Automatic audience-informed video summarization of hockey broadcasts

Rick Wu
Supervised by Prof. Martin D. Levine

Master of Science (Electrical Engineering)
Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada

August 2020

A thesis submitted to McGill University in partial
fulfillment of the requirements of the degree of
Master of Science

Abstract

In the past decade, online video has risen to become one of the most common ways of sharing and consuming media. With so much video content readily-available, browsing or processing these data is difficult due to the long duration of a typical video. Video summaries, on the other hand, provide an effective way for viewers to enjoy the highlights of a video without having to watch the entire video.

In this work, we propose a system that *automatically* summarizes hockey broadcasts into highlight videos. We leverage the audience of a game—both the broadcasters and the crowd—to decide which shots should be included in the final highlight reel. We argue that since the game is directly sourced by the crowd and the broadcasters, our direct approach is more effective than similar techniques in the literature which rely on external annotators.

Our system accomplishes this by first extracting video and audio features using convolutional neural networks (CNN) and searching for key indicators of excitement (e.g. cheering or celebration). Lastly, the best highlights are dynamically selected under a set of constraints (such as output duration) and composited into the final summary. Our results demonstrate that this technique accurately extracts the most exciting moments of a hockey game while avoiding the bias from using predetermined metrics or external annotations.

Abrégé

Dans le passé, les vidéos en ligne ont pris de l'ampleur pour devenir une des façons les plus communes pour partager et consommer les médias. Avec autant de contenu vidéo disponible, la consultation ou le traitement de ces données est difficile vu la longueur d'un vidéo typique. Les résumés vidéos, d'un autre côté, fournissent une façon efficace pour les consommateurs d'apprécier les points importants d'un vidéo sans devoir le regarder au complet.

Dans ce travail, nous proposons un système qui résume automatiquement les émissions de hockey en un court vidéo des points forts. Nous tirons avantage du public d'un match, à la fois des spectateurs de l'émission et la foule, pour décider quelles parties devraient être incluses dans la bande des points forts. Nous affirmons que puisque la partie s'inspire directement de la foule et des spectateurs, notre approche directe est plus efficace que les techniques similaires dans la littérature qui se fit à des commentateurs extérieurs.

Notre système accomplit cela d'abord en extrayant les caractéristiques vidéo et audio en utilisant des réseaux neuronaux convolutionnels (RNC) et en cherchant pour des indicateurs clés d'enthousiasme (p.ex. des acclamations ou des célébrations). Enfin, les meilleurs points forts sont sélectionnés de façon dynamique selon un ensemble de contraintes (comme la durée du résultat) et sont montés pour former le résumé final. Nos résultats démontrent que cette technique extrait avec précision les meilleurs moments d'une partie de hockey tout en évitant les influences des métriques prédéterminées ou les commentateurs externes.

Acknowledgements

I would like to acknowledge my supervisor, Prof. Martin D. Levine, who has guided and supported me throughout the completion of this thesis. I am sincerely indebted to him for his mentorship, patience, knowledge, and tireless dedication towards his students.

As required by the McGill thesis preparation guidelines, I formally declare that Prof. Martin D. Levine has assisted me in the editorial process of writing this thesis (in addition to his guidance as my supervisor). Aside from guidance from a supervisor, I have not received external assistance in the writing, editing, collection of data, or any other relevant procedure.

Contents

1 Introduction

1.1 Challenges of video summarization

Video summarization is the task of condensing an input video into a summary that is composed of a subset of the original content. Such summaries provide an excellent way for viewers to enjoy or preview the contents of a video without the full-time commitment. As such, video summaries are used for television shows, sports broadcasts, advertisements, live-streaming, and video previews on streaming websites such as YouTube or Netflix. The process of creating these summaries is a manual and time-consuming process; as a result, automatic video summarization is an appealing alternative.

This task is challenging for several reasons, but most prominently because it is difficult to formulate in a manner which is readily solved by conventional techniques in computer vision and machine learning. After all, what is the best way for a video summary to be composed? One might argue that only the most interesting or exciting moments of a video should be retained; another may argue that a summary should be comprehensive and reflect the entirety of the video. The problem, however, lies in attempting to define the "interestingness" of a video quantitatively. Even if we could do this, how would we design an algorithm which would generate video summaries that appeal to a broad audience? A segment of a video may be fascinating to one viewer and irrelevant to another. Video summarization is inherently an ill-posed problem, as there is an infinite number of ways to summarize a video, any of which may be equally valid depending on the viewpoint of the audience [1].

1.1 Challenges of video summarization

Furthermore, there is a significant variation in the structure and contents of videos in general. For instance, a sports broadcast tends to have a different duration, structure, and features than a first-person video filmed using a mobile phone. It is unlikely that two videos, especially those of greatly varying genres, would be best summarized by a single approach. As such, a given technique may be very effective at handling a specific type of video but fail to generalize to another type.

Typically, videos contain spatiotemporal, audio, and occasionally textual data. Each mode may provide insights about the content of the video and can be processed jointly or independently to compose a video summary. By incorporating multiple modalities into our system, we can detect more complex interactions within videos; for example, an event may be detected by using clues from both the visual and audio streams simultaneously. The trade-off, however, is that the inclusion of multiple modes of information comes at the expense of design complexity and computational cost. Each mode of information is distinctly formatted and must be carefully processed before they can be combined. In conjunction with the overhead of working with videos, which are incredibly dense in information, this often pushes computation time and cost beyond a level that is acceptable for practical application.

To tackle this problem, we use a supervised deep learning approach. Namely, we use convolutional neural networks **(CNNs)** to analyze both the visual and audio stream of the video. Given an appropriately annotated dataset, CNNs can learn to automatically extract features which correspond to certain actions or events occurring in a video. This can then be used to build a system which makes decisions on how to summarize a video based on the features detected. Normally, this would be done by a human.

Lastly, there lies the issue of obtaining enough data to train a deep neural network. In this case, we will be working with video data—in contrast to single images (video frames). Videos include the temporal dimension and are significantly denser in information as a result. Large open-source databases of videos (with annotations for summarization) are rare and usually contain far fewer data samples than datasets containing only images. Moreover, since people provide these annotations, they are potentially also biased, and may not truly represent the best moments of a video.

1.2 Video summarization for hockey broadcasts

While video summaries are incredibly useful for previewing content, there are significant challenges in *automatically* creating summaries that are interesting to a broad audience. In this work, we focus on the automatic summarization of sports broadcasts, specifically ice hockey. Working with sports broadcasts presents its own set of challenges. However, it also allows us to mitigate many of the earlier discussed problems by leveraging two invaluable resources—the crowd and the play-by-play announcers. By utilizing the spectators (composed of both the crowd and the announcers), we can design a system that detects the highlights of the game and intelligently produces a summary.

In a typical video summarization pipeline, the video is analyzed and searched for individual key moments of interest; these highlights are then linked together to form the final summary. In general, this works quite well, and many techniques found in the literature follow this framework. However, among the existing methods for finding highlights and forming summaries, most techniques do not emphasize the composition of the video. Once the highlights are detected, they are usually concatenated into a summary without further consideration. This approach may be effective in some contexts, but we argue that it is not enough to produce compelling sports broadcast summaries.

Several additional issues arise when generating hockey summaries. How should the final length of the video be constrained? Out of the candidate video clips, which best complement each other and should be chosen to form the summary? Where should these clips start and end? Including a clip that begins a few seconds too early leaves the viewer without context; a clip that ends a few seconds too soon leaves the viewer dissatisfied. Special care must be taken when forming the summary in order to generate high-quality results.

To address these issues, we design our system using two principal components: the **highlight detector** and the **highlight selector**. The highlight detector uses the reaction of the audience as a collective indicator of exciting moments in the game. By analyzing the visual and audio streams of a video, we can search for cues that correspond to the occurrence of an exciting event, for instance, the cheering of the crowd or the excitement in the announcers' voice. Furthermore, by analyzing multiple modes of the video, we can better detect highlights that may not be obvious when only viewing the video or listening to the audio on their own. Ulti-

mately, using the audience as a basis for driving the summarization process allows us to bypass the bias found in human annotations or manually designed features. Once the highlights of the broadcast are found, the highlight selector (i.e., a deep network) chooses which of them should be included in the summary. This is done in a manner which considers constraints such as the maximum or minimum duration of the summary, since there are often many more highlights in a hockey game than there is space to include in a summary. The overall flow of information through the video summarization system is depicted in Figure 1.1.

In our implementation, the highlight detector is based on 3D CNNs adapted for a multi-modal input using fused neural networks (Chapter 3), and the highlight detector is based on an adversarial neural network (Chapter 4). Both the components are trained individually and then joined sequentially to form the overall system; the input to the system is a full hockey broadcast, and the output is a highlight video. The rationale behind this choice of technology will be discussed at length for both the highlight detector and the highlight selector in Chapters 3 and 4, respectively.

To generate our training dataset, we use publicly available NHL hockey broadcasts. We process these hockey broadcasts into a format that can be used to train the deep networks (i.e., CNNs) which comprise both the highlight detector and the highlight selector (to be discussed). Moreover, to pretrain our network, we also utilize Audio Set [2], which is an extensive collection of annotated audio clips collected from millions of videos on YouTube. Since this dataset includes samples of cheering, we can use it for detecting reactions from the crowd (with minimal modifications). This approach allows us to train our system with minimal need for additional annotation; furthermore, it also opens up the possibility of extending the technique to other sports or genres of video with audiences.

In short, our main contributions are as follows: (1) we propose an audience-informed system for automatic summarization of hockey broadcasts; (2) we investigate the application of multi-modal 3D CNNs for highlight detection in hockey broadcasts; (3) we design a highlight selector that chooses which of the detected highlights should be included in the final summary.

1.2 Video summarization for hockey broadcasts

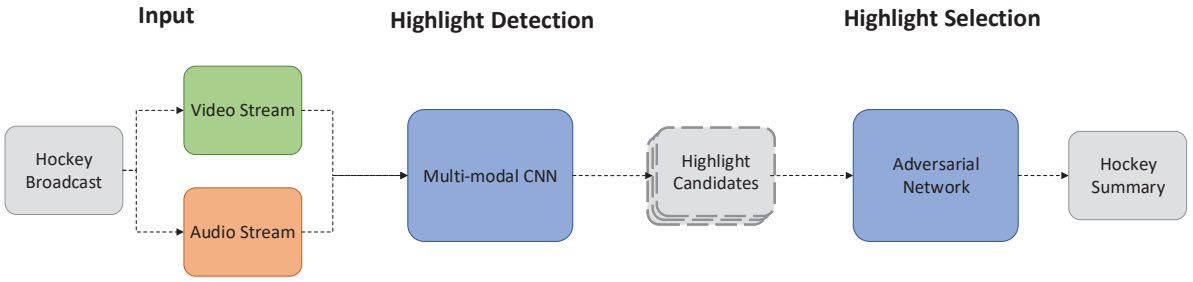

Figure 1.1: An overview of our proposed video summarization system. First, a **highlight detector** (based on a multi-modal CNN) finds the most *exciting* moments of a hockey broadcast based on cues from the audience and play-by-play announcers. Typically, there will be far more highlights in a hockey game than there is space to include in a summary. Therefore, once the highlights of the hockey broadcast are found, a **highlight selector** (based on an adversarial network) chooses which of these highlights should be included in the final summary.

2

Literature Review

There are several schemes for video summarization. Although these approaches may vary greatly, they all attempt to address a fundamental problem: how can we employ a computer program to determine what parts of a video are *interesting* to a viewer? In this chapter, we will examine various techniques from the literature, with an emphasis on those applicable to sports broadcasts.

2.1 Overview

The vast majority of video summarization techniques are based on machine learning, which is a family of algorithms that utilize training data to make predictions or decisions without explicit programming. Most machine learning techniques can broadly be classified into three categories: supervised, unsupervised, and semi-supervised learning. In the context of video summarization, supervised learning approaches take advantage of annotated datasets, where videos are paired with a mask (or frame importance scores), which act as ground truth for training a model. On the other hand, unsupervised learning techniques seek to extract information from a video, typically using explicitly designed features, which are then classified or used to assign an importance score to each frame. Semi-supervised learning techniques utilize a combination of labelled and unlabelled data, and usually have the advantage of requiring less annotated data than strictly supervised methods.

A typical video summarization framework selects which frames of a video should be in-

cluded in the summary based on a set of criteria (or some decision-making process). In general, a video is decomposed into a set of subshots, before selecting which subshots should be linked together to form a summary—shots selected in this manner are commonly referred to as **highlights**.

The criteria chosen to decide which subshots should be selected as highlights are highly varied, but in general, attempt to determine how *interesting* or *important* a given subshot is. For example, a technique may look for the presence of important entities [3] in a shot (such as people) to detect highlights. Another technique may optimize for metrics such as visual or semantic redundancy by examining low-level features such as histograms, motion, or colour [4, 5]. More recent approaches may forego explicitly designing any criteria and instead rely on deep neural networks (trained with annotated data) to predict which subshots are interesting [6, 7]. To compute these criteria, one may examine the visual, audio, or textual information present in a video , as discussed in Chapter 1; each of these modes may contain cues or features which are useful for computing how important a frame or shot is.

Another consideration is the scope of application for a given technique. Some approaches may be designed for general use and to be broadly applied to videos with varying genres, structure, content, and duration (e.g., for videos found on YouTube or social media). Other techniques focus on specific genres of videos, such as sports broadcasts, surveillance videos, or user-recorded content, by exploiting domain knowledge. For instance, sports broadcasts may have certain camera angles or sounds corresponding to the occurrence of specific moments of interest. Ultimately, the choice of criteria depends on the application; e.g., it may be desirable to maximize interest when summarizing a sports broadcast into a highlight reel or it may be optimal to minimize redundancy when summarizing documentaries. In this chapter, we will be examining both general and domain-specific approaches, with an emphasis on sports broadcasts.

2.2 Supervised video summarization

The most straightforward approach is to formulate the problem as a supervised learning task. In this approach, an input video is represented as a sequence of frames or subshots (including the audio and text), and the output is a mask indicating which of them should compose the

summary. Generally, a feature extractor is used to extract features from the input, and another model (such as a classifier) determines which parts of the video should be included in the summary. The system is then trained by using an annotated dataset. In short, the summarization task is reduced to binary classification or regression by assigning a label or score for each frame or subshot of a given video. Under this framework, the most important considerations are the choice of features to extract, the design of the model(s), and the training procedure.

A promising starting point is to take advantage of the existing methods for automatically extracting and classifying features from videos (e.g., for video classification or event recognition) using supervised deep learning [8, 9, 10, 11]. In recent approaches, LSTMs [12, 13, 14] and CNNs (2D/3D) [6, 15, 16, 14] are commonly used as both the feature extractor and the classifier. A typical framework for summarizing videos in this manner is illustrated in Figure 2.1 and Figure 2.2. In contrast to their classical counterparts, which are based on low-level cues (e.g. colour, histograms, motion) or explicitly designed features, these networks automatically learn from a dataset to extract relevant features from the video. In terms of training these networks, popular datasets include TVSum [17] and SumMe [18] which each contain approximately 25-50 videos from a variety of genres (news, documentaries, egocentric, etc.). These datasets are also commonly used to benchmark various techniques against each other. For our application, however, we are not aware of any public benchmarks or datasets for evaluating the summarization of hockey broadcasts.

There are various trade-offs among the choices of model to use. For instance, the architecture of LSTMs gives them the capacity to consider both long-term and short-term relationships in the data better than CNNs alone. Consequently, Zhang et al. [12] argue that visual analysis alone is not enough to determine redundancy, and that long-term dependencies are essential for interpreting high-level semantics in a video (leading to higher quality summaries). 3D CNNs, on the other hand, typically only examine temporal data within a small window determined by the size of the convolution filters in the network architecture; despite this, [6, 19, 11] have shown that they are effective at summarizing videos. More recent architectures have addressed this weakness and have implemented mechanisms to model long-term dependencies across time in CNNs as well; for example, the attention-based CNN proposed by Chen et al. [20]. Ultimately, the choice of feature extractor is flexible and can be interchanged depending on the constraints of the application.

2.2 Supervised video summarization

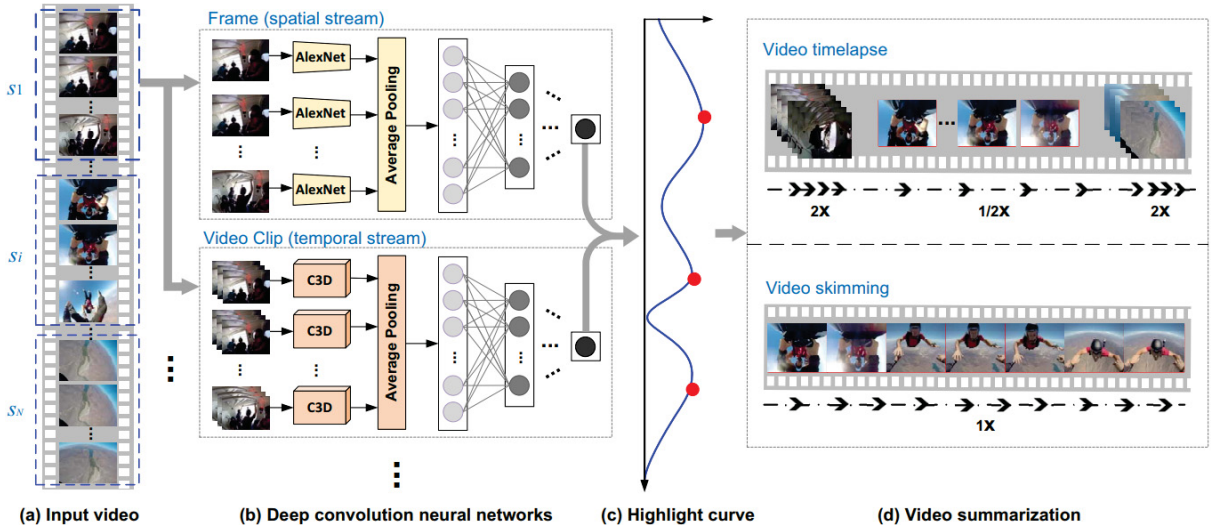

Figure 2.1: A supervised summarization system using CNNs proposed by Yao et al. [6]. A video is split into a set of video frame chunks, $\boldsymbol{s}$, and fed into two fused CNNs designed to analyze the spatial and spatiotemporal streams. The CNN system learns (from a dataset) to predict a highlight score for each chunk in the video. This procedure is repeated for each chunk and a highlight curve is generated for the entire video; the most interesting moments are then selected as highlights.

2.2 Supervised video summarization

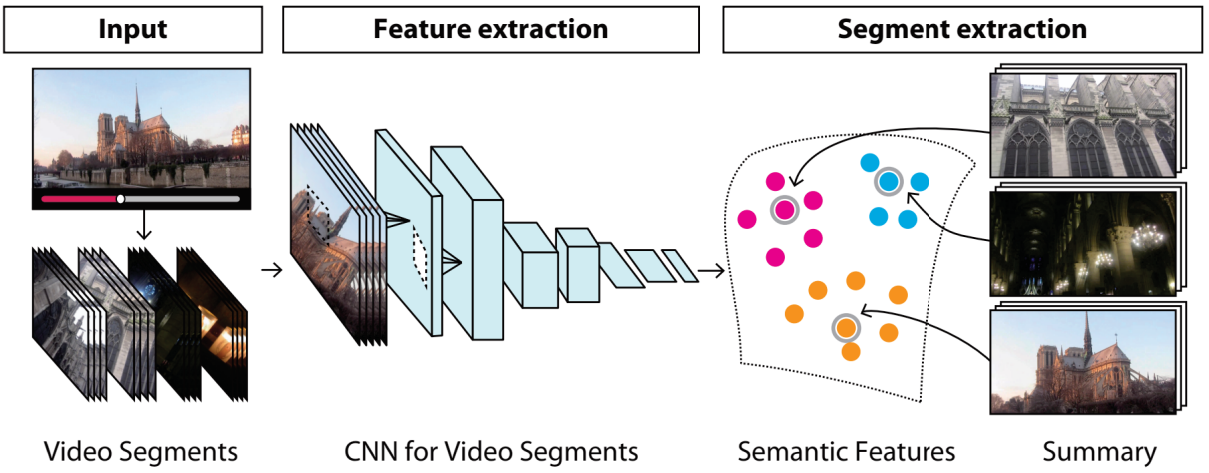

Figure 2.2: A pipeline for summarization using semantic clustering proposed by Otani et al. [15]. A video is split into a set of video frame segments and fed into a CNN. The CNN embeds the video segments into a semantic feature space, which are then subsequently clustered. Finally, video segments are chosen from a variety of clusters to form a summary which minimizes redundancy and monotony.

Aside from the choice of model, the formulation of the training objective is also important for properly training neural networks. For example, Yao et al. [6] highlights an issue with training models for summarization through classification. Since choosing subshots is inherently a ranking problem, they argue that this task is better characterized by regression than classification (since binary classification doesn't distinguish which subshots are most interesting *relative* to each other). To work around this, they propose a pairwise ranking scheme, in which the classifier is trained by being presented two subshots and tasked with identifying the maximally interesting subshot of the pair. Using this method, Yao et al. [6] show that a fusion of 2D and 3D CNNs can effectively learn to rank highlights in user-generated videos. Yet another way to model the summarization problem is through clustering. Otani et al. [15] train a system of neural networks to extract features that represent the semantic content of the video; these are then clustered and used to form the summaries (by choosing subshots from a variety of semantic clusters). This is a simple approach that excels in dealing with videos where semantically redundant content is abundant.

Unsurprisingly, supervised learning approaches have also demonstrated success when applied to sports broadcasts. For example, Tejero et al. [7] show that CNNs can learn to detect

highlights after being trained on the frames and the optical flow of an annotated dataset of Kendo videos. Similarly, Lin et al. [21] use a combination of LSTMs and CNNs to summarize baseball videos. For golf broadcasts, Merler et al. [22] uses an assortment of CNNs to simultaneously extract features from the visual, audio, and text streams. Their analysis of the information is very extensive and involves a combination of cheering detection, broadcaster voice tones, text analysis, expression recognition, TV graphics, game statistics, etc. In each of the previous examples, a highlight score versus time graph is generated for the video, and the most interesting moments are used to form a summary. With the exception of [22], these approaches suffer from the same weakness as many of the other supervised techniques and is prone to a shortage of annotated data or biases in the annotations.

As demonstrated by the previous examples, the choice of model, and how we choose to formulate the summarization problem plays a crucial role in controlling the characteristics of neural networks after training. This is especially useful when dealing with specific genres of videos, i.e., hockey broadcasts, where we know beforehand what type of issues will be most prominent. Using this, we can choose which aspects of the hockey game we wish our summarization system to place emphasis on by carefully selecting which models we use and how we formulate the training objective.

2.3 Unsupervised and semi-supervised video summarization

In unsupervised video summarization, the formulation of the problem is similar—given a video, find a mask which tells us which frames should form the summary. As discussed in the chapter overview, the video is divided into a set of subshots, and a metric is computed to quantify the "importance" of the shot. Importance can be defined in a number of ways but usually consists of measuring notions such as the presence of important entities, diversity (redundancy), motion, saliency, temporal spacing (spacing of the selected shots throughout the video), etc. [23, 24, 25, 3, 26, 27]. Overall, there is no general metric that is better than all of the others, and the best metric to use depends on the task.

Early unsupervised approaches tended to use low-level cues as the primary unit of consideration. Wolf [28] analyzes motion in films by first computing optical flow on the videos and then summarizing the shots by looking for moments of pause, meant to place emphasis for

the viewer. Similarly, Ma et al. [29] propose a framework that utilizes motion, attention, and saliency to find highlights in videos of varying genres (home videos, sports, television, etc.). Yihong et al. [30] use a method based on feature dimensionality reduction (through singular-value-decomposition) and clustering; they form a feature matrix by sampling frames from the video and analyzing the colour histograms. Afterwards the features are clustered, and a subshot is selected to represent each cluster.

In the context of sports broadcasts, Baoxin et al. [31] and Tjondronegoro et al. [32] propose unsupervised methods for summarization in sports videos. Their frameworks exploit domain knowledge in soccer videos to automatically detect events such as whistles, cheering or changes in cinematography. They examine low-level features such as the proportion of soccer field pixels to total pixels (to determine breaks in gameplay), or the intensity of specific frequencies in the audio stream. Similarly, Ekin et al. [33] proposes a framework that uses a mixture of low-level features (e.g. shot changes, dominant colours) and semantic features (e.g. goal detection, referee detection) to summarize soccer videos; their pipeline is depicted in Figure 2.3. While these techniques are simple and can detect certain events in a game, they require explicitly designing each feature and generally are not robust to variations in the broadcast.

In terms of semi-supervised learning, another technique is to incorporate feature extractors that have been trained on relevant datasets beforehand, e.g. objects, or entities, to extract higher-level information. For instance, Lu et al. [5] proposes a story-driven approach to summarization by analyzing the connection between objects. They argue that significant events in a video may be connected and form a coherent story. To this end, they introduce a story metric that quantifies the ability of significant objects in one subshot to *influence* objects of another subsequent subshot in a *coherent* manner. After detecting the objects in the video, a graph is constructed which models the probability that objects appear with respect to one another—these probabilities are then used to compute a numeric value for influence. For example, in an egocentric video, the appearance of a bowl may be connected to the appearance of food later in the video. This is a very intuitive notion for egocentric videos, where events may be connected in predictable sequences. With that being said, it may be difficult to extend this technique to non-egocentric videos where there isn't a clear sequence of events, and the same objects are constantly re-appearing throughout the video without correlation.

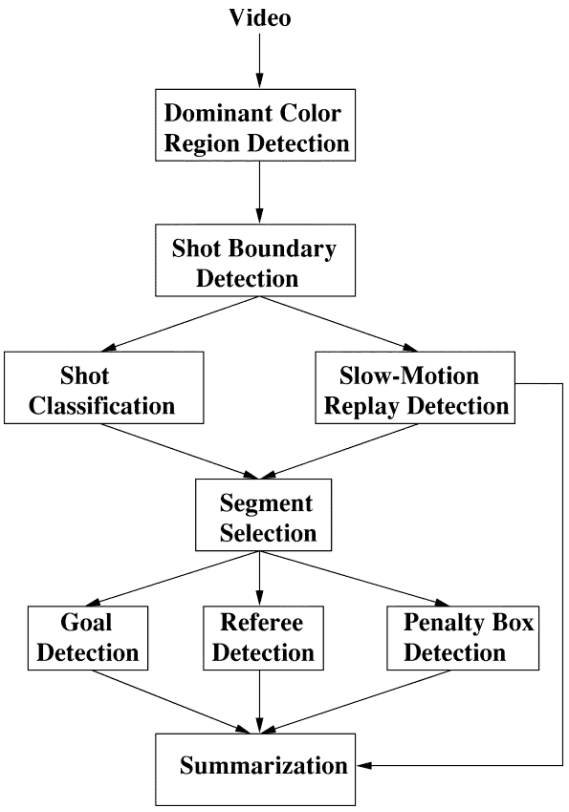

Figure 2.3: An unsupervised pipeline for summarization of soccer broadcasts by Ekin et al. [33]. This system relies on several modules explicitly designed to detect certain events in a soccer broadcast (shot boundaries, goals, slow-motion replays, etc.) using low-level cues such as motion, colour, or lines. A rule based system is used to determine which of the events should be included in the summary (e.g., all goals should be included, etc.).

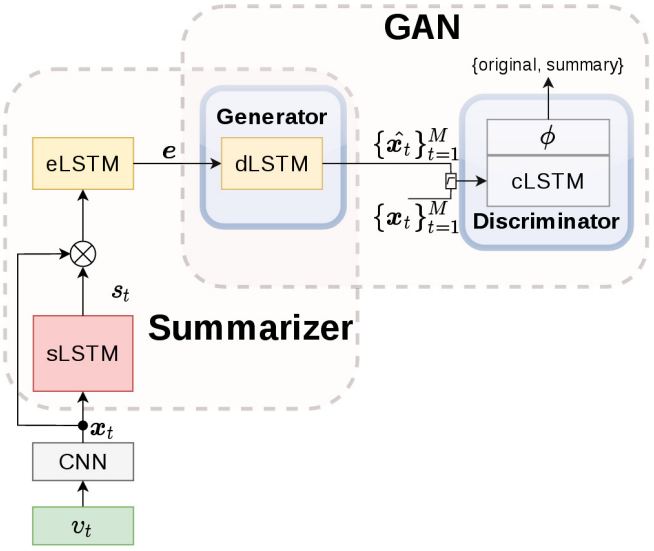

Figure 2.4: The adversarial LSTM system proposed by Mahasseni et al. [35]. A set of video frames $\boldsymbol{x}$ (indexed by time t) is passed into a selector LSTM that chooses a subset of frames, $\boldsymbol{s}$, which is then encoded by another LSTM to yield a feature vector, $\boldsymbol{e}$. The system is trained in an adversarial manner; the generator attempts to reconstruct $\boldsymbol{x}$ from its encoded representation $\boldsymbol{e}$, and the discriminator is tasked with distinguishing $\hat{\boldsymbol{x}}$ from $\boldsymbol{x}$.

Keane et al. [34] evaluate the events in hockey broadcasts and deploys an entropy-based metric as well as a "cumulative event interest score" to locate the most interesting moments in hockey broadcasts. They argue that selecting the most exciting events in a video may not maximize interest to a viewer and that a succession of medium excitement events in a short period may be more relevant. To this end, they propose accumulating and combining the impact of individual events within some window to determine the excitement at a given time as a "cumulative event interest score." They also argue that simply choosing the most exciting events would lead to monotonous summaries, and consequently use an entropy-based metric which penalizes the selection of multiple events of the same type. While this approach is effective, it requires that each event occurring within the hockey broadcast is labelled and detected. Due to the duration of a hockey broadcast, this requires manually annotating and classifying thousands of events per video—this is often impractical and greatly limits the scope of application.

More recently, generative adversarial networks (GANs) have also been investigated for video summarization. GANs functions by casting two neural networks against each other in

a game; for instance, training a CNN tasked with detecting artificial images against a CNN tasked with generating artificial (but realistic) images. In the context of video summarization, Mahasseni et al. [35] propose a system of LSTMs trained in an adversarial fashion to summarize videos. Their system is abstracted into three main components, the summarizer, the generator, and the discriminator, as depicted in Figure 2.4. First, the summarizer consists of two networks: a selector LSTM tasked with selecting highlights (a set of frames) from a video, and an encoder LSTM tasked with encoding the frames chosen by the selector LSTM into a fixed-length representation (in some feature space). Next, the generator takes the encoded output from the summarizer and uses yet another LSTM to decode and reconstruct the original video from its encoded representation. Finally, the discriminator consists of an LSTM tasked with discriminating between the original videos, and their reconstructions from the generator. In short, a set of loss functions are defined such that this adversarial system of networks is trained to maximally confuse the discriminator. This occurs when the discriminator can no longer distinguish the difference between the videos reconstructed by the generator, and their original counterparts. The adversarial network effectively learns to summarize videos and attains state-of-the-art performance on several datasets, even outperforming supervised techniques. This is a very interesting result, as it demonstrates that we can learn networks that extract and reconstruct videos without having to explicitly design any features or mechanisms as in most of the preceding techniques.

2.4 Conclusion

In summary, video summarization can be solved by many techniques, which vary depending on the domain of application and desired results. Moreover, we have seen numerous machine learning approaches used for video summarization. Techniques ranging from explicitly designed features to CNNs, LSTMs, and GANs, work well for video summarization, each with their own advantages and disadvantages. Lastly, we have seen the effectiveness of highlight-driven video summarization for sports broadcasts. This intuitive approach creates summaries by detecting the most exciting moments in the game. In the next section, we will discuss the design of the first component of our video summarization system: the highlight detector.

3

Highlight Detection

In this chapter, we discuss the methodology used to design our video summarization system. We begin by discussing the procedure we used to curate and process our dataset of hockey broadcast videos. Next, we detail the first component of our system, the highlight detector, and the procedure used to train it. Finally, we present a set of experiments that compare and contrast various approaches and parameters that we considered when designing the highlight detector (e.g., CNN architecture choice or preprocessing parameters). The results of these experiments then guide the final design of our highlight detector.

3.1 Overview

Our goal is to design a video summarization system for hockey broadcasts that produces compelling highlight videos in a minimally biased and practical manner (e.g., not requiring an excessive amount of annotated data or computation time). While there are numerous approaches to video summarization (as discussed in Chapter 2), highlight detection is an especially effective method for sports broadcasts, where there are usually specific moments in the broadcast which are of significant interest to a viewer. To this end, we design a highlight detection module tasked strictly with automatically finding these critical moments of interest.

Our approach primarily considers the detection of highlights to produce video summaries; this is an effective approach for sports broadcasts where a handful of exciting moments in the broadcast can sufficiently portray a hockey game. That being said, this type of approach is not

always effective for video summarization in general. In other domains and applications, it may be beneficial to include non-highlights into a video summary. This may be done to provide context and improve the overall coherence of the resulting summary. Although, we refer to "summarization" and "highlight detection" interchangeably in the context of this thesis, these terms are *not* synonymous in general.

The initial design decisions are then to choose which criteria (or procedure) to select highlights with, which model(s) to use, and which method to train the system with. For our first decision, we chose to look for cues corresponding to highlights as the primary driver behind our detection process. To facilitate this, we build a **dataset of highlights** that we can use to train a model. We accomplish this by taking advantage of the *spectators* of the game (the crowd and the announcers) to dictate which parts of the hockey broadcast should be considered a highlight. By capturing the spectators' excitement, we have collected an ensemble of opinions on whether that particular moment of the game is exciting or not. Moreover, using the spectators allows us to mitigate the bias found in human-annotated datasets, which typically only have a few annotators per data sample. Using this approach, we collect and annotate a dataset (approx. 1500 videos) of hockey clips where the spectators demonstrate excitement (cheering or excitement in the announcers' voices). This dataset contains clips from multiple NHL hockey seasons and is used to train and evaluate our highlight detection system.

Supervised deep learning techniques are a practical and straightforward solution to video summarization. Furthermore, we have also seen that it is beneficial to analyze multiple modes (i.e., video and audio), when detecting highlights (as discussed in Chapter 2). Consequently, we build our highlight detector using a set of CNNs designed to analyze both the video and audio streams simultaneously for cues corresponding to highlights. We achieve this by using fusion to combine several neural networks into a single system. In conjunction with our dataset of highlights, this lets us train a highlight detector by posing the problem as a binary classification task; each frame of the video is predicted to be either a highlight or non-highlight. This is then later used to drive the overall summarization process.

For the training procedure, we utilize an approach based on transfer learning. The video and audio cues we are searching for in hockey broadcasts are not unique to this domain. Because of this, we pretrain our neural networks on data samples from existing datasets containing cheering or speech samples. After pretraining, we can then train our networks on the dataset of hockey

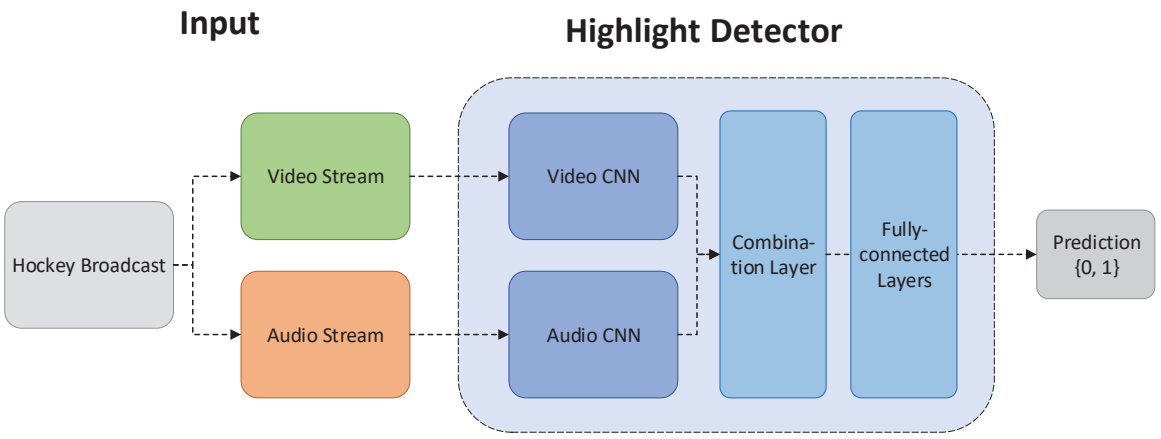

Figure 3.1: Overview of the highlight detector architecture. A hockey broadcast is split into its constituent video and audio streams which are then fed into a fused neural network. Within the highlight detector a CNN processes each input modality, which are then finally joined by a combination layer. This combined output is finally fed into a fully-connected layer to produce the final classification prediction.

highlights that we have curated (using a standard machine learning pipeline). Effectively, this permits us to achieve high performance (i.e., accuracy), while simultaneously keeping the highlight dataset relatively small in size. Altogether, the resulting system is straightforward and does not require an extensively annotated dataset to train.

Lastly, we run a set of experiments to analyze the effectiveness of various design decisions. Most prominently, this includes the choice of neural network architecture, the training procedure, and the choice of hyperparameters. We then use the results of these experiments to choose the best design for our highlight detection system.

3.2 Dataset and preprocessing

3.2.1 Gathering the hockey broadcast dataset

First, we gather a dataset of full-length hockey broadcasts from the NHL. To accomplish this, we downloaded hockey broadcasts using the NHL's replay-on-demand online streaming ser-

vice[1]. In total, 565 full-length broadcasts were gathered from regular-season and play-off games played during the 2017-2018 and 2018-2019 NHL hockey seasons. Additionally, the NHL produces and releases a highlight video (5 to 8 minutes in duration) for each broadcast. We have downloaded these summaries and paired them with their corresponding broadcasts.

The dataset amounts to approximately 1700 hours of video footage (at 29.97 frames-per-second, totalling 183 million frames). Technically, each video has a resolution of 640x360 pixels (360p), 29.97 frames per second, 128 kilobytes-per-second stereo audio (sampled at 48 kilohertz), and MP4 encoding at an average bit-rate of 1288 kilobytes-per-second. The average duration for a video is 3 hours for a full broadcast, and 7 minutes for the NHL produced summaries. The entire dataset contains significantly more samples than required to train a highlight detector; instead, we work with a subset of approximately 150 videos to derive our highlight dataset.

3.2.2 Preprocessing and cleaning

The first step in our preprocessing procedure is to remove irrelevant portions of the hockey broadcasts, i.e. the commercial breaks. Due to the formatting of the downloaded broadcasts, commercial content is absent from the videos. Instead, a placeholder image is displayed, without any audio signal, while the broadcast is on commercial break. In order to detect and remove the commercial breaks, we analyze the audio track of the video. The audio is separately extracted, and the intensity is processed with a moving average filter (using a window of 2 seconds). The resulting signal is thresholded to find segments of silence exceeding 15 seconds in duration. On average, approximately 1 hour of empty footage was removed from each broadcast using this method. At this point, the videos are ready to be annotated into the highlight and non-highlight partitions (to be discussed shortly).

After the annotation process, the videos are formatted for use with a machine learning pipeline. That is, the video and audio streams are individually extracted and processed into a format appropriate for input into a deep neural network. In this case, due to the extensive amount of data that needs to be processed and analyzed, the video and audio tracks are downsampled in advance. For each broadcast: (1) the video frames are extracted at 5 frames-per-

[1]NHL replay-on-demand service: https://nhllive.com/en/

second and down-sampled to a resolution of 224-by-224 pixels; (2) the audio is down-sampled to a 16-kilohertz single-channel signal and the loudness normalized.

3.2.3 Annotating the highlight dataset

We assemble a secondary dataset of highlights from the overall hockey broadcast dataset. This is accomplished by manually viewing the hockey broadcasts (and their NHL-produced summaries) to find exciting moments of gameplay. For our application, we deem that a moment in the game is exciting if there is visible or audible excitement from the spectators of the game. Prominent examples of this are audible cheering or excitement in the announcers voice. After finding these exciting segments, we record the start and end time of the exciting moment.

Likewise, to annotate the negative samples, we review the same broadcasts and find moments where the spectators are not demonstrating excitement. The start and end time of these moments are then recorded. Since negative samples greatly outnumber positive samples, only a *random* subset of the negative samples is included in the final dataset of highlights. This is done to avoid producing a highly imbalanced dataset which introduces challenges during the training and evaluation stages of building the highlight detector. Altogether, the dataset consists of approximately 750 positive samples and 750 negative samples. These samples are extracted from 151 hockey games played during the 2017-2018 and 2018-2019 NHL hockey seasons (i.e., a subset of the broadcasts we downloaded).

3.3 A framework for detecting highlights in sports video

3.3.1 Visual analysis with convolutional neural networks

We detect highlights in the broadcast by computing the probability that, at any given time, t, there is an exciting event (without distinction as to what type of event is occurring). This is accomplished by using a system of deep neural networks, which takes a video clip as an input (e.g., a 5-second clip of a proposed highlight) and outputs the predicted probability of being a highlight on the range $[0, 1]$. There are several considerations when designing such a system; namely, the preprocessing of the video and audio streams, the architecture of the CNN, and ultimately how the CNN is trained.

3.3 A framework for detecting highlights in sports video

The choice of preprocessing procedure ultimately impacts what type of CNN architecture can be chosen due to computational constraints. In order to capture the temporal aspect of the video, the input to the CNN is composed of 3D chunks of data. These chunks typically contain a few seconds of continuous footage, generated by iterating over the video with a sliding window. In this approach, the size and stride of the moving window are hyperparameters. This procedure is illustrated in Figure 3.2.

In general, it is impractical to feed every single frame into the CNN during training or inference—the amount of time captured in each input segment must be balanced with the time resolution of the sliding window. For instance, a 5-second input-segment (window) at 30 frames-per-second would require an input size of 150 frames. Spanned over 2 hours, with a stride of 1 second, this amounts to approximately 1 million frames per video. Such a substantial input is often impractical or impossible to work with due to computer memory limitations or computation time. Instead, it is more practical to sample frames from the video at a reasonable frequency. For example, 5 seconds of footage sampled at 5 frames-per-second, formatted as a sliding window with a stride of 2-3 seconds yields a much more manageable amount of information to process per video.

The next consideration is the choice of CNN architecture(s), as shown in Figure 3.1. While effective options are abundant, the main concerns, in this case, would be the ability to handle temporal and multimodal data while staying within reasonable computation constraints (i.e., something possible to compute on consumer-grade hardware). To handle temporal data, the two main architectures we have seen are CNNs and LSTMs, which both have been demonstrated to perform well on several computer vision tasks. For the CNNs, further modifications are usually needed for them to support sequential inputs. We use a temporal-pooling based approach to accomplish this due to effectiveness versus the number of extra parameters required (as demonstrated in [9, 36]). This is done by taking a CNN, which generally only supports a 2D input, and passing each frame of the input video clip through the network. Each frame passed through the network generates an N-dimensional embedding; the embeddings from each input frame are then stacked (on the temporal dimension) and subject to a $\max$ operator. Finally, the resulting N-dimensional output is passed through a series of fully-connected layers (i.e., a feedforward network) to produce the final output, as in a typical 2D CNN architecture.

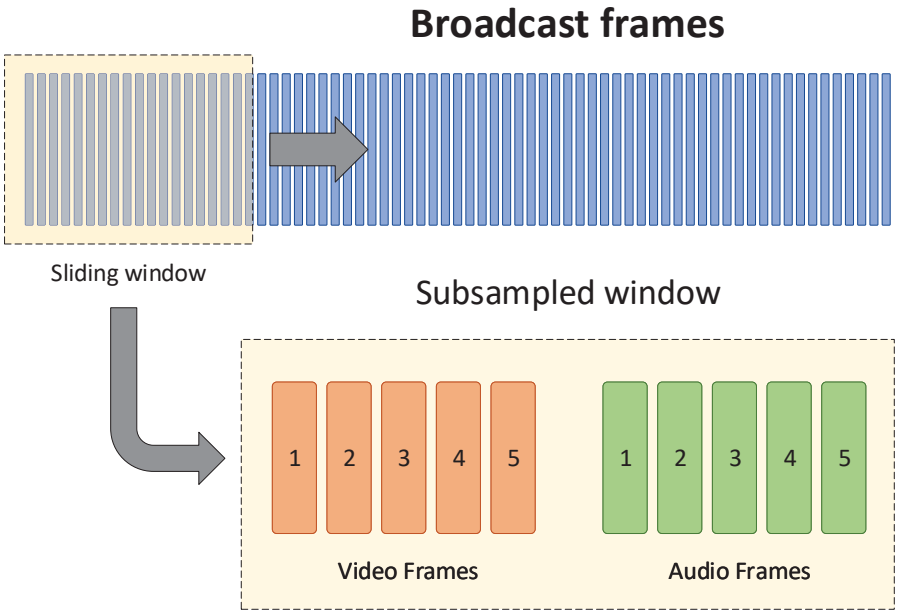

Figure 3.2: The sliding window procedure used to iterate over the frames of an input video. As part of the preprocessing step, a sliding window passes over the video, and frames falling within its range are *subsampled* (at even intervals) to produce a *chunk* of data. The chunk of data is composed of video and audio frames that are synchronized in time and are eventually used as input to the highlight detector.

3.3.2 Audio analysis with convolutional neural networks

Similar to the visual case, we employ CNNs to analyze the information in the audio stream. The first step is to convert the preprocessed 1D audio signal into a 2D format that can be used with a 2D CNN. To implement this, we convert the audio signal into a 2D format by computing the Mel spectrogram [37] of the audio signal; this technique has seen widespread in audio analysis and has been demonstrated to work well as an input to CNNs [38, 39, 40, 2]. The resulting spectrogram is a 2D image in which the horizontal axis represents time, the vertical axis represents frequency, and the pixel value represents intensity. This image's dimensions are left as a hyperparameter, e.g., 96-by-64 pixels for each second of audio. An example of this is shown in Figure 3.3.

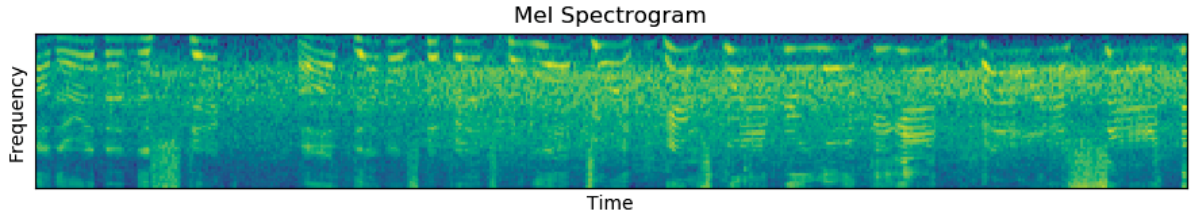

Figure 3.3: An example of a Mel spectrogram used to train our highlight detector. The spectrogram is a 480-by-64 pixels image generated from 5 seconds of audio, where the horizontal axis represents time, and the vertical axis represents frequency. The image is generated from a 1D audio signal and can be used to classify sounds into different categories.

Unlike the case with the video stream, there is significantly less information to process, especially after computing the Mel spectrograms. As such, there are no additional sub-sampling steps or hyperparameters to consider in the preprocessing. Instead, we compute the Mel spectrograms for the entire video and divide them into segments that align with the sliding window used for the video stream. The resulting video and audio inputs are synchronized in time. The audio CNN is then trained using a typical machine learning pipeline for images.

3.3.3 Multimodal analysis with fused convolutional neural networks

Our approach utilizes a multimodal input to utilize both the visual and auditory information available in the video. In this case, we analyze both modes simultaneously to detect highlights in the hockey broadcast. We accomplish this by using fusion to join multiple CNNs, each individually handling a different mode of information. For our experiments, we address two conventional methods of fusion: early fusion and late fusion. These two types of fusion are summarized in Figure 3.4.

Late fusion

The most straightforward approach is through **late fusion**, where several (typically unimodal) models separately process the input before joining together. In late fusion, each mode of the input (e.g., video and audio) is passed into a corresponding model that produces an output vector, yielding $v_1, v_2, ..., v_n$. The outputs are joined together using a **combination layer**: this may involve concatenating, adding, or multiplying the output vectors to produce a combined vector, $p = f_c(v_1, v_2, ..., v_n)$, where f_c denotes the combination layer. This combined vector

can then be used as a feature vector for another model, which finally makes a prediction based on the combined information. For our application, we utilize two CNNs, one for the video and the other for the audio.

This approach is not without drawbacks. Fusing multiple models comes at a price; namely, computation cost and computer memory. Late fusion potentially doubles the number of parameters in our model; this can make it challenging to train due to overfitting and instability (caused by having significantly more parameters). Furthermore, the extra networks required also exacerbate computer memory restrictions, which typically limit the capacity (i.e., parameters or depth) of neural networks. Nevertheless, fused neural networks remain an effective way of taking existing single-mode frameworks and adapting it for multimodal support without extensive changes to the conventional machine learning pipeline.

Early fusion

In **early fusion**, multimodal data is typically transformed into a joint representation before being passed as an input to a model. That is, unlike the late fusion scheme, the "combination layer" is situated at the start of the architecture instead of the end. Conventional methods of joining together multiple modes are concatenation or using a transformation to project all modes into a common feature space. Moreover, once the input is combined into a joint representation, this representation can be used with a *single* model (as opposed to multiple unimodal models running in parallel in late fusion).

For our experiments, we use an encoder CNN as the combination layer. It takes the usual video and audio input and produces a single condensed output (i.e., a 2D vector) which combines the information from both modalities. This vector is then fed into a single CNN, and the entire system (including the encoder) is trained end-to-end. To implement this, we upsample the audio spectrograms to match the dimensions of the video frames (224-by-224 pixels); the video and audio frames are then concatenated and used as the input to the encoder. This approach allows us to avoid having to design a transform (or some other procedure) to produce a joint representation of the video and audio data. Instead, the encoder is automatically trained along with the rest of the system.

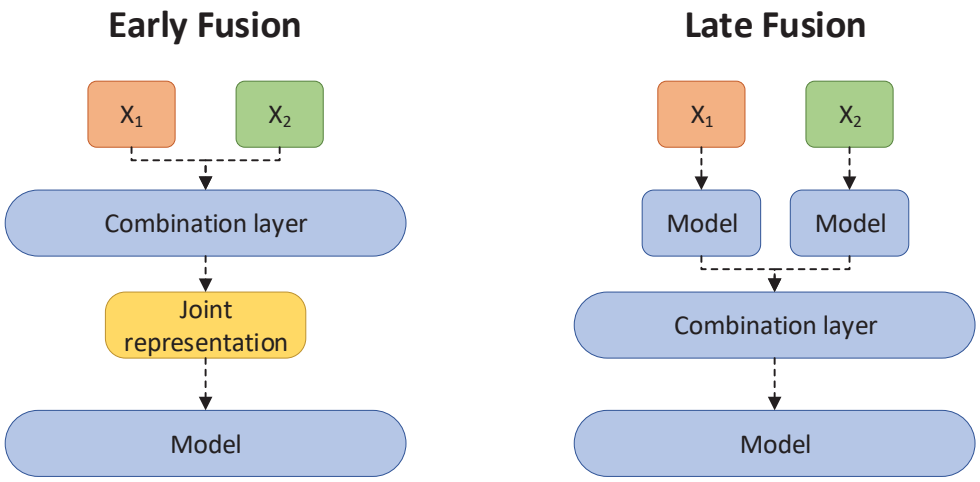

Figure 3.4: Comparison between the early fusion and late fusion techniques. In early fusion, the input, X_1 and X_2, are combined into a single representation (i.e., a feature vector) by a combination layer before being input to a model. In contrast, late fusion passes the inputs through *multiple* models (for each mode) before joining them in a combination layer; the combined representation is finally passed into another model to make a prediction.

3.4 Methodology and experiments

3.4.1 Highlight detector architecture

Combining all of the components we have talked about so far, Figure 3.1 depicts a summary of the system we have designed. Our highlight detector is composed of two main branches, each of which is a CNN designed to handle video and audio information. The first branch is a modified CNN that takes 3D chunks of visual information as input (through temporal max-pooling). The second branch is a standard CNN that takes a 480-by-64-pixels spectrogram as an input. Like the video stream, this branch takes as an input a sliding window with the same hyperparameters as the first branch (i.e., the branches have inputs synchronized in time). With the general pipeline in place, we explore several combinations of CNN architectures, as well as fusion techniques to determine which one performs best.

First, we run a set of experiments comparing the performance of different CNN architec-

tures. For the models under consideration, we chose to experiment with the VGG [41, 40, 2], ResNet [42], and Inception [43] architectures. Each of these models has an extensive record of being effective for image classification tasks and is ubiquitously used as benchmarks or baselines. We also examine the early fusion and late fusion variants of the architecture in each of these experiments.

In our last set of experiments, we perform an ablation study on our highlight detector. We examine the system's effectiveness when presented with only one of the two modes of information: video-only and audio-only. These experiments are intended to verify and quantify the benefit of using fused CNNs instead of a single CNN, in addition to examining the performance of video-only versus audio-only.

The exact structure and architecture of the highlight detectors we experiment with are given in Section A.1 in the Appendix.

3.4.2 Training procedure and hyperparameters

The training of the system is also imperative to overall performance. We must consider the choice of training objective (i.e., the loss function), the optimizer, the organization of the dataset, and all of the hyper-parameters accrued throughout the design process.

First, to partition the dataset, we follow standard machine learning practices. The dataset (made of approximately 1500 data samples) is first split into three randomized partitions: the training set (60%), the validation set (20%), and the testing set (20%). We train our highlight detector using the training set and then evaluate our design's performance on the validation set (which is then used to adjust hyperparameters). Once we have completed our experiments, we use the test set to report our final results.

In terms of hyperparameters for training the CNNs, we need to choose settings for the input size, the optimizer, and the training loop. For the input, we sample the video with a sliding window that has a width of 5 seconds, a resolution of 5 frames-per-second, and a stride of 2 seconds. In all of our experiments, we use the Adam [44] optimizer with an initial learning rate of $5\text{x}10^{-4}$. For the training loop, we use the binary-cross-entropy loss (**BCE**) function, a batch size of 16, and 30 epochs of training in each experiment; the model with the overall best loss is kept.

To implement late fusion, we use a combination layer that consists of a two-layer feedforward network with an input layer size of 1152 features, 512 features in the hidden layer, and an output size of 1. As for early fusion, we use a combination layer that concatenates an upsampled copy of the audios spectrogram, and the video frames (as discussed in Section 3.3.3); both the spectrogram and video frames have dimensions of 224x224 pixels to produce a concatenated input with dimensions of 428x224 pixels.

3.4.3 Evaluation

To evaluate our highlight detector's performance, we consider two main metrics: accuracy and binary-cross-entropy (BCE) loss. Since we are building a highlight detector, including shots that are false positives, would be highly detrimental to the quality of the resulting summaries. As such, we are interested in ensuring that the system achieves a high degree of precision, i.e., it should have a low number of false positives. With that being said, we still need to maintain adequate recall to ensure that enough highlights are being detected per game to form a summary with sufficient duration. Because of this, and since our dataset is balanced, we use accuracy as our primary indicator of performance. Accuracy simultaneously accounts for the importance of true positives and true negatives while being easily interpretable compared to a metric such as F-score. On the other hand, BCE measures the difference between our models' predicted probability versus the ground-truth. By minimizing for BCE during training time, we penalize the model heavily for making incorrect classifications with a high degree of confidence.

3.5 Results

In total, we experiment with 4 architectures in two different fusion configurations. Additionally, we perform a set of ablation experiments in which the input is limited to only video or audio. The results of all the experiments are summarized in Table 3.1.

First, we find that using a single model, video-only or audio-only approach is viable. Our experiments show that ResNet34 attains 96.74% accuracy and a BCE loss of 0.1024 when only(!) video data are available to the model. Similarly, we see that InceptionV4 attains similar results in the same situation. This performance level exceeds expectations, as the highlight clips were annotated based only on auditory cues (e.g., cheering). Nevertheless, the video-only

3.5 Results

CNNs are also capable of learning to detect visual cues associated with the auditory cues such as cheering or excitement in the announcers' voices.

Similarly, we see strong performance from the audio-only models as well. For these experiments we use 3 non-pretrained models (VGG19, ResNet34, InceptionV4) and 1 model (VGG19) pretrained on AudioSet [2]. Of the non-pretrained models, ResNet34 is the strongest and attains an accuracy of 97.36%. On the other hand, VGG19 leads overall with an accuracy of 98.22% and a BCE loss of 0.0664. As expected, using a model pretrained on a large dataset of sounds yields a significant increase in performance; in this case, a decrease of error rate from 2.64% to 1.78%.

Turning to the multimodal experiments, we analyze the same architectures (duplicated for the video and audio streams) in both the early fusion and late fusion configurations. We find that ResNet34 in the late fusion configuration performs best and attains an accuracy of 98.54% accuracy. In terms of the best BCE loss, ResNet34 in the early fusion configuration results in a loss of 0.0541. Compared to the unimodal experiments, nearly all of the architectures see a significant increase in performance. Moreover, both early fusion and late fusion are viable choices for highlight detection.

With that being said, when comparing the performance of multimodal models (that use both video and audio) to unimodal models which only utilized audio data, we did not see a large improvement in performance. For example, the multimodal variant of ResNet34 (using both audio and video) has only 0.9% more accuracy than the audio-only version of ResNet34, despite utilizing substantially more data. We suspect that this is due to using the *reaction* of the audience and the play-by-play announcers as the primary cue for defining a highlight. Since this cue is based on audio (i.e., cheering or excitement in the announcers' voices), it follows that the video data do not provide an abundance of additional information that isn't already present in the audio.

Lastly, we experiment with a mixture of architectures, i.e., using a different model for each of the video and audio streams. In this set-up, we find that a combination of ResNet34 for the video stream, and VGG19 for the audio stream performs best, with an accuracy of 98.82% and BCE loss of 0.0309. Our results demonstrate that using both video and audio improves the highlight detector's classification accuracy and BCE loss by a significant margin. The highlight

detector learns to find indications of excitement in both the video and audio, as well as cues which might only be apparent when analyzing both modes together, but not either in isolation.

As an aside, we note that although we use a balanced dataset for our experiments (i.e., our dataset has a similar number of positive and negative samples), this distribution of data is usually not found in practice. A typical hockey broadcast will contain significantly more negative segments (i.e., segments of the game that are not highlights) than positive segments, and therefore we should ensure that our highlight detector maintains a high degree of accuracy in such a scenario. In this case, when examining the accuracy of the models we experimented with, we observed that the highlight detector maintained approximately the *same* level of accuracy when dealing with both positive and negative samples. Therefore, we can expect our models to be highly accurate even if the data under consideration are unbalanced (i.e., the distribution of the data is skewed towards positive or negative samples).

3.6 Conclusion

In this chapter, we propose an audience-driven highlight detector for highlight detection. We show that by analyzing the audience of hockey broadcasts, we can effectively train a highlight detector with a relatively small dataset. Our experiments demonstrate the viability of using multimodal CNNs through a variety of neural network fusion techniques. Through experimentation, we settled on a combination of a modified ResNet34 and VGG19 architectures designed to handle visual and auditory data, respectively. This configuration detects highlights in hockey broadcasts with a high degree of accuracy while maintaining a relatively simple and intuitive design.

With a functional highlight detector now completed, the next step in designing our system is finding a way to create *compelling summaries* using the detected highlights. We seek to improve upon the conventional approach in highlight-driven video summarization, where highlights are just concatenated without further consideration. To this end, we propose a module explicitly designed to pick and choose which highlights should actually be included in the final summary.

3.6 Conclusion

Table 3.1: Highlight detector architecture versus accuracy and BCE loss

Model	Fusion	Acc. (%)	Loss
VGG19 (video only)	N/A	81.65	0.4068
ResNet34 (video only)	N/A	**96.74**	**0.1024**
InceptionV4 (video only)	N/A	96.15	0.1234
VGG19 (audio only)	N/A	64.50	0.5891
ResNet34 (audio only)	N/A	97.63	0.0698
InceptionV4 (audio only)	N/A	96.15	0.1324
VGG19 (pretrained, audio only)	N/A	**98.22**	**0.0664**
VGG19 (video + audio)	Late	98.19	0.0653
ResNet34 (video + audio)	Late	**98.54**	0.1028
InceptionV4 (video + audio)	Late	97.04	0.1131
VGG19 (video + audio)	Early	64.45	0.6756
ResNet34 (video + audio)	Early	97.92	**0.0541**
InceptionV4 (video + audio)	Early	96.45	0.1009
ResNet34 (video) + VGG19 (pretrained, audio)	Late	**98.82**	**0.0309**
ResNet34 (video) + InceptionV4 (audio)	Late	96.44	0.1059
InceptionV4 (video) + VGG19 (pretrained, audio)	Late	98.81	0.0418

4

Summary Formation

4.1 Overview

Given that the highlight detector is complete, we can take a full-length hockey broadcast and decompose it into a set of highlight clips. In general, there will be *more* highlight clips than there is space to include in the summary, i.e., hockey summaries are usually between four and eight minutes in duration, and there is no guarantee that all of the detected highlights will fit in this space. Moreover, while the highlight detector can find highlights, it does not consider which highlights complement or conflict with each other when placed in the same summary. Therefore, we require some sort of **selection** or **filtering** process to choose which highlights should actually be used to form the final summary.

To this end, the second component of our video summarization system is the highlight selector. It takes a set of **highlight candidates** from the highlight detector and selects which of the candidates should form the final video summary. This second component directly complements the highlight detector, and together they form our complete video summarization system. In short, the highlight selection problem is: given a limited duration for a summary, how do we best determine which highlights should be included to create a *compelling* hockey summary?

Evaluation

The highlight selection task is challenging because it is difficult to explicitly define what characteristics in a candidate are conducive to creating high-quality hockey summaries. After all, what

makes a summary pleasing to a viewer? A conventional approach usually considers some form of **accuracy** metric (e.g. recall, precision, and F-score) measured against a human-annotated dataset. However, only having a high degree of "accuracy" does not guarantee that a summary will be pleasing to view. Other considerations usually include **representation** and **diversity**. A quality summary should portray an accurate representation of the game to the viewer—as such, it should include most (if not all) of the key events that occur during the hockey game. Furthermore, a good summary should also represent a variety of content found within the broadcast to avoid monotony; this typically refers to diversity in visual, semantic, or temporal content. For instance, a good hockey summary should feature a diverse set of events that takes place at different times throughout a game. Lastly, we also consider the notion of **coherence** when generating our summaries. A good summary should portray events with context and avoid abruptly changing shots before an event is concluded. For example, a goal in a hockey game should be presented with some "set-up" leading to the goal and show some of the aftermath. Therefore, we should strive for a highlight selector that produces summaries satisfying these criteria (through the selection of the right set of candidates).

Ranking candidates using value functions

For our baseline approach, we consider a straightforward technique that uses an *explicitly* designed value function to assign a value to each of the candidates. The highest scoring candidates are then selected and joined together into a summary (i.e., the candidates are ranked by a value function, and only the best are chosen). This is very similar to the metric-driven video summarization approaches discussed earlier in Section 2.3, where a value is assigned to each moment (or segment) of a video based on its features. The key distinction is that we are *not* using a value function to determine what and where the highlights are (that is the job of the highlight detector), but only as a means of filtering some of them out.

A typical value function measures features such as the presence of important entities; for example, motion, visual diversity (among the selected candidates), etc. A more sophisticated value function may also attempt to account for the interaction between different video clips (e.g., do specific clips redundantly portray the same content, or do certain clips complement each other?). In general, the main consideration behind this **value-based** approach is figuring out which features in a video the value function should measure.

Adversarial learning

As opposed to the above, we examine a more recent and advanced technique based on deep learning. Rather than explicitly designing a value function and having to figure out which features to measure, we instead use a deep network that automatically learns to filter the highlight candidates. Typically, deep networks learn to summarize videos by training on an annotated dataset. The deep network is presented an input video and is tasked with finding which shots should form a summary. This summary is then evaluated by comparing it to summaries generated by human annotators. While this is an effective technique, it isn't applicable in our situation because we are working with an unlabelled dataset. Moreover, this would counter what we are trying to achieve by relying on the crowd to dictate highlights, rather than using human-made annotations.

For this reason, we use an approach that can function without labelled data. While there are several types of unsupervised learning techniques that could work for our application, we decided to investigate **adversarial learning** due to recent works demonstrating its effectiveness (e.g., Mahasseni et al. [35] and Rochan et al. [45]). The idea behind this approach is to create a system of deep networks that have objective functions that compete with one another, and using this as a means for training the components of the system (this will be discussed in Section 4.3). Training the system in this manner allows us to *bypass* the need for an annotated dataset of hockey summaries and *bypass* the need to design any explicit value functions (such as in the value-based approach).

Chapter organization

To summarize, the highlight selection task is non-trivial, and there are many different ways of interpreting the problem. In this chapter, we examine both of the approaches discussed above, namely: (1) the value-based approach and (2) the adversarial-learning-based approach. For both of these techniques, we formulate the problem, discuss various design decisions, and explore each technique's strengths and weaknesses. The results of these experiments (and, therefore, the overall video summarization system) will be discussed in Chapter 5.

4.2 Creating summaries by using value functions

For our first experiment, we address the highlight selection task using a conventional value function-based technique. In this approach, a set of highlight candidates is received from the highlight detector, and then a highlight selector chooses which candidates to keep by assigning a value to each of the candidates under consideration. This effectively allows us to *rank* the "quality" of the highlight candidates, and therefore we can select the best subset from which to form a summary. Before delving into the highlight selector's details, we indicate the flow of information through the complete video summarization system is as follows:

1. We pass the hockey broadcast under consideration through the highlight detector and obtain a *graph*, $P(t)$, that plots the probability of a highlight occurring versus the video's time (down to a resolution of 0.2 seconds). This graph is processed with a moving average filter with a window of 1 second to reduce noise in the signal. An example of a signal from the highlight detector (and its corresponding graph) is depicted in Figure 4.1.
2. This graph is thresholded (at $P(t) = 0.50$) and converted into a binary mask indicating the locations of the predicted highlights throughout the entire broadcast. In the mask, a value of one indicates a highlight, and a value of zero indicates a so-called "non-highlight".
3. We assume that each *continuous nonzero segment* in the binary mask is considered to be a potential highlight candidate. The broadcast is subsequently decomposed into a set of highlight candidates according to this definition. The resulting set of video clips is the output of the highlight detector *and consequently* the input to the highlight selector.
4. A subset of the highlight candidates is chosen by the highlight selector and concatenated into the final summary. In this case, the candidates are evaluated and ranked by a value function, and the best candidates are selected.

With this framework in place, the primary consideration is figuring out how to design a value function that assigns a meaningful value to each of the candidate clips. Effectively, we can tailor the characteristics of the summary produced by our system through the design of the value function used by the highlight selector. To this end, we investigate three different value

functions, each of which places emphasis on different characteristics of the candidates under consideration.

4.2.1 Probability-weighted candidates

The most straightforward value function to use (in this context) would be to assign each candidate a score based on its probability of being a highlight, the computation of which will be discussed shortly. Using this method, the highlight selector chooses the candidates with the strongest *individual* probability of being a highlight and joins them into a summary. Of note, since the highlight probability is from the highlight detector, this value directly incorporates the information from the spectators (i.e., the crowd and the announcers).

Due to its simplicity, this value function does not consider if specific candidates complement or conflict when placed in the same summary. Moreover, the value function does not consider other properties of the highlight candidates, e.g., the duration of the event, or the time at which the event occurs during the broadcast. For these reasons, we use this value function as our **baseline** to compare against the effectiveness of more sophisticated techniques.

Recall that there is a highlight probability associated with each moment in time of the broadcast (predicted by the highlight detector). Since each highlight candidate is formed by a continuous segment of video footage, we compute the highlight probability for a candidate by averaging the highlight probability associated with each second of that clip. We refer to this value as the **probability score** of a highlight candidate. For instance, if a highlight candidate occurs within the first ten seconds of the hockey broadcast, its probability score is computed by averaging the ten highlight probabilities predicted by the highlight detector for that period of time.

The computation of the probability score is summarized by Equation 4.1 and the overall value function by Equation 4.2. As for the notation: (1) c_i represents the i-th highlight candidate in a broadcast (bounded between the times $t_{i,1}$ and $t_{i,2}$); (2) $S(c_i)$ represents the probability score for the i-th candidate; (3) $P(t_n)$ represents the highlight probability at a given time t_n during the broadcast; and (4) the value function is denoted by $V(c_i)$.

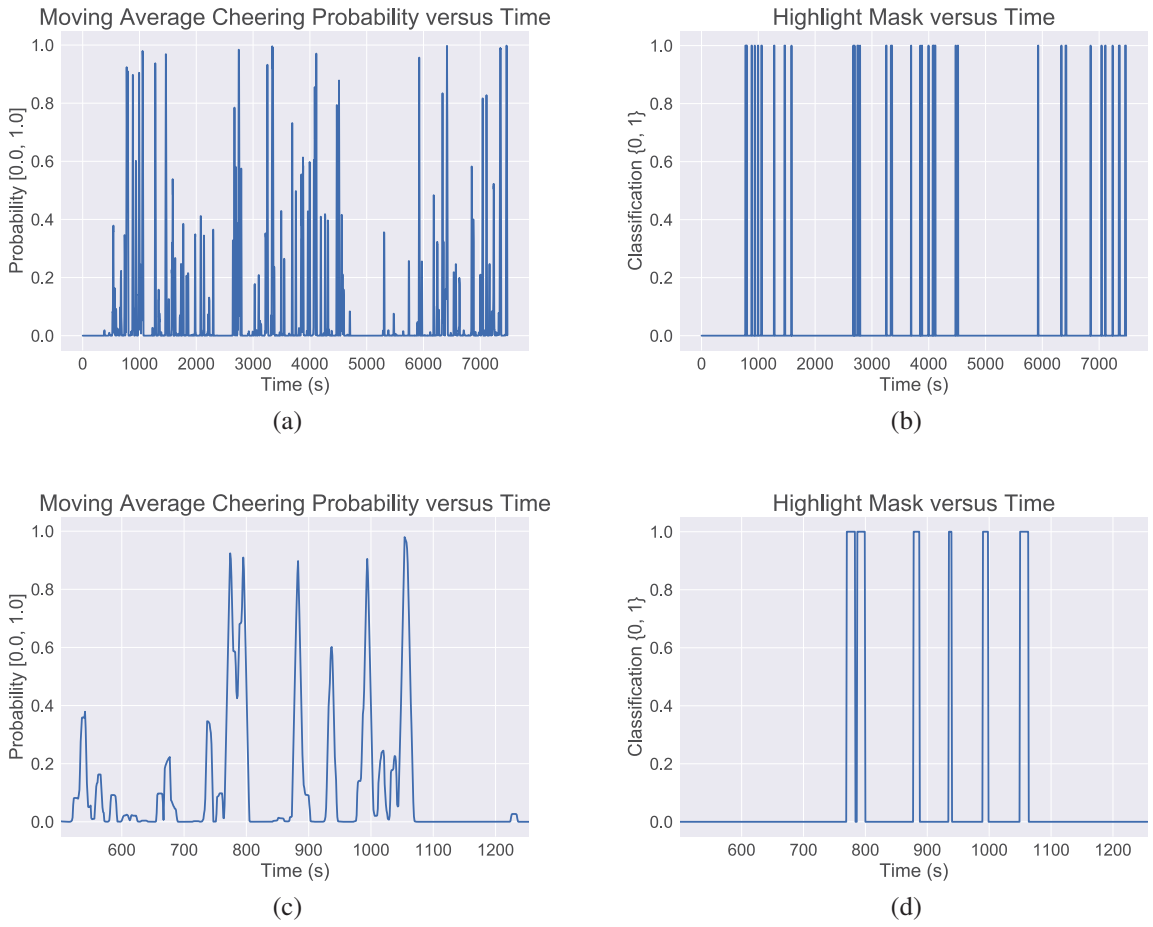

Figure 4.1: An example of the *cheering* probability versus time signal received from the highlight detector (for a single broadcast). The cheering probability signal is converted into a binary mask by applying a threshold; the resulting mask is split into continuous segments, which form a set of highlight candidates. Figures 4.1(a) depicts the moving average of the signal, and Figure 4.1(b) shows the mask obtained by thresholding the probability at $p = 0.5$. Figures 4.1(c) and 4.1(d) show a close-up look of the plot in Figure 4.1(a) and 4.1(b).

$$S(c_i) = \frac{1}{t_{i,2} - t_{i,1}} \sum_{t_n = t_{i,1}}^{t_{i,2}} P(t_n) \tag{4.1}$$

$$V(c_i) = S(c_i) \tag{4.2}$$

4.2.2 Duration-weighted candidates

For our next experiment, we consider a value function that weighs the candidates based on their duration; this is represented as a value function by Equation 4.3. The rationale behind this is that high duration highlights contain more content than their low duration counterparts, and therefore should be weighed more heavily. This, of course, is a heuristic, but, nonetheless it still makes an interesting experiment.

$$V(c_i) = S(c_i) * \text{duration}(c_i) \tag{4.3}$$

In contrast to the baseline probability-only approach, this value function places a significant emphasis on the duration of the individual candidates. It, therefore, penalizes highlight candidates with low durations, even if they have high probability scores. Consequently, the summaries formed using this technique lean heavily towards candidates that simultaneously have a strong probability score and a high duration. We also expect that choosing candidates with longer durations will result in summaries with fewer false positives since high duration candidates are the result of the highlight detector making several consecutive positive predictions. It is also more likely that the resulting summary will be more coherent since fewer candidates are forming the summary (and therefore, there are fewer transitions between scenes).

As an aside, by taking this approach, the highlight selection task becomes an optimization problem (known as the knapsack problem [46]) where one tries to fit the maximum value of items into a limited space (given that each item has a different size and value). In this analogy, the *limited* space is represented by the maximum time allocated to the summary, while the size of the item is the duration of the candidate, and the value is proportional to the candidate's probability score.

4.2.3 Proximity-weighted candidates

Lastly, we consider a value function that uses the heuristic that several exciting events occurring in close succession are more exciting than a single isolated event. Of course, this is not true in general, but such a heuristic has been previously shown to be effective for summarizing hockey broadcasts [34]. By analyzing when the highlight candidates occur during the game, we can add a bias to highlights that are clustered together. This "proximity term" is computed by comparing the distances between each of the highlight candidates. In doing so, the summaries formed by our highlight selector will favour highlights occurring near each other, even though the individual highlights may not have a very high probability score.

Before running this experiment, it is unclear how this type of weighing of candidates will impact the characteristics of the resulting summary. We suspect that there may be an increase in the number of false positives (due to less emphasis on the probability score). However, it is also more likely that the candidates selected will complement and provide context for each other (since they occur close together during the game).

To compute this value function, we add a bias based on the distance in time for every combination of highlight candidates pairs (i.e., the first candidate compared to the second, then the first candidate compared to the third, and so forth). This is expressed by Equation 4.4, in which the i-th highlight candidate (centered at time t_i) is compared against every other j-th candidate occurring at time t_j. The choice of the distance function (and how it tapers off with increasing distances) is relatively arbitrary. In our case, we kept things simple, and computed the bias using the reciprocal of the squared distance (in time) between two highlights. Lastly, the proximity term is scaled using a hyperparameter, $\alpha = 0.01$, in order to keep the magnitude of the bias relatively small compared to the probability score. As desired, the proximity term in the resulting value function is maximized when several candidates are within close proximity (in time) with one another.

$$V(c_i) = P(c_i) + \underbrace{\alpha \sum_{j=0}^{N-1} \frac{1}{(t_i - t_j)^2}}_{\text{proximity term}} \tag{4.4}$$

4.2.4 Methodology

Highlight selection process

Having designed the value functions, the next step is figuring how to apply them to generate a summary. We use an iterative approach to implement the candidate selection process, which is summarized as follows:

1. First, we assign a value to each highlight candidate under consideration using a value function. After assigning the values, all of the candidates are ranked and sorted accordingly.
2. We set an **initial threshold** equal to the mean value of all of the candidates. All candidates with values above this threshold are tentatively included in the summary.
3. If the resulting summary has a duration that falls within given minimum and maximum constraints, we terminate the highlight selection process and keep the summary. In general, we enforce a minimum summary duration of five minutes and a maximum duration of eight minutes.
4. Otherwise, if the summary has a duration that does not meet these standards, we raise or lower the threshold slightly (e.g., by ten percent) and repeat this procedure until a satisfactory summary is produced. For example, if the initial threshold produces a summary with an excessive duration, the threshold is iteratively raised until a summary is produced with a satisfactory duration.

The advantage of using this approach is that we effectively have a *dynamic* threshold for what candidates are included in the summary. If this were not the case, it would be difficult (and ineffective) to use a static threshold, since the value of candidates across different broadcast is highly varied.

Dataset

For this section of experimentation, we retain the same dataset used to train and evaluate the highlight detector from Chapter 3. Although we are not training any models for the optimization-based approaches, we retain the same partitioning of training and validation data from the high-

light detector. For the test evaluation (to be discussed in Section 5.2), we use a new partition of testing data that does not overlap with the previous dataset.

As previously discussed (in Section 3.2.1), this dataset was composed by a set of full-length NHL hockey broadcasts played during the 2018-2019 and 2019-2020 hockey seasons. For each of the broadcasts, the highlight detector generates a set of highlight candidates; together, these highlight candidates form the dataset used to experiment with the highlight selector. We refer to this dataset as the **highlight candidate dataset**. Moreover, for each of these broadcasts, the NHL produces and releases a professionally produced summary—we have also gathered these summaries and paired them with their original broadcasts. We henceforth refer to this dataset as the **NHL summary dataset** and will be using it for training the adversarial networks.

4.3 Creating summaries by using adversarial neural networks

Until this point, the value functions we have discussed have relied on explicitly designed features to measure the value of highlight candidates. A significant downside, however, is that it is usually difficult or unclear how these features should be designed in order to generate high-quality summaries. For instance, if we are trying to automatically generate hockey summaries that appear professionally-produced (i.e., high-quality), it is difficult to *define* what features or characteristics make an excellent hockey summary. In reality, a professionally-produced summary contains a multitude of fine details that would be impractical to measure using a handful of explicitly designed features. Consequently, a solution to this problem is not trivial by using value functions or other similar techniques.

An alternative solution to this problem is to use a deep network in conjunction with a human-annotated dataset. This approach would follow a typical supervised learning pipeline, which involves training a deep network to select highlight candidates based on a database of annotations. In this context, however, trying to solve this problem with supervised learning would be impractical due to the requirement of a large and extensively annotated dataset of hockey broadcasts. Since there are no public datasets of this nature, we would have to generate one ourselves; this would be an extremely time-consuming and expensive process. For instance, to annotate a sports broadcast, we would have to sort through several hours of content and label which parts should be included in a summary.

4.3 Creating summaries by using adversarial neural networks

Instead, we formulate the problem in a manner where we can use deep networks to implicitly learn how to produce summaries with characteristics similar to professionally produced hockey summaries *without using* annotations. To accomplish this, we leverage existing techniques for image generation, most notably, generative adversarial networks and adapt them for videos instead. Our approach is built upon previous works that have shown adversarial learning to be effective for video summarization, e.g. Mahasseni et al. [35] and Rochan et al. [45].

Adversarial networks

Adversarial networks function by casting two neural networks with opposing tasks against each other and using this conflict as a means of training both networks simultaneously. A typical application for this is artificial image generation, but it can also be extended to many other applications, such as video summarization.

For example, if we were trying to generate artificial images of human faces, we would build a system composed of a **generator** network tasked with generating images of human faces, and a **discriminator** network tasked with distinguishing between generated and real images. When trained using a dataset of real human faces, the generator and discriminator would both learn how to perform their respective tasks by "competing" against each other. At first, the "fake" samples created by the generator would not be very plausible and have a very different distribution of features than the "real" samples. However, as the network progresses during training, the generator creates increasingly more realistic images capable of fooling the discriminator. Likewise, the discriminator becomes increasingly adept at distinguishing between "real" and "fake" images. This feedback cycle trains both networks to perform their respective tasks, and eventually, the "fake" samples created by the generator have characteristics similar to the "real" samples. This concept has been previously demonstrated (e.g., Goodfellow et al. [47], Karras et al. [48], among many others) to be capable of training a generator that can produce very realistic images.

For our application, we seek to utilize adversarial learning and adapt it for generating hockey summaries instead of images; this procedure is depicted in Figure 4.2. In the place of a dataset of real images, we utilize the dataset of NHL summaries that we gathered earlier (discussed in Section 3.2.1). Recall that for each broadcast in our hockey broadcast dataset, we have paired it with a summary professionally produced by the NHL. While these NHL sum-

maries may not be perfect, they set a high standard for automatically generated summaries. Therefore, in this analogy (compared to image generation), we use the NHL summaries as the "real" samples for training the generator and discriminator. Here, the key idea is that we are using this procedure to train a generator network that can select highlight candidates that produce "fake" hockey summaries with the characteristics of "real" professionally-produced summaries. Beyond this, the labelling of certain summaries as "real" or "fake" does not actually mean anything in this context.

Using this, we design a system of deep networks where we task one network with generating a "fake" summary (by selecting a subset of the highlight candidates) and another network with the task of distinguishing our generated summaries from "real" NHL-produced summaries. We refer to these two networks as the generator and the discriminator (i.e., in the same manner as their image generating analogues). The pair of networks are trained using an adversarial objective function where the discriminator is penalized for failing to distinguish between the generated and real summaries, and the generator is rewarded for fooling the discriminator.

In short, we are interested in using adversarial learning to train a generator that can automatically produce summaries (through the selection of highlight candidates) with characteristics *similar* to professionally produced summaries created for NHL hockey games. Additionally, by using adversarial learning, we can forego having to explicitly design any metrics to measure the "quality" of a summary, as well as bypassing the need to produce an extensively labelled dataset. After training, we can combine the generator with the highlight detector (that we designed in Chapter 3), to form an end-to-end video summarization system. The highlight *detector* finds the most exciting moments in the broadcast, and the highlight *selector* picks the subset of highlights that form the best summary. The overall architecture being discussed in this section is summarized in Figure 4.3.

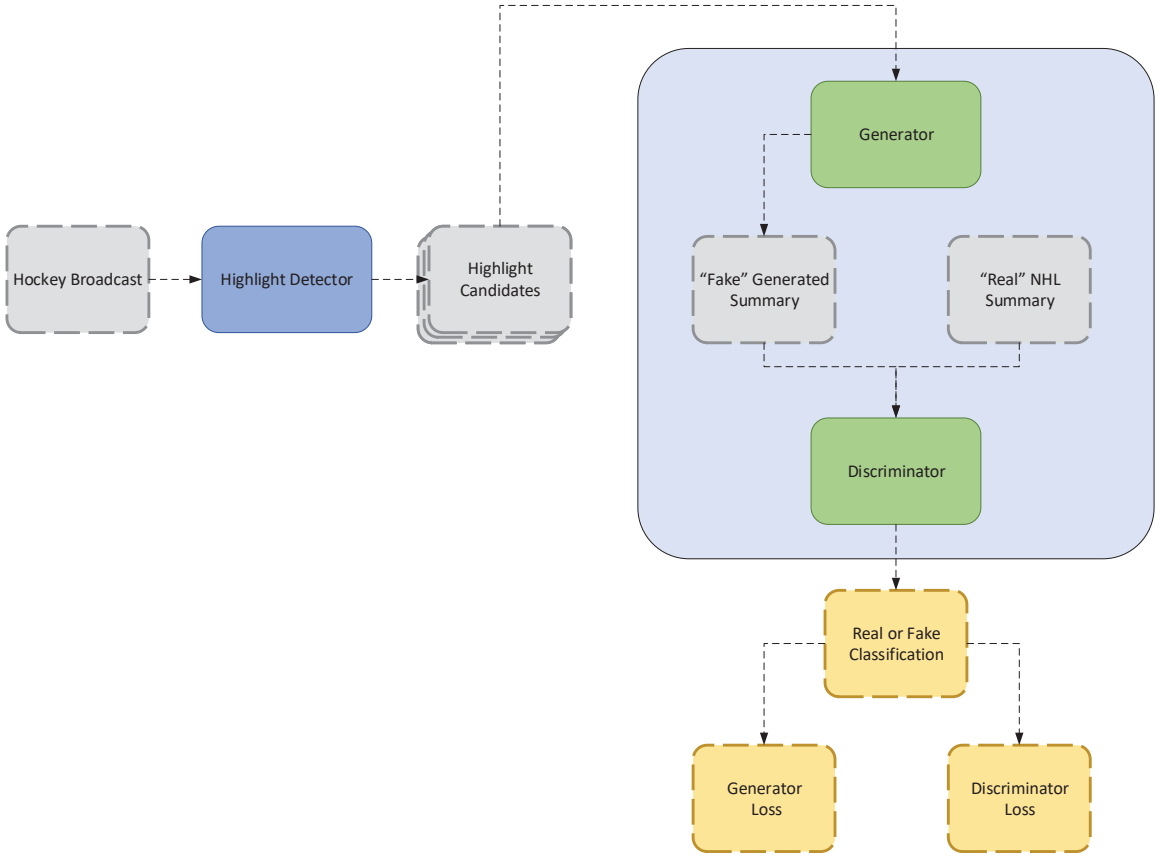

Figure 4.2: A sketch depicting the flow of information through an adversarial system for video summarization. First, a "fake" summary is generated by passing highlight candidates (determined by the highlight detector) through a generator. The generator creates this summary by selecting a subset of the highlight candidates. For the "real" data sample, we use a summary produced by the NHL for the *same* broadcast. Both "real" and "fake" samples are presented to a discriminator, which predicts a classification for both samples. From this classification, the loss functions are computed for the deep network and backpropagated to adjust the parameters of the entire system. The system is trained in this manner until the generator can produce "fake" summaries with similar characteristics to "real" NHL summaries.

4.3.1 Generator

The first component of the overall adversarial system is the generator. At a glance, its function is to produce a summary from a subset of highlight candidates presented to the system. These

highlight candidates are generated in the same way as in the value-based approach discussed previously (i.e., by analyzing the highlight probability graph generated from the highlight detector). The generator itself is composed of two deep networks, where each component has its own purpose. Namely, (1) the **highlight selector**, and (2) the **autoencoder**. The highlight selector is tasked with filtering the highlight candidates, and the autoencoder is included to facilitate the training of the overall adversarial system (composed of a generator and a discriminator).

Highlight selector

The first component of the generator is the **highlight selector**. The highlight selector's task is to *analyze the frames from each of the highlight candidates, and output a mask indicating which of the candidates should be chosen*. The chosen candidates are then concatenated to generate a summary. It turns out that the highlight selection task is very similar to a video classification task. Therefore, the design of the highlight selector shares a very similar design to the highlight detector discussed in Chapter 3.

The highlight selector's objective is analogous to an image classification problem, with the main distinction being that we are working with frames from multiple videos, instead of single images. Therefore, like the highlight detector, we can solve this problem by using a CNN adapted to temporal data such as videos. Using the same method as before, we adapt the CNN to handle 3D inputs by using temporal pooling (discussed in Section 3.3.1)—this technique is reasonably effective, simple, and computationally efficient. Moreover, we have already successfully used this technique for building the highlight detector.

The highlight selector's input is composed by extracting the frames from each of the candidate clips and concatenating them into a 3D matrix of frames. We use a matrix to represent the data because the highlight selector considers *multiple* candidates simultaneously. For example, if there are N highlight candidates proposed by the highlight detector, we sample L frames (spaced equally in time) *from each of the candidates* and stack them together into a N by L matrix of frames. After passing this input through the highlight selector (i.e., a CNN), a mask indicating which of the N candidates should be kept in the final summary is produced; this is then used to generate a summary from the highlight candidates. The exact details and formatting of the data will be discussed shortly in Section 4.3.3.

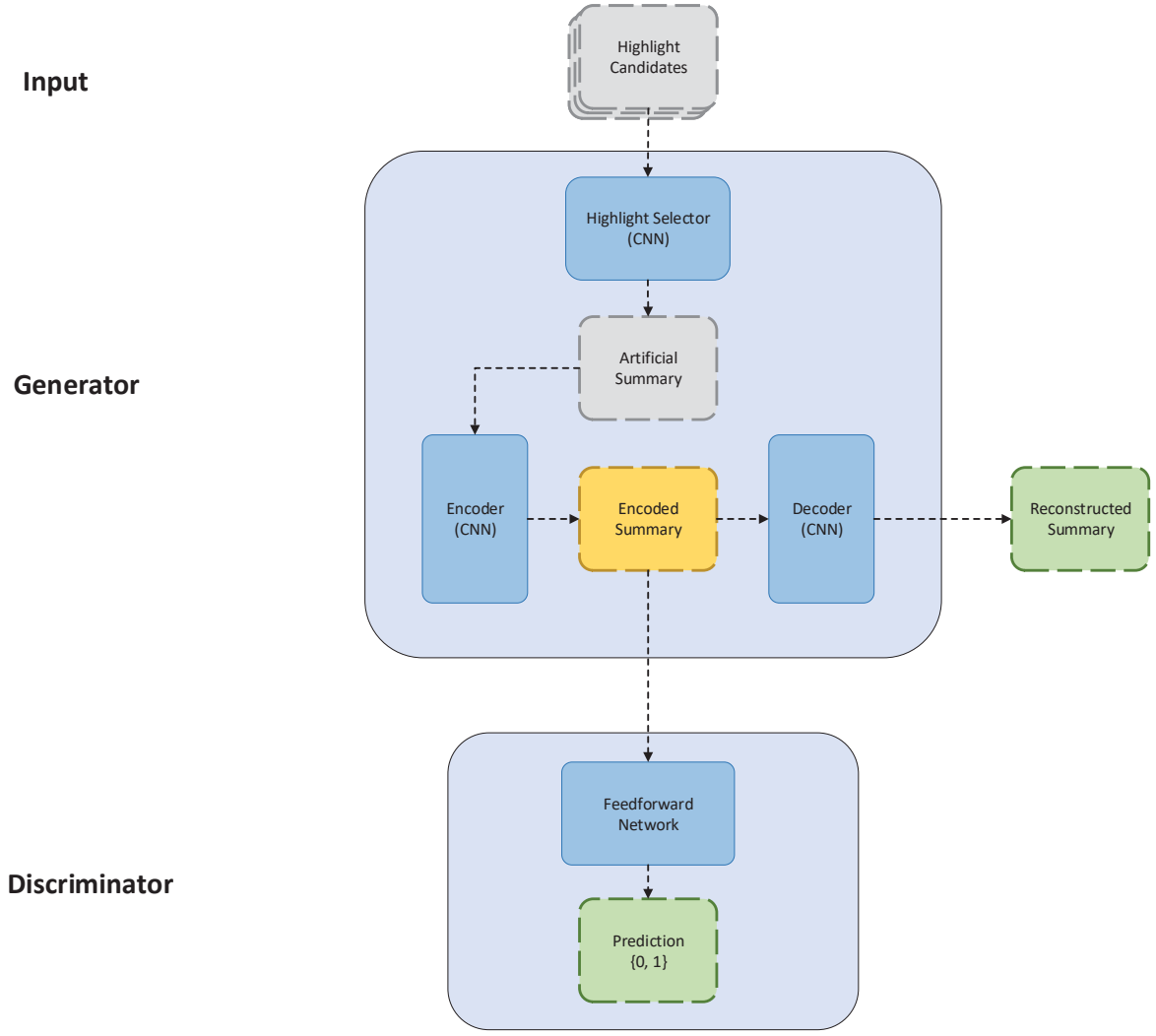

Figure 4.3: The overall architecture for the adversarial summarization system. It takes a set of highlight candidates from which a subset is chosen by the highlight selector to form a summary. The artificial summary is passed through an autoencoder that produces an encoding (i.e., an embedding). The discriminator, composed of a feedforward network, attempts to classify the encoding as real or fake. Furthermore, the encoding is passed through a decoder to create a reconstruction of the summary; this is used to compute the reconstruction loss during training time.

Encoder

Due to the varying durations of the summaries generated by the highlight selector, the generator is also tasked with encoding the generated summaries into an embedding (in some arbitrary feature space of hockey summaries). As such, the second component of the generator is the **encoder**. The encoding procedure simplifies the process of comparing data from different summaries by allowing us to design the generator to produce a fixed-size output (which would not be guaranteed otherwise). Furthermore, the encoding step significantly reduces the computational cost and memory usage of the discriminator (to be discussed shortly), making it easier to work with large inputs such as hockey broadcasts.

To accomplish this, we build and train the encoder using an autoencoder architecture. Autoencoders function by using an encoder and decoder pair trained in unison to learn compact representations of data automatically. That is, by using an autoencoder, we can train an encoder without requiring any additional data. In this architecture, an encoder is tasked with compressing an input into an embedding, and a decoder is tasked with reconstructing the embedding back into the original input.

Compressing the input into an embedding creates a *bottleneck* of information within the system and forces the encoder to learn compact representations of the input. On the other hand, the decoder is included to ensure that these compact representations retain the information of the original input. This procedure is summarized in Figure 4.4. The overall system is then penalized during training time based on the difference between the original input and the reconstructed output. By training the system in this manner, the encoder automatically learns to create compressed representations of the input (that retain most of the inputs original information) due to the autoencoder architecture's bottleneck of information. As for the choice of models, we base both the encoder and decoder off of CNNs, since they are well suited for tasks involving images.

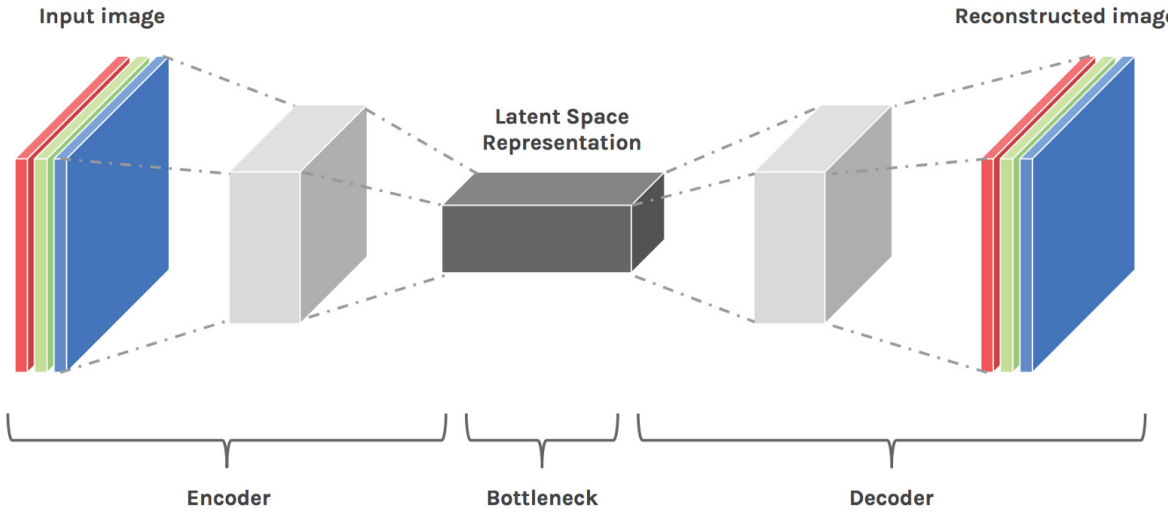

Figure 4.4: An example of a typical autoencoder architecture [49]. An input image is processed by a model and compressed into an encoding, which creates a bottleneck of information in the system. This encoding is subsequently processed by another model tasked with reconstructing the original input from its compressed representation. By training the system to produce an output which closely resembles the original input, the autoencoder learns to create compact representations of the input due to the bottleneck of information in the system.

To summarize, the encoder takes a summary generated by the highlight selector and produces a compact embedding to represent the video. This embedding is then presented to the discriminator, which is tasked with classifying it as a real or artificial. Using an encoder allows us to design the discriminator to accept a fixed-size input, simplifying the design process, and reducing the computational requirements of the overall system. Lastly, by using an autoencoder architecture, we can automatically train the encoder without any additional datasets.

4.3.2 Discriminator

The second component of the overall adversarial system is the **discriminator**. The discriminator is tasked with taking an encoded summary (produced by the generator's encoder component) and classifying it as either real or generated. In other words, the discriminator is a classifier. Since most of the computation is performed by the generator, the discriminator is very simple in comparison; it consists of a single feedforward network that takes a feature vector (embedding) as an input and produces a single value indicating the probability that the input is the

encoding from a real video summary.

During training, several sets of highlight candidates (from multiple broadcasts) are passed through the generator to create a set of generated embeddings. This is then mixed with a pool of real embeddings generated by passing NHL-produced summaries through the generator's encoder. Next, the discriminator is presented with embeddings from both real and generated summaries and is tasked with distinguishing between the two. The discriminator is penalized for making misclassifications, and the generator is penalized when the discriminator makes correct classifications. Typically, this training procedure is very noisy, and the generator and discriminator's performance will continually oscillate due to the nature of their conflicting objectives. The overall adversarial system is trained in this manner until the generator can generate summaries that fool the discriminator with a high degree of accuracy.

Ideally, upon convergence during training, the generated summaries will have similar characteristics to the real summaries; if so, we have successfully trained a generator (and thus highlight selector) that creates realistic appearing summaries. This is not always the case, and there are several failure modes associated with training adversarial networks. For example, the discriminator may learn much more quickly than the generator, and "dominate" its adversary, resulting in the system failing to train properly. Alternatively, the system's performance may oscillate and diverge, resulting in poor performance from both the generator and the discriminator. In these cases, the hyperparameters associated with each of the components are adjusted (through trial and error), and the training procedure is repeated. The final set of hyperparameters we used are reported in Section 4.3.3. Finally, upon completion of training, the trained highlight selector is extracted from the generator and joined with the highlight detector to form the complete video summarization system.

4.3.3 Methodology

The exact structure and architecture of the adversarial networks we experiment with are given in Section A.2 in the Appendix.

Loss functions

The generator and discriminator characteristics are determined by the choice of training objectives (i.e., loss functions) used during the training procedure. We keep things simple for our application and use a standard binary cross-entropy loss (BCE) function for both the generator and discriminator. The BCE loss penalizes the generator when the discriminator makes a correct prediction and penalizes the discriminator when it makes an incorrect prediction. As for the autoencoder, we use a mean-squared-error (MSE) loss function to quantify the reconstruction error. The MSE loss function measures the autoencoder's capability to reconstruct the original input presented to the system. Recall that due to the information bottleneck present in the autoencoder architecture, this type of loss function encourages the encoder to learn compact representations (embeddings) of the input which preserve the information of the original input.

$$\begin{aligned} s &= S(x) \\ e &= A_e(s) \\ \hat{y} &= D(e) \\ \hat{x} &= A_d(e) \end{aligned} \tag{4.5}$$

Before we define the loss functions of the adversarial system, we first assign a symbol to all of the components and variables we have discussed so far. We denote the candidate frames (formatted as a feature matrix) as x, the reconstructed feature matrix as $\hat{x}$, the generated summary as s, the ground truth label as y (i.e., real or fake), the discriminator's predicted classification as $\hat{y}$, and the encoded representation of the input as e. The encoder is denoted by $A_e(s)$, the decoder as $A_d(e)$, the highlight selector as $S(x)$, and the discriminator as $D(e)$.

Equation 4.5 summarizes the flow of information through the adversarial system using the above notation. Moreover, using this notation, the generator and discriminator have their loss

functions defined by Equation 4.7 and Equation 4.8, respectively. Both of these loss functions are essentially just a standard BCE loss function (as discussed above). Additionally, the generator's loss function includes a reconstruction loss (i.e., mean-squared-error loss). As discussed in Section 4.3.1, this reconstruction loss is included to train the autoencoder component of the generator.

Finally, the overall loss function of the system is given by Equation 4.6 (see below), where the loss functions for the generator and discriminator are denoted by $\mathcal{L}_G$ and $\mathcal{L}_D$, respectively. This is an adversarial objective where the minimization of a component's loss function maximizes its counterpart's loss function.

$$\mathcal{L}_{\text{adv}} = \mathcal{L}_G + \mathcal{L}_D \tag{4.6}$$

$$\mathcal{L}_G(x, \hat{x}, y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) + \underbrace{||(x - \hat{x})||_2}_{\text{reconstruction error}} \tag{4.7}$$

$$\mathcal{L}_D(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{4.8}$$

Figure 4.5: The loss functions for the adversarial system. Equation 4.6 represents the overall training objective composed by the generator and discriminator's loss functions. The generator's loss function is given by Equation 4.7, and the discriminator's loss function is given by Equation 4.8. Both of these loss functions are based on BCE, which penalizes the networks for making incorrect predictions (i.e., when y differs from $\hat{y}$).

Training procedure

The adversarial system is trained using a standard deep learning pipeline. A batch of training samples are presented to the system, and an output is generated for each sample. The loss function is evaluated using the output, and an optimizer adjusts the parameters of the system.

In this case, the highlight candidates for each broadcast in the dataset are precomputed using the highlight detector to form the input dataset. This data is then divided into batches and randomly fed into the generator, which produces a generated summary and a corresponding embedding for each sample in that batch. Next, the NHL-produced summary corresponding to

that samples broadcast is fed into the encoder to produce an embedding from a "real" summary. The discriminator is then presented with embeddings from real and generated videos, and then tasked with making a classification for the inputs. The loss functions are subsequently evaluated based on the discriminators predictions, and then the loss is backpropagated to update the parameters of the system (as discussed in Section 4.3.1).

This procedure is repeated until the system's validation loss functions either converge or begin to diverge, at which point the training is stopped. We use a stochastic gradient descent (SGD) optimizer for training all system components, with a learning rate of $1\text{x}10^{-}2$ and a batch size of 1 or 2 (due to memory constraints).

Dataset

We use a dataset of 200 NHL hockey broadcasts (randomly) sourced from the highlight candidate dataset for our experiments. Each broadcast is paired with their NHL-produced summary video from the NHL summary dataset (as previously discussed in Section 4.2.4). The videos used in these experiments do not overlap with any of the videos used to train or evaluate the highlight detector designed in Chapter 3. Moreover, for the test evaluation (to be discussed in Section 5.2), we use another new partition of testing data to report our final results.

Lastly, this data are partitioned into a training and validation set. The training set is used to train the adversarial system, and the validation set is used to guide design decisions (i.e., we adjust the hyperparameters of the system based on the results of using the validation set). Upon completing all of our experiments, the test set is evaluated and used to report the final results.

Note that since we have access to NHL-produced summary videos, we could have used these as ground-truth for training a highlight selector in a supervised fashion (rather than the adversarial formulation we have discussed). In doing so, however, we would no longer be designing an "audience-informed" video summarization system, since we would no longer be using the reaction of the crowd and announcers to dictate what should be a highlight. This, of course, would be counter to what we are trying to accomplish in this thesis.

Preprocessing, data formatting, and hyperparameters

The generator takes a set of highlight candidates formatted as a matrix of frames as input. Each candidate, c_n, is represented as a three-dimensional stack of frames (with a height of L). The frames from all of the N candidates (per broadcast) are then stacked into a feature matrix, $C = [c_1, c_n, c_{n+1}...c_N]$.

These frames are obtained by subsampling L frames from each of the candidates at a reduced framerate and resolution. Since each of these clips' duration is variable, each candidate c_n is zero-padded and truncated to maintain a fixed length of L frames-per-candidate. Similarly, the number of detected highlights per broadcast is also variable, and therefore we keep the N best candidates and discard the remainder in order to maintain a fixed number of candidates per broadcast. Altogether, we chose to use: (1) $L = 15$, and $N = 30$, and (2) a resolution of 112x112 pixels sampled at 2 frames-per-second, with 3 channels dedicated for colour. Using these parameters, each candidate feature matrix, c_n, has dimensions of [15x3x112x112]. The resulting matrix of RGB images (for each broadcast) has a fixed shape of [30x15x3x112x112]; this matrix is now ready to be input to the generator.

The generator takes this [30x15x3x112x112] feature matrix and produces a binary mask indicating which of the 30 candidates should be selected using its highlight selector component. This mask is then multiplied with the candidate feature matrix, to produce a feature matrix in which specific columns are zeroed out. In doing so, the information of the unchosen candidates from the feature matrix, and therefore this processed matrix represents a summary generated from a subset of the candidates.

The next step is to encode the summary feature matrix data into an embedding, which can finally be input to the discriminator. The encoder accomplishes this and produces an M-dimensional embedding from the input feature matrix. For our application, we set $M = 256$, resulting in a 256-dimensional encoding for each summary produced by the generator. Finally, this embedding is input to the discriminator, which predicts the probability (on the range [0, 1]) that the input is a real summary, instead of a generated summary.

Conclusion

In this chapter, we discuss the design of the highlight selector component. Its purpose is to take a set of highlights (determined by cues from the audience and play-by-play announcers) and select which of them should be included in the generated summary. This is done because there are far more highlights in a hockey broadcast than there is space for in a hockey summary. We formulate two different approaches to the highlight selection problem: a value-based approach, and an adversarial learning-based approach. These techniques are fundamentally different, and therefore present their own set of strengths and weaknesses.

The value-based technique works by ranking the highlight candidates based on a value function and only selecting the best highlights to include in a summary. This technique is intuitive and straightforward—it allows the user to design the objective functions as they please, and therefore its results are also easily interpretable. However, the downside is that it is unclear as to what properties make a good video summary. Consequently, it is not easy to explicitly design objective functions that are truly conducive to generating good summaries.

The adversarial learning-based technique functions by using a system of deep networks that work in unison to select highlight candidates that produce compelling summaries automatically. This approach is motivated by recent works demonstrating that this technique is very useful for video summarization tasks. Another advantage is that we do not have to explicitly design any features which indicate the quality of a summary, and instead, we can leave that task for the network to learn on its own. While attractive in theory, adversarial-learning techniques tend to be much more complicated than conventional summarization techniques, and more challenging to train than a typical deep network.

In the next chapter, we examine the performance of both of these approaches evaluated on a testing set of data. Specifically, we evaluate how the two techniques differ in specific criteria such as accuracy, diversity, representation, and coherence—all of which are essential to forming an excellent video summary.

5

Video Summarization: Results

5.1 Overview

Thus far, we have discussed the design of the highlight detector and highlight selection components of the video summarization system. The highlight detector (Chapter 3) finds the most exciting moments of the broadcast based on the crowd and the play-by-play TV announcers. This is followed by a highlight selector (Chapter 4) that decides which of these should be kept in the final summary. The resulting end-to-end system is summarized in Figure 5.1. In this chapter, we present and discuss the results of our complete video summarization system when evaluated on a test set (composed of NHL hockey broadcasts). We accomplish this by using the video summarizer to produce a set of hockey summaries (from the testing set), and then analyzing which variations of our system (to be discussed shortly) produce the "best" summaries.

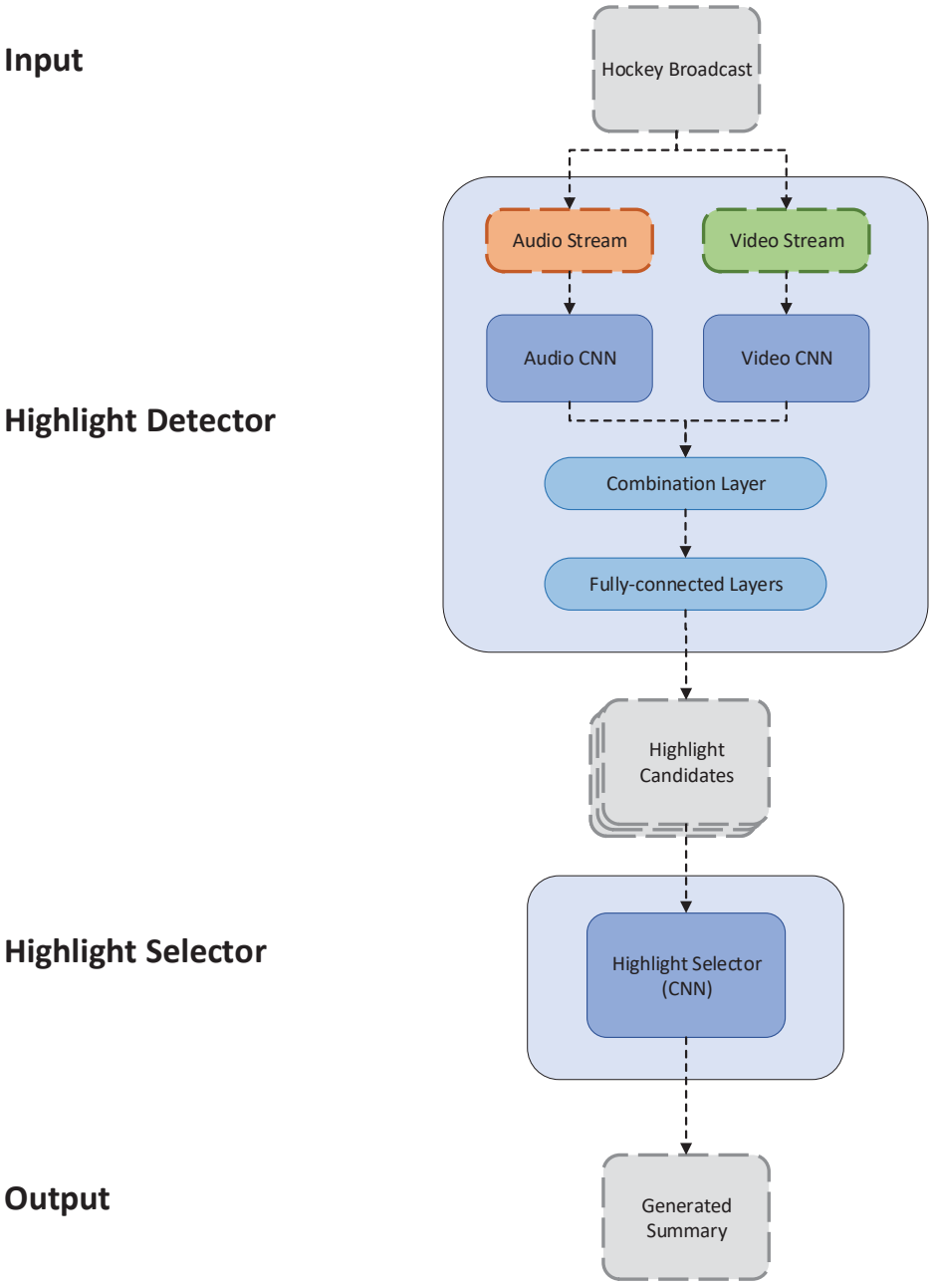

Figure 5.1: The complete end-to-end video summarization system. First, a set of highlight candidates are detected by a highlight detector (Chapter 3) based on information from the crowd and the play-by-play announcers. This set of highlight candidates are then filtered by a highlight selector (Chapter 4), which combines the selected highlights together to form a summary.

We begin by discussing the criteria and methodology that we use to evaluate the results of the summarization system. One of the most significant challenges was how to determine a meaningful way to evaluate the quality of a summary that could be explicitly defined. While a human can view a summary and subjectively conclude whether it is interesting or not (as a whole), it is unclear which *features* or *characteristics* of the summary led to that conclusion. This, of course, makes measuring the performance of a video summarization system extremely difficult, if not impossible.

Typically, to evaluate the quality of an automatically generated summary, most existing methods (e.g., [6], [7], [12], and [24]) will consider the overlap (or **accuracy**) between a particular automated summary under consideration and human-generated annotations that serve as ground-truth. We follow a similar approach, except with the distinction that our ground-truth is not derived by human annotators, but instead is based on the excitement of both the crowd and the play-by-play announcers. Since accuracy only gives us only one perspective on the quality of the summary, we also consider two additional criteria: representation and coherence, both of which are essential to forming a compelling hockey summary.

In total, we *analyze* five different variations of our video summarization system; these are comprised of the three value functions discussed in Section 4.2 and two different versions of the adversarial network discussed in Section 4.3. Finally, we use the three criteria mentioned above to evaluate the performance of each one of these variations of the video summarization system.

5.2 Evaluation

The quality of the summaries generated by any of the methods we have discussed so far (i.e., the value functions or adversarial networks from Chapter 4) is highly subjective and difficult to define analytically. To deal with this, we propose a set of criteria to measure the various characteristics of automatically generated summaries. In the absence of a single metric to *directly* (and automatically) measure the concept of "quality", this set of criteria will be employed to indicate how the characteristics of the summaries change with respect to the different techniques.

Recall from the literature review in Section 2.3, there are several existing metrics that attempt to measure the quality of a summary; common examples include accuracy, diversity,

redundancy, saliency, and representation [15, 26, 27, 29]. The choice of metric to use is somewhat *arbitrary*, but ultimately depends on the context of the application. In this case, we choose criteria that (we believe) are conducive to high "quality" hockey summaries.

The first criterion we consider is **"accuracy"**. We measure this by examining the generated summaries for the presence of false positives (i.e., footage without excitement from the crowd). Moreover, we also consider the presence of non-gameplay footage when measuring the accuracy of a summary (e.g., interviews, analyst desk commentary, etc.). These clips may contain excitement from the crowd but are still undesirable because they do not show the hockey game. Accuracy is mostly non-subjective and can be consistently measured. It is also a commonly used metric in the literature when analyzing the performance of a summarization technique [6, 7, 12, 13, 26]. For these reasons, this will be the metric that we primarily consider when judging the performance of a technique.

Following that, we also attempt to measure the **"representativeness"** of a summary. The rationale behind this is that a hockey summary should accurately portray the original hockey broadcast's content. For example, a good hockey summary should include most of the goals scored (or any other major event). Similarly, a hockey match is typically played across three gameplay periods, and therefore a representative summary should also display some content from *each* of the periods (i.e., the summary should be sufficiently *diverse*). Unlike accuracy, measuring the "representativeness" of a summary is subjective and difficult to report in a meaningful manner. This metric is also much less common in the literature, but it is still occasionally used to measure the quality of video summaries, e.g. [26] or [27]. Because of this, we take care to be precise and unambiguous with our definitions to minimize bias in the evaluation. Despite these drawbacks, including metrics other than just accuracy helps us better analyze how the summaries change with respect to different techniques.

Lastly, we consider if the generated summaries are **"coherent"** when viewed. A high-quality summary should be logical and consistent—any events that are displayed should be presented with some degree of context. For instance, if a highlight features a goal being scored in a hockey game, the summary should also include the moments leading up to that event. Likewise, any events being portrayed in the summary should be shown in its entirety (i.e., without being cut off). Similar to trying to measure "representativeness", this metric is highly subjective and difficult to report. As a result, this type of metric is rarely seen in the literature, with

some exceptions such as [5]. Regardless, it is a highly important characteristic for creating compelling hockey summaries and is therefore worth examining.

In total, the rubrics we use for evaluating the characteristics of the summaries is split into three sections: (1) accuracy, (2) representation, and (3) coherence. We evaluate all three criteria for each of the summaries generated using the testing set (discussed in Section 5.3).

5.2.1 Accuracy

The first metric we evaluate is the **"accuracy"** of the video summarization system. We compute this by examining whether there are any false positives or non-gameplay footage in the summaries. The definition of false positives follows the same criteria initially discussed in Section 3.2.1; that is, a false positive is a shot that does not contain any *audible or visible* excitement from the crowd or the play-by-play announcers. For example, a shot which portrays a hockey player skating across the ice, without any reaction from the crowd or announcers, would be considered a false positive. As for non-gameplay footage, we define this as footage included in the summary which does not show the hockey game. Common examples of this are interviews or mid-game events that take place in the broadcast between the hockey gameplay periods. While such footage may exhibit excitement from the crowd, they are undesirable because they do not represent the hockey game. Since this metric can be reliably measured with minimal bias (i.e., it is mostly independent of the evaluator), we use this is as the primary metric for judging the capability of a technique.

False positives and non-gameplay footage are highly detrimental to the viewing experience of a video—even the presence of a few erroneous clips (i.e., clips that are not highlights) can make a viewer uninterested and confused. It follows that we should select a variation of our video summarizer that is highly accurate.

Recall that the complete video summarization system is a two-step procedure formed by the combination of a highlight detector and a highlight selector. While the highlight detector we have designed is highly accurate (i.e., we observed approximately 98% accuracy in the experiments of Section 3.5), false positives still inevitably appear in the summaries.

This is because the highlight detector does not analyze the entire hockey broadcast at the same time, but instead only a small segment at a time. As previously discussed (in Section

3.3), the hockey broadcast is split into many "chunks" which each represent a few seconds of gameplay. Each of these chunks is determined to be either "highlight" or "non-highlight" by the highlight detector, and therefore the highlight detector may have to make thousands of predictions per hockey broadcast. *Consequently, even if the highlight detector is highly accurate, there will still be several false positives per broadcast*. With that being said, the number of false positives is also mitigated by the highlight selector in the second stage of the summarization process. The highlight selector acts as a secondary filter for false positives by filtering the highlights detected by the highlight detector.

For our evaluation, we specify that a summary with two or fewer false positives is satisfactory. We determine this for each summary generated using the testing set and repeat the procedure for each variation of the video summarization system we are considering. This process is implemented by visually inspecting each of the summaries generated using the testing set. Although this may seem unnecessary, we choose to compute this metric visually to stay consistent with the other metrics that we will be soon discussing (i.e., representation and coherence), which are not feasible to automate.

5.2.2 Representation

Next, the summaries produced by our system should be **"representative"** of the entire hockey broadcast they are summarizing. That is, the summary should capture most (if not all) of the *significant* events that occur during the game and leave the viewer with an *accurate depiction of the game*. For example, the most significant events of a hockey game are the goals (in general), and therefore a representative summary of a hockey broadcast should portray most of the goals that occur during a game. If this were not the case, the viewer might not receive an authentic depiction of the hockey game.

Additionally, a hockey game is played across several gameplay periods. Following the same reasoning as above, a good summary should ensure that each of these periods is covered to some extent. For instance, even if one gameplay period was significantly more interesting than the others, the summary should still minimally cover all of the periods to represent the game properly.

Unlike the case with accuracy, we have not explicitly designed the video summarization

system to generate representative summaries. Specifically, neither the highlight detector nor the highlight selector is designed (or trained) with objective functions that reward any of the criteria discussed above. Regardless, we expect that different variations of the video summarizer will produce summaries that differ in this characteristic (which, therefore, makes this criterion worth observing).

We define that a summary is satisfactorily representative if it features at least one clip from each of the gameplay periods in the broadcast, and at least one clip for the majority of the goals scored during the game (there are typically three periods in a hockey game). As in the case of determining accuracy, this characteristic is manually evaluated for each summary under consideration (i.e., each summary generated using the testing set). We perform the evaluation visually due to the lack of a practical way to automate this process (i.e., automating the evaluation would be a significant challenge in itself).

5.2.3 Coherence

Lastly, and perhaps the most difficult to determine, is the notion of creating **"coherent"** video summaries. A hockey summary should tell a clear and logical story about the hockey broadcast it is summarizing when viewed. In other words, the summary should be free of footage that is presented without context or abruptly interrupted. This property, like the two previously discussed characteristics, is essential to forming compelling hockey summaries.

For example, if a hockey team scores a goal on the opposing team, the video should show a few seconds of the build-up to that moment, and also a few seconds of the aftermath. Similarly, when the summary portrays an event taking place (such as a play on the opposition's goal), the footage should show the event in its entirety, without being abruptly cut short. Without these crucial moments before and after an event, the viewer is left confused without context or is left unsatisfied. Similar to the presence of false-positives in a summary, the inclusion of incoherent footage is highly detrimental to the viewing experience of a hockey summary, and should be avoided as much as possible.

Similar to the case with representation, we do not explicitly design the video summarization system to generate summaries that are "coherent". With that being said, we expect that the summaries generated using different variations of our video summarizer, will produce summaries

that vary greatly in their level of "coherence".

We define that a summary with two or fewer incoherent clips is satisfactory for the purposes of our evaluation. As in the case with the previously discussed criteria, this is visually evaluated for all summaries under consideration due to the lack of a practical way to automate this process.

5.3 Results

To report our results, we visually review the summaries generated from the hockey broadcasts in the testing set and record whether each summary satisfies the criteria discussed above (i.e., accuracy, representation, and coherence). This procedure is repeated for each variation of the video summarization system we are considering; our evaluation results are summarized in Table 5.1.

Furthermore, we examine two different approaches for forming hockey summaries, split into a total of five variations. The first value-based approach consists of the three value functions discussed in Section 4.2. The second adversarial-based approach consists of two versions of the adversarial network as discussed in Section 4.3. All of the approaches use the highlight detector designed in Chapter 3 but differ in the choice of highlight selector, as is discussed in Chapter 4. As for the testing dataset, we use a set of 30 broadcasts from the 2018-2019 NHL hockey season. These broadcasts are a separate partition of hockey games and have not been used to previously train or validate any of the video summarization system components.

We assign a score for each characteristic (i.e., criterion) based on the fraction of the summary videos that satisfy the constraints discussed above. For example, if every video produced by a given technique is free of false positives (and non-gameplay footage), it would score a perfect 100 on the accuracy category. By examining the outputs produced by each of these approaches, we find that each technique produces summaries with greatly varying characteristics.

Summarization method versus criteria scores

Technique	Accuracy	Representation	Coherence	Average
Probability-weighted (baseline)	53.3	50.0	50.0	51
Duration-weighted	53.3	60.0	**66.7**	**60**
Proximity-weighted	50.0	**66.7**	30.0	49
Adversarial (ResNet18)	53.3	36.7	60.0	50
Adversarial (ResNet34)	**56.7**	33.3	63.3	51

Table 5.1: The results from evaluating five variations of our video summarization system on a testing set. Three variations are based on value-functions, and the other two are based on adversarial learning. Each variation is used to produce a set of generated summaries from a testing set of NHL broadcasts. The resulting summaries are then visually reviewed, and a score for each criterion is computed out of 100, based on the fraction of the summaries that are satisfactory.

5.4 Discussion

5.4.1 Value based approaches

We found that the value-based approaches performed quite well and definitely above expectations. We experimented with three different value functions: probability-weighted, duration-weighted, and proximity-weighted. Each variation emphasizes different criteria when forming the summaries.

The simplest of these techniques is the **probability-weighted** approach, in which highlight candidates are selected solely based on the probability of being a highlight predicted by the highlight detector (i.e., the highlight score, as previously discussed). As such, no emphasis is placed on any of the properties of the candidates themselves (e.g., when the highlight occurs or the duration of the highlight). For this reason, we consider this first experiment as the **baseline**—it effectively reflects how the highlight selector behaves if it were to only consider the highlight scores of each candidate, and not the overall composition formed by the selected highlights. This contrasts with the other value-based approaches (and the adversarial approach) which consider the *relationship* of a highlight candidate with respect to one another. Therefore,

by comparing the performance of this approach to the others, we can determine if the highlight selector is learning to find desirable relationships between highlight candidates (which ultimately help form higher quality video summaries).

Overall, we found that this probability-weighted approach works quite well despite its simplicity. The approach produces summaries with a low number of false positives and a relatively low amount of non-gameplay footage. As for its weaknesses, this approach struggles the most with representation and coherence. We found that most hockey broadcasts had specific gameplay periods that were predicted (by the highlight detector) to be significantly more interesting than others. As a result, the summaries produced by this approach tended to overrepresent certain portions of the broadcast. Consequently, many of the summaries produced by this technique failed to portray the game accurately; for example, by missing many of the goals scored in the game, or not having any footage from entire gameplay periods. We consider this technique as our baseline approach, which attains an average score of 0.50 across all of the criteria.

The next technique we looked at was the **duration-weighted** approach. In this case, the value function of the highlight selector weighs the candidates based on their duration in addition to its highlight score. Compared to the probability-weighted approach, this changed the composition of the resulting summaries drastically. Most notably, the summaries were formed by significantly fewer candidates (where each tended to have a higher duration). This had the effect of smoothing out the video since there were much fewer transitions in the overall summary (i.e., much less jumping around from clip to clip). As a result, these summaries scored much higher on the coherence criterion, since it was more likely that events portrayed in the summary were presented in their entirety (as opposed to being abruptly cut-off). Moreover, the summaries produced by this approach maintained a high degree of accuracy (i.e., a low number of false positives) and is tied with the probability-weighted approach in performance. On average, this is the highest performing technique, with an average score of 0.60 across all of the criteria.

Lastly, we examined a **proximity-weighted** approach. The rationale behind this is that several highlights occurring in close proximity to one another may be more interesting than a single highlight occurring in isolation. In practice, however, we found that the summaries produced by this approach were not compelling. We found that valuing candidates based on their proximity introduced many false positives in the summaries. This is due to the inclusion of many low

probability highlights (which turned out to be false positives) occurring near each other. Consequently, the proximity-weighted approach scores poorly on the accuracy and coherence criteria compared to the two other approaches we have examined. Of the value-based approaches, this technique has the lowest average score of 0.49.

5.4.2 Adversarial learning based approaches

In our second set of experiments, we examined the use of an adversarial network to produce video summaries. We experimented with two versions of the highlight selector, where the first is based on ResNet18 [42] and the second is based on ResNet34 [42] (i.e., two different CNNs). These two deep networks share approximately the same overall architecture and differ in the number of layers in the network. While there are several other CNN architectures to consider (e.g., VGG [41] or Inception [43]), we found that these other networks required computational resources (i.e., memory usage) in excess of commodity hardware, or had significant issues converging during the training procedure.

We ultimately obtained mixed levels of success with this approach. The adversarial approaches performed respectably on all categories except for representation, where it struggled compared to all of the other techniques we have tried so far. Overall, the summaries produced using the adversarial approach had a low number of false positives and were free of non-gameplay footage. As for the other qualities, this approach performed similarly to many of the earlier discussed techniques and suffered from the over-representation of certain portions of the broadcast. Thus, the summaries failed to portray the entirety of the hockey broadcast accurately, and missed many critical events during the game (i.e., goals), leading to a low representation score. We did not observe a significant difference in the summaries produced by the ResNet18-based and ResNet34-based highlight detectors—the two variations achieved an average score of 0.50 and 0.51, respectively. In comparison to the value-based approaches, this makes the adversarial techniques approximately equal to the baseline (i.e., the probability-weighted approach discussed above).

While adversarial approaches are attractive on paper, as it allows us to train a video summarizer without annotated data or explicitly designed features, there are also several practical drawbacks. The first of which is the unstable nature of training adversarial networks, which

often require extensive experimentation and tuning to train properly. Most of this tuning, unfortunately, is done by trial and error.

The next drawback is the lack of interpretability of the network; it is unclear which parameters (or design choices) affect the characteristics of the output. This, of course, makes the system difficult to apply in practice. For example, if the video summarizer was producing summaries with a certain undesirable quality, it is unclear which parameter would need to be adjusted—this is not the case when using value functions. Despite these drawbacks, the overarching technology is still very promising and worth further investigation. Future work could experiment with different architectures, problem formulations, and incorporate techniques to improve interpretability or the training procedure's stability.

5.4.3 Conclusion

In this chapter, we examined the results of our complete video summarization system. This system was formed by joining the highlight detector designed in Chapter 3, and the highlight selector designed in Chapter 4. The highlight detector finds potential highlights in the hockey broadcast by detecting cues of excitement from the crowd and the play-by play announcers. Due to the limited amount of time allocated to a hockey summary, the highlight selector decides which of these highlights should be included in the resulting summary.

For our experiments, we use our most accurate highlight detector (as concluded in Section 3.5), and then pair it with five different variations (split into two approaches) of the highlight selector. This corresponds to the three value-based variations, and two adversarial-based variations discussed in Section 4.2 and Section 4.3, respectively. We evaluate each of these variations by inspecting the summaries generated by each technique using a testing set of NHL broadcasts. To facilitate this process, we considered three criteria that are conducive to compelling hockey summaries: (1) accuracy, (2) representation, and (3) coherence.

Overall, we found that all of the approaches we examined had their strengths and weaknesses, with neither being clearly superior to the other in all aspects. Using an explicitly-designed value function is by far the most straightforward and interpretable of the two approaches (i.e., value-function based or adversarial learning based). It allows us to specify which characteristics should be present in the resulting summaries and optimize for them accordingly.

5.4 Discussion

Altogether, we found that the duration-weighted approach produced the best hockey summaries overall, according to this set of criteria. Unlike the other variations we examined, the duration-weighted approach performs consistently across all criteria, instead of excelling for a certain criterion and performing poorly on another. With that being said, the adversarial-based approaches demonstrated that they are viable and competitive with the value-based approaches. The adversarial approaches allow us to generate summaries without having to explicitly design any features or value functions but suffer from several practical issues such as a lack of interpretability (i.e., it is difficult to determine what parameters affect the output) and the instability of the training process. Future work could address some of these issues and significantly improve the viability of adversarial learning-based approaches.

6

Conclusion

In this work, we have investigated several techniques for the automatic summarization of hockey broadcasts. These summaries allow a viewer to quickly enjoy or preview the contents of a hockey broadcast without having to commit several hours of their time. While highlight videos are an effective and accessible way for viewers to watch hockey broadcasts, they are typically manually generated—a time consuming and expensive process. For this reason, we designed an end-to-end video summarization system, which *automatically* finds and composes the most exciting moments of a hockey broadcast.

The first challenge we addressed was determining the process for finding the most exciting moments of the hockey broadcast. Typically, this is accomplished by training a model to detect highlights (e.g., a deep network) using an annotated dataset of videos. However, these annotations are usually generated by humans, which can lead to significant bias due to the subjective nature of determining what constitutes a highlight. Instead, our approach leverages the *spectators* of the hockey game (i.e., the audience and the play-by-play announcers) to determine which parts of a hockey broadcast should be considered a highlight.

To do this, we design a highlight detector that analyzes the broadcast for cues such as cheering from the crowd or excitement in the announcers' voices. By using information from the human spectators, we have effectively collected an *ensemble* of opinions on the current level of excitement in the game. This allows us to bypass the need for a human-annotated dataset, which is time-consuming and expensive to manually annotate, especially for high duration videos such as hockey broadcasts. At the same time, we mitigate potential bias in the annotations by using

the collective opinion of a crowd, instead of a handful of annotators.

We build this highlight detector using a multi-modal CNN that simultaneously utilizes information from both the *visual* and *auditory* streams of the video broadcast. By analyzing *multiple modes* of a video, we can detect cues from the audience that may not be obvious when only looking at only the video or listening to the audio on its own. Through experimentation, we find that using multiple modes (to train a deep network) yields a significant performance increase over just using visual or audio data alone. To train the highlight detector, we leverage existing datasets that contain audio samples from relevant categories such as cheering or human speech. In doing so, we can *pretrain* our highlight detector on these datasets, and significantly reduce the amount of data required to train our highlight detector. Using this procedure, we experimented with numerous CNN architectures and produced a highlight detector that is able to classify video clips with an accuracy of 98%.

Next, we address the issue of generating compelling video summaries. In contrast to many other approaches, which typically detect highlights and join them into a summary without further processing, we explicitly design a highlight selector to choose which highlights should actually be included in the final summary. We examine two different approaches to constructing this highlight selector. The first approach frames the task as an optimization problem. Each highlight candidate is assigned a value based on its properties (e.g., duration or proximity to other candidates) by using a value function that is subsequently optimized. The second approach takes a more indirect approach and utilizes a system of deep networks trained using adversarial learning. Through adversarial learning, we train a generator network that learns to produce hockey summaries with similar characteristics to professionally produced NHL hockey summaries.

We analyze the effectiveness of both of these approaches using a rubric composed of both quantitative and qualitative criteria (i.e., accuracy, representation, and coherence). Our findings indicate that neither approach is clearly superior to the other, with both techniques presenting their own set of strengths and weaknesses (in terms of performance and practicality). The value-based technique is straightforward, interpretable, and practical, while still being able to generate compelling hockey summaries. The adversarial learning-based approach, on the other hand, foregoes the need to explicitly design a value function, and instead can automatically learn to generate hockey summaries by leveraging NHL-produced hockey summaries.

Conclusion

In conclusion, we present a video summarization system for hockey broadcasts that detects *highlights* based on information from the audience of the hockey game. We demonstrate the effectiveness of using multi-modal CNNs to search hockey broadcasts for cues of excitement from the audience, and how this can be used for detecting highlights. Next, we present two different approaches for *selecting* which of the highlights detected in a broadcast should be included in the final summary. We examine how the characteristics (e.g., representation and coherence) of a hockey summary changes with respect to the choice of highlight selection technique. Altogether, this work presents an end-to-end video summarization system capable of generating compelling hockey summaries in a practical manner.

Bibliography

[1] A. Kanehira, L. Van Gool, Y. Ushiku, and T. Harada, "Viewpoint-Aware Video Summarization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (Salt Lake City, UT), pp. 7435–7444, IEEE, June 2018. 12.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017. 556.

[3] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1346–1353, June 2012. 546.

[4] Y. Li and B. Merialdo, "Multi-video summarization based on AV-MMR," in *2010 International Workshop on Content Based Multimedia Indexing (CBMI)*, pp. 1–6, June 2010. 26.

[5] Z. Lu and K. Grauman, "Story-Driven Summarization for Egocentric Video," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2714–2721, June 2013. ISSN: 1063-6919.

[6] T. Yao, T. Mei, and Y. Rui, "Highlight Detection with Pairwise Deep Ranking for First-Person Video Summarization," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 982–990, June 2016. 144.

[7] A. Tejero-de Pablos, Y. Nakashima, T. Sato, N. Yokoya, M. Linna, and E. Rahtu, "Summarization of User-Generated Sports Video by Using Deep Action Recognition Features," *IEEE Transactions on Multimedia*, vol. 20, pp. 2000–2011, Aug. 2018. 18.

[8] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 221–231, Jan. 2013. 3268.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, June 2014. 4000.

[10] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action Recognition using Visual Attention," *arXiv:1511.04119 [cs]*, Feb. 2016. arXiv: 1511.04119.

[11] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *arXiv:1705.07750 [cs]*, Feb. 2018. arXiv: 1705.07750.

[12] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video Summarization with Long Short-Term Memory," in *Computer Vision - ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), Lecture Notes in Computer Science, (Cham), pp. 766–782, Springer International Publishing, 2016. 261.

[13] Z. Ji, K. Xiong, Y. Pang, and X. Li, "Video Summarization with Attention-Based Encoder-Decoder Networks," *arXiv:1708.09545 [cs]*, Apr. 2018. 45.

[14] R. Agyeman, R. Muhammad, and G. S. Choi, "Soccer Video Summarization Using Deep Learning," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 270–273, Mar. 2019. 3.

[15] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkilä, and N. Yokoya, "Video Summarization using Deep Semantic Features," *arXiv:1609.08758 [cs]*, Sept. 2016. 48.

[16] M. Rochan, L. Ye, and Y. Wang, "Video Summarization Using Fully Convolutional Sequence Networks," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), vol. 11216, pp. 358–374, Cham: Springer International Publishing, 2018. 28.

[17] Yale Song, J. Vallmitjana, A. Stent, and A. Jaimes, "TVSum: Summarizing web videos using titles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, MA, USA), pp. 5179–5187, IEEE, June 2015. 220.

[18] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating Summaries from User Videos," in *Computer Vision – ECCV 2014*, vol. 8695, pp. 505–520, Cham: Springer International Publishing, 2014. 329.

[19] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury, "Weakly Supervised Summarization of Web Videos," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice), pp. 3677–3686, IEEE, Oct. 2017. 30.

[20] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "A^2-Nets: Double Attention Networks," *arXiv:1810.11579 [cs]*, Oct. 2018. 55.

[21] C. Lin and Y. Chen, "Sports Video Summarization with Limited Labeling Datasets Based on 3D Neural Networks," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Sept. 2019. 0.

[22] M. Merler, K.-N. C. Mac, D. Joshi, Q.-B. Nguyen, S. Hammer, J. Kent, J. Xiong, M. N. Do, J. R. Smith, and R. S. Feris, "Automatic Curation of Sports Highlights Using Multimodal Excitement Features," *IEEE Transactions on Multimedia*, vol. 21, pp. 1147–1160, May 2019. 10.

[23] Chong-Wah Ngo, Yu-Fei Ma, and Hong-Jiang Zhang, "Automatic video summarization by graph modeling," in *Proceedings Ninth IEEE International Conference on Computer Vision*, (Nice, France), pp. 104–109 vol.1, IEEE, 2003. 131.

[24] Z. Xiong, A. Divakaran, A. Divakaran, K. A. Peker, K. A. Peker, R. Radhakrishnan, R. Radhakrishnan, R. Cabasson, and R. Cabasson, "Video Summarization Using Mpeg-7 Motion Activity And Audio Descriptors," in *ISO/IEC 21000-7 FDIS, "Information Technology - Multimedia Framework - Part 7: Digital Item Adaptation*, Kluwer Academic Publishers, 2003. 77.

[25] D. Liu, Gang Hua, and Tsuhan Chen, "A Hierarchical Visual Model for Video Object Summarization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2178–2190, Dec. 2010. 104.

[26] A. G. del Molino, C. Tan, J.-H. Lim, and A.-H. Tan, "Summarization of Egocentric Videos: A Comprehensive Survey," *IEEE Transactions on Human-Machine Systems*,

vol. 47, pp. 65–76, Feb. 2017. Conference Name: IEEE Transactions on Human-Machine Systems.

[27] X. Li, B. Zhao, and X. Lu, "A General Framework for Edited Video and Raw Video Summarization," *IEEE Transactions on Image Processing*, vol. 26, pp. 3652–3664, Aug. 2017. 41.

[28] W. Wolf, "Key frame selection by motion analysis," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 2, pp. 1228–1231 vol. 2, May 1996. 616.

[29] Y.-F. Ma, L. Lu, H.-J. Zhang, and M. Li, "A user attention model for video summarization," in *Proceedings of the tenth ACM international conference on Multimedia*, MULTIMEDIA '02, (Juan-les-Pins, France), pp. 533–542, Association for Computing Machinery, Dec. 2002. 638.

[30] Y. Gong and X. Liu, "Video summarization using singular value decomposition," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, pp. 174–180 vol.2, June 2000. 224.

[31] B. Li, H. Pan, and I. Sezan, "A general framework for sports video summarization with its application to soccer," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, vol. 3, pp. III–169, Apr. 2003. 74.

[32] D. Tjondronegoro, Y.-P. P. Chen, and B. Pham, "Sports video summarization using highlights and play-breaks," in *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, MIR '03, (Berkeley, California), pp. 201–208, Association for Computing Machinery, Nov. 2003. 52.

[33] A. Ekin, A. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *IEEE Transactions on Image Processing*, vol. 12, pp. 796–807, July 2003. 1029.

[34] E. Keane, P. Desaulniers, L. Bornn, and M. Javan, "Data-driven lowlight and highlight reel creation based on explainable temporal game models," *Sloan Sports Conference*, p. 15, 2019. 0.

[35] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised Video Summarization With Adversarial LSTM Networks," pp. 202–211, 2017. 137.

[36] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond Short Snippets: Deep Networks for Video Classification," *arXiv:1503.08909 [cs]*, Apr. 2015. arXiv: 1503.08909.

[37] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, pp. 357–366, Aug. 1980. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing.

[38] L. Muda, M. Begam, and I. Elamvazuthi, "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques," vol. 2, no. 3, p. 6, 2010. 794.

[39] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions," *arXiv:1712.05884 [cs]*, Feb. 2018. 500.

[40] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN Architectures for Large-Scale Audio Classification," *arXiv:1609.09430 [cs, stat]*, Jan. 2017. arXiv: 1609.09430.

[41] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015. arXiv: 1409.1556.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[43] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv:1602.07261 [cs]*, Aug. 2016. arXiv: 1602.07261.

[44] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.

[45] M. Rochan and Y. Wang, "Video Summarization by Learning from Unpaired Data," *arXiv:1805.12174 [cs]*, Apr. 2019. arXiv: 1805.12174.

[46] A. Shaheen and A. Sleit, "Comparing between different approaches to solve the 0/1 Knapsack problem," *International Journal of Network Security*, vol. 16, pp. 1–10, July 2016.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[48] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *arXiv:1710.10196 [cs, stat]*, Feb. 2018. arXiv: 1710.10196.

[49] J. Depois, "Latent space visualization - Deep Learning bits #2." Library Catalog: hackernoon.com.

[50] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," *Proceedings of ICML*, vol. 27, pp. 807–814, 06 2010.

Appendices

A

Experiment Parameters

In this appendix, we give the exact deep network architectures used to conduct the experiments in Chapter 3 and Chapter 4.

In total, we use three different preconfigured CNN architectures in our experiments: VGG [41], ResNet [42], and Inception [43]. Typically, we connect these CNNs using a series of fully-connected or activation layers. We use the shorthand notation "FC-N" to represent a fully-connected layer with "N" nodes. Likewise, we use the notation "ReLU" to represent a rectified-linear-unit [50] activation layer.

A.1 Chapter 3: Highlight detector architectures

The architecture for the highlight detector in the early fusion and late fusion configurations are given as follows. Detailed discussion regarding the design, intent, and function of these networks can be found in Chapter 3.

A.1 Chapter 3: Highlight detector architectures

Table A.1: Early-fusion highlight detector architecture

[Video]	[Audio]
Combination Layer (i.e., concatenation)	
3D-ResNet or 3D-VGG or 3D-Inception	
Temporal pooling	
FC-512	
ReLU	
FC-256	
ReLU	
FC-1	

Table A.2: Late-fusion highlight detector architecture

[Video]	[Audio]
3D-ResNet or 3D-VGG or 3D-Inception	ResNet or VGG or Inception
Temporal pooling + FC-512	FC-128
Combination Layer (i.e., concatenation)	
FC-620	
ReLU	
FC-512	
ReLU	
FC-1	

A.2 Chapter 4: Adversarial neural network architectures

The architecture for the highlight selector, encoder, decoder, and the discriminator are given as follows. Detailed discussion regarding the design, intent, and function of these networks can be found in Chapter 4.

Table A.3: [Generator] Highlight selector architecture

[Video]
3D-ResNet18 or 3D-ResNet34
FC-15360 (30 * 512)
ReLU
FC-512
ReLU
FC-256
ReLU
FC-30

Table A.4: [Generator] Encoder architecture

[Video]
3D-ResNet18 or 3D-ResNet34
Temporal pooling
FC-512
ReLU
FC-512

A.2 Chapter 4: Adversarial neural network architectures

Table A.5: [Generator] Decoder architecture

[Video embedding (512)]
FC-512
ReLU
FC-2205
ReLU
FC-4410
ReLU
FC-8820
Reshape (8820 → 15x3x14x14)
4x Bilinear Upsampling (15x3x14x14 → 15x3x224x224)

Table A.6: [Discriminator] Feedforward network architecture

[30x Video embedding (512)]
FC-15360 (30 * 512)
ReLU
FC-1024
ReLU
FC-512
ReLU
FC-1