions

The TwitchChat dataset is a large-scale livestream chat dataset designed to empower the research community to begin exploring this domain, given the apparent differences from typical text domains. Several key challenges exist, for example, further understanding these vector spaces and research into models which can generate vector spaces with strong semantic or sentimental links between tokens, potentially uncovering the meaning of livestream-specific tokens and emotes. Likewise, given that these spaces are clusterable, it may be possible to explore the homogeneity of these clusters, e.g. through Hopkins Statistic [13]. Additionally, tracking tokens from each cluster over time may uncover information about what is happening in the stream and how the audience reacts. Another avenue of research could be to explore the implication of context on token use, especially given that distinct communities form around channels [68, 181].

This data is beneficial for understanding how highlight detection models can be developed using chat data. For instance, the models in Chapter 4 use pretrained networks. However, these do not exist currently for chat data due to the lack of attention from the Machine Learning research communities at the time of writing. This large-scale dataset can be used to pretrain such a model. Furthermore, the comparison between different SGNS models suggests that modelling the temporal aspect of chat is important, especially when modelling

less popular tokens. The next section in this chapter will use the Twitch-Chat dataset, as well as certain features, e.g. which tokens are popular when pretraining models can predict highlights from chat data.

Finally, an evaluation of the dataset shows that careful work in designing testing metrics for modelling chat data is required. While the crowd-sourced relatedness test enabled vector spaces to be compared, it appears less suitable for evaluating livestream vectors than spaces learnt through more typical text. This would likely result in significant research output but is beyond the scope of this thesis.

5.3 Textual Highlight Detection

The previous section explored the unique nature of livestream chat data and presented a curated livestream chat dataset. Analysis was performed via word vectorisation models. These vector models are extremely useful for understanding the nature of the dataset on a token basis. However, while these vectors are learnt using context, they are not necessarily immediately useful for modelling large segments of text because word vectors are designed to find a vector space for single words and not for sentences/messages. In order to perform textual highlight detection, longer-form text needs to be analysed, i.e. messages or sets of messages. This is not without its challenges. Not only is livestream text data challenging to model for the reasons discussed above, but long-form sections of data are temporal and thus require significantly more complex models than current language models.

For livestream highlight detection, a unique architecture is likely required for several reasons. In particular, unlike word vectors where input data needs to consider the distance between two tokens, highlight detection needs to operate on a 'window' of message data. Data from standard text, i.e. prose or social media text, can be easily converted into a vector of tokens. However, a window of livestream text is conceptually a vector of messages, each of which is a vector of tokens. Additionally, each message contains auxiliary data, e.g. the timestamp and author, which may be helpful when modelling. It is technically possible to ignore the auxiliary data and flatten all the messages into a single token vector, either across the window or on a per-second basis. Indeed this is the approach of many prior works into livestream highlight detection, e.g. [69]. However, this reduces the information passed to the network. For example, an increase in message frequency may indicate a highlight event

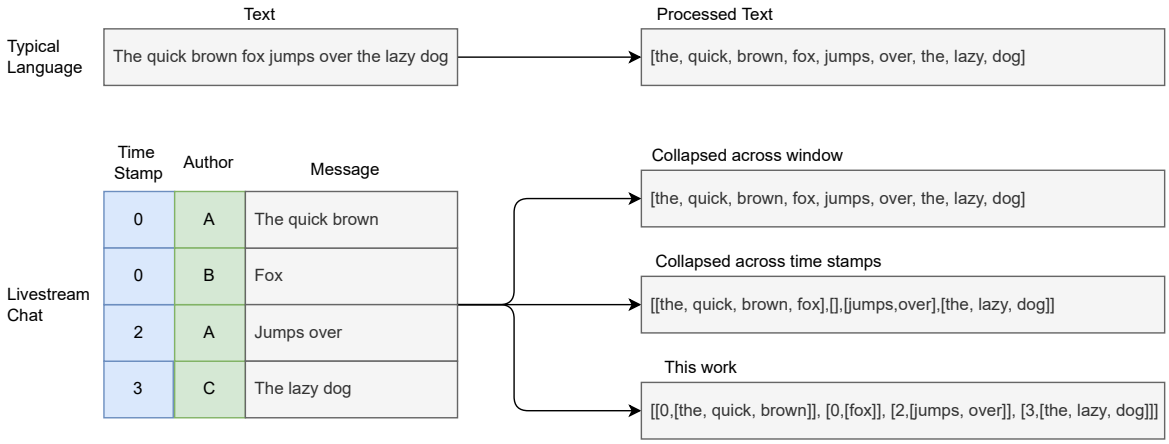


Figure 5.11: Comparison between data processing on typical language data, competing livestream processing, and this work.

in the stream, but this information is lost when the input data is collapsed. Therefore, the model presented in this study deliberately models both messages and timestamps to capture as much information as possible. Figure 5.11 demonstrates the difference between typical natural language data, how other livestream models often process data, and the method used for the model presented in this section.

For this work, the author of a message is not considered. This is for several reasons regarding the number of authors, i.e. commentating viewers, and the expected impact that the author has on predictive performance. Streams can be vast. For example, in the TwitchChat dataset, the largest number of viewers recorded was $165,371$, although the platform has seen larger numbers [206]. This makes it very difficult to build models which utilise authorship, as the number of additional variables to account for would be too large. Furthermore, it is unlikely that explicit authorship modelling will improve performance significantly given that livestream chats rarely have more than a handful of coherent voices [$32, 57, 189$], i.e. it does not matter a considerable amount exactly which author is writing a message.

5.3.1 Methodology

A novel transformer architecture, the 'Time-Aware Stacked Transformer Encoder' (TASTE) model, is proposed and compared to several competing chat-based highlight detection methods in this study to tackle the challenge of livestream highlight detection. The TASTE model is a stacked transformer architecture which uses two transformer blocks, each with a different purpose. The first transformer block, the 'message transformer', is akin to the transformer

layers in a typical language model encoder. It takes a single message, i.e. a vector of tokens, and produces a latent feature vector. This transformer block uses a learnt positional embedding to encode the position of each token and performs pooling via the 'CLS pooling' method proposed in [45]. This is where a 'CLS', i.e. classification token, is prepended to each message and the latent vector produced for this token is carried forward to the next stage of the model. The vectors for each token in a message are discarded. The second transformer block is the 'stack of messages transformer'. This takes a stack of latent feature vectors produced by the message transformer and encodes each with the time stamp that the message was sent. This is achieved using a Sine Encoding because the Sine Encoding produces an absolute encoding, i.e. encoding the particular time the message was sent, and a relative encoding, i.e. the time difference between two messages. This is important because the absolute position of a message is useful for mapping input message time to output label time while modelling the relative position between messages allows for the inherent modelling of features such as message frequency. This second transformer block allows the model to be 'time-aware', i.e. the architecture explicitly models the time messages were sent. This is one of the benefits of TASTE over competing approaches, which often collapse the time dimension of inputs and therefore lose information. After the stack of messages transformer has been applied, the resulting vectors, one for each message in the input, are then pooled via global average pooling (GAP) to produce a single latent vector which describes the chat activity over a period of time. GAP is used rather than CLS pooling as it is impossible to prepend the equivalent of the CLS token to the stack of messages input because the inputs are latent vectors for each model rather than a token embedding. This single latent vector is then passed to the downstream classification task, in this case, predicting highlights. This classification task is modelled through a series of fully connected neural network layers with a number of output heads equal to the number of labels. In this case each label represents if a highlight was occurring during a given time stamp or not. This model only requires the encoder architecture, unlike standard LM models, which are pretrained with an encoder and decoder, because it is designed for a classification task. Figure 5.12 demonstrates the TASTE architecture.

The TASTE model is trained by taking a stack of messages covering a particular time frame, in this case, $c + s + c$ where s is the sample window and directly corresponds to the number of output labels and c is a context window parameter to include data before

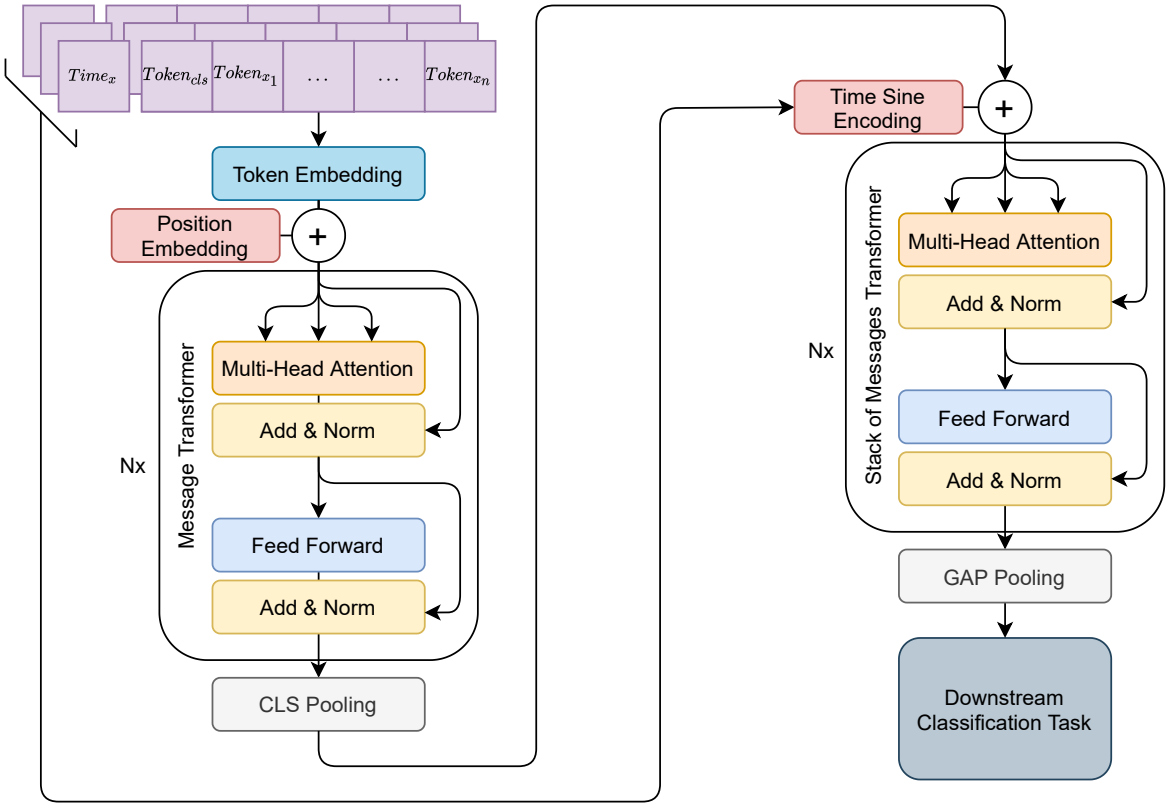


Figure 5.12: The TASTE architecture.

and after s. For ease of notation the entire window $c + s + c$ will be referred to using w. For the experiments in this chapter the parameters $c = 4$ and $s = 15$ are used (TASTE), inspired by [69], as well as $c = 2$ and $s = 5$ to highlight the impact of using a smaller window (TASTE Small). In practice, any values can be used, although larger windows will result in more memory consumption and thus may not be appropriate depending on the compute capacity available. Each message consists of two parts, the message body, a list of tokens, and the message timestamp, an integer value. For each input to the model, the timestamps are normalised to between 0 and w. This is done for two reasons. Firstly, it simplifies the sine encoding as, during initialisation, a fixed upper threshold needs to be set. Normalising the values removes the concern for test-set or deployment samples being outside the bounds of the sine encoding. Secondly, this allows the model to learn a mapping between input message time and label. This stack of messages, relating to w seconds of real-time, is then trained against a vector of binary labels of length s. This input-output mapping is demonstrated in Figure 5.13. Because a variable number of messages are sent in the w window, the model is trained using a fixed upper bound for the number of messages, set

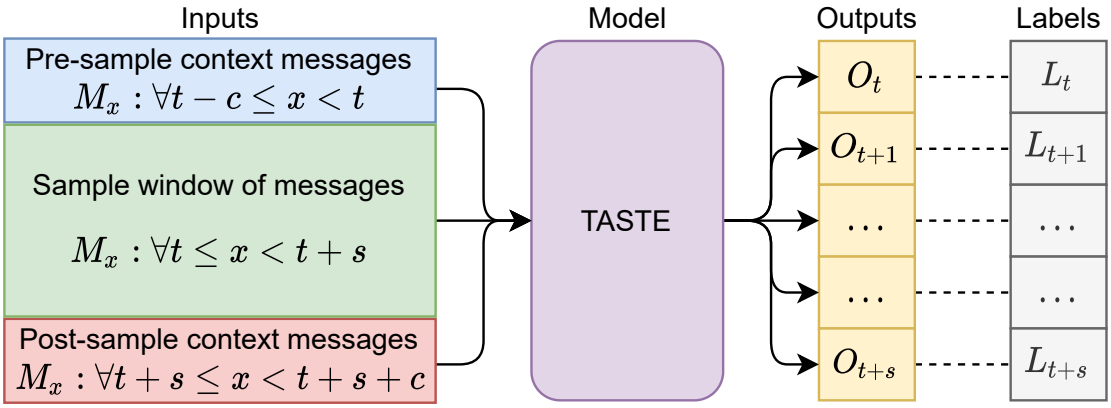


Figure 5.13: The input data to label mapping for TASTE.

extremely high, and each sample is padded to this size. The decision to set this upper bound very high, in this case, 320 messages, is motivated by the fact that there is no obvious way to safely truncate the message stack while retaining the ability to predict across a window of time. The maximum length for a message can be set to allow for truncation and is set at 64 tokens. Truncating token vectors is established in the literature, and therefore, this study applied suffix truncation for messages with length > 64, although these long messages represent only 0.0005% of the training data.

5.3.1.1 Single-Second Transformer Model

An additional transformer model is trained on data from each second of the game. Every message from a given timestamp is collapsed into a single token vector, as illustrated in Figure 5.11. Then the TASTE message transformer, CLS pooling and dense classification layers with a single output is trained to predict if this vector of tokens represents a highlight or not. This model, therefore, removes nearly all context for the highlight prediction and thus examines the assumed need for such data.

5.3.1.2 Pre-Training

The message transformer is pre-trained on three data sets. Firstly, the TwitchChat dataset presented in Section 5.2. Secondly, the Worlds dataset discussed in Section 5.3.2. Finally, data from a prior work, [58], again discussed in Section 5.3.2. Pre-training was carried out using the Masked Language Model (MLM) approach proposed in [46]. MLM pre-training works by first applying a mask to input token vectors, i.e. messages. This is done proba-

bilistically. For each message in the pre-training data, each non-CLS token is selected for masking with a probability of 15%. If a token is selected, it is masked in three ways. 80% of the time, the token is replaced with a blank MASK token. 10% of the time, the token is replaced with another token from the vocabulary. Finally, 10% of the time, the token is not masked at all but treated as such to bias the model when initially learning [46]. The pre-training task is to take this masked message and recover the original message. This allows the model to learn the general relationship between tokens in the corpus.

5.3.1.3 Highlight Selection

Like the audio-visual 'Auto-Highlight' model described in Chapter 4, TASTE utilises a selection threshold. However, the TASTE training task is supervised, and thus it is possible to better fit the threshold hyper-parameter by analysing the ratio of highlight to non-highlight segments in the training data. In the case of the data used in this chapter, around 12.5% is labelled as highlights. Interestingly this is identical to the heuristic value used in Chapter 4, although that was selected through casually observing the dataset rather than empirical analysis. Because TASTE is trained using supervised learning, it is possible, rather than using a threshold, to use the training output boundary to determine if a segment is a highlight or not, similar to the binary classification model discussed in the previous Chapter 4. Therefore both approaches will be tested and the results discussed in Section 5.3.4. The thresholding approach will be the default approach because it is the standard technique in highlight detection literature.

5.3.1.4 Comparison Approaches

Several competing approaches are presented to establish the competing state-of-the-art models for textual highlight detection. These models are detailed below.

5.3.1.4.1 Message Density Model Firstly, and most simply, the frequency of messages is used as a highlight indicator. This model acts under the assumption that changes in the number of messages being sent are directly related to how interesting what is happening in the stream is. First, a window, w, is determined to predict highlights using message density. In keeping with the other models presented in this section, the window used for this chapter is 23 seconds, i.e. a 15 second highlight segment s and two 4 second context windows, c.

Then the density for a given 1 second timestamp t is calculated by retrieving the number of messages sent for all windows which cover the timestamp in the highlight window, D(x) = $|m_i : \forall x - c < i < x + h + c|$, and then finding the average, $tsd = \frac{1}{h} \sum D(x) : \forall t - h < x < t$. Once a time stamp density, tsd, is calculated, the top $n\%$ time stamps with the highest densities are selected as highlights. As with other approaches $n = 12.5$. This approach has been proposed in several prior works, for example, [69].

5.3.1.4.2 biGRU-DNN Model The first deep learning comparison approach is the biGRU-DNN method proposed in [69]. Unfortunately, no publicly available code exists for this model, nor did the authors respond to requests for code and data to be shared. Therefore, the model version in this thesis has been implemented using the author's description in [69]. The model operates on a window of data. However, unlike TASTE, where each message is modelled, messages sent in the exact second are concatenated, i.e. the 'collapsed across time stamps' format from Figure 5.11. All tokens are converted to word vectors with $l = 50$. For this purpose, a word vector model, as discussed in the previous section, with an embedding space of 50 is pre-trained using the TwitchChat dataset. Then all tokens sent in a particular second are pooled by finding the mean value for each vector element. This results in a $w \cdot l$ latent space. This tensor is then passed through two Bi-Directional Gated Recurrent Unit (biGRU) layers. Finally, a set of feed-forward layers are used with a single $[0, 1]$ output value. This output value is used to determine whether or not the input data constitutes a highlight. This is a brief explanation of the system; a more thorough discussion is presented in [69].

5.3.1.4.3 L-Char-LSTM The second deep learning approach is the L-Char-LSTM proposed in [58]. This is another model based on recurrent units. Importantly, L-Char-LSTM operates on the character level, i.e. the input is a series of characters rather than a series of tokens. The authors choose to utilise a character-level model because, as found with the TwitchChat dataset, they recognise the challenges of misspellings and emotes. The model's vocabulary is reduced by training on just ASCII characters, 128 tokens. This reduces the size of the embedding part of the network. However, it has the consequence that words are not entirely understood, so analysis similar to that carried out with the TwitchChat dataset is impossible. Furthermore, it requires the recurrent layers to correctly model the meaning of the order of tokens, given that each token has no inherent meaning, unlike word tokens.

This approach is in contrast to the preprocessing and lemmatisation approach taken with the TwitchChat dataset, although initially, it is unclear which approach is more appropriate. The authors have released source code as well as trained models and data. Thus the L-Char-LSTM model used in this thesis is taken directly from the author's released work.

5.3.2 Data

Three datasets are utilised for this study. Firstly, the TwitchChat dataset is used when pretraining word vector models (e.g. for the biGRU-DNN). Secondly, the text data from [58], a total of 218 games, is used as training data. Finally, a new dataset is gathered from the 2020 *League of Legends* World Championship. This was gathered for two reasons. Firstly, due to technical reasons, it was impossible to collect the video data associated with the chat data from [58]. It was impossible to compare a visual model, e.g. the Auto Highlight model proposed in Chapter 4, to textual highlight detection. Secondly, at the time of data gathering, it was the most recent large-scale tournament and therefore represented the most recent *League of Legends* esports presentation. The world's data is split into test and train data. All group-stage games, 49 games, are added to the 218 games from [58] to form the training data. The test set for this work is all of the knockout games from the tournament, 27 games in total. For all of these games, ground truth highlight annotations are gathered through crowd-sourced annotations, either through the method discussed in [58] for games in that dataset or through the method detailed in Chapter 4 for games from the 2020 *League of Legends* World Championship. For the biGRU-DNN and L-Char-LSTM models, the raw data was processed as required by those models. The data was prepossessed and lemmatised for both transformer architectures using the techniques outlined in Section 5.2.1.2.

5.3.3 Experiment

Several experiments are carried out to explore the efficacy of the proposed TASTE transformer and further analyse the predictions that the model makes. The first task is to compare the transformer-based models proposed in this thesis with competing approaches. Six models are presented; the single-second transformer, two TASTE models, and the three competing approaches. Two TASTE models are presented to examine the impact that window size has. Furthermore, several hyperparameters and other features of the TASTE model are examined. All models are implemented in Python using a range of popular machine learning libraries.

PyTorch [153], Keras [35], and Tensorflow [1] were all used to compose and train models, with Numpy [70], and Scikit Learn [154] used for a range of auxiliary functions. All models were trained on using 2 PCs. The first has a Ryzen 5 1600 CPU, 16GB of RAM and an Nvidia 2080ti GPU. The second had an Intel i7 9700 CPU, 64GB of RAM, and 2x Nvidia Titan RTX GPUs. The models in this study are trained in a supervised setting; therefore, the key metrics reports are Accuracy, Precision, Recall and F1 Score.

5.3.4 Results

5.3.4.1 Experiment 1 - Comparison of Textual Highlight Models.

Table 5.4 details the performance for each model on the test set. Understandably, the Density model did not perform particularly well. This is to be expected. It is the least complex model and fails to consider the content of messages. Perhaps more interestingly, the single-second transformer performed poorly despite its more complex architecture. This is likely because, like the density model, it is not provided with enough information to make a reliable decision. Not only does the single-second transformer model not have the context of how many messages are being sent, but it is also expected that there may be some 'lag' between a highlight moment occurring and the chat reacting due to participants needing to type messages. Finally, the biGRU-DNN model performed very poorly. Upon further reflection on the model, this is likely due to the judicious use of pooling and collapsing functions, resulting in a model which does not have enough information to learn an effective mapping. It was also observed that the biGRU-DNN model struggled to converge during training. Unfortunately, this model could not be improved without further information from the authors. In terms of the models which performed better, the L-Char-LSTM model appears slightly worse than the TASTE Small model, with the full TASTE model performing much better. It is unsurprising that the larger TASTE model outperforms the smaller one across all metrics. However, it is interesting to observe that it is much better the larger window is because this suggests that highlight cues from text data often lag by over 6 seconds. Furthermore, the performance of both TASTE models compared to the L-Char-LSTM model indicates the value of using the custom transformer architecture.

Figure 5.14 show the distribution of F1 Scores across the test set for all models, and Table 5.5 details the Min, Max and Median F1 Scores. From these, it is clear that there is a large variance in how models performed across the test set. This is different to the performance

Table 5.4: Results from Experiment 1. Bold denotes a model outperformed all other models in the metric.

Model	Accuracy	Precision	Recall	F1
Density	0.775	0.212	0.173	0.190
biGRU-DNN [69]	0.751	0.118	0.097	0.107
L-Char-LSTM [58]	0.577	0.200	**0.580**	0.300
Single-Second Transformer	0.773	0.202	0.166	0.182
TASTE Small	0.809	0.349	0.286	0.314
TASTE	**0.838**	**0.463**	0.378	**0.416**

Table 5.5: F1 scores distributions across the dataset for Experiment 1. Bold denotes a model outperformed all other models in the metric.

Model	Min F1	Max F1	Median F1
Density	0.038	0.479	0.192
biGRU-DNN [69]	0.033	0.210	0.095
L-Char-LSTM [58]	**0.149**	0.365	0.287
Single-Second Transformer	0.104	0.248	0.179
TASTE Small	0.061	0.447	0.332
TASTE	0.061	**0.542**	**0.431**

of audio-visual models, i.e. Chapter 4, where models performed more consistently, except for an outlier game. Unfortunately, it is unclear if this is because there is a higher level of variance and noise within chat data leading to inconsistent performance prediction or if this is merely an artefact resulting from plotting F1 Score rather than Average Precision. What is clear is that the biGRU-DNN model is consistently poor, likely a result of the training issues discussed above. Surprisingly, there is one sample where the density model performed very well, perhaps because the game was particularly engaging, and the high-density segments were more often due to exciting gameplay rather than other factors. While the best performing in terms of total F1 Score across the dataset and Median F1 score, the TASTE model appears less consistent. It is not entirely clear why this is, although Section 5.3.5 discusses some of the model's prediction errors.

Finally, Table 5.6 shows both the Wilcoxon signed-rank test and Vargha-Delaney A measure results comparing these models. As with the models presented in Chapter 4, most models are statistically distinct, measured through Wilcoxon signed-rank test, in their predictions. The most interesting result is that the Density and Single-Second models are highly similar in terms of both statistical metrics. At first, this appears puzzling, given that the

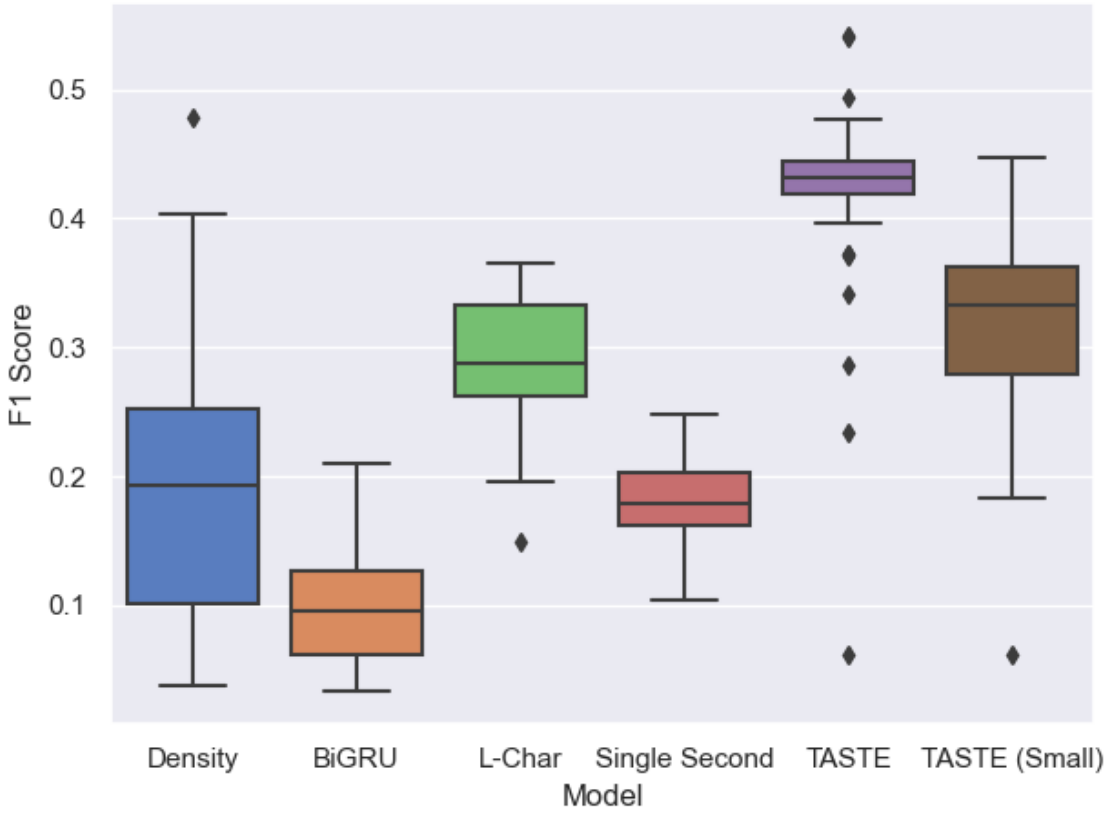


Figure 5.14: Box and Whisker plots for all models evaluated in Experiment One.

Density model is provided with only the number of messages being sent and not their content, while the Single-Second model only has the content of messages and not the frequency. However, this similarity may be due to factors influencing both models. For example, the Single-Second model receives vectors of all tokens sent at a given time stamp. If many messages are being sent, this vector is likely to be very long, and the model may be learning that the length of a message is important to highlight prediction, i.e. implicitly learning a density-based prediction. Secondly, there is an observed 'cheering' effect in esport tournament broadcasts. Often this is indicated through a small subset of tokens being used rapidly and repeatedly. This may be both observable through the content of messages and an increase in density; hence both models perform similarly. Finally, the TASTE model shows a substantial improvement in effect size compared to all other models, and even the smaller TASTE model is much better than all models other than the L-Char model from [58]. This is further evidence of the power of the TASTE approach compared to competing approaches.

Table 5.6: Statistical testing results for Experiment One. The upper right triangle details Vargha-Delaney A measure. Values less than 0.5 indicate column outperforms row and values more than 0.5 indicate row outperforms column. † denotes a small effect size ($a < 0.44 \vee a > 0.56$). ‡ denote a medium effect size ($a < 0.36 \vee a > 0.64$). ⋄ and bold denotes a large effect size ($a < 0.29 \vee a > 0.71$). The lower left triangle details p-values calculated via a Wilcoxon signed-rank test. Bold denotes significant values, i.e $p < 0.05$.

	Density	BiGRU	L-Char	Second	TASTE	TASTE Small
Density	-	**0.77**$^{\diamond}$	**0.19**$^{\diamond}$	0.51	**0.09**$^{\diamond}$	**0.14**$^{\diamond}$
BiGRU	$\mathbf{2 \times 10^{-3}}$	-	**0.01**$^{\diamond}$	**0.12**$^{\diamond}$	**0.03**$^{\diamond}$	**0.04**$^{\diamond}$
L-Char	$\mathbf{2 \times 10^{-4}}$	$\mathbf{6 \times 10^{-6}}$	-	**0.96**$^{\diamond}$	**0.10**$^{\diamond}$	0.39†
Single Second	9×10^{-1}	$\mathbf{3 \times 10^{-5}}$	$\mathbf{6 \times 10^{-6}}$	-	**0.04**$^{\diamond}$	**0.07**$^{\diamond}$
TASTE	$\mathbf{7 \times 10^{-6}}$	$\mathbf{6 \times 10^{-6}}$	$\mathbf{3 \times 10^{-5}}$	$\mathbf{6 \times 10^{-6}}$	-	**0.86**$^{\diamond}$
TASTE Small	$\mathbf{6 \times 10^{-5}}$	$\mathbf{8 \times 10^{-6}}$	2×10^{-1}	$\mathbf{9 \times 10^{-6}}$	$\mathbf{6 \times 10^{-6}}$	-

5.3.4.2 Experiment 2 - Impact of Thresholding

The initial decision threshold was set at 12.5% of the video's length, in line with the amount of stream content annotated as highlights in the training data. However, the model is imperfect and thus makes false-positive errors, i.e. segments are selected despite not being labelled as a highlight, and false-negative errors, i.e. highlight segments, are not detected by the model. It may be possible to improve the models' performance metrics by relaxing or tightening this threshold. Figure 5.15 displays the Accuracy, Precision, Recall and F1 Score across a sweep of thresholds from 6.75% to 50% in increments of 6.75. Selected thresholds, where various metrics scored highest, are further detailed in Table 5.7. As expected, when the threshold is lowered, i.e. fewer segments are selected as highlights, accuracy and precision increase but recall decreases. If the model selected the entire stream as a highlight, it would achieve perfect recall but have low precision. Likewise, if the model were only allowed to select a single segment, assuming that segment was a ground truth highlight, it would achieve perfect precision but low recall. Selecting the ideal threshold value is about finding a balance between these two extremes, represented in part by F1 Score. That said, there is an additional concern not modelled by Precision, Recall, and F1 metrics; video length. For example, the threshold 12.5% was initially selected because it is precisely the video length the average viewer is likely to expect, given that it was empirically decided from the training data. Likewise, it is clear that selecting a large threshold value, e.g. 50% of a video, is unlikely to be helpful in a real-world setting because the video will be too long. The best performing threshold value in

Table 5.7: Selected video threshold. The full sweep of thresholds is shown in Figure 5.15.

Threshold	Accuracy	Precision	Recall	F1
6.75%	**0.851**	**0.527**	0.233	0.323
12.5%	0.838	0.463	0.378	0.416
19.25%	0.810	0.403	0.507	**0.449**
50%	0.601	0.254	**0.830**	0.389

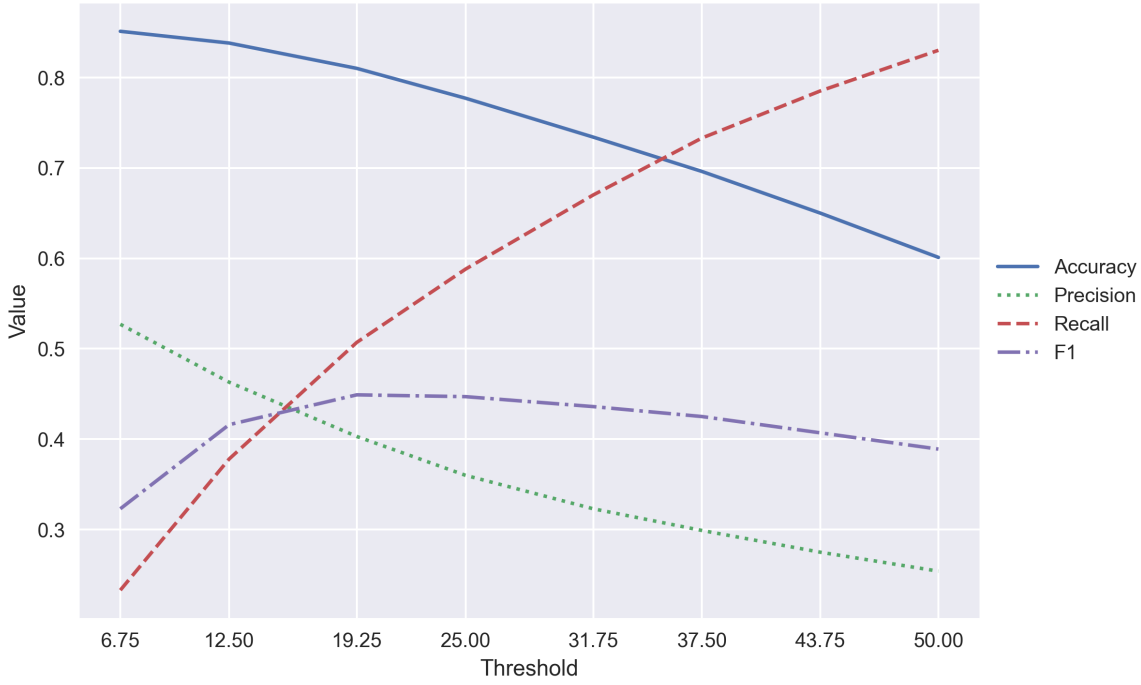


Figure 5.15: Accuracy, Precision, Recall and F1 Score across a range of video thresholds.

terms of F1 Score is 19.25%, slightly higher than the initially set value because it sacrifices minimal precision, i.e. makes a smaller number of additional false-positive predictions while recalling around 12.9% more ground truth highlight moments. It seems evident from these results that some threshold value between 12.5% and 19.25% is likely the most appropriate. In a deployed model, the user could easily control this parameter based on whether they feel slightly longer videos are worthwhile.

TASTE is trained in a supervised setting, and therefore it is possible to, rather than threshold across a video, threshold across the output from the network, much like the classification models discussed in Chapter 4. This would allow the model to dynamically change how much of a particular video is considered a highlight, i.e. in theory, in a less interesting stream, fewer highlights would be selected. Figure 5.16 displays the Accuracy, Precision, Re-

Table 5.8: Selected model threshold. The full sweep of thresholds is shown in Figure 5.16.

Threshold	Avg. Video Threshold	Accuracy	Precision	Recall	F1
0.1	76.5%	0.369	0.188	**0.938**	0.313
0.3	27%	0.763	0.345	0.609	**0.441**
0.4	**13.7%**	0.833	0.450	0.404	0.426
0.5	6%	**0.853**	0.552	0.217	0.312
0.6	2%	0.850	**0.573**	0.069	0.129

call and F1 across a sweep of thresholds from 0.1 to 0.6 in increments of 0.1. Where various metrics scored highest, selected thresholds are further detailed in Table 5.8. The average percentage of segments selected as highlights is also detailed in Table 5.8. The model threshold 0.3 scores highest in F1 Score. On average, this is equivalent to 27% of a video being selected. The model value 0.4 is the closest to the 12.5% video threshold initially applied. From this, it is clear that the model performs better on the test set when the threshold is slightly relaxed. Naturally, as discussed above, the relaxation increased the video length. In some respects, the length of a highlight video is just as important to model evaluation as if the video captures all annotated highlight clips or does not include dull moments because the goal of a highlight video is to be succinct. Video length is accounted for explicitly in the video thresholding discussed above because the same percentage of segments are selected for each video. However, this is not the case in model thresholding hence the average being reported. The distribution of video length is discussed in more detail below.

The test set has largely been discussed as a single set of video segments. However, it is a set of videos, each of which is constructed of segments. Therefore it is also possible to plot the distributions of metrics for each video. This is done for two threshold values, a video threshold of 12.5%, i.e. the original empirically determined value, and a model value of 0.4125, which averages to the same 12.5% of segments in the test set selected as highlights. Figure 5.17 and Table 5.9 demonstrate these distributions. Accuracy is omitted from these plots because it is the least useful metric given its lack of focus on the minority highlight class. Instead, the number of video segments selected by each method is plotted. This shows the difference in the number of segments selected by each technique. The model threshold technique has a wide distribution, whereas the video threshold technique distribution is much tighter, despite both having similar averages. The tighter distribution of video threshold segments was expected, given that most matches have roughly similar lengths, and the model

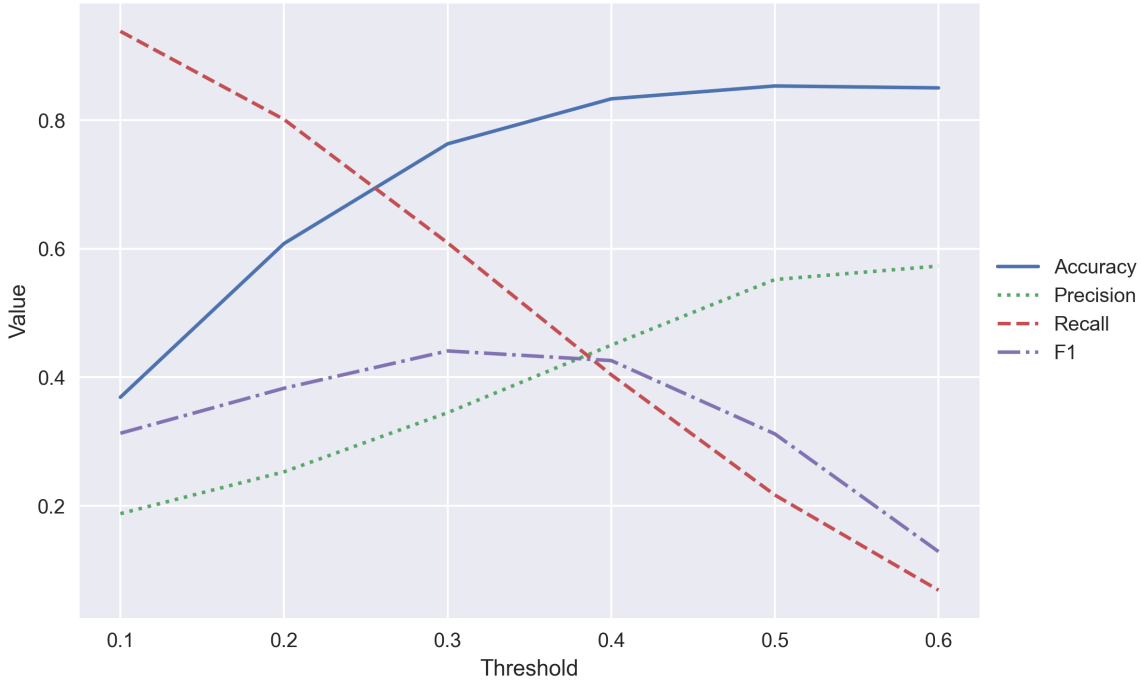
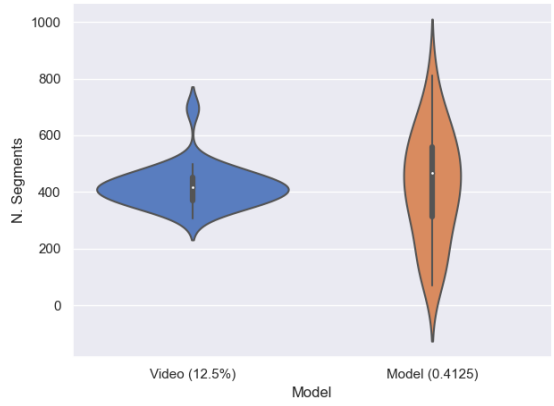


Figure 5.16: Accuracy, Precision, Recall and F1 Score across a range of model thresholds.

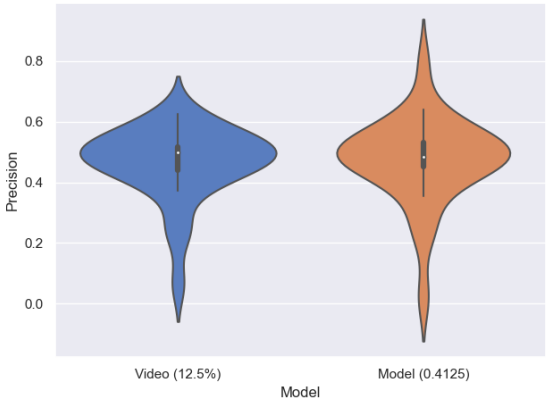
Table 5.9: Metric distributions for both video and model thresholding techniques across the test set.

Threshold	Metric	Minimum	Maximum	Mean	Median
Video	Segments	307	696	420.407	418
Model	Segments	72	812	420.185	467
Video	Precision	**0.066**	0.626	0.463	**0.499**
Model	Precision	0.023	**0.792**	**0.472**	0.485
Video	Recall	**0.057**	0.571	**0.378**	0.386
Model	Recall	0.010	**0.613**	0.370	**0.404**
Video	F1 Score	**0.061**	**0.543**	**0.412**	**0.447**
Model	F1 Score	0.013	0.551	0.394	0.427

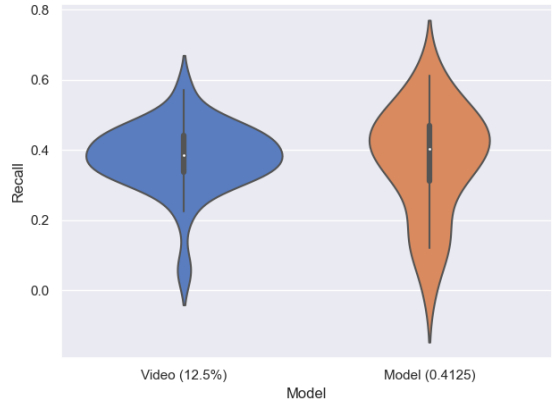
predictions are tied to this length. What is more interesting is how wide the distribution of segments is selected by the model threshold technique, resulting in some very short or long highlight videos. To explore this further, Figure 5.18 shows the distribution of the percentage of a video selected as a highlight for both the ground truth labels as well as the model threshold approach. The actual ground truth labels have a much tighter distribution than the predictions. Note that the video threshold results are not presented because there is no distribution to speak of. The model always selects the same percentage as highlights.



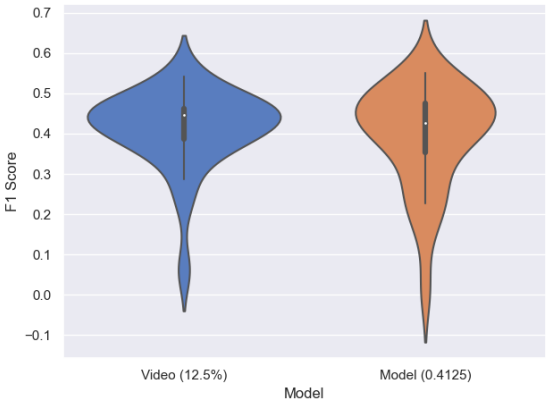
(a) N. segments selected distribution across test set.



(b) Precision distribution across test set.



(c) Recall distribution across test set.



(d) F1 Score distribution across test set.

Figure 5.17: Metric distribution for all videos in the test set for equivalent video and model thresholds (model threshold 0.4125 relates to an average video threshold of 12.5%) .

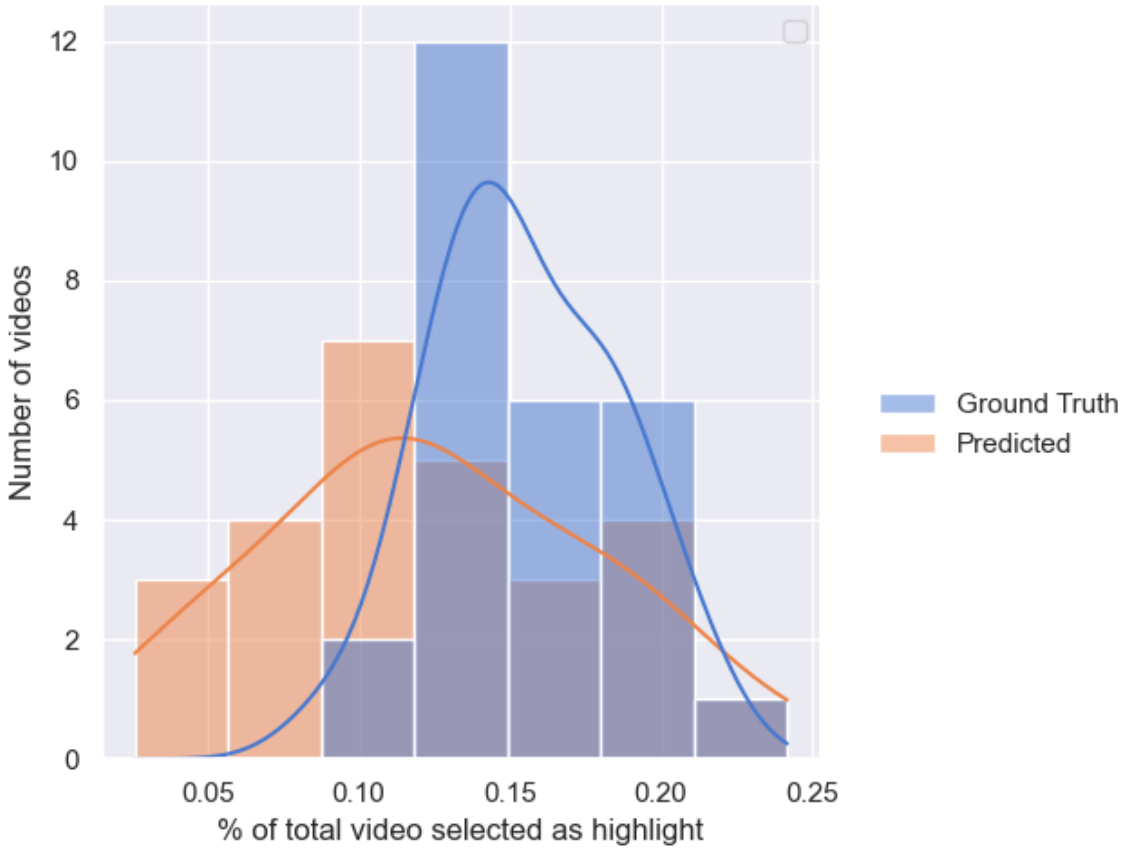


Figure 5.18: Histogram of the percentage of each video in the test set selected as a highlight when using model thresholding with the threshold set at 0.4125.

In general, it seems that using a model threshold results in a model with higher variance. For some videos, the model performs very well, perhaps those most similar to the training data, but the model performs more poorly for others. The video threshold appears to have less variance. Judging the models on F1 Score video threshold is clearly better and is likely the safer choice for a deployed system given the distribution of segments selected; the model threshold may generate videos which are either very long or very short, although judging by Figure 5.18 they are likely to be shorter than a human editor. Ultimately, it is impossible to firmly argue that thresholding in one way or the other is best. Both offer positives and negatives. That said, video thresholding will likely be more reliable in a deployed setting. Finally, it is essential to note that each model's highlight signal does not change when experimenting with thresholding techniques. The only change is what part of the signal is classified as a highlight.

5.3.4.3 Iterative Threshold

Another thresholding approach, building off the video thresholding above, is to iteratively relax the threshold boundary while having some metric for filtering out segments selected as highlights that are likely to be low quality. For example, such a metric may be segment length. If the segment is very short, it may not make sense to the viewer and thus should be removed from the highlight output. When applying iterative thresholding, it would still be important to predict the same percentage of highlights, in this case, set at 12.5% as in the original experiments. However, the iterative relaxing would lower the model output threshold at which highlights are recognised while enforcing a minimum length for video segments. Thus, segments which are broadly exciting but may have a less interesting middle section are more likely to be recognised, while very short segments which may, in a vacuum, seem interesting but lack interesting context are likely to be omitted. This technique is motivated by the observation that human editors are likely to perform a similar pruning operation when editing a highlight video. Iterative thresholding is performed via Algorithm 1. Briefly, the initial threshold predictions are made. Next, all highlight segments shorter than the minimum highlight length parameter are removed. Then, if enough segments have been selected, the algorithm finishes. Otherwise, the threshold is relaxed to allow select one more prediction and the process is repeated.

Algorithm 1 Iterative Thresholding

```
1:  procedure ITERATIVE THRESHOLDING(predictions, threshold, minSegmentLength)
2:      cuttoff ← count(predictions) × threshold
3:      sortedPredictions ← sortDescending(predictions)
4:      while True do
5:          iterThreshold ← sortedPredictions[cuttoff]
6:          highlights ← all predictions > itterThreshold
7:          processedHighlights ← ProcessHighlights(highlights, minSegmentLength)
8:          if count(processedHighlights)/count(predictions) > threshold then
9:              return processedHighlights
10:         else
11:             cuttoff ← cuttoff+1
12: procedure PROCESS HIGHLIGHTS(highlights, minSegmentLength)
13:     segments ← all highlight segments in highlights
14:     segments ← all segments > minSegmentLength
15:     return segments
```

Minimum segment lengths ranging from 1 second, i.e. no iterative thresholding is applied, to 120 second long segments are explored to test this iterative thresholding approach. The

Table 5.10: Selected iterative threshold in values. The full sweep of thresholds is shown in Figure 5.19.

Minimum Length	Accuracy	Precision	Recall	F1
1	0.838	0.463	0.378	0.416
5	0.838	0.462	0.378	0.416
8	0.838	0.464	0.38	0.418
33	**0.847**	**0.498**	**0.418**	**0.455**
120	0.788	0.387	0.295	0.335

accuracy, precision, recall and F1 score for these values is shown in Figure 5.19, as well as selected values in Table 5.7. From this, setting a minimum length for segments shows an improvement in F1 Score, up to a point. The best performing Iterative Threshold model has a minimum length of 33 seconds. However, this is likely impractical; analysing the length of segments in the training data shows that the most common segment length is 5 seconds, with a median length of 13 seconds, a lower quartile of 8 seconds and an upper quartile of 23 seconds. Therefore values of 5, i.e. the mode, or 8, the lower quartile, are likely to be more appropriate as minimum lengths and produce more realistic results. These values are also reported in Table 5.7. Unfortunately, these more realistic values have only a minimal impact on the performance metrics. This is not to say that utilising iterative thresholding would not produce highlight videos which appear more realistic to the viewer, just that the standard testing metrics across the whole dataset do not change greatly. Testing if using iterative thresholding would make more realistic videos is extraordinarily hard to model via metrics and thus would require extensive testing with human evaluators.

5.3.4.4 Experiment 3 - Time-Series Manipulation

Two techniques for improving the model's performance via time-series prediction manipulation are reported. Both are applied to the standard TASTE model, and unless otherwise stated, the standard threshold hyperparameter, i.e. 12.5%, is retained.

Firstly, manual observation of the output from the TASTE model suggests that performance benefits may be achieved if the time series prediction is offset by a fixed number of seconds. Such a technique makes sense for textual highlight detection from audience reaction because there is an expected time delay between an event happening in the game and the chat reaction. In theory, this is modelled with the input data context window, but it may be that additional alignment is fruitful. This offsetting is particularly interesting in the context

Figure 5.19: Accuracy, Precision, Recall and F1 Score across a range of iterative threshold minimum segment lengths.

of the results from the iterative thresholding experiment. These results show that grouping highlight segments into long, i.e. $> 30 second$ segments results in performance increases. This may be a result of the need to offset. For example, long highlight segments enforce context from before or after the local maxima of the highlight signal. Correctly offsetting the signal to align better with the in-stream event may result in a more realistic prime iterative threshold value.

Figure 5.20 details the Accuracy, Precision, Recall and F1 Score that the TASTE model achieved across a range of offset values, from -30 seconds to $+30$ seconds. From this, it is clear that offsetting the predictions by a negative amount, i.e. aligning reactions with the triggering in-stream event, has a very positive effect on prediction. The best performing offset in the F1 Score is -15, which achieved an F1 Score of 0.529, a huge improvement over the standard output. This plot also clearly shows that offset values > 0 harm prediction, which is natural given that chat messages rarely indicate that an exciting moment is about to happen.

As discussed previously, offsetting and iterative thresholding are likely to interact. Both iterative thresholding and offsetting are applied simultaneously to examine this. For sim-

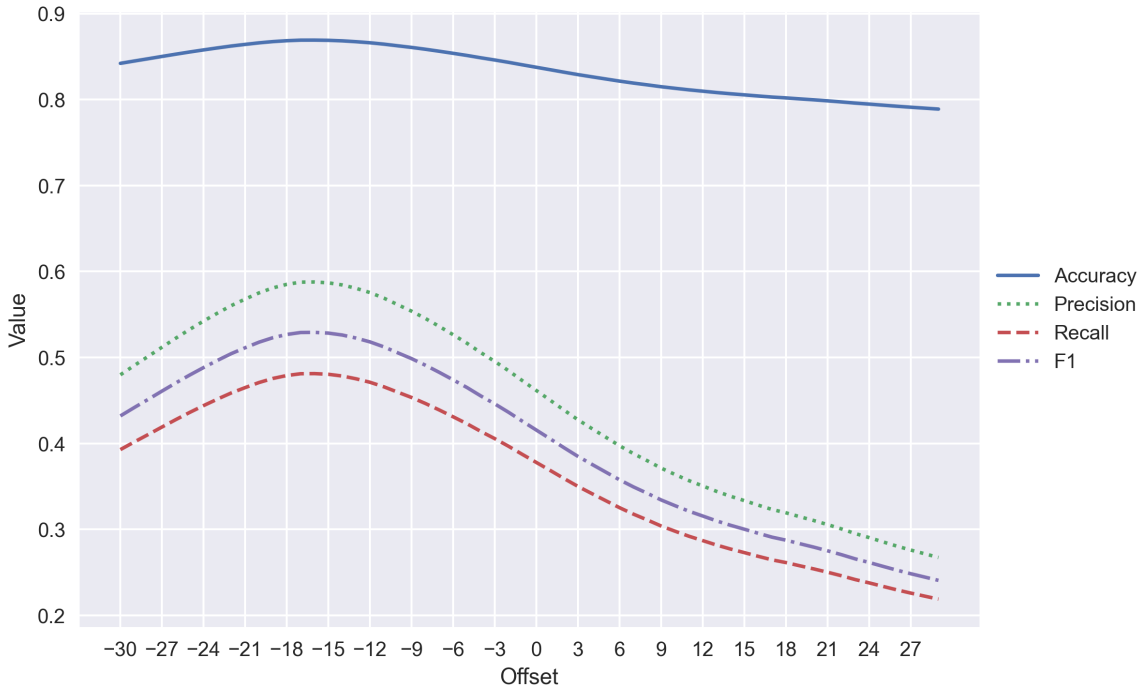
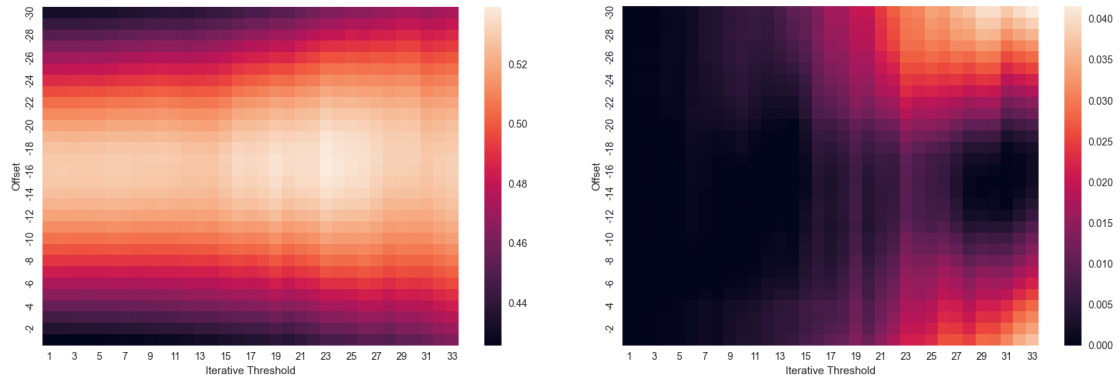


Figure 5.20: Accuracy, Precision, Recall and F1 Score across a range of offsets.



(a) F1 Score heatmap.

(b) Differential heatmap.

Figure 5.21: F1 Score and F1 Score differential heatmaps show model performance when both Iterative Thresholding and Offsetting are applied. Higher values in (a) indicate better performance overall, higher values in (b) indicate values for a larger impact when applying Iterative Thresholding.

plicity, only offset values between -30 and 0 are experimented with, and iterative threshold values between 1 and 33. Figure 5.21 (a) shows the F1 Score across a range of offset and iterative threshold values. The best performing combination of these two approaches was to offset by -17 and apply an iterative threshold of 23, which achieved an F1 score of 0.539.

Figure 5.21 (b) shows the change in F1 Score when applying iterative thresholding and

offsetting simultaneously. This change is the delta between no iterative thresholding and applying it. In general, iterative thresholding has only a small impact on performance, although it is noticeable in certain situations. Firstly, large iterative threshold values are helpful when an inappropriate offset is selected, e.g. large or small values. However, this is not a particularly useful discovery because this effect does not make up for the performance deficit of the poorly selected offset. Secondly, it is clear that iterative thresholding values ¿ 15 seconds are generally better at 'correcting' this poorly chosen offset value. Finally, there is a very subtle improvement when considering the best-performing offsets. For example, offset -15 performed best using a minimum segment length of 6. However, the improvement is minimal, improving F1 by 0.001. While iterative thresholding has minimal performance improvement when offsetting, it does not negatively impact results, and thus a small value, e.g. $\sim 4- \sim 10$, may help provide context to the viewer and thus be desirable.

The second time-series manipulation technique is to apply Time-Series smoothing using a Gaussian smoothing kernel, as in Chapter 4. While the TASTE model can understand significantly more context than the audio-visual model from Chapter 4 because it receives up to 23 seconds of chat data, there may still be some benefits to smoothing for longer context-awareness. This may be especially useful given the assumption that there is a lag between highlight moments and audience reaction. That said, this lag is unidirectional, i.e. audience reaction always lags behind in-stream events. However, the smoothing technique was proposed to include context from surrounding events in the context of audio-visual prediction and thus has a symmetric kernel. Therefore, it is unlikely that smoothing will prove as effective as it was in Chapter 4.

A sweep of potential candidates is proposed to test the most appropriate value for the smoothing kernel. Kernel sizes ranging from 3 to 120 are experimented with. Accuracy, Precision, Recall and F1 Score across the sweep of values is shown in Figure 5.22. From this, the optimal kernel length for TASTE is 77, with an F1 Score of 0.432. In a vacuum, this seems like a more negligible improvement than the initial iterative thresholding experiments. However, further investigation showed that while promising on a non-offset time series, an iterative threshold is unnecessary when applied to an offset signal. Iterative thresholding is conceptually different to smoothing because it was designed to model tendencies in segment length selections. However, it appears to function better as a way to force context to be included, similar to the goal of smoothing. Since iterative thresholding is not particularly

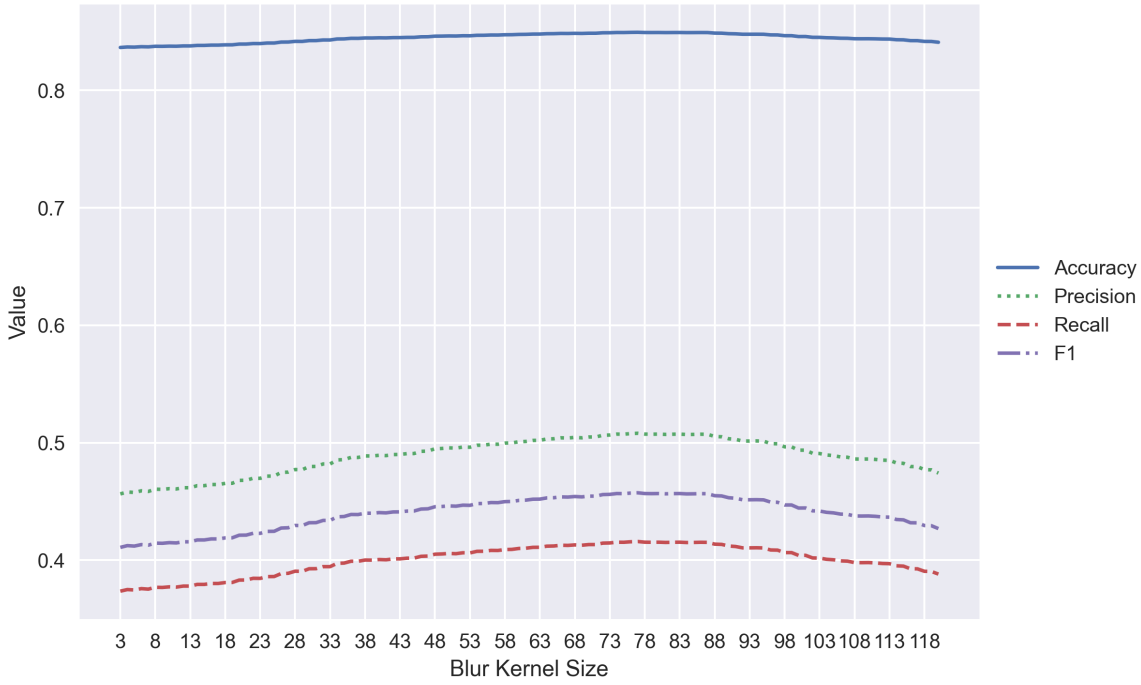


Figure 5.22: Accuracy, Precision, Recall and F1 Score across a range of blur kernel sizes for the TASTE model with no offsetting.

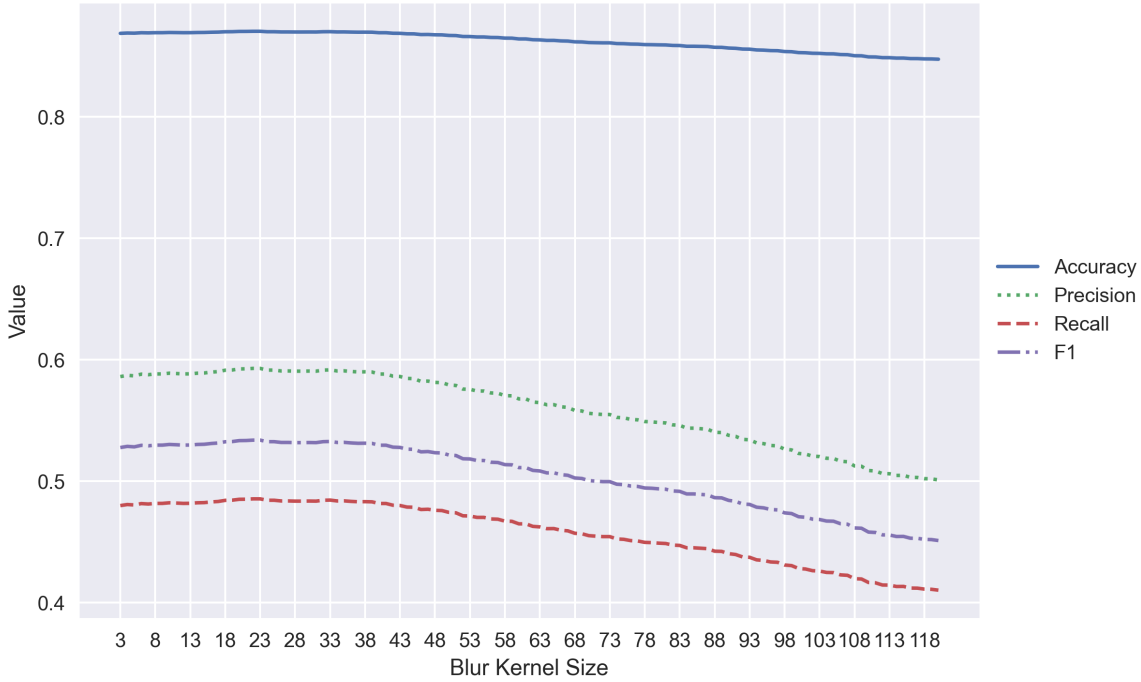


Figure 5.23: Accuracy, Precision, Recall and F1 Score across a range of blur kernel sizes for the TASTE model with a -15 offset.

Table 5.11: Blurred vs Non-Blurred.

Model	Kernel Size	Accuracy	Precision	Recall	F1
Standard	-	0.838	0.463	0.378	0.416
Smoothed	77	0.86	0.480	0.393	0.432
Offset	-	0.869	0.587	0.480	0.528
Offset + Smoothed	22	**0.870**	**0.593**	**0.485**	**0.534**

effective when the time series is correctly aligned, it seems valuable to test smoothing with an offset signal. Therefore, the performance of various kernel sizes is also considered when applied to a signal offset by -15. Interestingly, the best performing kernel size for the offset signal is 22, much smaller than the best kernel size for the non-offset signal, Figure 5.23. This is likely because, as with iterative thresholding, smoothing is less helpful after offsetting has been applied and the signal aligned.

Table 5.11 details comparison between applying smoothing and offsetting. Offsetting is the most useful time-series manipulation approach. As expected, the improvements seen by implementing smoothing are not as dramatic as in Chapter 4. That said, smoothing still has clear utility for the offset and non-offset time series. Therefore it is likely to include both techniques if deploying a TASTE model. Iterative thresholding provides no real noticeable performance benefit when using both offsetting and smoothing, at least in terms of F1 Score.

5.3.5 Discussion

Livestream chat is capable of contributing solid indicators of in-stream highlight moments. That said, careful consideration needs to be made regarding the model architecture. For instance, the various competing approaches presented in this work have apparent performance deficits due to loss of information, e.g. through collapsing messages into single-timestamp vectors. TASTE performs very well and is similar in performance to the Auto-Highlight model presented in Chapter 4. This is an exciting result given that, unlike that model, which gets its cues from what is happening in the stream, TASTE is only provided with time-delayed audience reactions.

5.3.5.1 Exploring prediction errors

Figure 5.24 provides a visual example of how the model makes predictions for a single game. Most ground truth highlights are correctly identified, but there are examples of incorrectly

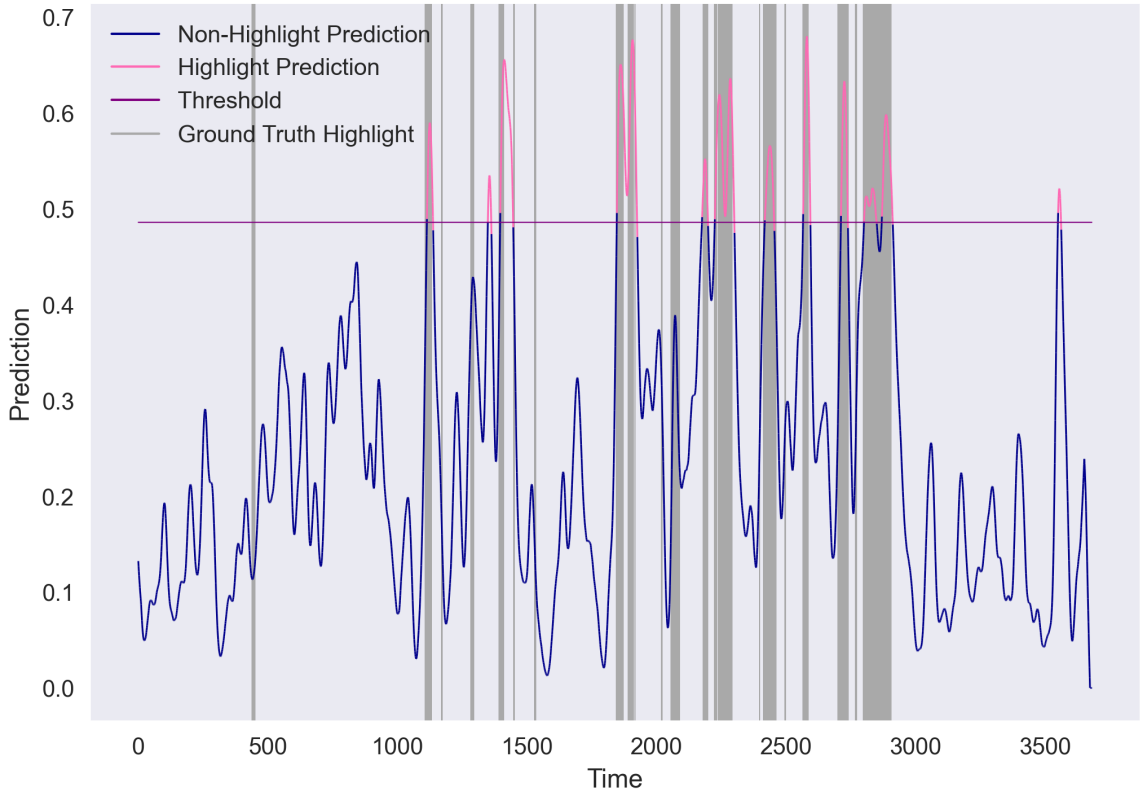


Figure 5.24: Time Series Highlight Prediction using and offset of −15, smoothing, and an iterative threshold minimum length of 6 seconds.

selected and non-selected elements. The ground truth highlight which occurs shortly before the 500 second point is an interesting example. This segment shows the character selections made by the players before the game starts. This is included within a highlight video made by human editors not because it is exciting or interesting but because it provides context to viewers. However, because this moment is not noteworthy in itself, there is little audience reaction, and thus the TASTE model cannot recognise it. This is one potential weakness of audience-based highlight detection, given that an audio-visual model trained on the editing style of human editors is more likely to recognise the importance of this moment.

Conversely, the moment selected as a highlight just after the 3500 second point is interesting to the model and thus a moment which provoked a strong audience reaction. However, it occurs after all ground truth highlight segments and thus occurs after the end of the game. Interestingly, manual inspection of this segment appears to show the opposite of the missed segment discussed above. Here a piece of post-game analysis is shown detailing statistics from the game. These statistics were exciting or interesting enough to trigger a significant

reaction from the audience, e.g. because a particular team played well. This content could be interesting to the highlight viewer, despite being omitted from the original video. This may be an example of the TASTE model being able to detect moments outside of those selected by a human editor but are still attractive, at least to the live audience. The Auto-Highlight model is trained on audio-visual, which is the same information the human editor has access to and thus is likely to more closely replicate the style of human-edited highlights, regardless of if moments prove to be exciting to the live audience or not.

Finally, there is a large ground truth highlight segment at around 2000 seconds which the TASTE model misses. Further, it appears that TASTE outputs generally low values for this segment, i.e. it is evaluated as reasonably uninteresting. This segment contains combat between the two teams, but where both teams were cautious, and only one player died. It is likely, therefore, that the audience did not react strongly to this segment. Ultimately this segment of the game was relatively unexciting to the live audience. However, the editor felt it was important to include it, perhaps due to one player dying or because it indicated a shift in the game's momentum. Naturally, such moments are impossible to detect from chat data alone if they do not evoke an audience reaction.

5.3.5.2 Exploring the deployability of TASTE in a 'real-world' setting

Like the Auto-Highlight model discussed in Chapter 4, the TASTE model performs well. Thus it is both interesting and essential to assess its deployability in a real-world setting, i.e. can the be model run sufficiently quickly that it is useful as part of a highlight detection product. To test this, the TASTE model's execution time and memory requirements are examined and compared to competing approaches. Once again, this is deliberately performed on consumer-grade hardware, the same NVIDIA 2080ti equipped machine as in Chapter 4, to allow fair comparisons between textual and audio-visual highlight models, as well as to indicate if these approaches are likely to be able to be operationalised in a real-world setting. Once again, the memory requirements for the models implemented in Keras (BiGru-DNN, Single Second Transformer, and both TASTE models) are approximated. The L-Char-LSTM implementation is the one released by [58] and is thus implemented in PyTorch.

Table 5.12 details the number of weights each model has, their training and inference memory requirements, and the average time it takes for the model to predict a single second segment at run time. Interestingly, the L-Char-LSTM model is parameter and memory light

Table 5.12: Weights, memory, and time requirements for each neural network model. Memory requirements are listed in MB and are an approximation.

Modality	Weights	Training Memory	Inference Memory	Average Time (seconds)
BiGru-DNN	3,408,953	1334.15	6.07	0.0022
L-Char-LSTM	382,210	-	1.53	0.0277
Single Second Transformer	3,567,873	21.31	7.54	0.0003
TASTE	5,746,319	536.79	27.87	0.0167
TASTE Small	5,741,445	289.10	14.60	0.0062

but takes a long time to execute. This is likely because the model is applied to characters and thus processes more inputs for a message. The BiGru-DNN model is memory expensive when training because the default batch size is large compared to the TASTE models. This is not an issue during inference because predictions are made for one sample at a time. Besides the L-Char-LSTM model, the TASTE architecture takes the longest time to execute a sample during inference and requires the most memory. However, the execution time is still very fast, taking about as long as just the Audio Auto-Highlight model and an order of magnitude faster than any Visual Auto-Highlight model. Therefore, TASTE is easily capable of real-time prediction and thus suitable for deployment.

5.3.6 Conclusions

This study presented the TASTE architecture and comparisons to other state-of-the-art approaches. TASTE performs exceptionally well at livestream highlight detection from text data cues. This is due to its ability to more completely model the temporal aspect of text chat compared to comparison approaches. However, despite the added complexity of the TASTE architecture, it still executes very quickly, performing fast enough for real-time prediction and operating faster than most Auto-Highlight models. This is because, unlike audio-visual models, TASTE does not require input feature extraction and can operate on raw messages, albeit with light prepossessing. Furthermore, as with Chapter 4, time-series post-processing techniques are proposed, improving performance again.

5.4 Highlight Detection from Textual Data Conclusions

This chapter has presented two studies which outline the challenges and opportunities of working with Livestream chat data. The Twitch-Chat dataset provides the first publicly

available large-scale dataset of audience chat logs from a wide range of livestreams. Furthermore, it allows for the analysis of livestream data, particularly its form, i.e. temporal, time-sensitive, and with many authors, as well as its contents, i.e. many tokens, e.g. emotes, unique to the platform. The word vector models trained as part of that study show the strange nature of the vector space and the challenge in evaluating livestream chat models, given how many of the popular Twitch chat tokens are not in common usage.

However, the word vector models are unlikely to be useful for highlight detection, hence, the proposal of the TASTE architecture. TASTE uses state-of-the-art transformers in a custom architecture designed to model the unique format of livestream chat data. TASTE performs very well at detecting highlights, especially when the audience reaction signal is correctly aligned with the triggering event. Furthermore, TASTE operates quickly enough to be suitable for deployment in a live setting using readily available consumer-grade hardware if desired.

Chapter 6

Towards Unified Audio, Visual & Chat Models

6.1 Introduction

Until now, chat data has been modelled separately to video, i.e. audio-visual, data. This is for several reasons. For instance, the process for obtaining audio-visual data is different to chat data. At several points in this thesis, research could only be carried out on audio-visual or chat data. For example, audio-visual data was sometimes gathered post-broadcast from video-sharing platforms to alleviate the necessity for gathering data live, which would have taken months to gather a suitably sized dataset. Unfortunately, this post-broadcast format does not contain chat data, and thus it is impossible to explore audience reactions with this data. Additionally, the TwitchChat dataset was gathered by monitoring the Inter Relay Chat channels associated with each stream, and thus it was impossible to recover the broadcasts occurring via the same method. Furthermore, these full-length broadcasts are incredibly long and broadcast in high definition, sometimes 4K, resulting in datasets with exorbitant disk requirements.

Building a unified highlight detection model that includes cues for the video feed and audience reaction can be expected to yield higher quality highlight videos than just the broadcast or chat data. Chapter 4 shows how valuable audio-visual data is, Chapter 5 demonstrates the utility of Chat data, and the benefits of multimodal fusion have been shown throughout the thesis. Therefore, it naturally follows that modelling all modalities is likely to produce the most accurate models. However, fusing these three domains is trickier than

audio-visual fusion, which was utilised in Chapters 3 and 4. This challenge is mainly due to the complexities of handling the time-lagged audience reaction alongside the broadcast feed. Therefore comprehensive study into audio-visual-textual fusion for highlight detection is, unfortunately, outside of the scope of this thesis. That said, rudimentary fusion approaches may yield surprisingly effective results. This chapter presents the beginnings of research into this audio-visual-textual highlight detection task utilising model fusion in Section 6.2 and outlines future directions for unified highlight detection in Section 6.3.

6.2 Unification through Model Fusion

Chapters 4 and 5 detail successful methods for modelling audio-visual and textual data, respectively, for highlight detection in game-driven broadcasts. This study utilises these models to explore a unified audio, visual, and textual model. It also compares this unified to models using fewer modalities. Additionally, analysis is carried out to explore the relationship between highlight signals from different modalities, given that they are trained independently on different data views of the same events.

This study shows that even when using a straightforward fusion method, there is the potential for considerable improvements to prediction, with the best performing model achieving a 20% increase over both Auto-Highlight and TASTE, each of which outperforms comparable state-of-the-art approaches. Furthermore, segments which are falsely rated highly by one modality, but are not rated highly by the other modalities, are shown to be less represented in the final model, reducing false-positive predictions.

6.2.1 Methodology

Model fusion is one approach utilised in Chapter 4 and describes modelling each modality separately and averaging the highlight signals. It is a particularly attractive technique for fusing these domains because performing fusion on richer data, i.e. feature fusion, is challenging given the inherent time lag and variations in feature representation in chat data compared to audio-visual data. Three predictive models are used. Firstly, the Auto-Highlight visual ranking model from Chapter 4, uses a stack of RGB and Flow frames. Secondly, the Auto-Highlight audio ranking model operates on a segment of audio data. Finally, the TASTE model from Chapter 5, models text data. To fully explore the relationship between modal-

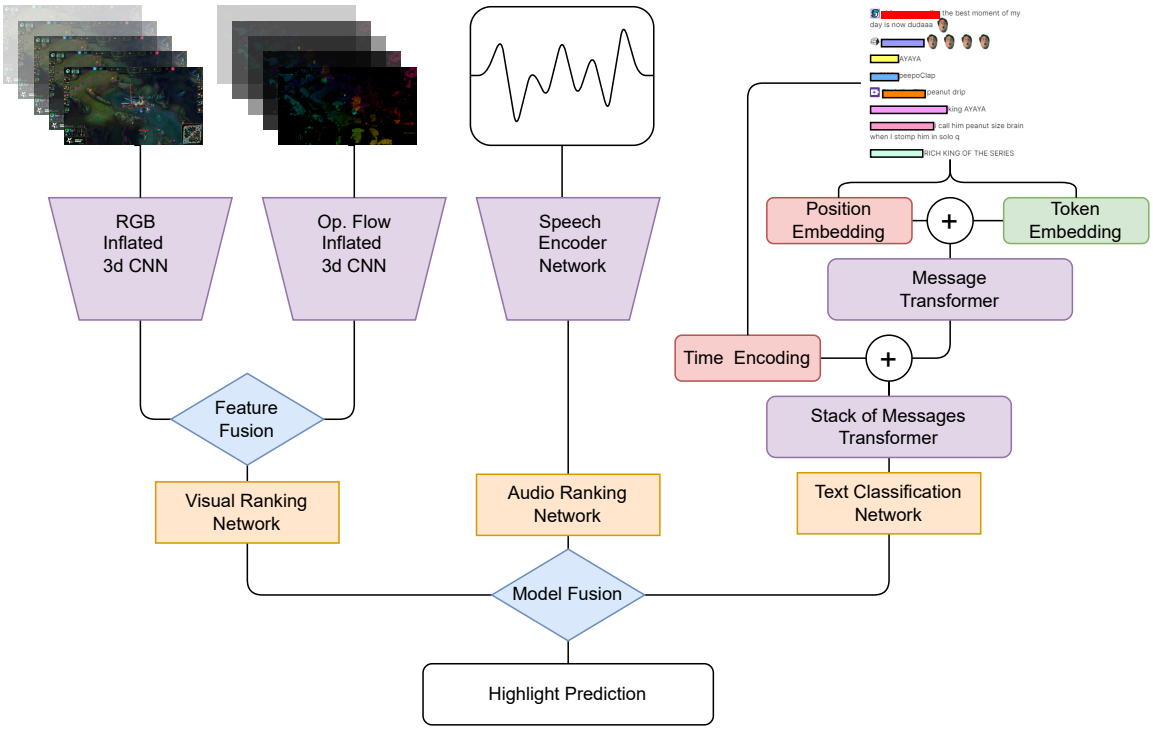


Figure 6.1: Unified Audio, Visual and Textual modelling through model fusion. The left hand section represents the visual model from Chapter 4, the middle section represents the audio model from Chapter 4, and the right hand section represents the text model from Chapter 5.

ities, the Auto-Highlight visual and audio networks are treated separately for this chapter. Performing model fusion on these two networks produces the same result as the hybrid-fusion Auto-Highlight model.

The model weights used are taken verbatim from the studies in Chapters 4 and 5. However, the dataset is the 'worlds' dataset from Chapter 5. This means that the Auto-Highlight models have no fine-tuning on the worlds group games and thus are not exposed to the world championship setting, i.e. the broadcast set or presenters. This study, therefore, additionally acts as a test of the generalisability of the Auto-Highlight model to an unseen setting. However, this has the downside that it biases comparisons between TASTE and Auto-Highlight, assuming that fine-tuning on the group stages, as with TASTE, noticeably impacts predictive performance. Therefore, this study does not claim that audio-visual modelling is better than chat or vice versa but, rather, explores how effective unifying the two approaches through model fusion is.

6.2.2 Data

For this study, the 'worlds' dataset described in Chapter 5 is used to evaluate models. This is because it is the only dataset presented in this thesis that contains audio, visual and textual data for the same broadcasts. The data used in Chapters 3 and 4 is audio-visual only, and the TwitchChat dataset contains only text data. All models are evaluated on the knockout games from the worlds data, i.e. the evaluation set from the second study in Chapter 5. Therefore the performance of the text-only model, TASTE, in this study is identical.

6.2.3 Experiment

The performance of model fusion across the two Auto-Highlight models and TASTE is compared to the performance of these models without fusion. This experiment also details the correlations between the three inputs to the model fusion mechanism. As with prior studies, all models are implemented in Python using a range of popular machine learning libraries. PyTorch [153], Keras [35], and Tensorflow [1] were all used to compose and train models, with Numpy [70], and Scikit Learn [154] used for a range of auxiliary functions. All models were evaluated using a PC with a Ryzen 5 1600 CPU, 16GB of RAM and an Nvidia 2080ti GPU.

6.2.4 Results

Table 6.1 shows the performance across the whole test set for seven models. Three unimodal models are tested, Audio (Auto-Highlight), Visual (Auto-Highlight) and Textual (TASTE). Additionally, the hybrid fusion Auto-Highlight model is presented. Three model fusion models which fuse audio-visual domains with textual data are also reported.

These results show that model fusion textual data with other modalities is highly effective despite the simple model-fusion mechanism. For instance, including textual information alongside visual data improved the F1 score by 0.118 compared to visual data alone, a 23.3% increase, and 0.090 compared to only textual data, a 16.8% increase. Likewise, combining audio and textual data sees an improvement of 0.161 over the audio-only model, and 0.059 over the textual-only model, for increases of 37.2% and 11% respectively. Unsurprisingly, the Audio-Visual-Textual model performs the best with a 0.018 improvement in F1 score over the Visual-Textual model and percentage improvements of 48.6% over audio-only, 26.9% over

Table 6.1: Comparison of modality performance for the entire test set.

Model	Accuracy	Precision	Recall	F1
Audio	0.842	0.479	0.392	0.432
Visual	0.862	0.562	0.460	0.506
Auto-Highlight	0.872	0.600	0.491	0.540
TASTE	0.870	0.593	0.485	0.534
Visual-Textual	0.896	0.693	0.568	0.624
Audio-Textual	0.887	0.659	0.540	0.593
Audio-Visual-Textual	**0.900**	**0.713**	**0.584**	**0.642**

Table 6.2: Results for Experiment One. Minimum F1 Score, Maximum F1 Score, Mean F1 Score, and Medium F1 Score are shown for a selection of models across each video in the dataset. [1] [218].

Modality	Min. F1	Max. F1	$\bar{m}$ F1	$\tilde{m}$ F1
Audio	0.178	0.699	0.433	0.456
Visual	0.377	0.613	0.504	0.505
Auto-Highlight	0.361	0.687	0.539	0.539
TASTE	0.074	0.719	0.535	0.548
Audio-Visual-Textual	**0.410**	**0.779**	**0.640**	**0.637**

visual only, and 20.2% over textual only. This model also outperformed all other models in every metric presented in Table 6.1, i.e. Accuracy, Precision, and Recall.

As expected, the Audio-Visual-Textual model is the best of the unified broadcast-audience models. Therefore, because this study is primarily exploring the performance change compared to established models from previous chapters and for brevity, this will be the only video-textual model discussed. Table 6.1 showed the performance of a selection of models across the entire dataset. However, this does not highlight how a model performs on individual videos. For example, it may be possible that a model performs extremely well with one video and poorly on another. However, such a finding is not captured in this table. Therefore Table 6.2 demonstrates how models perform across the test set on an individual video basis. As discussed in previous chapters, F1 score is one of the more powerful metrics for describing highlight performance, and thus this table details the distribution of F1 Scores.

These results are exciting because they demonstrate that the audio-visual-textual model improves performance on average and that the worst performance on a video is better than other models, as is the best performance. This is further demonstrated by Figure 6.2 which

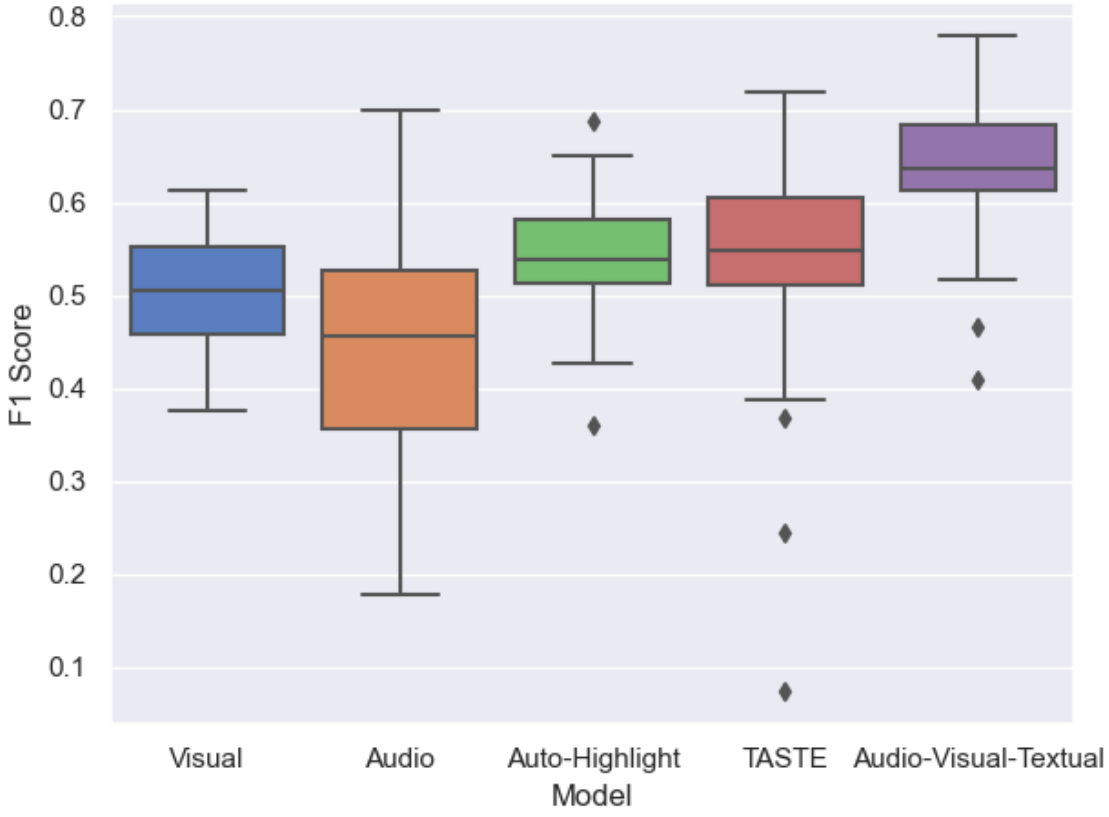


Figure 6.2: Box and Whisker plots for a selection of models presented in Experiment One.

shows box and whisker plots for all videos in the test set. This shows that some models, namely the TASTE model, have a wide range, i.e. sometimes the model performs very poorly, and sometimes it performs very well. However, in general, the models which utilise more than one modality appear to have smaller distributions of F1 Scores, especially compared to TASTE or the audio Auto-Highlight model. This is further evidence, as discussed in the literature that multimodal models are likely to reduce prediction noise.

Finally, Table 6.3 details the Vargha Delaney A measure and Wilcoxon signed-rank test values for pairs of models. Unsurprisingly, there is a large effect size and a statistically significant difference in prediction for the audio-visual-textual compared to all other models. This further highlights how important the improvement to performance is when all modalities are used, given that this study uses a simplistic model fusion approach. Unsurprisingly, because TASTE is the best performing non-unified model, it is the model where unification has the smallest effect on performance. However, even still here, the effect size would be considered 'large'. Likewise, the audio-only model, which is the worse performing model in

Table 6.3: Statistical testing results for Experiment One. The upper right triangle details Vargha Delaney A measure. Values less than 0.5 indicate column outperforms row and values more than 0.5 indicate row outperforms column. † denotes a small effect size ($a < 0.44 \vee a > 0.56$). ‡ denote a medium effect size ($a < 0.36 \vee a > 0.64$). $\diamond$ and bold denotes a large effect size ($a < 0.29 \vee a > 0.71$). The lower left triangle details p-values calculated via a Wilcoxon signed-rank test. Bold denotes significant values, i.e $p < 0.05$.

	Audio	Visual	Auto-Highlight	Taste	Audio-Visual-Textual
Audio	-	$0.361^{\dagger}$	$\mathbf{0.252}^{\diamond}$	$\mathbf{0.252}^{\diamond}$	$\mathbf{0.082}^{\diamond}$
Visual	1×10^{-1}	-	$0.345^{\ddagger}$	$0.318^{\ddagger}$	$\mathbf{0.091}^{\diamond}$
Auto-Highlight	$\mathbf{5 \times 10^{-5}}$	2×10^{-2}	-	$0.581^{\dagger}$	$\mathbf{0.159}^{\diamond}$
TASTE	1×10^{-2}	9×10^{-2}	6×10^{-1}	-	$\mathbf{0.217}^{\diamond}$
Audio-Visual-Textual	$\mathbf{7 \times 10^{-6}}$	$\mathbf{7 \times 10^{-6}}$	$\mathbf{4 \times 10^{-5}}$	$\mathbf{2 \times 10^{-5}}$	-

terms of F1 score, is also improved upon with a large effect size by using any other model except the visual-only model.

6.2.5 Discussion

The implications of Table 6.2 and Figure 6.2 are important for understanding how the fusion mechanism influences prediction, especially the range of F1 scores across the test set. For instance, the model with the smallest range, and therefore the most robust to variance in different videos, is the visual-only model. On the other hand, both the audio and TASTE models demonstrate a wider range. This may be for several reasons. For instance, there may be a variety of commentators, and some may have commentating styles with clearer highlight signals than others. Likewise, some matches may attract particularly vocal audiences, e.g. because of the teams involved, making highlight detection more or less difficult. On the other hand, the visual cues are more consistent between matches. Fusing audio or textual data with visual data reduced the range of F1 scores per video compared to these modalities but, naturally, increased the range compared to just visual data. The audio-visual-textual has an F1 Score range of 0.369, which is worse than the visual and auto-highlight models. However, the model performs very well in general compared to these models, and the worst-performing game for the unified model scored higher than the visual model.

It is interesting to consider how these individual modality models, i.e. Audio, Visual and Textual (TASTE), correlate with each other and the final unified model. Figure 6.3 displays a correlation matrix for these models. Perhaps unsurprisingly, the unified model correlates

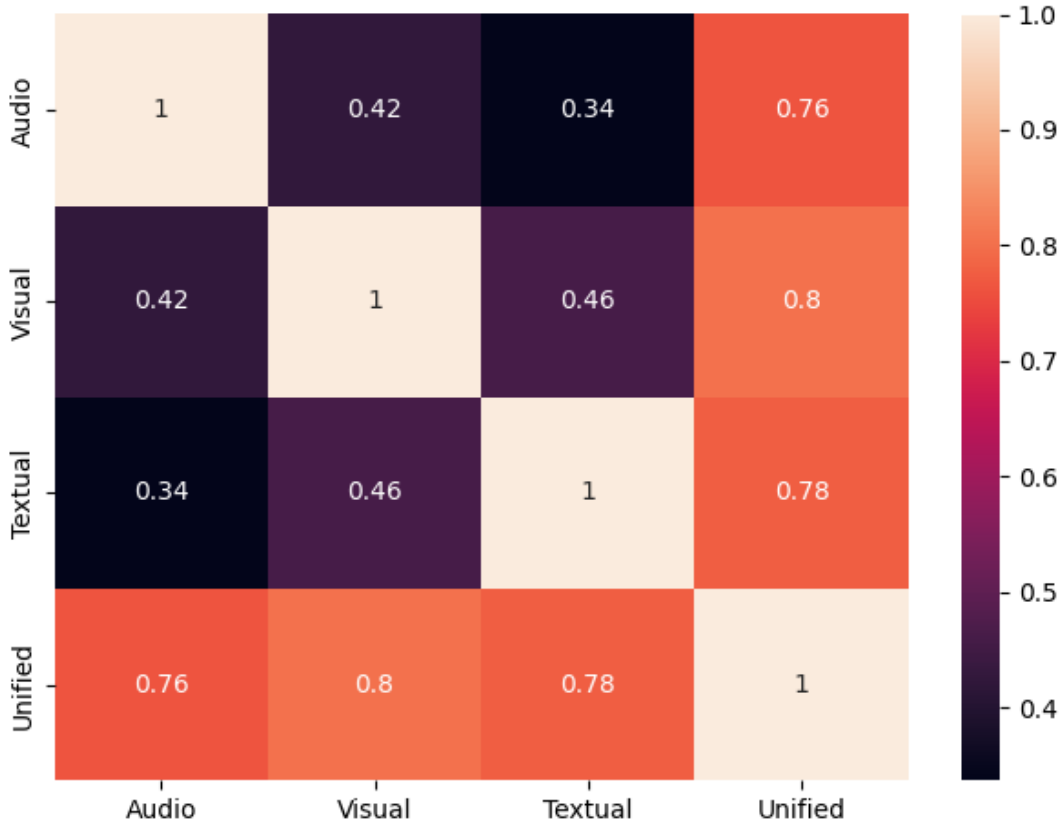


Figure 6.3: Correlation matrix between output values for the unified model as well as each of it's constituent models.

strongly with its constituent models, even though these models do not correlate strongly with each other. The weakest correlation is between the audio model and TASTE. This may be because these are the modalities most susceptible to noisy data. It is unlikely that noisy false positive/negative highlight cues would be correlated across these modalities, and thus when one makes a false prediction, the other is unlikely to do.

Finally, Figure 6.4 demonstrates the time series prediction for a selected video from the test set. The grey columns represent the ground truth labels, and the pink sections of the highlight signal for each model denote segments selected as highlights. A pink line segment which is not aligned with a grey column denotes false positive predictions, whereas ground truth segments where the signal is not above the threshold, i.e. pink, represent false-negative predictions. From this, it is clear how segments rated very highly from one of the constituent models are often not represented in the unified signal if the other models do not also rate that segment highly. For example, there is a segment at about 3100 seconds which

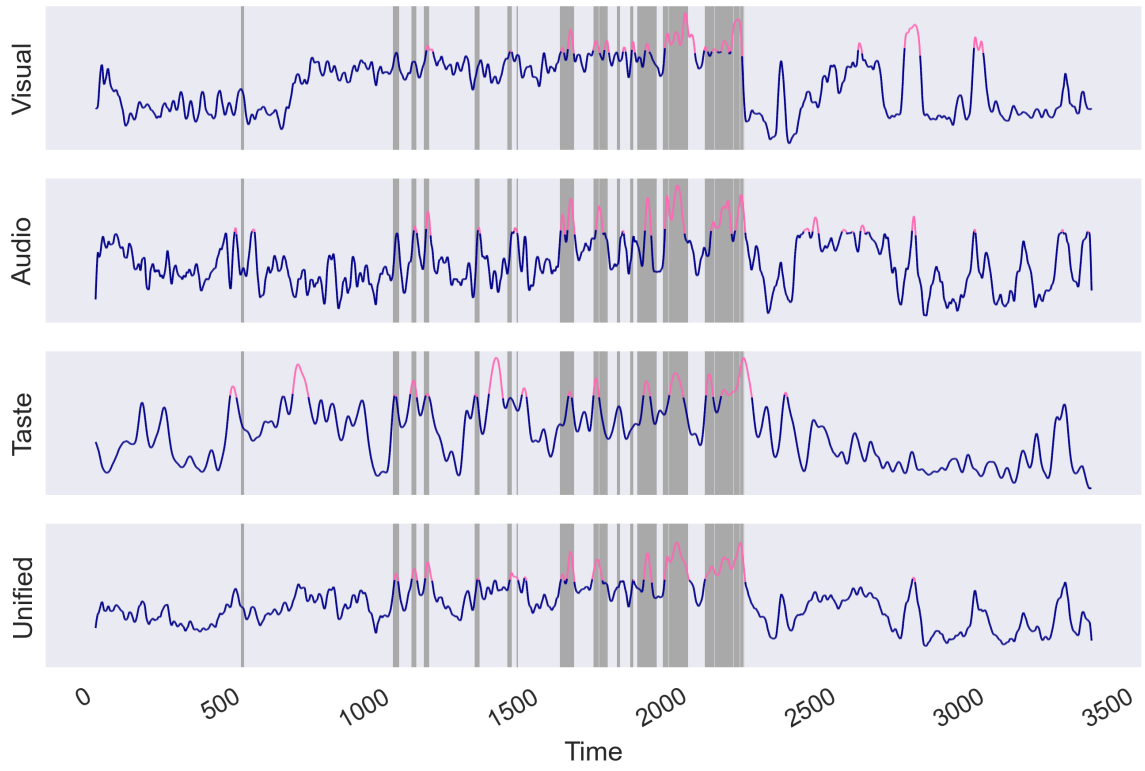


Figure 6.4: Time Series Highlight Prediction for each modality for a selected game.

the visual model ranks highly. However, the other models do not, and thus it is correctly not selected as a highlight in the unified model. This may be a replay segment, e.g. during post-match analysis, which would appear visually to be a highlight but perhaps lacks excited commentator speech or audience reaction. Incorrectly identifying these moments was an issue with the original Auto-Highlight model because, in general, it lacked context and thus could not determine if a segment was part of the match or surrounding analysis. Incorporating chat data into the model enables it to model context through audience reaction. For example, the audience is less likely to react strongly to a replay than to a live game event, even if visually, these segments are identical.

6.2.6 Conclusion

Model fusion for audio-visual-textual modalities is an effective technique for building highlight detection models, significantly outperforming all other models presented in the thesis. This is a somewhat surprising result. It is expected that modelling highlight signals from the video feed alongside audience reactions would improve results. However, model fusion

is a straightforward mechanism, and thus the extent to which it improves performance is unexpected. While modelling of all potential modalities has only been explored preliminarily in this study, in many ways, the huge performance increase is one of the more important findings for this thesis. That said, it is likely that more complex fusion mechanisms discussed in the next section will yield even more fruitful results.

6.3 Future Directions for Highlight Detection

The work discussed in the previous section marks the last research study contributing to this thesis. However, this body of research by no means exhaustively explores this challenge. Thus, it is pertinent to discuss potential future directions for modelling the interesting moments in livestreams.

6.3.1 Improved Fusion Approaches

The main mechanisms for fusion in this work have been concatenation and aggregation. Concatenation is used in early and feature fusion models because it is a straightforward way to join a set of latent feature vectors that does not lose any information. However, concatenation also fails to generate new information, i.e. it does not model combinations of features from different modalities. Aggregation is helpful for model fusion as, unlike concatenation, it reduces multiple features to a single value, which is required when fusing model outputs because the result of the aggregation becomes the output for the model. However, like concatenation, this is a simplistic mechanism. That said, it also benefits from being easily applied, e.g. in settings which are not strictly supervised, i.e. Chapter 4.

Chapter 3 explored using Tensor-Train fusion, which sought to more intelligently fuse latent features by modelling higher-order interactions between features from different modalities. While it appeared to yield performance benefits, at least in some tasks, it added some amount of complexity, both in terms of the number of weights in the model and the challenges of developing models which optimally utilise the Tensor-Train layer, given the number of hyper-parameters involved. While this was an interesting application of Tensor-Train for multimodal fusion, it was also clear from this study that exploring the most fruitful multimodal fusion techniques is a complex issue outside this thesis's scope.

Therefore, future work into intelligent fusion mechanisms will likely improve modelling

results or contribute to a better understanding of how best to fuse multimodal data. For instance, one of the most significant challenges is modelling context, which is handled via a series of transformer blocks in the TASTE model. This may well prove to be a fruitful technique for modelling and fusing across audio-visual and textual data. Transformers could be utilised in a supervised setting by replacing the stack of messages transformer from the TASTE model with a transformer that includes input from audio-visual data. The transformer would allow the model to naturally learn the inherent time lag between audio-visual and chat data and attend to cues at different time stamps in the various modalities. The key challenge here would be ensuring that the sampling rate for all modalities is compatible.

6.3.2 Enhanced Audio-Visual Modelling

The Autohighlight audio-visual hybrid fusion model performs well compared to other state-of-the-art ranking models. However, like them, it selects highlights based only on short segments and does not explicitly model events outside these segments, which could provide additional context. While applying smoothing introduces consideration for the context of a given sample, this is not modelled in the original ranking model. It is challenging to model context in a weakly-supervised setting. While the context in a given segment is present in the broadcast dataset, as, in the original unedited broadcast, this is not the case for the highlight dataset because it has been edited. Careful consideration is required to include context data into these models to improve prediction, as doing so will likely yield positive results. There are no clear solutions to this problem; thus, while improving this is likely to improve audio-visual modelling significantly, it is also likely to represent an extensive research undertaking.

Additionally, the pretrained feature extraction used for audio-visual models is generic feature extractors, e.g. trained on kinetics. While these models produce a set of features useful for prediction, a pretrained feature extractor focused on esports titles, either on a per-title basis or genre-based, could improve results. There are several methods for achieving this, but one currently popular approach is self-supervised learning. For instance, it may be helpful to utilise a training scheme similar to the successful 'jigsaw' scheme but using temporal stacks of frames to build latent feature extractors that describe esports-specific features, e.g., user interface elements. This would then, in theory, improve the ranking/classification models by providing them with more relevant information. Custom feature extraction would also allow

the highlight detection research to ensure the sampling size is such that cross audio-visual-textual modelling through a transformer is possible.

6.3.3 Highlight Detection Though the Lens of Computational Creativity

It is arguable that the highlight models in Chapters 4 and 5 create highlights as a pastiche of the video editors who made the original highlight videos in the training data. While this aids in creating human-like highlight videos, it fails to fully explore the potential for highlight video generation using computational creativity techniques. For example, this thesis primarily considers highlight detection as essentially a classification problem where there exist some ground truth highlight labels. However, highlight generation is subjective. Therefore this ground truth may not be the optimal set of highlight segments or several distinct sets of optimal highlight clips may exist. Many areas of computational creativity seek to allow the algorithm to find exciting moments without strictly training them to detect human-style moments, something akin to the Novelty based method in Chapter 3. Pursuing this avenue of research may yield surprising, unique, but still highly entertaining highlight videos.

One potential application of this would be to utilise the active community of esports fans on social media sites to improve detection models. This could work by posting generated highlight clips to social media channels and judging the responses it receives as a form of fine-tuning. This would allow the model to transform from predicting the most similar segments to the source highlight videos to predicting the segments most enjoyed by esports fans. This could also lead to personalised highlight systems where viewers are shown a highlight video conditioned on the kind of content the particular viewer has historically enjoyed.

Alternatively, techniques like those presented in this thesis may serve as a jumping-off point for human-in-the-loop co-creativity systems where the system aids a human editor rather than replacing them. Such a system could work by, e.g. presenting the editor with a set of segments the model ranks highly. The human editor could then select, dismiss, or edit segments based on their judgement. This would speed up the human editing process and fine-tune the model over time.

6.3.4 Other Titles

Finally, this thesis studies only two titles. *Playerunkown's Battlegrounds* was deliberately chosen because it was expected that the novelty approach would be particularly suitable for the game due to its properties, i.e. largely low-intensity gameplay with moments of high-intensity action which correlates strongly to a highlight moment occurring. The rest of the thesis focuses on *League of Legends* because it is popular and thus has abundant data. In particular *League of Legends* was a very attractive game because it has many worldwide professional leagues making it very straightforward to study many game-driven settings. Furthermore, *League of Legends* has a community of highlight video editors, which enabled the semi-supervised approach used for Chapter 4.

However, numerous other games are commonly streamed, e.g. the games included in the TwitchChat dataset. These games and the streamers playing them would benefit from automatic highlight detection. Further study into whether the techniques that have proved successful in this thesis generalise to other games would be helpful. In general, deep learning methods are designed to be general models. However, there may be games with robust and low noise highlight signals, e.g. a particular message that appears on the screen strongly correlated with highlight moments. Equally, there may be games with nosier highlight signals, for example, a strategy game where a particular event could be a highlight or not based on the game's history. Therefore, developing a dataset containing various games would help understand the kinds of games with which these techniques work well.

Chapter 7

Conclusions

7.1 Introduction

The research chapters present this thesis's contributions in exploring deep learning methods for highlight detection. This chapter, therefore, concludes the thesis. To achieve this, Section 7.2 revisits all the research goals presented in Chapter 1 and evaluates them. The limitations of this research are then discussed in Section 7.3. Finally, 7.4 presents the closing remarks.

7.2 Revisiting Research Goals

The below sections explore whether the original research goals have been achieved during this thesis.

Explore dataset properties, preprocessing, and labelling required to successfully model the most exciting moments in livestreamed esports broadcasts.

Data has consistently been an issue with building models in this domain. This is mainly due to the lack of supervised training and evaluation data. Several solutions were explored. For instance, Section 3.2 approached the highlight problem via unsupervised novelty modelling. This approach was deliberately chosen because the labelling issue could be sidestepped if unsupervised approaches were suitable. While this study was successful with the studied game, it did not generalise to other games, and thus fully unsupervised approaches are not suitable for generalised methods for highlight detection. The other study into personality-

driven highlight detection used supervised learning but, like Section 3.2, did so via proxy signals for highlights. While the training was partially successful, especially for predicting game events, the requirement for annotation resulted in a smaller than ideal dataset. A significant effort is required for successful supervised learning, either human if hand-annotating or computationally if frame-matching. Weakly-supervised learning, a compromise between supervised and unsupervised methods, proved very useful for audio-visual highlight detection. However, *League of Legends* is one of the most popular esports titles. This means there is an abundance of this crowd-sourced highlight data available. There may not be such an abundance of this data for other esports titles, especially those aiming to grow their audience.

Text data from the audience chat system presented significant challenges. Not only are there labelling obstacles as with audio-visual data, but the text chat format is unique, as outlined in Chapter 5. Significant study into understanding the data format, for example, the temporal nature and the extensive use of platform-specific tokens, was required before a predictive model could be applied. This culminated in the release of the TwitchChat dataset [165]. Once this was performed, the TwitchChat dataset could be used for model pretraining. Fortunately, there also existed an easily accessible dataset of chat data annotated for highlight moments, making supervised highlight models for this data relatively straightforward, at least in terms of preparing the input data.

Develop methods for utilising video data to perform audio-visual highlight detection suitable for both personality-driven livestreams and game-driven livestreams.

While each audio-visual study in this thesis used slightly different approaches, there was some commonality between the architectures. For instance, CNNs were very useful for modelling the visual aspects of streams, either using 2D CNNs on a per-frame basis or 3D CNNs on video segments. It is less clear what the best method for audio modelling is; for example, frequency-based approaches have shown promise, as did directly modelling raw data with a CNN.

While personality-driven and game-driven broadcasts are fundamentally constructed from the same data sources, they share minor differences. The primary difference is that personality-driven livestreams tend to contain additional visual modalities in the form of the

streamer's webcam. This was handled by adding extra visual inputs to the models, which was easily achieved because the feature extraction techniques were trained separately, and the fusion methods can accept an arbitrary number of input modalities.

Therefore, while the data-label formulation was different for all three audio-visual studies, the primary techniques, especially visual techniques, share a common theme. Thus, Deep Learning techniques clearly work well for modelling this complex data.

Develop methods for utilising chat to perform Natural Language based highlight detection.

In contrast to the audio-visual modelling discussed above, chat modelling was significantly complex. This was primarily due to the unknown qualities of livestream chat data compared to more typical prosaic English text, for example, the large quantity of non-standard tokens precluding the use of any pretrained vectorisation techniques. Thus a significant amount of research time was spent investigating these properties and developing the TwitchChat dataset, which, while extremely valuable to this research, did not directly contribute highlight detection methods.

Fortunately, natural language techniques have been the subject of considerable research attention, and one technique, transformers, has become the prevailing method for modelling text. The TASTE architecture discussed in Chapter 5 builds upon the standard Transformer encoder architecture to incorporate the temporal nature of livestream. It is clear from the performance of the TASTE model that this transformer-based approach is potent for predicting highlights from text data. Therefore, while this area was not explored through many studies, as with audio-visual data, it is clear that at least one technique, transformers, is successful.

Explore the potential for multi-modal models using Audio, Visual and Language data.

Chapter 6 explores the potential for fusing audio, visual and textual data into a single prediction and outlining potential future directions for this research. It is evident from this study that fusing across video cues, i.e. audio-visual data, and audience cues, i.e. textual data, has a hugely positive effect on predictive performance. Not only does the unified model outperform Auto-Highlight and TASTE in average performance metrics, but it is also

appears more robust than non-visual models. Overall this research goal can be considered to have been met, although only lightly.

Fusing across all modalities is a complex problem that requires satisfactorily modelling the individual modalities. This is not a simple task, especially considering the challenges with handling livestream data, hence the focus in this thesis on modelling the modalities well versus modelling all modalities simultaneously. There was only space within this thesis to explore preliminary techniques for audio-visual-textual fusion. The model fusion mechanism used for this study is relatively straightforward. Therefore, given that such a straightforward technique provides such a considerable performance improvement, it is likely that the more complex techniques outlined in the future direction section will yield even better predictive models.

7.3 Limitations

There are several limitations to this work. Firstly, the majority of this thesis has only applied the highlight detection techniques to one game, *League of Legends*. Therefore, the approaches discussed in this thesis may not generalise well to other games. This was already the case with the unsupervised novelty detection approach applied to the game *PlayerUnknown's Battlegrounds*, hence abandoning that line of research after the initial study. However, the other methods used in this thesis were chosen because they are well known for being generalisable to different problems. Therefore, while techniques in this thesis have not been tested on other games, there is some expectation that they will work. *League of Legends* was chosen because of the abundance of available livestream data, allowing the training processes to leverage large-scale datasets. Therefore, the primary concern concerning the generalisability of the work in this thesis, bar novelty detection, is not the models themselves but the availability of suitable training data.

One challenge in evaluating highlight detection models is that no apparent metrics neatly explain the models' performance. Precision-based metrics are standard in the literature, which is understandable given that including dull moments is harmful to deploying a highlight detection model. However, highlight detection in game-driven esports broadcasts also requires the match's narrative to be captured. Therefore, arguably in this context, Recall is equally important, and thus F1 Score has been adopted in several studies. However, none

of these metrics satisfactorily measure a model's effectiveness at highlight detection because they fail to capture how realistic an edited video is. For instance, consider a video with a long ground-truth highlight segment. If most of this is correctly detected as a highlight, but several short sub-segments are incorrectly classified as non-highlight, the model may score highly on the above metrics yet fail to produce a realistic video. One potential solution is to perform human-based evaluation of the produced highlight models. This would ensure that the videos are evaluated in a scenario similar to the use case for such techniques. However, human evaluation was not used for these studies because doing so is not straightforward. There is a significant overhead in performing human-focused evaluation, especially considering the subjective nature of highlight generation. Therefore, while the commonly used metrics are not ideal compared to human evaluation, they are significantly more practical.

Finally, Chapter 6 explored preliminary experimentation into models which use audio, visual, and chat data. It is clear from these results that using all modalities is very powerful. However, the study does not explore fusion methods in-depth, especially considering using audio-visual data with chat data. Furthermore, as outlined in the same chapter, there is potential for more complex fusion methods, especially those which can more intelligently fuse chat data with the other modalities. However, these approaches are not covered in this thesis. Therefore, while the models in the thesis are largely successful, especially the audio, visual, and chat model fusion model, there is potential for significant improvement.

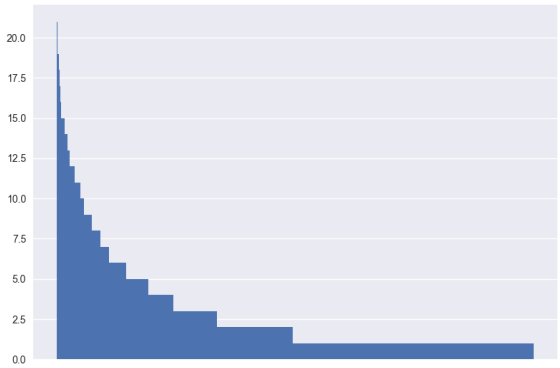
7.4 Closing Remarks

This thesis has explored using a suite of techniques for detecting the most exciting moments in a livestream broadcast under the guise of 'highlight detection'. It is clear from the research that deep, multi-modal models are effective at modelling these moments, at least in the games studied. Furthermore, the generalist characteristics of these models suggest that they are likely to perform similarly for other games. In the course of this research, the value of both the broadcast itself, i.e. the audio-visual data, and the reaction from the audience, i.e. textual chat data, became evident. Therefore, the most successful approach utilises all available modalities, i.e. Chapter 6.

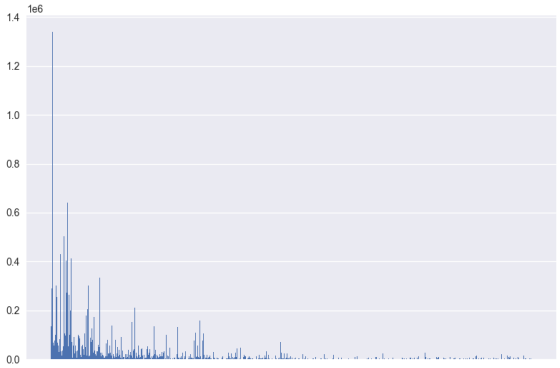
Appendices

Appendix A

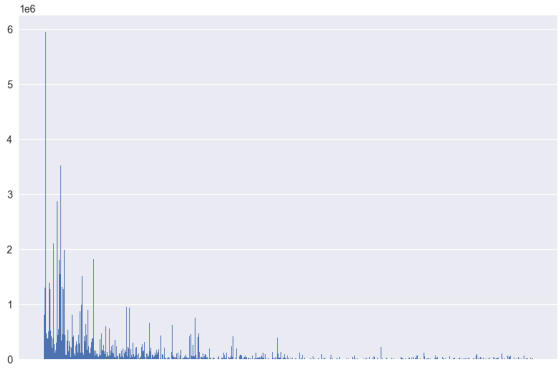
Additional Plots for the TwitchChat Data-Set



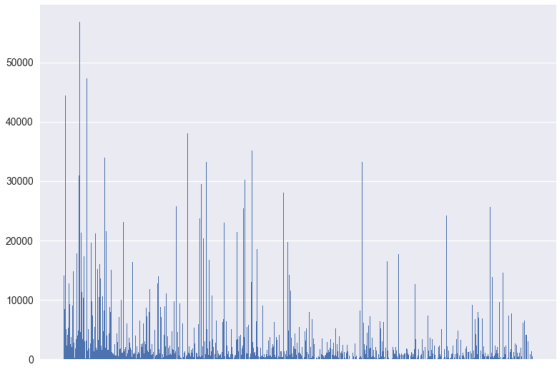
(a) Document Count



(b) Message Count

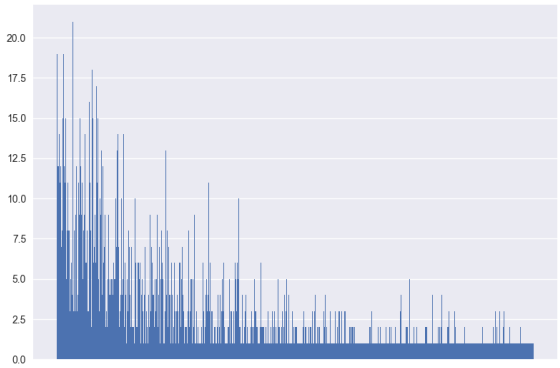


(c) Token Count



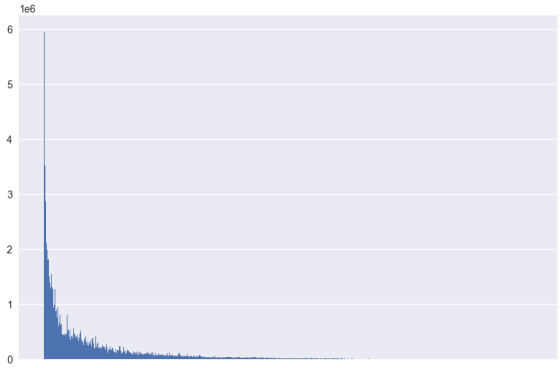
(d) Median Viewer Count at Start of Stream

Figure A.1: Features for each streamer sorted by document count.

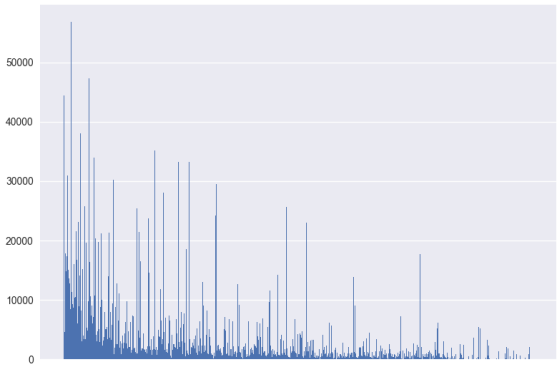


(a) Document Count

(b) Message Count

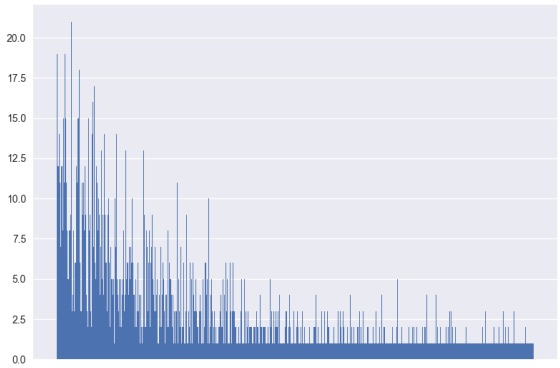


(c) Token Count



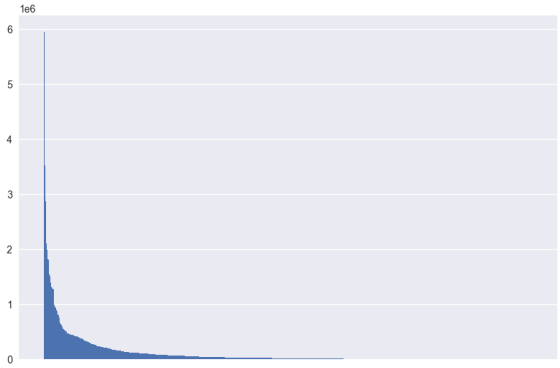
(d) Median Viewer Count at Start of Stream

Figure A.2: Features for each streamer sorted by message count.

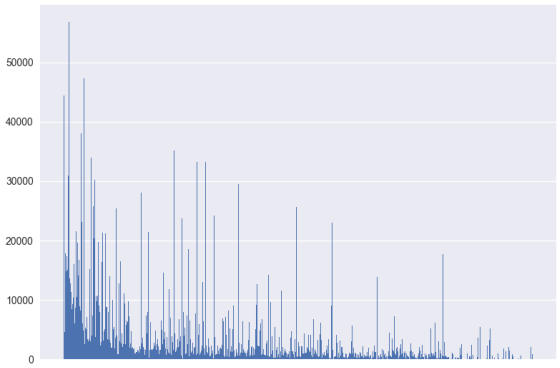


(a) Document Count

(b) Message Count

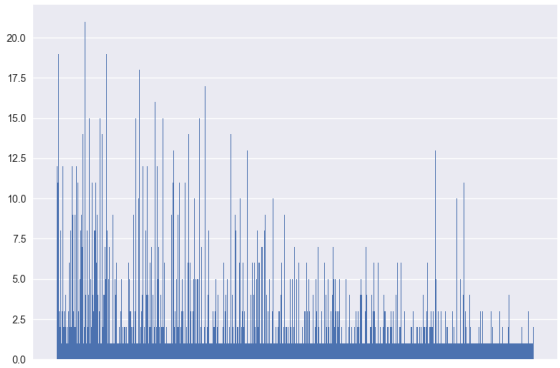


(c) Token Count

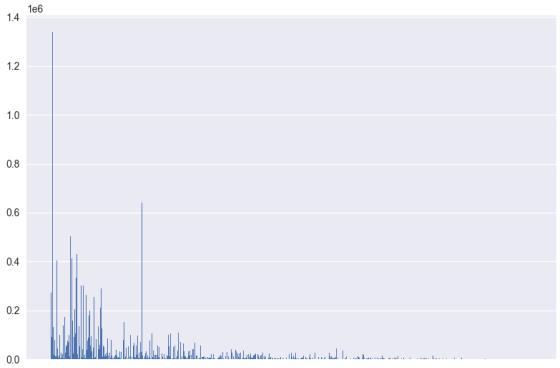


(d) Median Viewer Count at Start of Stream

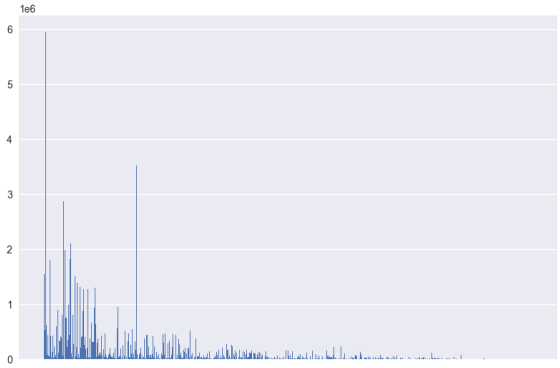
Figure A.3: Features for each streamer sorted by token count.



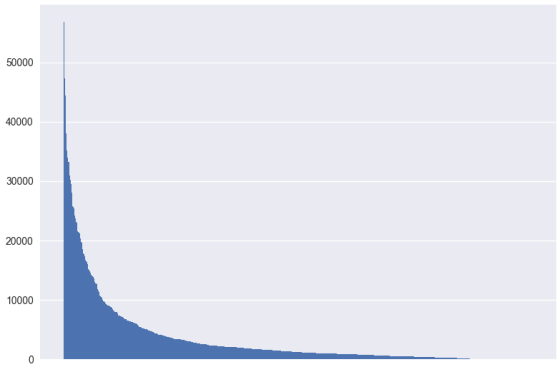
(a) Document Count



(b) Message Count

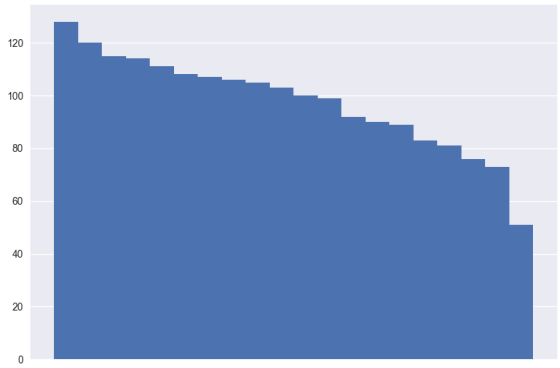


(c) Token Count

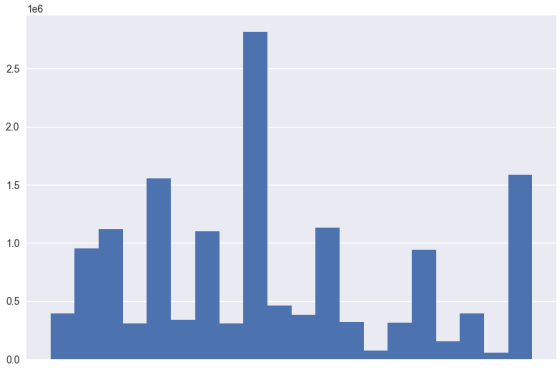


(d) Median Viewer Count at Start of Stream

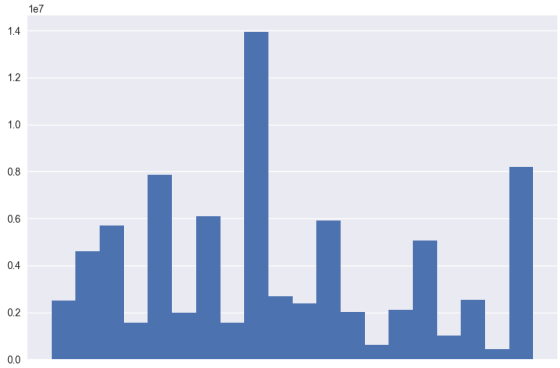
Figure A.4: Features for each streamer sorted by median viewer count.



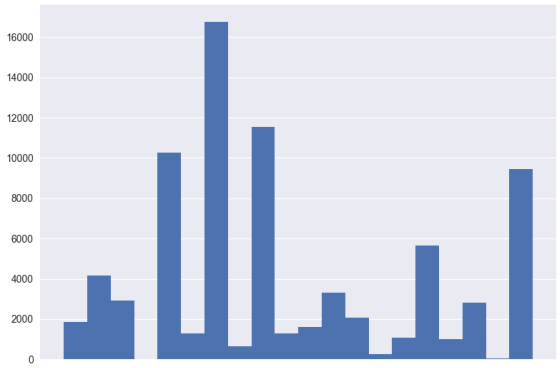
(a) Document Count



(b) Message Count

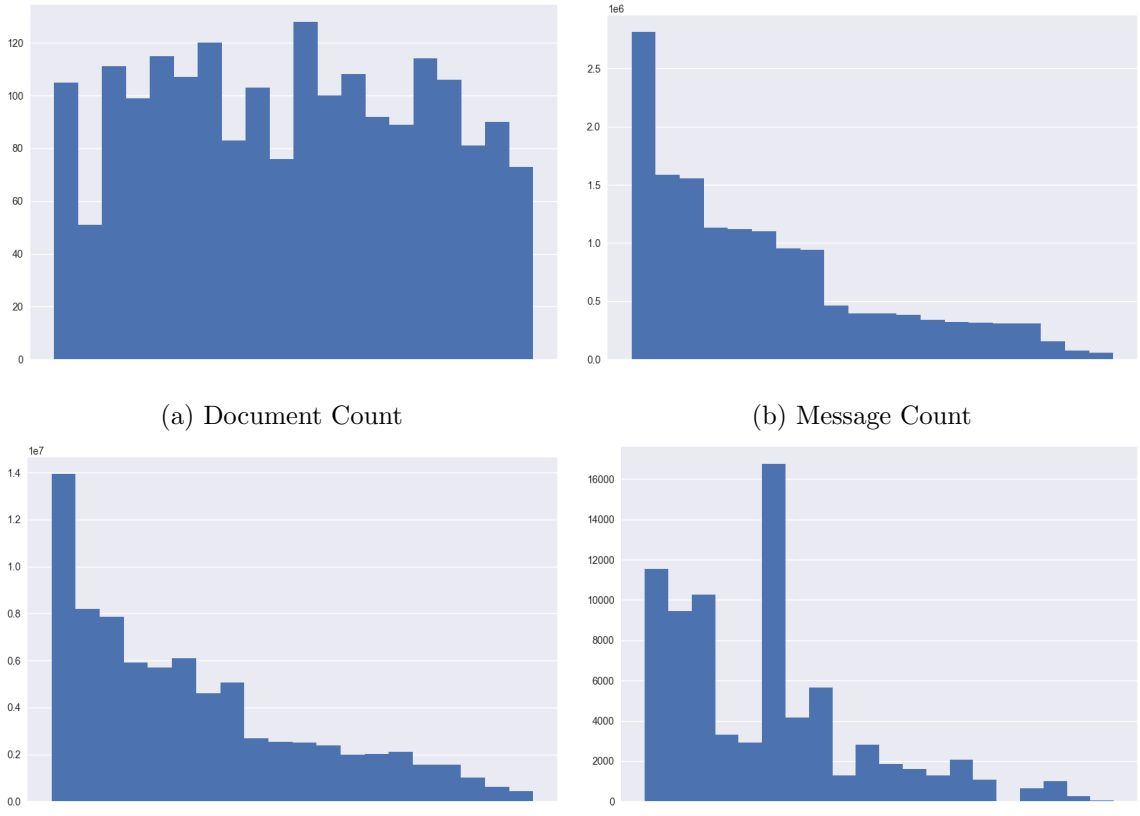


(c) Token Count



(d) Median Viewer Count at Start of Stream

Figure A.5: Features for each game sorted by document count.



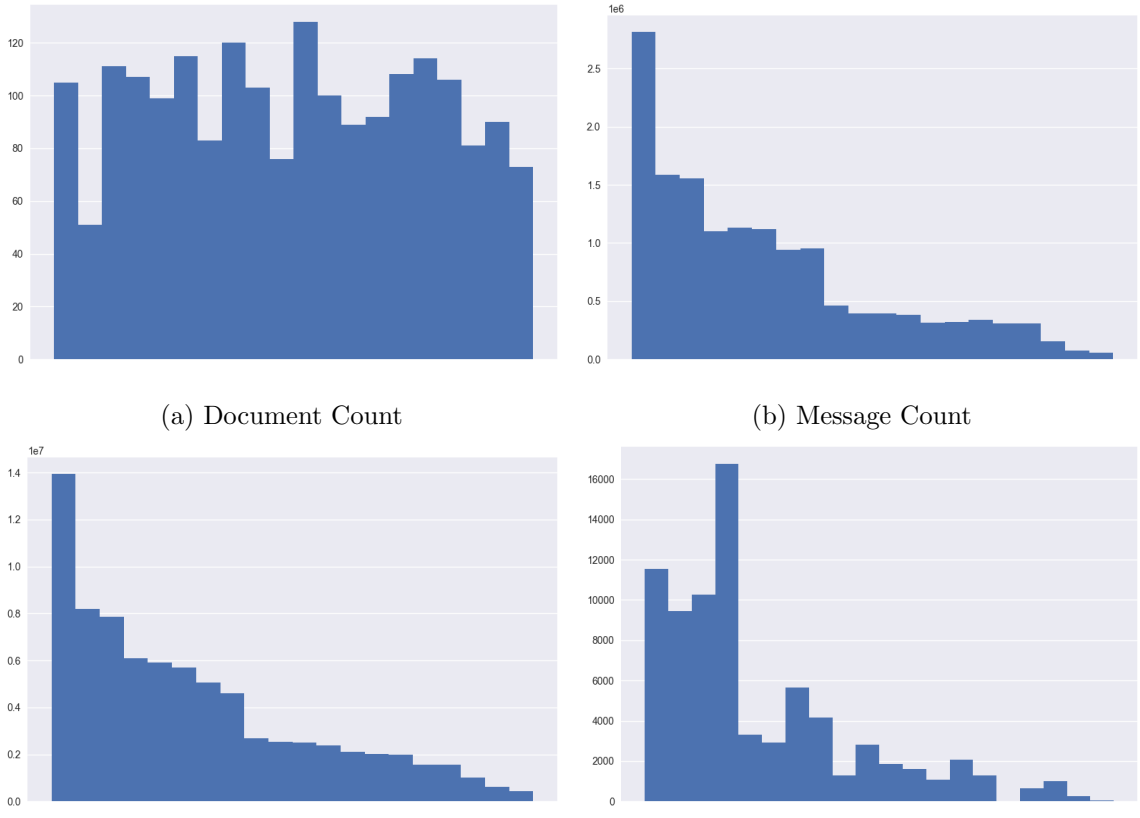
(a) Document Count

(b) Message Count

(c) Token Count

(d) Median Viewer Count at Start of Stream

Figure A.6: Features for each game sorted by message count.



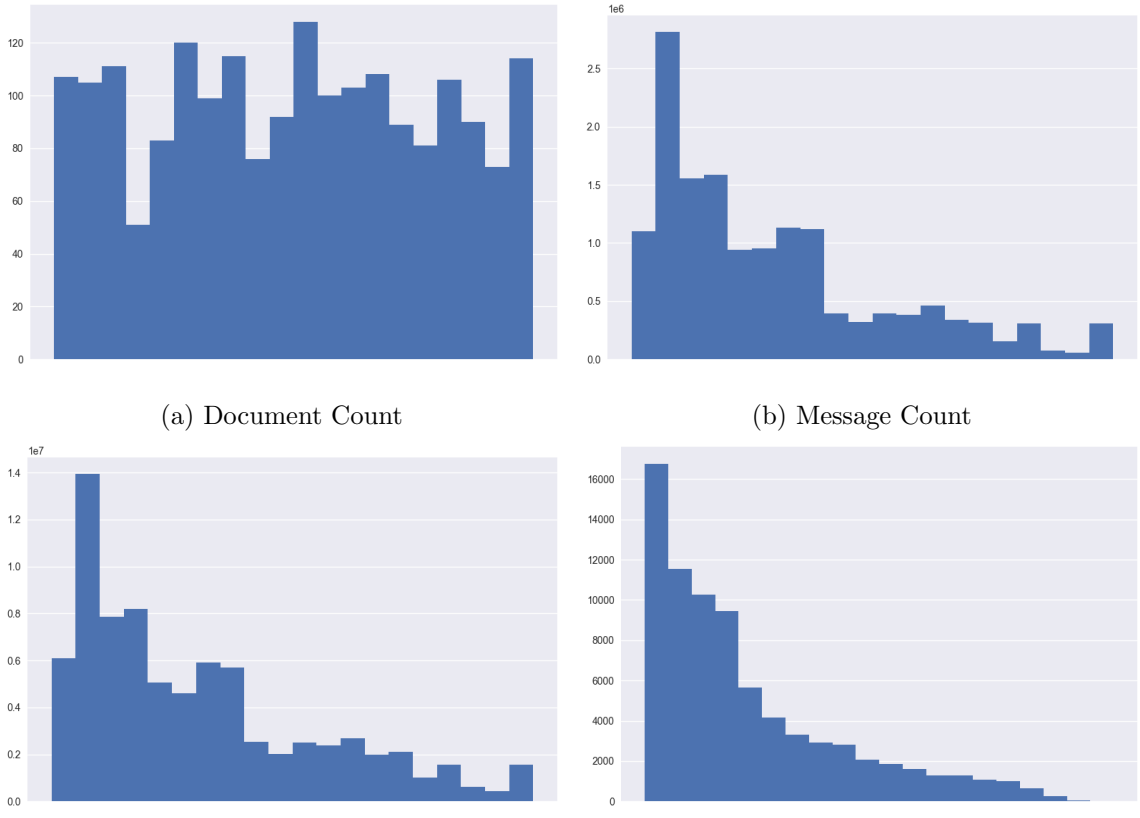
(a) Document Count

(b) Message Count

(c) Token Count

(d) Median Viewer Count at Start of Stream

Figure A.7: Features for each game sorted by token count.



(a) Document Count

(b) Message Count

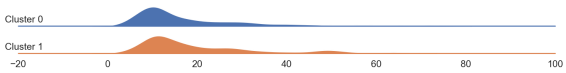
(c) Token Count

(d) Median Viewer Count at Start of Stream

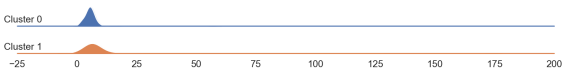
Figure A.8: Features for each game sorted by median viewer count.

Appendix B

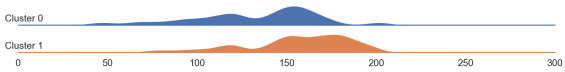
KDE Plots for the Word Vector Models



(a) Median Chat Speed

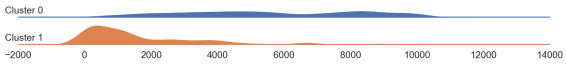


(b) Median Message Length

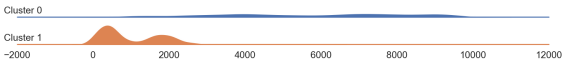


(c) Median Streamer Rank

(d) Median tf-idf

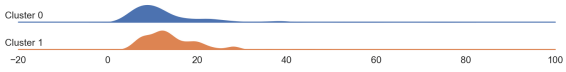


(e) Document Frequency Rank

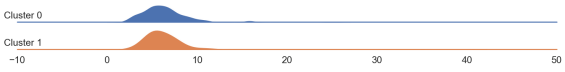


(f) Token Frequency Rank

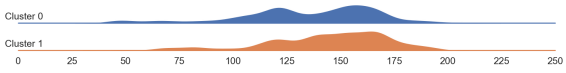
Figure B.1: KDE plots comparing the distributions of the 100 tokens closest to the centre of each cluster for DW-SGNS.



(a) Median Chat Speed

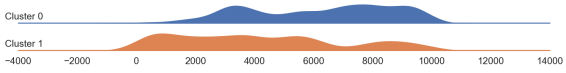


(b) Median Message Length

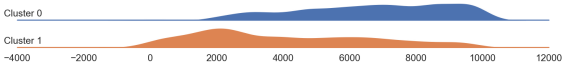


(c) Median Streamer Rank

(d) Median tf-idf

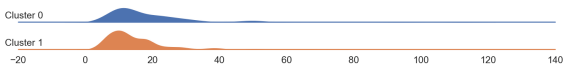


(e) Document Frequency Rank



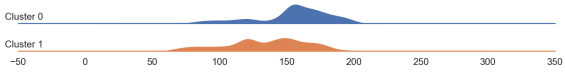
(f) Token Frequency Rank

Figure B.2: KDE plots comparing the distributions of the 100 tokens closest to the centre of each cluster for 2T-SGNS.

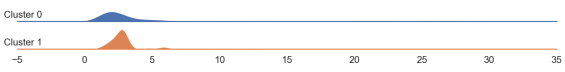


(a) Median Chat Speed

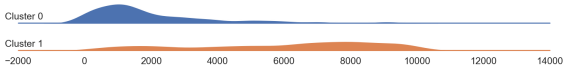
(b) Median Message Length



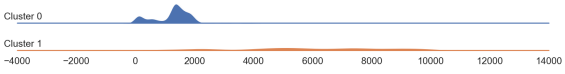
(c) Median Streamer Rank



(d) Median tf-idf



(e) Document Frequency Rank



(f) Token Frequency Rank

Figure B.3: KDE plots comparing the distributions of the 100 tokens closest to the centre of each cluster for 5T-SGNS.

List of References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Agarap, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).

[3] Aghdam, H. H., and Heravi, E. J. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*, 1st ed. Springer Publishing Company, Incorporated, 2017.

[4] Alghamdi, R., and Alfalqi, K. A survey of topic modeling in text mining. *International Journal of Advanced Computer Science and Applications 6*, 1 (2015).

[5] Alvernaz, S., and Togelius, J. Autoencoder-augmented neuroevolution for visual doom playing. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)* (Aug 2017), pp. 1–8.

[6] Alwassel, H., Mahajan, D., Korbar, B., Torresani, L., Ghanem, B., and Tran, D. Self-supervised learning by cross-modal audio-video clustering. In *Advances in Neural Information Processing Systems 34*. Curran Associates, Inc., 2020, pp. 1–18.

[7] Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. *J. Mach. Learn. Res. 15*, 1 (Jan. 2014), 2773–2832.

[8] Aouani, H., and Ben Ayed, Y. Emotion recognition in speech using MFCC with SVM, DSVM and auto-encoder. *2018 4th International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2018* (2018), 1–5.

[9] Asteriadis, S., Karpouzis, K., Shaker, N., and Yannakakis, G. N. Towards detecting clusters of players using visual and gameplay behavioral cues. *Procedia Computer Science 15* (2012), 140–147.

[10] Ayata, D., Yaslan, Y., and Kamaşak, M. Emotion recognition via random forest and galvanic skin response: Comparison of time based feature sets, window sizes and wavelet approaches. In *2016 Medical Technologies National Congress (TIPTEKNO)* (2016), pp. 1–4.

[11] Bachorowski, J.-A. Vocal expression and perception of emotion. *Current Directions in Psychological Science 8*, 2 (1999), 53–57.

[12] Bakarov, A. A Survey of Word Embeddings Evaluation Methods. *arXiv:1801.09536 [cs]* (Jan. 2018). arXiv: 1801.09536.

[13] Banerjee, A., and Dave, R. N. Validating clusters using the hopkins statistic. In *2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No.04CH37542)* (2004), vol. 1, pp. 149–153 vol.1.

[14] Barbieri, F., Espinosa Anke, L., Ballesteros, M., Soler, J., and Saggion, H. Towards the Understanding of Gaming Audiences by Modeling Twitch Emotes. In *Proceedings of the 3rd Workshop on Noisy User-generated Text* (Copenhagen, Denmark, 2017), Association for Computational Linguistics, pp. 11–20.

[15] Barbieri, F., Espinosa-Anke, L., Ballesteros, M., Soler-Company, J., and Saggion, H. Towards the Understanding of Gaming Audiences by Modeling Twitch Emotes. *Proceedings of the 3rd Workshop on Noisy User-generated Text* (2017), 11–20.

[16] Belova, A., He, W., and Zhong, Z. E-sports talent scouting based on multimodal twitch stream data. *CoRR abs/1907.01615* (2019).

[17] Birjali, M., Kasri, M., and Beni-Hssane, A. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems 226* (2021), 107134.

[18] Blashki, K., and Nichol, S. Game geek's goss: linguistic creativity in young males within an online university forum (94/\/\3 933k'5 9055oneone). *Australian journal of emerging technologies and society 3*, 2 (Jan. 2005), 71–80.

[19] Blom, P. M., Bakkes, S., Tan, C. T., Whiteson, S., Roijers, D., Valenti, R., and Gevers, T. Towards personalised gaming via facial expression recognition. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (2014), AIIDE'14, AAAI Press, pp. 30–36.

[20] Bonometti, V., Ringer, C., Hall, M., Wade, A., and Drachen, A. Modelling early user-game interactions for joint estimation of survival time and churn probability. In *2019 IEEE Conference on Games (CoG)* (2019), pp. 1–8.

[21] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[22] Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower Provost, E., Kim, S., Chang, J., Lee, S., and Narayanan, S. Iemocap: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation 42* (12 2008), 335–359.

[23] Busso, C., Deng, Z., Yildirim, S., Bulut, M., Lee, C. M., Kazemzadeh, A., Lee, S., Neumann, U., and Narayanan, S. Analysis of emotion recognition using facial expressions, speech and multimodal information. *Proceedings of the 6th international conference on Multimodal interfaces - ICMI '04* (2004), 205.

[24] Busso, C., Lee, S., and Narayanan, S. Analysis of emotionally salient aspects of fundamental frequency for emotion detection. *IEEE Transactions on Audio, Speech and Language Processing 17*, 4 (2009), 582–596.

[25] Cao, N. T., That, A. H. T., and Choi, H. I. An effective facial expression recognition approach for intelligent game systems. *International Journal of Computational Vision and Robotics 6*, 3 (2016), 223.

[26] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (Los Alamitos, CA, USA, may 2018), IEEE Computer Society, pp. 67–74.

[27] Carnein, M., Assenmacher, D., and Trautmann, H. Stream clustering of chat messages with applications to twitch streams. In *Advances in Conceptual Modeling* (Cham, 2017), S. de Cesare and U. Frank, Eds., Springer International Publishing, pp. 79–88.

[28] Carreira, J., and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 4724–4733.

[29] Chang, J., and Scherer, S. Learning representations of emotional speech with deep convolutional generative adversarial networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2017), 2746–2750.

[30] Chapelle, O., Schölkopf, B., and Zien, A., Eds. *Semi-Supervised Learning*. The MIT Press, 2006.

[31] Chen, W., and McDuff, D. Deepphys: Video-based physiological measurement using convolutional attention networks. *CoRR abs/1805.07888* (2018).

[32] Cheung, G., and Huang, J. Starcraft from the stands: Understanding the game spectator. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2011), CHI '11, ACM, pp. 763–772.

[33] Chiu, C. Y., Lin, P. C., Li, S. Y., Tsai, T. H., and Tsai, Y. L. Tagging webcast text in baseball videos by video segmentation and text alignment. *IEEE Transactions on Circuits and Systems for Video Technology 22*, 7 (2012), 999–1013.

[34] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 103–111.

[35] Chollet, F. keras, 2015.

[36] Chollet, F., et al. Keras, 2015.

[37] Chu, W. T., and Chou, Y. C. Event detection and highlight detection of broadcasted game videos. *HCMC 2015 - Proceedings of the 2nd Workshop on Computational Models of Social Interactions: Human-Computer-Media Communication, co-located with ACM MM 2015* (2015), 1–8.

[38] Chu, W. T., and Chou, Y. C. On broadcasted game video analysis: event detection, highlight detection, and highlight forecast. *Multimedia Tools and Applications 76*, 7 (2017), 9735–9758.

[39] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.

[40] Deng, J., Cuadrado, F., Tyson, G., and Uhlig, S. Behind the game: Exploring the twitch streaming platform. In *2015 International Workshop on Network and Systems Support for Games (NetGames)* (Dec. 2015), pp. 1–6.

[41] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[42] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[43] Deng, J., Tyson, G., Cuadrado, F., and Uhlig, S. Internet Scale User-Generated Live Video Streaming: The Twitch Case. In *Passive and Active Measurement* (Cham, 2017), M. A. Kaafar, S. Uhlig, and J. Amann, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 60–71.

[44] Deng, J., Zhang, Z., Eyben, F., and Schuller, B. Autoencoder-based unsupervised domain adaptation for speech emotion recognition. *IEEE Signal Processing Letters 21*, 9 (2014), 1068–1072.

[45] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (Oct. 2018). arXiv: 1810.04805.

[46] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186.

[47] Diwanji, V., Reed, A., Ferchaud, A., Seibert, J., Weinbrecht, V., and Sellers, N. Don't just watch, join in: Exploring information behavior and copresence on twitch. *Computers in Human Behavior 105* (2020), 106221.

[48] Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1422–1430.

[49] Doyran, M., Schimmel, A., Baki, P., Ergin, K., Türkmen, B., Akdag Salah, A., Bakkes, S., Kaya, H., Poppe, R., and Salah, A. Mumbai: multi-person, multimodal board game affect and interaction analysis dataset. *Journal on Multimodal User Interfaces 15* (02 2021).

[50] Ekman, P. Facial expression and emotion. *American Psychologist 48*, 4 (1993), 384–392.

[51] Ekman, P. Basic emotions. In *Handbook of Cognition and Emotion*, T. Dalgleish and M. Power, Eds. John Wiley & Sons, Ltd., 1999, ch. 3.

[52] Elkan, C., and Noto, K. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2008), KDD '08, Association for Computing Machinery, p. 213–220.

[53] Farnebäck, G. Two-frame motion estimation based on polynomial expansion. In *Image Analysis* (Berlin, Heidelberg, 2003), J. Bigun and T. Gustavsson, Eds., Springer Berlin Heidelberg, pp. 363–370.

[54] Fei, W., Ye, X., Sun, Z., Huang, Y., Zhang, X., and Shang, S. Research on speech emotion recognition based on deep auto-encoder. In *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)* (June 2016), pp. 308–312.

[55] Feichtenhofer, C., Pinz, A., and Zisserman, A. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[56] Flores-Saviaga, C., Hammer, J., Flores, J. P., Seering, J., Reeves, S., and Savage, S. Audience and streamer participation at scale on twitch. In *Proceedings of the 30th ACM Conference on Hypertext and Social Media* (New York, NY, USA, 2019), HT '19, Association for Computing Machinery, p. 277–278.

[57] Ford, C., Gardner, D., Horgan, L. E., Liu, C., tsaasan, a. m., Nardi, B., and Rickman, J. Chat speed op pogchamp: Practices of coherence in massive twitch chat. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI EA '17, Association for Computing Machinery, p. 858–871.

[58] Fu, C.-Y., Lee, J., Bansal, M., and Berg, A. Video highlight prediction using audience chat reactions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 972–978.

[59] Gan, Q., Wang, S., Hao, L., and Ji, Q. A Multimodal Deep Regression Bayesian Network for Affective Video Content Analyses. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 5123–5132.

[60] Garcia-Garcia, J. M., Penichet, V. M. R., and Lozano, M. D. Emotion detection: A technology review. In *Proceedings of the XVIII International Conference on Human Computer Interaction* (New York, NY, USA, 2017), Interacción '17, Association for Computing Machinery.

[61] Gholamalinezhad, H., and Khosravi, H. Pooling methods in deep neural networks, a review. *CoRR abs/2009.07485* (2020).

[62] Ghosh, S., Laksana, E., Morency, L., and Scherer, S. Learning representations of affect from speech. *CoRR abs/1511.04747* (2015).

[63] Giannakopoulos, T., Makris, A., Kosmopoulos, D., Perantonis, S., and Theodoridis, S. Audio-visual fusion for detecting violent scenes in videos. In *Artificial Intelligence: Theories, Models and Applications* (Berlin, Heidelberg, 2010), S. Konstantopoulos, S. Perantonis, V. Karkaletsis, C. D. Spyropoulos, and G. Vouros, Eds., Springer Berlin Heidelberg, pp. 91–100.

[64] Gravier, G., Demarty, C., Bredin, H., Ionescu, B., Boididou, C., Dellandréa, E., Choi, J., Riegler, M., Sutcliffe, R. F. E., Szöke, I., Jones, G. J. F., and Larson, M. A., Eds. *Working Notes Proceedings of the MediaEval 2016 Workshop, Hilversum, The Netherlands, October 20-21, 2016* (2016), vol. 1739 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[65] Gruber, N., and Jockisch, A. Are gru cells more specific and lstm cells more sensitive in motive classification of text? *Frontiers in Artificial Intelligence 3* (2020).

[66] Guzdial, M., and Riedl, M. Game level generation from gameplay videos. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 12*, 1 (Jun. 2021), 44–50.

[67] Gygli, M., Song, Y., and Cao, L. Video2gif: Automatic generation of animated gifs from video. *CoRR abs/1605.04850* (2016).

[68] Hamilton, W. A., Garretson, O., and Kerne, A. Streaming on twitch: Fostering participatory communities of play within live mixed media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2014), CHI '14, Association for Computing Machinery, p. 1315–1324.

[69] Han, H.-K., Huang, Y.-C., and Chen, C. C. A deep learning model for extracting live streaming video highlights using audience messages. In *Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference* (New York, NY, USA, 2019), AICCC 2019, Association for Computing Machinery, p. 75–81.

[70] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'io, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[71] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 770–778.

[72] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), pp. 131–135.

[73] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation 9* (12 1997), 1735–80.

[74] Hong, F.-T., Huang, X., Li, W.-H., and Zheng, W.-S. Mini-net: Multiple instance ranking network for video highlight detection. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII* (Berlin, Heidelberg, 2020), Springer-Verlag, p. 345–360.

[75] Huang, C. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer: Generating music with long-term structure. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019), OpenReview.net.

[76] HUANG, Z.-w., and MAO, Q.-r. Speech emotion recognition with unsupervised feature learning. *Frontiers of Information Technology & Electronic Engineering 16*, 5 (2015), 358–366.

[77] Hussain, S. A., and Balushi, A. S. A. A. A real time face emotion classification and recognition using deep learning model. *Journal of Physics: Conference Series 1432*, 1 (jan 2020), 012087.

[78] Hutto, C., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media 8*, 1 (May 2014), 216–225.

[79] Ilse, M., Tomczak, J. M., and Welling, M. Attention-based deep multiple instance learning. *CoRR arXiv:1802.04712* (2018).

[80] Ishii, R., Fujimaki, K., and Thawonmas, R. Fighting-game gameplay generation using highlight cues. *IEEE Transactions on Games* (2021), 1–1.

[81] Ishii, R., Ito, S., Thawonmas, R., and Harada, T. A fighting game ai using highlight cues for generation of entertaining gameplay. In *2019 IEEE Conference on Games (CoG)* (2019), pp. 1–6.

[82] Jain, N., Kumar, S., Kumar, A., Shamsolmoali, P., and Zareapoor, M. Hybrid deep neural networks for face emotion recognition. *Pattern Recognition Letters 115* (2018), 101–106. Multimodal Fusion for Pattern Recognition.

[83] Jenny, S. E., Manning, R. D., Keiper, M. C., and Olrich, T. W. Virtual(ly) athletes: Where esports fit within the definition of "sport". *Quest 69*, 1 (2017), 1–18.

[84] Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 1 (2013), 221–231.

[85] Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Chen, z., Nguyen, P., Pang, R., Lopez Moreno, I., and Wu, Y. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 4480–4490.

[86] Jiang, R., Qu, C., Wang, J., Wang, C., and Zheng, Y. Towards extracting highlights from recorded live videos: An implicit crowdsourcing approach. In *2020*

IEEE 36th International Conference on Data Engineering (ICDE) (2020), pp. 1810–1813.

[87] Johnson, M. R., and Woodcock, J. 'it's like the gold rush': the lives and careers of professional video game streamers on twitch.tv. *Information, Communication & Society 22*, 3 (2019), 336–351.

[88] Kahou, S. E., Bouthillier, X., Lamblin, P., Gulcehre, C., Michalski, V., Konda, K., Jean, S., Froumenty, P., Dauphin, Y., Boulanger-Lewandowski, N., Chandias Ferrari, R., Mirza, M., Warde-Farley, D., Courville, A., Vincent, P., Memisevic, R., Pal, C., and Bengio, Y. Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces 10*, 2 (Jun 2016), 99–111.

[89] Karpouzis, K., Yannakakis, G. N., Shaker, N., and Asteriadis, S. The platformer experience dataset. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)* (Sept 2015), pp. 712–718.

[90] Katsaggelos, A. K., Bahaadini, S., and Molina, R. Audiovisual Fusion: Challenges and New Approaches. *Proceedings of the IEEE 103*, 9 (2015), 1635–1653.

[91] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. The kinetics human action video dataset. *CoRR abs/1705.06950* (2017).

[92] Kaytoue, M., Silva, A., and Cerf, L. Watch me playing, i am a professional: a first study on video game live streaming. *Proceedings of the 21st international conference companion on World Wide Web* (2012), 1181–1188.

[93] Kaytoue, M., Silva, A., Cerf, L., Meira Jr, W., and Raïssi, C. Watch me playing, I am a professional: a first study on video game live streaming. In *Proceedings of the 21st International Conference on World Wide Web* (2012), ACM.

[94] Kessous, L., Castellano, G., and Caridakis, G. Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis. *Journal on Multimodal User Interfaces 3*, 1 (2010), 33–48.

[95] Kim, J., Wohn, D. Y., and Cha, M. Understanding and identifying the use of emotes in toxic chat on twitch. *Online Social Networks and Media 27* (2022), 100180.

[96] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014).

[97] Kishore, K. V. K., and Satish, P. K. Emotion recognition in speech using mfcc and wavelet features. In *2013 3rd IEEE International Advance Computing Conference (IACC)* (Feb 2013), pp. 842–847.

[98] Kobs, K., Zehe, A., Bernstetter, A., Chibane, J., Pfister, J., Tritscher, J., and Hotho, A. Emote-controlled: Obtaining implicit viewer feedback through emote-based sentiment analysis on comments of popular twitch.tv channels. *Trans. Soc. Comput. 3*, 2 (apr 2020).

[99] Kollias, D., Tagaris, A., and Stafylopatis, A. On line emotion detection using retrainable deep neural networks. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), pp. 1–8.

[100] Konstantinova, K., Bulygin, D., Okopny, P., and Musabirov, I. Online communication of esports viewers: Topic modeling approach. In *Advances in Computer Entertainment Technology - 14th International Conference, ACE 2017, London, UK, December 14-16, 2017, Proceedings* (2017), A. D. Cheok, M. Inami, and T. Romão, Eds., vol. 10714 of *Lecture Notes in Computer Science*, Springer, pp. 608–613.

[101] Kossaifi, J., Lipton, Z. C., Kolbeinsson, A., Khanna, A., Furlanello, T., and Anandkumar, A. Tensor regression networks. *Journal of Machine Learning Research 21*, 123 (2020), 1–21.

[102] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[103] Kunanusont, K., Lucas, S. M., and Pérez-Liébana, D. General video game ai: Learning from screen capture. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (2017), pp. 2078–2085.

[104] Lai, K., Yu, F. X., Chen, M., and Chang, S. Video event detection by inferring temporal instance labels. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2251–2258.

[105] Lalitha, S., Geyasruti, D., Narayanan, R., and Shravani, M. Emotion Detection Using MFCC and Cepstrum Features. *Procedia Computer Science 70* (2015), 29–35.

[106] Larson, M. A., Ionescu, B., Sjöberg, M., Anguera, X., Poignant, J., Riegler, M., Eskevich, M., Hauff, C., Sutcliffe, R. F. E., Jones, G. J. F., Yang, Y., Soleymani, M., and Papadopoulos, S., Eds. *Working Notes Proceedings of the MediaEval 2015 Workshop, Wurzen, Germany, September 14-15, 2015* (2015), vol. 1436 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[107] Lewis, J., Trinh, P., and Kirsh, D. A corpus analysis of strategy video game play in starcraft: Brood war. *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (2011), 687–692.

[108] Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, J. Learning from noisy labels with distillation. *CoRR abs/1703.02391* (2017).

[109] Liaw, C., and Dai, B. Live stream highlight detection using chat messages. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)* (2020), pp. 328–332.

[110] Lin, J. C., Wu, C. H., and Wei, W. L. A probabilistic fusion strategy for audio-visual emotion recognition of sparse and noisy data. *ICOT 2013 - 1st International Conference on Orange Technologies* (2013), 278–281.

[111] Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G. N., and Togelius, J. Deep learning for procedural content generation. *Neural Computing and Applications 33*, 1 (Jan 2021), 19–37.

[112] Liu, T., and Tao, D. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 3 (2016), 447–461.

[113] Ljubešić, N., and Fišer, D. A Global Analysis of Emoji Usage. In *Proceedings of the 10th Web as Corpus Workshop* (Berlin, 2016), Association for Computational Linguistics, pp. 82–89.

[114] Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[115] Loper, E., and Bird, S. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 2002), ETMTNLP '02, Association for Computational Linguistics, pp. 63–70.

[116] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops* (June 2010), pp. 94–101.

[117] Makantasis, K., Liapis, A., and Yannakakis, G. N. From pixels to affect: A study on games and player experience. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)* (2019), pp. 1–7.

[118] Makantasis, K., Liapis, A., and Yannakakis, G. N. The pixels and sounds of emotion: General-purpose representations of arousal in games. *IEEE Transactions on Affective Computing* (2021), 1–1.

[119] Martínez, H. P. H., and Yannakakis, G. Mining multimodal sequential patterns: a case study on affect detection. *... International Conference on Multimodal ...* (2011), 3–10.

[120] Mavromoustakos-Blom, P., Kosa, M., Bakkes, S., and Spronck, P. Correlating facial expressions and subjective player experiences in competitive hearthstone. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021* (New York, NY, USA, 2021), FDG'21, Association for Computing Machinery.

[121] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (2015), vol. 8.

[122] McInnes, L., Healy, J., Saul, N., and Grossberger, L. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software 3*, 29 (Sept. 2018), 861.

[123] McKeown, G., Valstar, M., Cowie, R., Pantic, M., and Schroder, M. The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent. *IEEE Transactions on Affective Computing 3*, 1 (Jan 2012), 5–17.

[124] Melhart, D., Liapis, A., and Yannakakis, G. N. The affect game annotation (AGAIN) dataset. *CoRR abs/2104.02643* (2021).

[125] Mendi, E., Clemente, H. B., and Bayrak, C. Sports video summarization based on motion analysis. *Computers & Electrical Engineering 39*, 3 (2013), 790 – 796. Special issue on Image and Video Processing Special issue on Recent Trends in Communications and Signal Processing.

[126] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings* (2013), Y. Bengio and Y. LeCun, Eds.

[127] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]* (Jan. 2013). arXiv: 1301.3781.

[128] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Red Hook, NY, USA, 2013), NIPS'13, Curran Associates Inc., p. 3111–3119.

[129] Miller, G. A. Wordnet: A lexical database for english. *Commun. ACM 38*, 11 (Nov. 1995), 39–41.

[130] Mimno, D., and Thompson, L. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, 2017), Association for Computational Linguistics, pp. 2873–2878.

[131] Miyakoshi, Y., and Kato, S. Facial emotion detection considering partial occlusion of face using bayesian network. In *2011 IEEE Symposium on Computers Informatics* (2011), pp. 96–101.

[132] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (Feb 2015), 529–533.

[133] Moodley, T., and van der Haar, D. Scene recognition using alexnet to recognize significant events within cricket game footage. In *Computer Vision and Graphics* (Cham, 2020), L. J. Chmielewski, R. Kozera, and A. Orłowski, Eds., Springer International Publishing, pp. 98–109.

[134] Musabirov, I., Bulygin, D., Okopny, P., and Konstantinova, K. Between an arena and a sports bar: Online chats of esports spectators. *CoRR abs/1801.02862* (2018).

[135] Nakandala, S., Ciampaglia, G., Su, N., and Ahn, Y.-Y. Gendered conversation in a social game-streaming platform. *Proceedings of the International AAAI Conference on Web and Social Media 11*, 1 (May 2017), 162–171.

[136] Nanni, L., Maguolo, G., Brahnam, S., and Paci, M. An ensemble of convolutional neural networks for audio classification. *Applied Sciences 11*, 13 (2021).

[137] Narayan, S. The generalized sigmoid activation function: Competitive supervised learning. *Information Sciences 99*, 1 (1997), 69–82.

[138] Nascimento, G., Ribeiro, M., Cerf, L., Cesário, N., Kaytoue, M., Raïssi, C., Vasconcelos, T., and Meira, W. Modeling and analyzing the video game live-streaming community. In *2014 9th Latin American Web Congress* (2014), pp. 1–9.

[139] Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1196–1204.

[140] Nematzadeh, A., Ciampaglia, G., Ahn, Y.-Y., and Flammini, A. Information overload in group communication: From conversation to cacophony in the twitch chat. *Royal Society Open Science 6* (10 2016).

[141] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. Multimodal deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (USA, 2011), ICML'11, Omnipress, pp. 689–696.

[142] Nguyen, N., and Yoshitaka, A. Soccer video summarization based on cinematography and motion analysis. *Multimedia Signal Processing (MMSP)* (2014), 1–6.

[143] Nicolaou, M. A., Gunes, H., and Pantic, M. Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space. *IEEE Transactions on Affective Computing 2*, 2 (2011), 92–105.

[144] Nicolaou, M. A., Panagakis, Y., Zafeiriou, S., and Pantic, M. Robust canonical correlation analysis: Audio-visual fusion for learning continuous interest. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2014), 1522–1526.

[145] Nicolaou, M. A., and Ringer, C. Streaming behaviour: Live streaming as a paradigm for multi-view analysis of emotional and social signals. In *Twitch Workshop, 13th International Conference on the Foundations of Digital Games* (2018), FDG '18.

[146] Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. Tensorizing neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Cambridge, MA, USA, 2015), NIPS'15, MIT Press, pp. 442–450.

[147] Olejniczak, J. A Linguistic Study Of Language Variety Used On Twitch . Tv : Desriptive And Corpus-based Approaches. *Redefining Community in Intercultural Context 4*, May (2015), 329–334.

[148] Oseledets, I. Tensor-train decomposition. *SIAM Journal on Scientific Computing 33*, 5 (2011), 2295–2317.

[149] Panagakis, Y., Kossaifi, J., Chrysos, G. G., Oldfield, J., Nicolaou, M. A., Anandkumar, A., and Zafeiriou, S. Tensor methods in computer vision and deep learning. *Proceedings of the IEEE 109*, 5 (2021), 863–890.

[150] Pankaj Malhotra, Lovekesh Vig, G. S., and Agarwal1, P. Long short term memory networks for anomaly detection in time series. In *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (2015).

[151] Park, J., Barash, V., Fink, C., and Cha, M. Emoticon style: Interpreting differences in emoticons across cultures. *Proceedings of the International AAAI Conference on Web and Social Media 7*, 1 (Aug. 2021), 466–475.

[152] Parkhi, O. M., Vedaldi, A., and Zisserman, A. Deep Face Recognition. *Proceedings of the British Machine Vision Conference 2015*, Section 3 (2015), 41.1–41.12.

[153] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[154] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[155] Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. A review of novelty detection. *Signal Processing 99* (2014), 215–249.

[156] Ping, Q. Video recommendation using crowdsourced time-sync comments. In *Proceedings of the 12th ACM Conference on Recommender Systems* (New York, NY, USA, 2018), RecSys '18, Association for Computing Machinery, p. 568–572.

[157] Ping, Q., and Chen, C. Video highlights detection and summarization with lag-calibration based on concept-emotion mapping of crowdsourced time-sync comments. In *Proceedings of the Workshop on New Frontiers in Summarization* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 1–11.

[158] Pires, K., and Simon, G. Youtube live and twitch: A tour of user-generated live streaming systems. In *Proceedings of the 6th ACM Multimedia Systems Conference* (New York, NY, USA, 2015), MMSys '15, Association for Computing Machinery, p. 225–230.

[159] Poria, S., Cambria, E., Bajpai, R., and Hussain, A. A review of affective computing : From unimodal analysis to multimodal fusion. *Information Fusion 37* (2017), 98–125.

[160] Poyane, R. Toxic communication during streams on twitch.tv. the case of dota 2. In *Proceedings of the 22nd International Academic Mindtrek Conference* (New York, NY, USA, 2018), Mindtrek '18, Association for Computing Machinery, p. 262–265.

[161] Rajesh, K. M., and Naveenkumar, M. A robust method for face recognition and face emotion detection system using support vector machines. In *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)* (2016), pp. 1–5.

[162] Recktenwald, D. Toward a transcription and analysis of live streaming on Twitch. *Journal of Pragmatics* (Feb. 2017).

[163] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping, 2015.

[164] Ren, R., Jose, J., and Yin, H. Affective sports highlight detection. *European Signal Processing Conference*, Eusipco (2007), 728–732.

[165] Ringer, C., Nicolaou, M., and Walker, J. Twitchchat: A dataset for exploring livestream chat. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 16*, 1 (Oct. 2020), 259–265.

[166] Ringer, C., and Nicolaou, M. A. Deep unsupervised multi-view detection of video game stream highlights. In *International Conference on Foundations of Digital Games* (2018), ACM.

[167] Ringer, C., and Nicolaou, M. A. Streaming behaviour: Livestreaming as a paradigm for analysis of emotional and social signals. In *2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)* (2019), pp. 182–185.

[168] Ringer, C., Nicolaou, M. A., and Walker, J. A. Autohighlight: Highlight detection in league of legends esports broadcasts via crowd-sourced data. *Machine Learning with Applications* (2021 (Under review)).

[169] Ringer, C., Walker, J. A., and Nicolaou, M. A. Deep unsupervised multi-view detection of video game stream highlights. In *2019 IEEE Conference on Games* (2019), IEEE.

[170] Rioult, F., Métivier, J.-P., Helleu, B., Scelles, N., and Durand, C. Mining Tracks of Competitive Video Games. *AASRI Procedia 8*, Secs (2014), 82–87.

[171] Risi, S., and Togelius, J. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence 2*, 8 (Aug 2020), 428–436.

[172] Robinson, R., Rubin, Z., Segura, E. M., and Isbister, K. All the feels: Designing a tool that reveals streamers' biometrics to spectators. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (New York, NY, USA, 2017), FDG '17, ACM, pp. 36:1–36:6.

[173] Rosenblatt, F. The perceptron - a perceiving and recognizing automaton. Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[174] Russell, J. A. A Circumplex Model of Affect. *Journal of Personality and Social Psychology 39*, 6 (1980), 1161–1178.

[175] Salunke, V. V., and Patil, C. A new approach for automatic face emotion recognition and classification based on deep networks. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (2017), pp. 1–5.

[176] Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (Lisbon, Portugal, 2015), Association for Computational Linguistics, pp. 298–307.

[177] Schuller, B., Valstar, M., Eyben, F., McKeown, G., Cowie, R., and Pantic, M. AVEC 2011 - The first international audio/visual emotion challenge. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6975 LNCS*, PART 2 (2011), 415–424.

[178] Schuller, B., Valster, M., Eyben, F., Cowie, R., and Pantic, M. AVEC 2012: the continuous audio/visual emotion challenge. *Proc. 14th Int'l Conf. Multimodal Interaction Workshops* (2012), 449–456.

[179] Schuller, B. W. Speech Emotion Recognition: Two Decades in a Nutshell, Benchmarks, and Ongoing Trends. *Communications of the ACM* (2018), 90–99.

[180] Schwan, J., Ghaleb, E., Hortal, E., and Asteriadis, S. High-performance and lightweight real-time deep face emotion recognition. In *2017 12th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)* (2017), pp. 76–79.

[181] Seering, J., Kraut, R., and Dabbish, L. Shaping pro and anti-social behavior on twitch through moderation and example-setting. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (New York, NY, USA, 2017), CSCW '17, Association for Computing Machinery, p. 111–125.

[182] Sekhavat, Y. A., Roohi, S., Mohammadi, H. S., and Yannakakis, G. N. Play with one's feelings: A study on emotion awareness for player experience. *IEEE Transactions on Games 14*, 1 (2022), 3–12.

[183] Shah, M., Chakrabarti, C., and Spanias, A. A multi-modal approach to emotion recognition using undirected topic models. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (June 2014), pp. 754–757.

[184] Shaker, N., Asteriadis, S., Yannakakis, G. N., and Karpouzis, K. Fusing visual and behavioral cues for modeling user experience in games. *IEEE Transactions on Cybernetics 43*, 6 (2013), 1519–1531.

[185] Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 464–468.

[186] Simonyan, K., and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems* (2014).

[187] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (2015).

[188] Singh, G., and Singh, B. Feature based method for human facial emotion detection using optical flow based analysis. *Research Cell: An International Journal of Engineering Sciences 4* (Sep 2011).

[189] Smith, T., Obrist, M., and Wright, P. Live-streaming changes the (video) game. *Proceedings of the 11th european conference on Interactive TV and video - EuroITV '13* (2013), 131.

[190] Soleymani, M., Asghari-Esfeden, S., Pantic, M., and Fu, Y. Continuous emotion detection using eeg signals and facial expressions. In *2014 IEEE International Conference on Multimedia and Expo (ICME)* (2014), pp. 1–6.

[191] Song, Y. Real-time video highlights for yahoo esports. *CoRR abs/1611.08780* (2016).

[192] Spijkerman, R., and van der Haar, D. Video footage highlight detection in formula 1 through vehicle recognition with faster r-cnn trained on game footage. In *Computer Vision and Graphics* (Cham, 2020), L. J. Chmielewski, R. Kozera, and A. Orłowski, Eds., Springer International Publishing, pp. 176–187.

[193] Sridevi, M., and Kharde, M. Video summarization using highlight detection and pairwise deep ranking model. *Procedia Computer Science 167* (2020), 1839–1848. International Conference on Computational Intelligence and Data Science.

[194] Stadelmann, T., Amirian, M., Arabaci, I., Arnold, M., Duivesteijn, G. F., Elezi, I., Geiger, M., Lörwald, S., Meier, B. B., Rombach, K., and Tuggener, L. Deep learning in the wild. In *Artificial Neural Networks in Pattern Recognition* (Cham, 2018), L. Pancioni, F. Schwenker, and E. Trentin, Eds., Springer International Publishing, pp. 17–38.

[195] Strupp, S., Schmitz, N., and Berns, K. Visual-based emotion detection for natural man-machine interaction. In *KI 2008: Advances in Artificial Intelligence* (Berlin, Heidelberg, 2008), A. R. Dengel, K. Berns, T. M. Breuel, F. Bomarius, and T. R. Roth-Berghofer, Eds., Springer Berlin Heidelberg, pp. 356–363.

[196] Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. Training convolutional networks with noisy labels, 2015.

[197] Sun, M., Farhadi, A., and Seitz, S. Ranking domain-specific highlights by analyzing edited videos. In *Computer Vision – ECCV 2014* (Cham, 2014), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Springer International Publishing, pp. 787–802.

[198] Sun, Y., Ou, Z., Hu, W., and Zhang, Y. Excited commentator speech detection with unsupervised model adaptation for soccer highlight extraction. *ICALIP 2010 - 2010 International Conference on Audio, Language and Image Processing, Proceedings* (2010), 747–751.

[199] Szwoch, M., and Pieniazek, P. Facial emotion recognition using depth data. *2015 8th International Conference on Human System Interaction (HSI)*, D (2015), 271–277.

[200] Tan, C. T., Rosser, D., Bakkes, S., and Pisan, Y. A feasibility study in using facial expressions analysis to evaluate player experiences. *Proceedings of The 8th Australasian Conference on Interactive Entertainment Playing the System - IE '12* (2012), 1–10.

[201] Taylor, T. L. *Watch Me Play: Twitch and the Rise of Game Live Streaming*. Princeton University Press, Oct. 2018.

[202] Thomas, T., Domínguez, M., and Ptucha, R. Deep independent audio-visual affect analysis. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2017), pp. 1417–1421.

[203] Trivedi, C., Liapis, A., and Yannakakis, G. N. Contrastive learning of generalized game representations. In *2021 IEEE Conference on Games (CoG)* (2021), pp. 1–8.

[204] Turchenko, V., and Luczak, A. Creation of a deep convolutional auto-encoder in caffe. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (2017), vol. 2, pp. 651–659.

[205] Turney, P. D., and Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research 37* (Feb. 2010), 141–188. arXiv: 1003.1141.

[206] TwitchTracker. Twitch statistics & charts. https://twitchtracker.com/statistics, 2021.

[207] Twitch.tv. Hardware recommendations. https://www.twitch.tv/creatorcamp/en/setting-up-your-stream/hardware-recommendations/, 2021.

[208] Twitch.tv. Twitch press center. https://www.twitch.tv/p/press-center/, 2021.

[209] Tzirakis, P., Trigeorgis, G., Nicolaou, M., Schuller, B., and Zafeiriou, S. End-to-end multimodal emotion recognition using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing PP* (04 2017).

[210] Tzirakis, P., Trigeorgis, G., Nicolaou, M. A., Schuller, B. W., and Zafeiriou, S. End-to-end multimodal emotion recognition using deep neural networks. *CoRR abs/1704.08619* (2017).

[211] Vargha, A., and Delaney, H. D. A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics 25*, 2 (2000), 101–132.

[212] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 6000–6010.

[213] Wang, L., Sun, Z., Yao, W., Zhan, H., and Zhu, C. Unsupervised multi-stream highlight detection for the game "honor of kings", 2019.

[214] Wiseman, S., and Gould, S. J. J. Repurposing Emoji for Personalised Communication: Why [pizza slice emoji] means "I love you". In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (Montreal QC, Canada, 2018), ACM Press, pp. 1–10.

[215] Woodcock, J., and Johnson, M. R. The affective labor and performance of live streaming on twitch.tv. *Television & New Media 20*, 8 (2019), 813–823.

[216] Wulf, T., Schneider, F. M., and Beckert, S. Watching Players: An Exploration of Media Enjoyment on Twitch. *Games and Culture 15*, 3 (May 2020), 328–346.

[217] Wöllmer, M., Kaiser, M., Eyben, F., Schuller, B., and Rigoll, G. Lstm-modeling of continuous emotions in an audiovisual affect recognition framework. *Image and Vision Computing 31*, 2 (2013), 153 – 163. Affect Analysis In Continuous Input.

[218] Xiong, B., Kalantidis, Y., Ghadiyaram, D., and Grauman, K. Less is more: Learning highlight detection from video duration. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 1258–1267.

[219] Xu, D., Ricci, E., Yan, Y., Song, J., and Sebe, N. Learning deep representations of appearance and motion for anomalous event detection. In *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015* (2015), X. Xie, M. W. Jones, and G. K. L. Tam, Eds., BMVA Press, pp. 8.1–8.12.

[220] Xu, H., and Chua, T.-S. Fusion of AV features and external information sources for event detection in team sports video. *ACM Transactions on Multimedia Computing, Communications, and Applications 2*, 1 (2006), 44–67.

[221] Yang, C., Paliyawan, P., Thawonmas, R., and Harada, T. Tgif!: Selecting the most healing tnt by optical flow. In *Proceedings of the Symposium Interpretable*

AI for Well-being: Understanding Cognitive Bias and Social Embeddedness co-located with Association for the Advancement of Artificial Intelligence 2019 Spring Symposium (AAAI-Spring Symposium 2019), Stanford, CA, March 25-27, 2019 (2019), T. Kido and K. Takadama, Eds., vol. 2448 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[222] Yang, Y., Krompass, D., and Tresp, V. Tensor-train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, JMLR.org, p. 3891–3900.

[223] Yao, T., Mei, T., and Rui, Y. Highlight detection with pairwise deep ranking for first-person video summarization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 982–990.

[224] Yu, F. X., Liu, D., Kumar, S., Jebara, T., and Chang, S.-F. $\propto$ svm for learning with label proportions. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (2013), ICML'13, JMLR.org, p. III–504–III–512.

[225] Zadeh, A., Chen, M., Poria, S., Cambria, E., and Morency, L.-P. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 1103–1114.

[226] Zeiler, M. D. ADADELTA: an adaptive learning rate method. *CoRR abs/1212.5701* (2012).

[227] Zeng, Z., Pantic, M., Roisman, G. I., and Huang, T. S. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 1 (Jan 2009), 39–58.

[228] Zhang, Z., Weninger, F., Wöllmer, M., and Schuller, B. Unsupervised learning in cross-corpus acoustic emotion recognition. *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings* (2011), 523–528.

[229] Zhao, J., Mathieu, M., Goroshin, R., and LeCun, Y. Stacked What-Where Auto-encoders. *CoRR 1*, i (2015), 1–12.

[230] Zhu, G., Huang, Q., Xu, C., Xing, L., Gao, W., and Yao, H. Human behavior analysis for highlight ranking in broadcast racket sports video. *IEEE Transactions on Multimedia 9*, 6 (2007), 1167–1182.

[231] Zipf, G. K. *The psycho-biology of language.* Houghton, Mifflin, Oxford, England, 1935. Pages: ix, 336.