

MULTI-LABEL MULTI-MODAL CLASSIFICATION OF MOVIE SCENES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

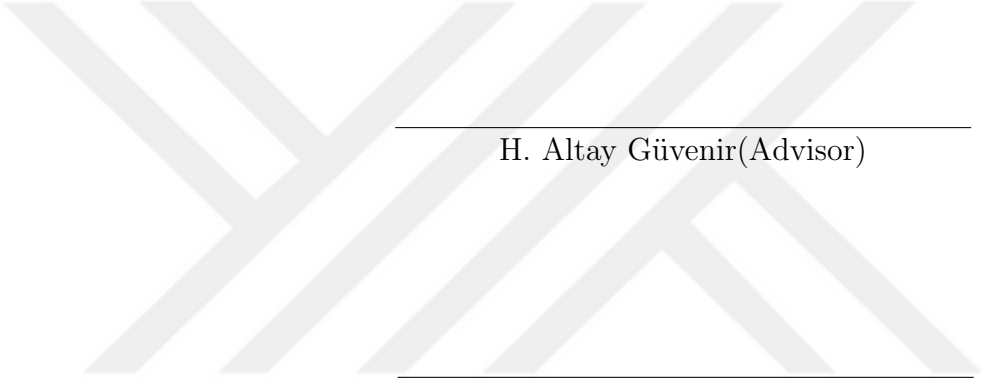
By
Irmak Türköz
September 2022

Multi-Label Multi-Modal Classification of Movie Scenes

By Irmak Türköz

September 2022

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



H. Altay Güvenir(Advisor)

Hamdi Dibekliolu

Nazlı İkizler Cinbiş

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

MULTI-LABEL MULTI-MODAL CLASSIFICATION OF MOVIE SCENES

Irmak Türköz
M.S. in Computer Engineering
Advisor: H. Altay Güvenir
September 2022

Promoting movies through their trailers provides valuable information that can help viewers and investors form expectations about the movie’s future success. Recent research confirmed that the audience prefers to watch movies through at-home-streaming services rather than at the theaters which resulted in movie trailers being shown privately. Moreover, advertisements created for different interest groups can provide a drastically improved experience for users and advertisers alike.

There have been few attempts to automatize the trailer generation process however, AI-generated trailers were considered less attractive than editors’ creations. Fortunately, the use of the most recent advancements in deep learning and the greater availability of datasets can accelerate the automated trailer generation process. Every movie produced is labeled with a set of genres that it represents. Thus, it is possible to generate multiple trailers of the same movie for different genres to offer personalized advertisements to the audience. To the best of our knowledge, personalized advertisements of movies via genre-specific trailers will be the first attempt in the automated trailer generation studies.

For this task, we needed a tool that extracts representative scenes of a particular genre from a given movie. Then, these scenes can be concatenated to form a draft of a trailer for each genre. The draft can be finalized through the creative post-production process. In this thesis, we developed a deep learning network that classifies scenes into a set of genres. In order to construct a training dataset to train this network, we compiled a set of scenes that are labeled with their representative genres.

Our network accomplishes a multi-label classification task with hyperparameters learned from experimental binary models. The learning process comprises the use of visual features, audio features, and their combination. The final result of the model is evaluated by comparing its classification performance with human perception.



Keywords: Movie Trailer Labeling, Scene Understanding, Deep Learning, Computer Vision.

ÖZET

FİLM SAHNELERİNİN ÇOK ETİKETLİ KARMA MODEL İLE SINIFLANDIRILMASI

Irmak Türköz
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: H. Altay Güvenir
Eylül 2022

Filmleri fragmanları aracılığıyla tanıtmak, izleyicilerin ve yatırımcıların filmin gelecekteki başarısı hakkında beklentiler oluşturmaya yardımcı olabilecek değerli bilgiler sağlar. Yakın zamanda yapılan araştırmalar, izleyicilerin sinema salonları yerine evde dijital yayın platformları aracılığıyla film izlemeyi tercih ettiğini doğruladı ve bu da film fragmanlarının kişiye özel olarak gösterilmesine neden oldu. Ayrıca, farklı ilgi grupları için oluşturulan reklamlar, hem kullanıcılar hem de reklam verenler için önemli derecede iyileştirilmiş bir deneyim sağlayabilir.

Geçmişte fragman oluşturma sürecini otomatikleştirmek için birkaç girişimde bulunuldu, ancak yapay zeka tarafından oluşturulan fragmanlar, editörlerin yarattıklarından daha az çekici olarak kabul edildi. Derin öğrenmedeki en son gelişmelerin kullanımı ve veri kümelerinin daha fazla kullanılabilirliği, otomatik fragman oluşturma sürecini hızlandırabilir. Üretilen her film, temsil ettiği bir dizi türle etiketlenmiştir. Böylece her filmin temsil ettiği her tür için ayrı fragmanını oluşturularak izleyiciye kişiselleştirilmiş reklam sunulabilir. Bildiğimiz kadarıyla, türe özel fragmanlar aracılığıyla filmlerin kişiselleştirilmiş reklamları, otomatik fragman oluşturma çalışmalarında ilk deneme olacaktır.

Bu görev için belirli bir filmde belirli bir türün temsili sahnelerini çıkaran bir araca gereksinim vardır. Bu sahneler her tür için bir fragman taslağı oluşturmak üzere birleştirilir. Taslak, yaratıcı post prodüksiyon sürecinden geçirilir. Bu tezde, sahneleri bir dizi türe ayıran bir derin öğrenme ağı geliştirdik. Bu ağı eğiten bir öğrenim veri seti oluşturmak için, temsili türleriyle etiketlenmiş bir dizi sahne derledik.

Ağımız, deneysel ikili modellerden öğrenilen hiper parametrelerle çok etiketli bir sınıflandırma görevini yerine getirdi. Öğrenme süreci, görsel özelliklerin, işitsel

özelliklerin ve bunların kombinasyonunun kullanımını içererek gerçekleştirdi. Modelin nihai sonucundaki sınıflandırma performansı, insan algısı ile karşılaştırılarak değerlendirildi.



Anahtar sözcükler: Film Fragmanı Etiketleme, Sahne Tanımlama, Derin Öğrenme, Bilgisayarlı Görü.

Acknowledgement

First of all, I want to express my immense gratitude to my supervisor Prof. H. Altay Güvenir, for generating the key ideas to our approaches, for his guidance throughout every step of this research, and always giving valuable feedback and support in the process.

I would like to express special thanks to Dr. Sami Arpa, the founder of Largo Films company, for valuable discussions in this research and for sharing his knowledge about movie industry¹.

I am also thankful for support given by undergraduate students of Bilkent University Machine Learning Group members of Lara Fenercioğlu, and Elif Pınar Balbal for their contributions to my thesis.

I want to express my gratitude to my dearest friend, Bartu Soykök for sharing his machine learning knowledge with me and supporting me in the every step.

Finally, I want to thank my parents for their support in any direction I have chosen, and my sister and her husband for their supervision.

¹Largo Films SA. EPFL Innovation Park, Building I. 1015, Lausanne/ SWITZERLAND. info@largofilms.ch

Contents

1	Introduction	1
1.1	Background Information	1
1.2	Motivation	4
1.3	Contributions	6
1.4	Organization of the Thesis	7
2	Literature Review	8
2.1	Related Work	8
2.1.1	Trailer to Genre Classification	9
2.1.2	Automated Trailer Generator	14
2.2	Foundation Dataset	16
2.2.1	MovieLens-20M	16
3	Preparation of Dataset	18
3.1	MovieLens20M Data Downloader	19

3.2	Scene Extractor	20
3.3	Removal of Post-production Components	21
3.3.1	Elimination Model Dataset	22
3.3.2	Elimination Model Architecture	24
3.3.3	Manual Scene Cleaning	25
3.3.4	Manually and Automatically Cleaned Datasets	27
4	Scene to Genre Classification	30
4.1	Visual Training	30
4.1.1	Data Preparation	30
4.1.2	Binary Model Architectures	36
4.1.3	Multi-Label Prediction Model Architecture	48
4.2	Audio Training	49
4.2.1	Audio Feature Extraction	50
4.2.2	Audio Model Architecture	52
4.2.3	Multi-Modal Architecture	53
5	Evaluation	56
5.1	Metrics and Cost Functions	56
5.1.1	Precision and Recall	57

5.1.2	F1-Score	59
5.1.3	Hamming Loss	61
5.2	Maximum Gap Thresholding	62
5.3	Survey on Human Perception	64
5.4	Model Results	68
5.4.1	Evaluation of Binary Models	68
5.4.2	Evaluation of Visual Only Multi-label Model	73
5.4.3	Evaluation of Audio Only Multi-label Model	76
5.4.4	Evaluation of Multi Modal Multi-label Model	76
6	Conclusion	81
6.1	Future Work	82
A	Survey	90

List of Figures

1.1	Personalized Trailer Representation	5
3.1	Cut Points Image Representation For One Sample of Dataset . . .	22
3.2	Eliminated Frame Examples	23
3.3	Kept Frame Examples	23
3.4	Training and Validation Loss of Elimination Model	24
3.5	Training and Validation Accuracy of Elimination Model	24
3.6	Elimination Model Architecture details	26
3.7	Elimination Model Visualization	29
4.1	Data Partition Visualization	32
4.2	ISFE Example	34
4.3	SEFE Example	35
4.4	Animation Model Architecture	38
4.5	Comedy Model 1 Architecture	39

4.6	Comedy Model 2 Architecture	40
4.7	Horror Model Architecture	42
4.8	Romance Model Architecture	43
4.9	Action Model 1 Architecture	44
4.10	Action Model 2 Architecture	45
4.11	Action Model 3 Architecture	46
4.12	Thriller Model 1 Architecture	47
4.13	Thriller Model 2 Architecture	48
4.14	Audio Model Architecture	53
4.15	Multi-Modal Model Architecture	55
5.1	Human Prediction Results Confusion Matrices	67
5.2	System Prediction Results	70
5.3	Multi-Modal Prediction Results Confusion Matrices	80
A.1	Survey results: Answer chart for question "What is your age group?"	90
A.2	Survey results: Answer chart for question "What is your gender?"	91
A.3	Survey results: Answer chart for question "How frequently do you watch movies?"	91
A.4	Survey results: Answer chart for question "Which of the following movie genres are your favorite?"	92

A.5 Survey results: Answer chart for question "Which of the following movie genres you NEVER prefer to watch?" 92

A.6 Example of a Question in Survey 93



List of Tables

2.1	Available Genres of MovieLens20M dataset in Literature	11
2.2	MovieLens dataset releases [1]	17
3.1	Manually Cleaned vs Automatically Cleaned Datasets	28
4.1	Train Test Split Genre Distribution of Audio Training	52
5.1	Maximum Gap Thresholding Executed on An Example	64
5.2	Human Prediction Scores	66
5.3	Binary Models Evaluation	69
5.4	Visuals Models Evaluation	75
5.5	Multimodal Prediction Scores	77
5.6	All Models Evaluation and Comparison	78

Chapter 1

Introduction

A movie trailer is a preview of a movie to attract attention or to tease the audience to encourage them to watch the movie. Although movie trailers were only shown in movie theaters a decade ago, now their reach has expanded to our mobile phones, television commercials, and even advertisement billboards. Movie-watching trends are transformed into more agile and mobile platforms offered by Netflix, Apple TV, Amazon Prime, Disney Plus, etc. Today, the audience can watch movies from their televisions, mobile phones, and many other devices.

In this chapter, we will first give some background information related to the movie advertisement by explaining what is a movie trailer and why it is significant for movie producers. Afterward, we will give our motivation to create an automated trailer generator and how can it be implemented. Lastly, we will explain our research subject.

1.1 Background Information

According to the a study held in June 2020, just 14 percent of adults said that they strongly preferred seeing a movie for the first time in a theater, and 36 percent said that they would much rather stream the film at home than visit

a cinema [2]. Since trailers are shown prior to the movie screening in theaters, a major reduce in theater visits results in audience watching trailers at their streaming services. Thus, there is a requirement of personalized advertisements in movie industry to excite its audience.

In the literature, trailers might be considered movie summaries. Even though both movies and trailers contain crucial parts of the movie, one main difference between video summaries and teaser trailers are not supposed to give any spoilers of the movie. In trailers, the story-line should remain mysterious so that it will not spoil the audience's enjoyment of the narrative, whereas a video summary explicitly reveals these important parts of story-lines. Thus, most of the trailers are produced from the initial parts of the trailer in order not to give spoilers to the potential audience.

Orest et al. summarize major key requirements of movie trailers as follows [3]:

- Trailers are responsible to give an overall idea of the movie without revealing major plot twists.
- Most trailers focus on introducing only main characters, instead of side characters to avoid the audience's distraction.
- The soundtrack, background music, and the character lines in the trailers should catch the mood movie.
- A good trailer should keep attention to emotions rather than the plot of the story.

Some factors explained above require trailer editors to have a rigorous study of the movie. The process of the manual trailer creation depends on the editors' knowledge, skills, budget, and subjectivity.

According to Debernitz and Karray, an estimated 65 percent of the movie's total budget is spent on production while the remaining 35 percent is allocated to

marketing and distribution on average relying on the data from Investopia¹ [4]. Joshi et al. explain that most of the movie advertising budget is incurred prior to the movie’s theatrical release[5]. The advertisements of the movies are primarily executed through trailer screening which also gives significant information about the potential studio profitability to investors[5].

Many platforms such as Google now use “Personalized advertising” techniques to show the relevance of the content for users and increase the marketability for investors. *Personalization* refers to the inclusion of information in the ad content that identifies or characterizes the recipient which has been found to carry out higher click-through rates and as a consequence, more sales [6]. Even though many applications now offer interest-based advertising, a personalized pre-release of a specific film for different interests of viewers crafted by editors would be costly and time-consuming.

An example of personalized advertisements offered in the movie industry is used in a streaming service called Netflix. Netflix platform provides recommendations based on user viewing history and personal data, to suggest movies and aid quick exploration. According to the survey conducted on Netflix audience out of the 90 respondents, most users (80%) said the significance of the availability of recommendations of content on the Streaming interface ranged from medium to high [7]. However, Netflix only offers certain trailers of a movie based on user preferences and does not show a movie trailer if it may not attract the audience. Since there is only one trailer mapped to a movie, a trailer is not recommended if the movie genre appealed to the user previously.

Recent studies have shown that everyday adults in the United States spend nearly six hours watching movies [8]. For the movie audience, trailers are the main component of pre-release advertising because it contains remarkable and attractive parts of movies within a short duration of time. Therefore, because trailers are the first preview for the audience, creating high-quality movie trailers involves high levels of human cognitive effort, time, and cost. Due to the increasing expense of multiple trailers created for one movie, most movies have only one

¹<https://www.investopedia.com/>

trailer generated. However, each user may have different interests, even for the same movie which means the audience may not show affection after showing a trailer that is not related to their interest [8].

1.2 Motivation

Trailer creation requires a lot of human effort and creative decision-making. The trailer construction process can take between 10 days to 2 years to complete [3]. Moreover, in high-cost movies, there can be up to six different trailer creation companies involved in the creation process. While these are valid for only one trailer, for most block-buster movies, producers release more than one trailer in the pre-release of the movie advertisement period, shown at different times to create a mysterious process by revealing a little bit of the movie in every trailer.

Fortunately, the most recent advancements in deep learning techniques may offer a sustainable option to customize trailers to the users' genre interests. Even though automated trailer creation is possible, post-processing is required as creativity and customizability requirements of a trailer are implied in Orest et al.'s research [3]. It is hard to distinguish the key plot-twist segments of the movies to prevent them to be shown in the final trailers. Moreover, a fully automated system may not analyze main characters correctly to only include these characters in the trailers. On the other hand, an automated system would be able to objectively suggest candidate scenes to be used in trailers. In this way, movie makers would not necessarily provide the whole movie to the trailer editors; and trailer editors would not waste a tremendous time on cutting scenes or combining them. Thus, the automated trailer generator process can indeed save time, money, and resources and provide objectivity regardless of the trailer editors.

As mentioned in section 1.1, there are personalized advertisements on the platform Netflix where a movie is recommended only to the audience whose profile contains the genre of the movie. Instead, an automated system can create a unique advertisement of the same movie for every genre interest in a very short

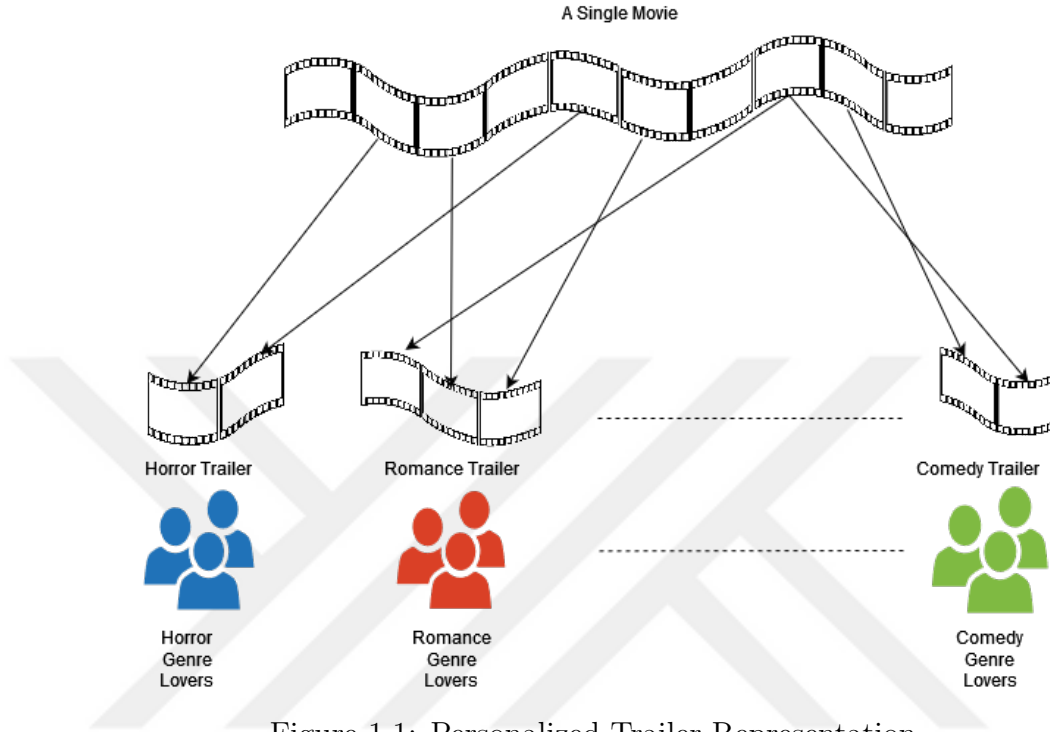


Figure 1.1: Personalized Trailer Representation

time. This would increase the grossing rate of the movie which would be profitable for the producers. The audience would be presented with the parts they might like in the movie from this trailer and even discover new genres from the other parts of the movie (shown in Figure 1.1). In conclusion, both the audience and the producers can get the advantage of the separately generated trailers for each genre.

A trailer is created by concatenating some of the carefully selected scenes from the movie and applying several post-production techniques. In this process before post-production, the selection of scenes becomes an important issue. We propose selecting scenes relevant to the genre for which the trailer is being created. For example, consider a movie that is labeled with the genres Comedy, Adventure, and Drama. In this case, we will create one trailer for the Comedy, another one for Adventure, and one other for Drama. In the creation of the Comedy trailer, we will select the scenes that are related to comedy, from the first parts of the movie. Therefore we have to be able to determine the genre of a scene clipped from the movie.

While our final goal is to fully automatize the trailer generation process, we have focused on selecting scenes by partitioning trailers and classifying these partitions into different genres to generate genre-based trailers. We believe that still creative human post-production will be required to advance a successful framework for movie trailer generation as Smith et al. also imply [9]. However, our system will ease the work of the post-production workers enormously as they will only add transitions, some visual effects, credentials, and sound to the extracted scenes from the movie we have given as the output. Since these factors may also require specific requirements by the producers, it would be more reliable and sustainable to be done by human trailer editors.

1.3 Contributions

Our contributions in this thesis are:

- Construction of a dataset for classifying scenes as their genres by:
 1. extracting scenes from trailers,
 2. removing uninformative scenes,
- Proposed an approach for classifying extracted scenes for genre classification by using:
 1. visual features,
 - (a) with binary label for each class,
 - (b) with multi-label classification;
 2. audio features,
 3. a combination of audio and visual features,
- Survey for humans' ability to make genre classification for evaluation of our system,

- Proposed a technique, called Maximum Gap Thresholding (MGT), for selecting the subset of labels to be predicted in Multi-Label Classification problems.

1.4 Organization of the Thesis

In this thesis, we have accomplished the task of creating an automated system to create personalized trailers that characterize a genre. For this task, we have built a deep learning-based system to classify movie trailers. The second chapter explains related work regarding the movie trailer creation process focusing on the trailer to genre classification and automatization of the creation. The third chapter presents the techniques we have used to create a dataset from MovieLens20M. In the fourth chapter, we explain our models for audio and visual features trained models to be later used in a combined multi-modal architecture. Finally, an evaluation of these models and comparison is provided in Chapter 5, and a conclusion is stated in Chapter 6.

Chapter 2

Literature Review

The idea of predicting interest in a movie trailer, or generating interest in a movie trailer has been studied by many researchers. In this section, we will present related work for our thesis subject including movie genre classification and automated trailer generation.

2.1 Related Work

Our main task is to create an automated trailer based on the genres to offer the audience a personalized movie advertising system, and to do so, we have to classify scenes of a movie to one or more of the 9 genres given in Section 4.1.3. Moreover, we studied a more focused research on Animation discrimination and published a conference paper [10]. In this section, we will explain previous work in our field of research. It is important to emphasize our research subject of generating movie trailers with respect to the genre has been never studied before, to the best of our knowledge.

2.1.1 Trailer to Genre Classification

Our fundamental research was to build on a reliable system to discriminate genres to later use in an automated trailer generation. It is a significant part of our research to classify the movie trailer scene genres correctly to be able to confidently use the movie scene classification to produce different genre-characterized trailers from the same movie.

One of the challenges of trailer classification is that classes are not mutually exclusive as is also explained by Boutell et al. where a natural scene may contain multiple objects such that the scene can be described by multiple class labels, and requires new metrics for evaluating individual examples and cost functions [11]. Solving this challenge helps us to train models that can solve multi-label input, multi-label output, and non-mutually exclusive label tasks. Mutually exclusive labels means that labels of a data samples are independent from each other.

Video shot boundary detection is a major step in some related work for trailer to genre classification. The main idea of video shot boundary is extracting the feature of video frames, and then detecting the shot type according to the difference the feature [12]. According to a novel approach given by Guru et al., there are two main types of shot boundaries or shot transitions, abrupt transitions and gradual transitions [13]. In abrupt transitions a sudden discontinuity in the frame sequence is seen, where as in gradual transitions a slow (change) transition is seen. Fade Out, Fade In, Wipe, and dissolve are the different techniques in gradual transitions.

Zhou et al. suggested a movie genre classification method with decomposition to scenes and extracted W-CENTRIST features from each scene to feed to the k-Nearest Neighbors categorization model [14]. For their research, they have used 4 genres namely action, comedy, drama, and horror. They have conducted their experiments on 1239 annotated trailers and for each of these trailers, they have extracted scenes with shot boundary detection. Later, they used a single keyframe as a static representation of a shot and used W-CENTRIST features, a variant of the CENTRIST descriptor that captures color information in addition

to intensity statistics, to describe a keyframe. Their experiment resulted in a best overall accuracy of 71%.

Subsequently, Simões et al. suggested Convolutional Neural Networks on an image-level classification supported by shot boundary detection [15]. They have used Gist, CENTERIST, and w-CENTERIST provided by Zhou et al.'s research [14] for their feature extraction and they have split each trailer movie into scene representatives or keyframes. They have fed these high-level features into a CNN with 4 classes as its outputs. They have acquired 62% accuracy on LMTD-4 (Labelled Movie Trailer Dataset 4).

On the other hand, Huang and Wang used a simpler classification method, Support Vector Machine (SVM) to combine both Audio and Video Features of the trailer [16]. This research proved the significance of audio features as they used a total of 75 image features and 202 audio features. Some examples of visual features are Shot number, Average shot number, Average color histogram etc. and some audio features examples are Spectral Centroid, Spectral flux, Zero crossing etc. They have also used the Self-Adaptive Harmony Search (SAHS) algorithm for these 277 features to find out which features are more relevant than the visual features in discriminating the 7 movie genres: action, animation, comedy, documentary, drama, musical, and thriller.

On a more recent note, Wehrmann et al. suggested a multi-label approach based on Convolutions Through Time (CTT) [17]. Their module used convolutional filters for learning temporal relations instead of using Recurrent Networks and takes into account high-level frame-based information extracted by deep neural networks. They have experimented with their system on LMTD and they have measured their success with micro average, macro average, and weighted average scores. They have used 9 genres, Action, Adventure, Comedy, Crime, Drama, Horror, Romance, Sci-Fi, and Thriller acquired a 0.66 weighted average score with a system that combines both audio and visual components of the whole trailer.

Shambharkar et al. took advantage of the newest technology available to classify 4 genres with the 3D Convolutional Neural Networks and acquired 82% accuracy [18]. Their research proved that 3D convolutions are much better than 2D convolution architectures such as VGG16 and Inception V3 because 3D-CNN gives the advantage of learning temporal and spatial features from video frames. They have windowed video trailers with 64 consecutive frames to feed their architecture and used softmax as their final layer.

Research Genre	Zhou et al. [14].	Simoes et al. [15].	Huang et al. [16].	Shambharkar et al. [18].	Wehrmann et al. [17].
Action	✓	✓	✓	✓	✓
Adventure					✓
Animation			✓		
Children's					
Comedy	✓	✓	✓	✓	✓
Crime					✓
Documentary			✓		
Drama	✓	✓	✓		✓
Fantasy					
Film-Noir					
Horror	✓	✓		✓	✓
Musical			✓		
Mystery					
Romance				✓	✓
Sci-Fi					✓
Thriller			✓		✓
War					
Western					

Table 2.1: Available Genres of MovieLens20M dataset in Literature

We have compared literature available genres in Table 2.1 with our foundation Dataset, MovieLens-20M which will be explained in Section 2.2.1.

2.1.1.1 Animation Discrimination

As the first experiment, we tackled with the animation discrimination task separately in our research, as it is not exactly a genre but a movie creation technique of modeling to create an illusion of a photo-graphic event used in cartoons whereas animated movies refer to a sequence of drawings. The problem of distinguishing cartoons from photo-realistic movies can be viewed as a case study of trailer genre classification as an immensely researched and yet-to-be-improved subject of computer understanding. Latest technologies in Computer Generated Images (CGI) have been creating more realistic animation styles that become challenging to distinguish by humans. An animation discrimination system can be further enhanced into a system to distinguish even the most photo-realistic yet computer-created movies. As Farid et al. stated, advancements in these realistic computer-generated releases blur the line between reality and fantasy, and they can even cause legal situations, such as engaging in sexually explicit conduct [19].

One of the first attempts to classify movie segments as either animated or non-animated is by Roach et al. for the classification of video fragments using motion only [20]. Their dataset consisted of only a few sequences, 8 cartoon sequences simplified as sketched cartoons and animated cartoons, and 20 non-cartoon sequences where they also classified non-cartoon genres as computer-generated cartoons, sci-fi, and real-life motions to simplify the problem.

A hand-crafted method was suggested by Ianeva et al. [21] with accounting pattern spectrum of parabolic size distributions to map color histograms, texture, color edges, brightness, and various image analysis filters, and trained with Probabilistic Gaussian Mixture Model and SVM. Even though they have accomplished satisfying accuracy with their trained key-frames, their model suffered from external key-frames with divergent visual content characteristics.

Eventually, Glasberg et al. suggested a method based on MPEG-7 features extracted from 100 representative video sequences of cartoons, and commercials with animated cartoon sequences [22] which accomplished to classify of a sequence

of 50 frames in one minute with 0.8 accuracy of animated scenes and 0.85 of non-animated scenes.

To distinguish not only cartoons created by the human hand but also computer graphics images, from natural photographs Ng et al. suggested a method with accounting wavelet, geometry, and cartoon features of an image with 3200 images belonging to 4 different classes and achieved 0.84 accuracy [23]. Color histograms and SVM were considered to classify computer graphics images from photographic images by Chen et al. which proved that HSV color channels are more informative than RGB color channels with slightly better accuracy and true positive rate [24]. To extract video level features, Chen et al. used color component and color kind based on region segmentation and proved that SVM is better performing than GMM, and Manifold Ranking [25]. They have also considered cartoon segmentation as their features, and they have extracted their frames from the movies with a seemingly primitive shot boundary analyzer as a change detector.

On the other hand, Sankar et al. also considered hybrid images consisting of both cartoon and photo-realistic segments of 557 features which can be reduced to 80 without a crucial loss in performance [26]. Moreover, Ionescu et al. worked with two sets of features namely temporal descriptors, like rhythm or action, and color descriptors are determined using color perception to achieve an average global correct classification up to 0.92 [27].

Initial attempts to use neural networks to classify cartoons as movie genre was presented by Montagnuolo et al. with accounting texture information, color histograms and temporal activity information based on the displaced frame difference which acquired average precision of 0.86 and an average recall of 0.85 [28]. Alternatively, Quan et al. accompanied CNN with four convolutional blocks followed by two fully connected layers to achieve 0.94 accuracy on their dataset of images with two labels: natural and computer-generated which did not include human-drawn cartoons [29]. More recently, Zhang et al. attempted to classify computer-generated images from RGB color channel and pixel correlation to acquire 0.94 accuracy on the ScNet dataset with a convolutional neural network

architecture [30].

Most of the related work in this area either considered images or motion separately. However, we have assembled a dataset and built a fused model that combines both spatial information using CNN and temporal information using GRU to attempt an improved performance on accuracy. We have also published a paper named “Detection of Animated Scenes Among Movie Trailers” [10].

2.1.2 Automated Trailer Generator

Automated trailer generation can be studied more widely with the newest technology available due to the slow process of video processing. There have been several methods suggested to summarize the movies, however, a summary of a video is different from a movie trailer because a trailer needs to attract an audience without giving spoilers whereas summarizing needs to offer key components of every part of the movie.

Irie et al. suggested Vid2Trailer, an automated trailer generator in which there are three major processing stages: symbol extraction, impressive component extraction, and reconstruction. Vid2Trailer accounts for mathematical algorithms and probability estimations for their affective content analysis and later they use an Artificial Intelligence (AI) for shot arrangement. To evaluate their results, they conducted a survey with 20 subjects. They have compared their results with Clustering-based Video Summarization(CVS), Attention-based Video Summarization(AVS), and Real Trailer(RT) and they have acquired a better score of interestingness than CVS and AVS, less score than the RT [31].

Xu et al. suggested an automatic trailer generation approach, which is densely connected with visual attractiveness. They have based their approach on three steps: a validated measure of trailer attractiveness; proposed an algorithm for selecting and re-arranging shots to maximize the level of attractiveness, and synchronized shots and music for upgraded viewer experience [32]. They surveyed

human perception and asked three questions to rate music rhythm, trailer attractiveness, and realistic appropriateness. Their survey concluded that their success rate is higher than that of Vid2Trailer approach [31, 32].

An approach for human-AI trailer collaboration suggested by Smith et al., to identify 10 candidates among all movie scenes as the best for being a part of a trailer [9]. In their proposal, the professional filmmaker would edit and arrange these moments to construct a comprehensive movie trailer. They based their approach on a specific movie, called Morgan. They applied Principal Component Analysis (PCA) method to the extracted features and the authors noticed that their Augmented Intelligence system somehow learned the genre domain of their given movie while deciding on the trailer-making process.

In another notable research, Orest et al. suggested unsupervised trailer generation using anomaly detection by combining audio and visual features [3]. Their method to create trailers without any previous knowledge about the target movie involves shot boundary detection, and feature engineering of visual and audio features to be fed into a linear model, proximity-based model, and ensemble model. For evaluation, they surveyed four questions to question the audience's rating. These questions are: (1) Give the overall rating for this trailer. (2) How strongly this trailer arouse interest in you? (3) How many spoilers does this trailer contain? (4) How likely would you watch the original movie after viewing this trailer? [3]. Moreover, they have compared their appropriateness, attractiveness, interest ratings on the Likert scale from 1 (the lowest) to the 7 (the highest). which they attained better rating results than their compared related work. Their appropriateness rated between 4-6 while video2trailer achieved 2-5, their attractiveness achieved 3-6 when video2trailer achieved 2-5 and their interest achieved 3-6 while video2trailer achieved 2-4.

2.2 Foundation Dataset

We have the idea of automatically generating the movie trailers from movies; however, movies are too long in length (approximately 2 hours each) and most of the movies are not publicly available. Therefore, we have considered trailers as our training data. Trailers can be also acknowledged as a summary of the whole movie and they contain significant moments from the movies to thrill its audience [8]. Thus, one of our main reasons with working trailers instead of movies is that all of the trailers are publicly available on the internet. The second reason is that trailers are short and downloadable as video data with their labels identified in a document. Since a trailer will be created at the end product, it is also meaningful to train a system on how to label the trailers rather than the long movies. In a sense, a system would be more successful learning the parts of the movies that excite the audience than the long and non-descriptive segmentation of movies.

2.2.1 MovieLens-20M

Many datasets did not provide trailers labeled with their genres. We investigated for the best fit for our research that directly gives a mapping between movie trailers (or URLs of movie trailers from YouTube) and their respective genres as labels. One of the biggest datasets we could derive was MovieLens20M.

The MovieLens project was started by University of Minnesota researchers and it was launched in 1997. The dataset of MovieLens was released in 1998, describing people’s expressed preferences for movies [1]. Later, the legal issues were transferred to GroupLens. The dataset has been updated with users interacting with the MovieLens online recommender system over time. The audience preferences take the form of tuples or stars, with each the result of a person expressing a preference (a 0-5 star rating) for a movie at a particular time [1]. Higher stars means the user liked the movie more and low stars means they did not like the movie. This star system was later changed into a 0.5-5.0 half-star scale in 2003. The history of the MovieLens dataset releases is given in Table 2.2

until the MovieLens20M release.

Name	Date Range	Rating Scale	Users	Movies	Ratings
ML 100K	9/1997 - 4/1998	1-5, stars	943	1,682	100,000
ML 1M	4/2000 - 2/2003	1-5, stars	6,040	3,706	1,000,209
ML 10M	1/1995 - 1/2009	half-star, 0.5-5	69,878	10,681	10,000,054
ML 20M	1/1995 - 3/2015	half-star, 0.5-5	138,493	27,278	20,000,263

Table 2.2: MovieLens dataset releases [1]

Although our task involves trailer genres, previous releases of the MovieLens20M did not provide proper information about the movies for our research. However, in 2018, GroupLens created another database containing 25,623 YouTube IDs and mapped Movie IDs to Youtube URLs. Fortunately, in a previous release, they have also provided mapping between genres and movie IDs. Thus, given a movie ID in the dataset, we can get YouTube URLs and movie genres to derive a trailer classification dataset. As it can be seen from the Table 2.2, releases of MovieLens was named based on the number of ratings.

Chapter 3

Preparation of Dataset

We consider one of our contributions as the construction of a dataset composed of animated and non-animated movie trailers since there is not a proper dataset for this specific task as we observed. We first started with MovieLens20M trailers mapped to their genres as mentioned in Section 2.2.1. Although there are 27000 movies in this dataset, while downloading the data from YouTube we omitted downloading some videos caused by video removal or user log-in required for permission. Moreover, due to the ban caused by the high number of HTTP requests, we ended up with 6760 of these trailers.

According to Ji et al.’s analysis of trends in the content of the Top 20 U.S. box office movies from 1967-2008, “adult”, “film-noir”, “game-show”, “news”, “reality-TV”, “short”, “documentary”, “talk-show” did not appear at all in the top 20 lists; thus, we decided to omit these genres from our dataset [33]. However, if the data with one of these labels have another valid label; we just remove the label and allow data to exist on our dataset with the valid labels.

As preliminary information, our dataset contains movie trailers with a 30 fps (frame per second) frame rate. Since we assume that a trailer may contain scenes belonging to various genres, we initially partition each trailer into a sequence of scenes by shot boundary analysis. In this implementation, we refer to the movie

trailer partitions generated by a scene detector as scenes. Each of these scenes is represented by a pair of frame indices, one for the beginning and the other for the end frame. For example, since every trailer has 30 frames per second, a pair of (360,600) refers to the scene which is a segment of the trailer between the 12th and 30th seconds.

The preparation of the dataset was followed by several steps: (1) The initial links were downloaded from the CSV file given by the MovieLens20M mapping to YouTube trailers [34]. (2) Downloaded trailers were fed into the shot-boundary detection unit. (3) The output of shot-boundary detection is later filtered automatically with removal of too short and too long scenes. (4) Each downloaded trailers is pre-processed with our model to eliminate the segments that do not contain visual information about its genre.

3.1 MovieLens20M Data Downloader

As mentioned in Section 2.2.1, we decided to use two CSV files provided by the MovieLens20M dataset which are named *movies.csv* (includes genres and names mapped to the movie indices in the MovieLens20M database) and *ml-youtube.csv* (includes YouTube links with respect to the movie id's in MovieLens20M database). We built our own data downloader which downloads links from YouTube, and also saves the downloaded movies with their ids and genres on a separate CSV file. Unfortunately, some of the links were removed from YouTube, or they were not accessible anymore due to privacy or age restrictions. We completed this process without logging in, therefore all of the links downloaded were publicly available at the time. Some example errors of corrupted videos are “This video is no longer available due to a copyright claim by x company”, “This video is no longer available because the YouTube account associated with this video has been terminated.”, “No video formats found”.

Our Data downloader uses Pafy¹ library which supports 2.6+ and 3.3+ versions

¹<https://pypi.org/project/pafy/>

of Python. Pafy has the option to get the best-quality resolution version of the trailers. All of the trailers have their own resolutions, but most of them are close to 480p. “p” here refers to *Progressive scanning* which is a format of displaying, storing, or transmitting moving images in which all the lines of each frame are drawn in sequence [35]. We used *MP4* format to save the video trailers. Due to the HTML request limitation of YouTube, only a certain amount of videos could be downloaded. A few of these video trailers were removed due to exceeding a regular trailer time (up to 10 minutes). In the final version of our database, we have 6760 video trailers. Moreover, an insider package of Pafy called *youtube-dl* was not updated to the latest version of YouTube. In the latest YouTube patch they removed the dislike count and it caused errors in *youtube-dl* package; thus, we removed *'dislike_count'* variable from *backend_youtube_dl.py* class.

3.2 Scene Extractor

The initial assumption of our thesis is based on the idea of every trailer including content for any of the genres. Based on this reasoning, we considered every video as its smaller parts; scenes, or segments which we use shot boundary detection to partition (see Figure 3.1). Although we label every scene as a set the same genres of the trailer; in the end product, we will label movies for each of their scenes with different sets of genres. The second reason we chosen to use scenes is that long sequences of video classification are rather more challenging and it might be hard to catch every information given in the whole trailer. Thirdly, when we divided our task into shorter scenes; we enlarged our dataset by the average number of scenes extracted from each of the trailers. However, we avoided using a different scene of the same trailer in the both training and testing step to avoid giving hints to our system which will be explained in Section 4.1.1.

The scene extractor divides a movie or trailer into intervals, mapped by the pairs of indices that indicate the beginning and the ending frame of a scene. For our scene detector algorithm, we used PySceneDetector ² which is a shot

²<http://scenedetect.com/en/latest/>

boundary detection library based on signal processing. There are three different shot boundary detection units in PySceneDetector (1) Content-Aware Detector (CAD) that detects jump cuts in the input video, (2) Adaptive Content Detector (ACD) which prevents false detections where there is fast camera motion, (3) Threshold Detector (TD) that compares the intensity/brightness of the current frame with a set threshold, and triggers a scene cut/break when this value crosses the threshold³. Although it is open-source and fast, PySceneDetect was built for long movie scene detection and can work poorly on short movie trailer scenes. For our task, we first experimented with CAD which is the fastest; however, CAD is poor in detecting fade-in and fade-out or smooth transitions. We experimented with TD, since we are working with trailers and not movies scenes were not segmented short enough for fast transitions. Furthermore, when we experimented with Adaptive Content Detector we noticed that it is a more advanced version of CAD and works better for fast transitions in the trailer scenes. Therefore, we combined ACD and TD in our shot boundary detection unit and we used ContentDetector with a threshold of 40.0 frames and a minimum scene length of 1 frame, and ThresholdDetector with a threshold of 12 frames, a minimum scene length of 1 frame, and fade bias of 1. The minimum scene length of 1 frame allows fade-in and fade-out to become scenes on their own. We filter these fade-in and fade-out noises later when we eliminate scenes with lengths less than 100 frames. However, there are still some poorly partitioned scenes that include fade-in and fade-outs in the final version. Another framework can be developed for trailer scene extraction however we focused on trailer genre classification in this thesis.

3.3 Removal of Post-production Components

We trained a separate deep learning algorithm to automatize the noise elimination process and avoid human labeling. The end product was successful in finding what we call noisy scenes or scenes that should not belong to an intermediate movie scene. Examples of to-be-eliminated scenes are film studio logos, commercials,

³<http://scenedetect.com/en/latest/reference/detection-methods/>

ID_Frame	Genres	Cut_Points	Number_Scenes
3360	['Drama', 'Romar	[[176, 533], [644, 766], [766, 908], [1058, 1205], [1205, 1316], [1316, 1418], [159	18

1st cut points between frame [176 - 533]



18th cut points between frame [2011-2190]



Figure 3.1: Cut Points Image Representation For One Sample of Dataset

credentials, and scenes composed of font or text animations that can be seen in Figure 3.2.

To eliminate unwanted scenes, we first created an image dataset composed of two classes; scenes to be removed and scenes to be kept. Afterwards, we trained a shallow Convolutional Neural Network (CNN) architecture. Once we ensured the reliability of the model, we automatically removed noisy scene intervals from the cut_points lists given example in Figure 3.1. We finalized our dataset construction process with the removal of scene boundaries of less than 100 frames and more than 10000 frames to eliminate errors caused by the scene extractor unit. We allowed the removal of these scenes since we have an adequate amount of data samples and noisy data elimination is significant for our training.

3.3.1 Elimination Model Dataset

To automatize elimination process, we trained a scene elimination deep learning model. In order to create a training database, we chose single frames from the scenes that might contain certain writings such as a close-up scene of a newspaper, or a signboard in the background to keep noteworthy information from movie scenes (see Fig. 3.3). We picked 858 *to_be_eliminated* samples and 804 *to_be_kept*

samples as the training data. All of these samples were extracted from the middle of the scenes, and we picked them by hand from automatically saved over 4000 images from the middle frames of the scenes. We prepared an image-level dataset for this task since it was rather simple to learn from stationary image, and temporal information was not required. Examples from our dataset are given in Figures 3.2, 3.3.

We used Tensorflow’s *image_dataset_from_directory*⁴ for this task with 0.2 validation split which is equal to 1294 files for training and 323 files for validation. We also applied for data augmentation to enlarge our dataset. We used RandomFlip, RandomRotation, and RandomZoom to increase the number of samples. We fixed the image size to 224 by 224 by 3, which represents 224 pixels horizontally 224 pixels vertically, and 3 Red, Green, and Blue (RGB) channels.

3.3.2 Elimination Model Architecture

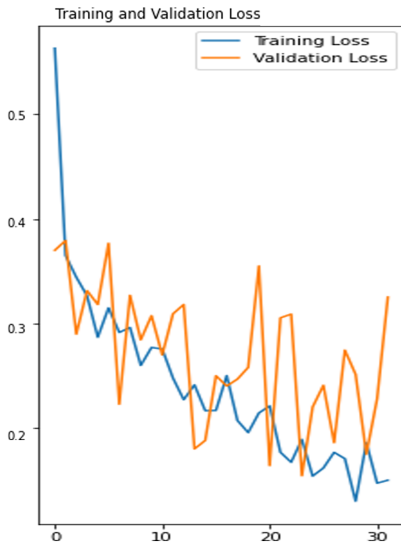


Figure 3.4: Training and Validation Loss of Elimination Model

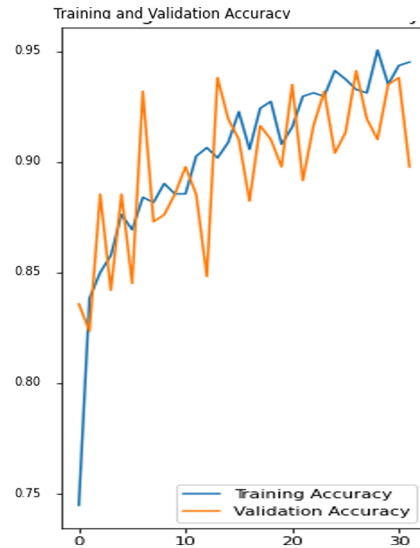


Figure 3.5: Training and Validation Accuracy of Elimination Model

We provided binary labels of our data samples, where we labeled to be kept

⁴https://www.tensorflow.org/api_docs/python/tf/Keras/utils/image_dataset_from_directory

images as 0 and to be eliminated labels as 1. We fed middle frame images to our architecture composed of convolutional blocks. Subsequently, a CNN ResNet with 4 sequential residual blocks of 16, 32, 64, and 128 units were used to train the model and acquired 0.9412 binary accuracy on the test data. Each block of CNN includes a two times Convolutional 2D layer followed by an activation layer with ReLu; a block is finalized by max pooling with pool size 2. The model architecture can be also seen in Figure 3.6. As it can be seen from Figures 3.4 and 3.4, the validation loss decrease rate was significantly reduced after around the 30th epoch. Although over-fitting after the 30th epoch may seem like fast learning, this task can be considered simple enough to be learned in approximately 30 epochs. We visualized layers for some of the noisy data to see what features our model learned. As it can be seen in Figure 3.7, our model was successive at finding text in the given frame.

After we acquired 0.9462 accuracy on the 29th epoch, we saved model weights to be used in scene elimination on the whole dataset. Later, we removed the noisy scenes if our pre-trained elimination model labels the middle frame of the scene with certainty of more than 0.6 as noisy class (or eliminate class). We also removed certain scene boundaries that have a length smaller than 100 frames and larger than 10000. Finally, we ended up with a new CSV file with filtered-out cut points, (see Figure 3.1) and acquired the train-ready data.

3.3.3 Manual Scene Cleaning

Although we pre-processed trailer scenes with the removal of the post-production components scene and we removed too long and too short scenes; there was still noise in the database. Even though an optimal system may learn the correct weights on hidden layers; manual noise-cleaning may boost the scene classification in terms of time, memory, and data size constraints. Therefore, we developed a simple interface on Jupyter Notebook⁵ for manual scene cleaning.

⁵<https://jupyter.org/>

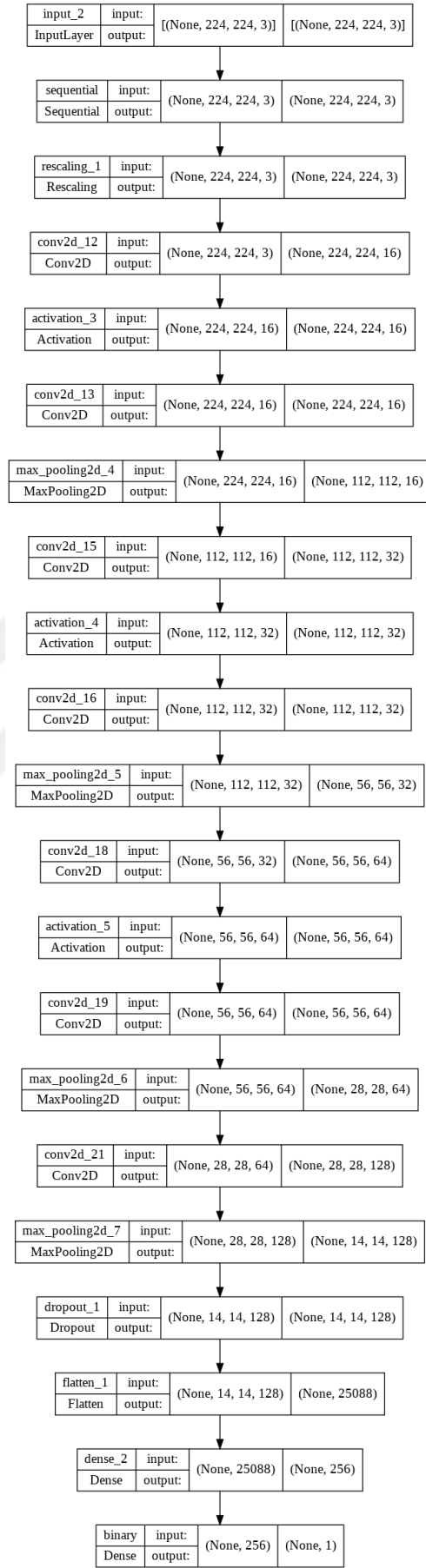


Figure 3.6: Elimination Model Architecture details

A manual scene cleaning requires human interaction to put each scene to the manually cleaned dataset. For each movie trailer’s scene, we show each scene with 10 frames extracted with 4.1.1.2 algorithm which gives samples of 10 frames from the scene. The user decides to keep or remove from the dataset given the knowledge of which genre this group of frames that forms a scene. If the user can not relate the given movie genre to these 10 frames, or these 10 frames are noisy data (the elimination system might miss some of the noisy frames) the user will pick the remove option. If the user decides the scene is valuable and informative, he will pick the keep option and the kept scenes will be recorded in a separate CSV. In this way, the user can save his progress and continue from the saved CSV the next time he uses the interface. Moreover, this system also allows users to work in parallel.

Due to lack of human labor as this is a master thesis, we could manually label only 700 trailers. The genre distribution of the manually created dataset is the same as the original dataset because we picked trailers to show in the interaction window randomly to not disturb the data distribution. We named this smaller part of the dataset the manually-cleaned dataset. We experimented with both manually cleaned and not cleaned versions in our training.

3.3.4 Manually and Automatically Cleaned Datasets

We used both versions of our dataset; namely, manually cleaned and automatically cleaned in our different experiments to evaluate human-labeling results. In the automatically cleaned version, we also discarded some trailers to prevent imbalanced data. For instance, some genres such as Drama and Comedy caused imbalanced data, thus we removed some trailers with these genres if they do not disturb the bottle-neck genres, those with low number of instances, such as Sci-Fi and Mystery. Since we use the scenes and not whole trailers, it is possible to reduce our data size without causing a small-scale training data.

On another note, the manually cleaned version was previously automatically cleaned. Our end products given in Sections 3.3.1 and 3.3.3 is given in Table

2*Genres	Manually Cleaned		Automatically Cleaned	
	Number of Trailers	Number of Scenes	Number of Trailers	Number of Scenes
Action	100	797	587	5122
Adventure	71	448	455	4349
Comedy	248	2077	787	9133
Drama	378	3427	1029	11440
Horror	73	500	456	4318
Romance	105	876	525	5768
Sci-Fi	56	380	407	3509
Thriller	133	1127	702	6248
Mystery	46	337	339	3529
Total	762	6335	2441	25749

Table 3.1: Manually Cleaned vs Automatically Cleaned Datasets

3.1. Noticeably, the total number of each genre is not equal to the total number of genres at the bottom of the table. This is due to the multi-label task we are given from MovieLens20M. Thus, a single sample may belong to one or more genres. Note that, we removed some labels from the initial dataset MovieLens20M. Removal of some labels was due to lack of data samples, incompatibility to the term “movie genre”, and used genres in related work research given in Table 2.1. We also discussed which of these genres are used in the movie industry with our industrial advisor Sami Arpa in our interview [36]. Finally, we decided on 9 genres given in Table 3.1 : Action, Adventure, Comedy, Drama, Horror, Romance, Sci-Fi, Thriller, and Mystery. Even though we discarded the Animation genre in this phase, we created another small sample from MovieLens20M and solved Animation-genre as a separate problem which will be explained in Section 4.1.2.1. Each scene is labeled as the labels of the trailer it is taken from. For example if a movie assigned to Action, Comedy and Romance genres, then each scene extracted from this movie’s trailer is labeled as Action, Comedy and Romance. This labeling method can conflict with our assumption that a trailer may belong to several genres. In given example, first scene of this movie trailer can belong to Comedy genre whereas second scene may belong to Action and Romance genres. Therefore, labelling scenes with the same labels we get from trailers might cause noise in our data. Nevertheless, given enough data samples trained system should be reliable.

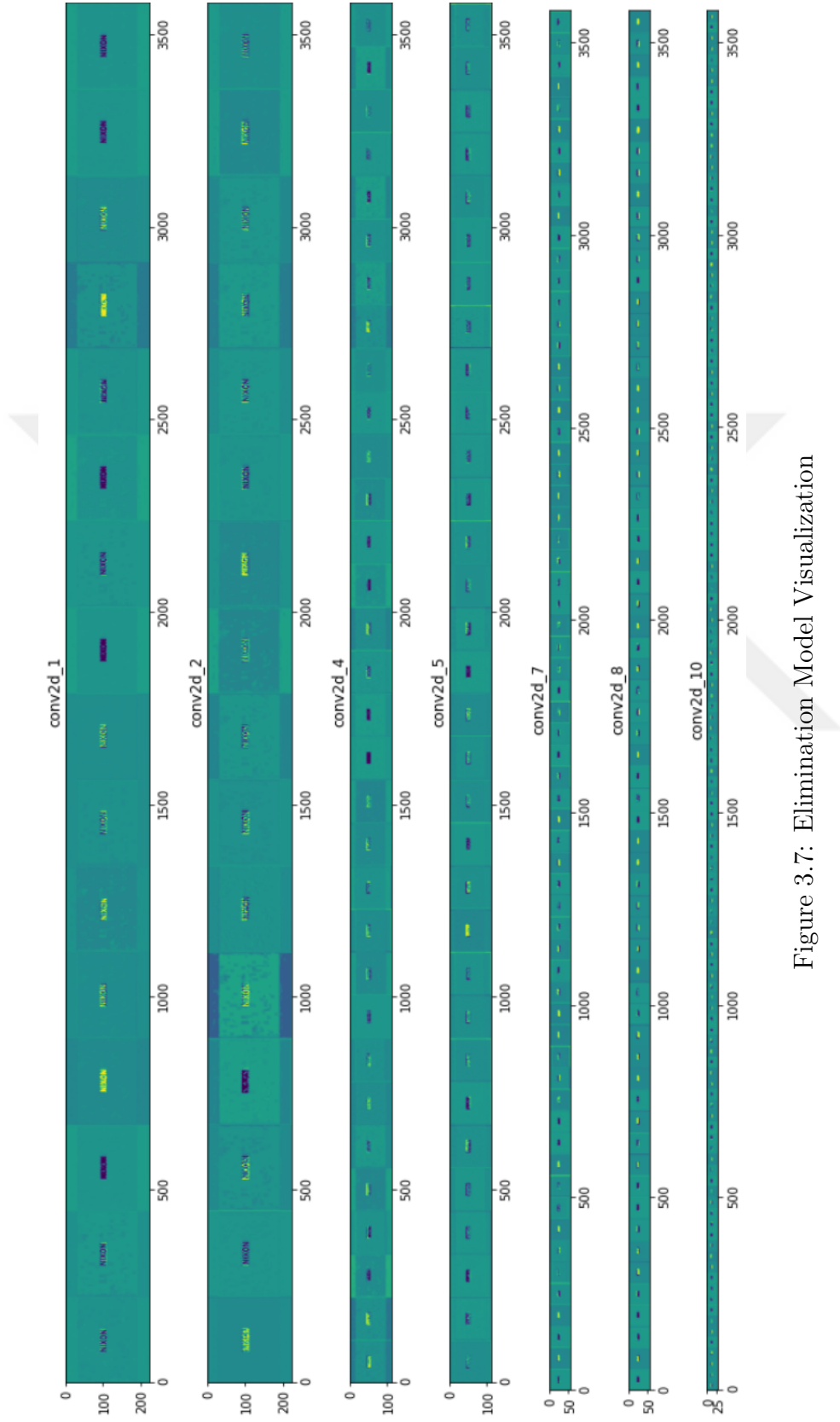


Figure 3.7: Elimination Model Visualization

Chapter 4

Scene to Genre Classification

A trailer video contains both visual and auditory content that proves to be informative about the genre of the trailer. Thus, we have divided the trailer classification task into two main components: audio and visual training. We later combined these networks into a multi-modal system to get our final results.

4.1 Visual Training

In this part, we focused on the visual features of the trailers and experimented with different algorithms to prepare the data, and to decide on an architecture we found out that different architectures work best with different genres. In the final step, we combined all of the experimented architectures into one multi-label output prediction system.

4.1.1 Data Preparation

Video processing can be considered as a tedious problem by computer engineers. Every frame is represented by a large number of pixels, which is the product of the

height and the width of an image. Further, it is an even slower process considering that every second of a video is represented by 30 frames. For our model, we prepared our data generator in which we optimized the video processing speed during the learning period with a library called Decord¹. Decord is a video slicing library based on a wrapper on top of hardware accelerated video decoders. It is designed to provide efficiency during training with video data for deep learning [37].

Our data generator is designed to generate several frames from each scene we pre-extracted in our dataset. The generator generates inputs to our model with dimensions of the number of frames, image width, image height, and three channels (Red, Green, Blue, or RGB). These frames are extracted from each scene, and we experimented with different values, and different algorithms to extract frames which will be explained in Sections 4.1.1.2 and 4.1.1.1.

Since we have data samples as scenes, a scene of a trailer may appear in the training subset and another scene may appear in validation data. This would give hints to the testing phase from the training phase, and we would not be successful in evaluating our model if we already given a genre of a very similar scene in the training step. Thus, we partitioned data in a way that a scenes from a trailer do not appear in both training and validation data. We ensured a trailer’s scenes appear only in either the training subset or validation subset by dividing the data from the data sample we given to the generator as can be seen in Figure 4.1.

In this thesis, we used validation and test terms interchangeably. Because we do not back-propagate with validation or use a technique such as cross-validation, our validation data samples are never taught to models before the testing / evaluation step.

¹<https://github.com/dmlc/decord>

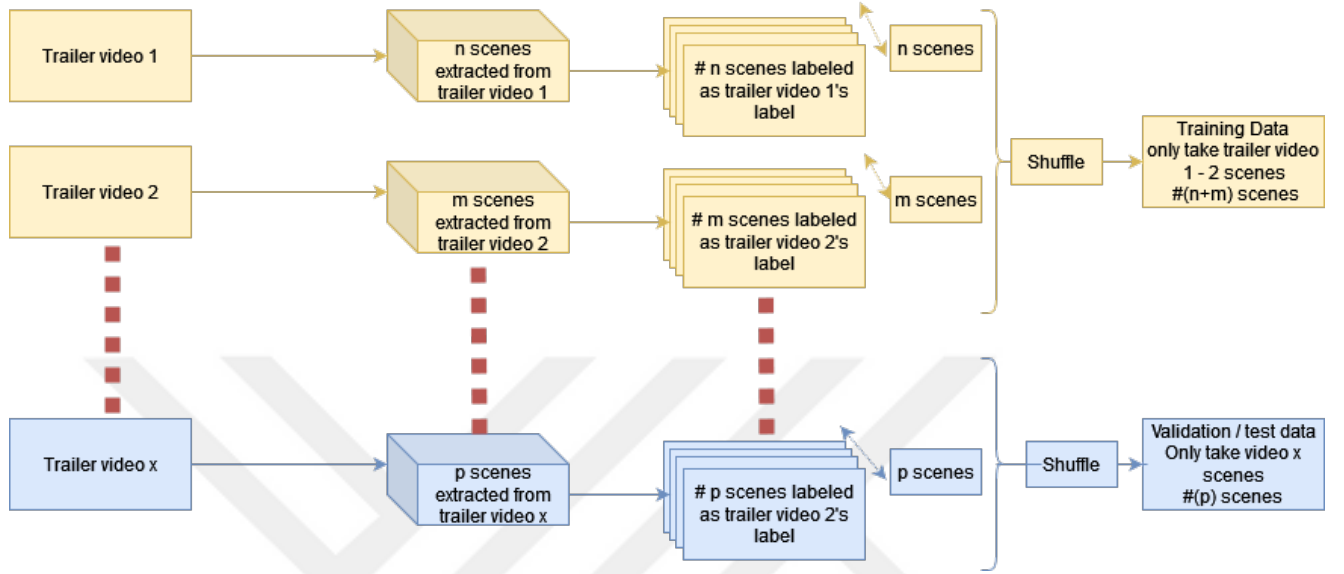


Figure 4.1: Data Partition Visualization

We considered two different frame extraction algorithms. In Section 4.1.1.1 we explain the first method for frame extraction where we considered equal intervals when we required sensitivity to the same interval lengths in different scenes. In the second method, we described in Section 4.1.1.2 we have the whole scene considered with intervals which vary for each of the scenes due to their changing length. The first method allows us to find the changes in equally distanced frames of different scenes. The second method allows sensitivity to the length of each scene and transitions between the scenes. We experimented with both of the methods to find out which works the best for our architectures which will be explained in Section 4.1.2. Intermediate Sectioned Frame Extraction with Equal Intervals (ISFE) algorithm is found to be more usable while learning slow changes such as in the Animation genre. The Start to End Frame Extraction with Unequal Intervals (SEFE) algorithm is more efficient in learning the changes throughout the whole scene. For example, in an action genre, there are sudden changes and the SEFE algorithm proved to be convenient.

4.1.1.1 Intermediate Sectioned Frame Extraction with Equal Intervals (ISFE)

Since we split each trailer into several scenes, we considered each of the scenes as a different batch of the same genre. For each scene, we extracted a set of sequential frames to be used in our model. There are various ways to implement the frame extraction method, however, we used the division of each scene into a constant number of frames so that we can record the motion of scenes of different lengths. Our method involves a simple partitioning of the scene to an equal number of the frames explained in equation 4.3.

$$par_{start} = \frac{scene_{end} + scene_{start} - frame_rate^2}{2} \quad (4.1)$$

$$par_{end} = \frac{scene_{end} + scene_{start} + frame_rate^2}{2} \quad (4.2)$$

$$frame_list = (X_{par_{start}}, X_{par_{start}+frame_rate}, \dots, X_{par_{end}}) \quad (4.3)$$

$scene_{start}$: scene start frame index

$scene_{end}$: scene end frame index

par_{start} : partition start frame index

par_{end} : partition end frame index

$frame_rate$: number of frames to be produced (given by user)

$frame_list$: the frame list we end up with

In the ISFE equation, par_{start} refers to the frame number that is the starting point of the partition, which is calculated by the scene endpoint and the scene start points for each of the scenes (Equation 4.3). This equation also allows us to partition the middle section of the scene with an optimum frame rate to reduce any misinformation frames caused by scene changes, such as fade-in or fade-outs. par_{end} refers to the frame number which partition ends, and these frames were chosen with equal distances of frame rate. $Frame_rate$ corresponds to frames that will be extracted with this algorithm as a result.



Figure 4.2: ISFE Example

4.1.1.2 Start to End Frame Extraction with Unequal Intervals (SEFE)

In the second frame extraction method SEFE, we considered the whole scene with equally partitioned segments to make frame_rate outputs. In this algorithm, we take the scene begin frame (*scene_start*) index and scene end frame (*scene_end*) index as the starting and ending points of the frames. We consider a hop length as a distance between any two extracted frames. Taking into account that each of the scenes has different lengths, hop length (or size) will differ from one scene to another. However, the outputs will be equal to frame_rate times image dimensions. We also added a removed fraction of frames (10 frames) from the start frame and subtracted 10 from the end frame. This allowed us to eliminate the most commonly seen fade-in fade-out noise caused by the scene transitions. Because we divide the start and end index frames by frame_rate, and frame numbers indices should be integers; we rounded down the value to the nearest integer for the hop_size. This may later cause the size of output to become one more than frame_rate. To prevent this, we took only the first frame_rate part of the frame_list given in Equation.4.7.

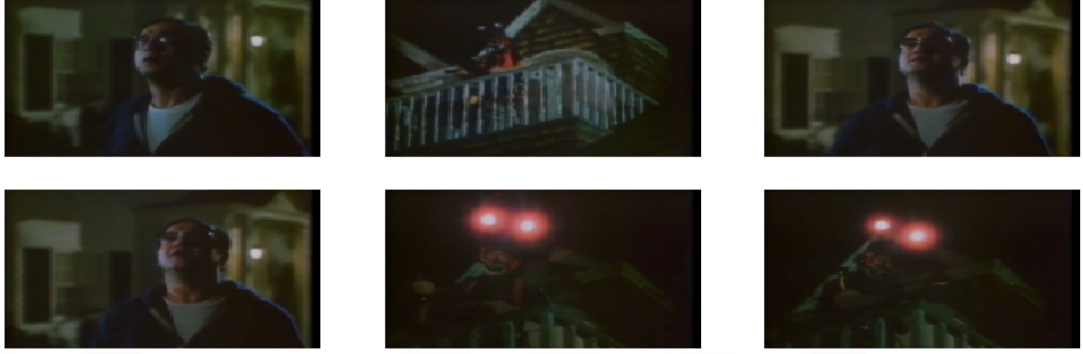


Figure 4.3: SEFE Example

$$par_{start} = scene_{start} + 10 \quad (4.4)$$

$$par_{end} = scene_{start} - 10 \quad (4.5)$$

$$hop_size = \frac{par_{end} - par_{start}}{frame_rate} \quad (4.6)$$

$$frame_list = (X_{par_{start}}, X_{par_{start}+hop_size}, \dots, X_{par_{end}}) \quad (4.7)$$

$scene_{start}$: scene start frame index

$scene_{end}$: scene end frame index

par_{start} : partition start frame index

par_{end} : partition end frame index

hop_size : intervals between frame indexes to equally segment the scene

$frame_rate$: how many frames will be produced (given by user)

$frame_list$: the frame list we end up with

In the SEFE equation, par_start refers to the frame number that is the starting point of the partition, which is calculated by the scene end point and the scene start points for each of the scenes (Eq. 4.7). hop_size refers to the equal distance we calculate from partition start and partition end frames. Lastly, we took the first frame as par_start and the last frame as par_end and extract the scenes with equal distances of hop_size .

4.1.2 Binary Model Architectures

In this section, we will explain our experimental model architectures. We worked with used genres in related work in Table 2.1 to apply a fitting algorithm to each different genre. For Animation, Comedy, Horror, Romance, Thriller, and Action genres we trained separate binary classification models to find out optimal values of hidden layers, hyper-parameters, and optimal frame rate algorithm and frame rate value which determines the number of frames to be extracted from each scene. Even though binary models proved to have different architectures and hyper-parameters, we combined 9 classes in a single model and combined a multi-output model with the knowledge of optimal parameters in other binary discriminators. In all of our architectures, we used the sigmoid function as the final activation layer because the sigmoid function is capable of solving multi-label classification problems.

We need to emphasize, our binary models were not used in the end product of multi-modal multi-label classifier. Our main goal with the binary models is to find the best hyper-parameters for the visual model especially. Moreover, in these experiments we provided, we only included best of the models. In our experiments, we early stopped training when over-fitting acquired in first 10 epochs. Although we kept parameters constant and changed a single parameter in each experiment, we did not include every model of trials to be brief and concise. We included only the models with better performance than random-forest (randomly classifying) and improved metric scores which will be explained in Section 5.1.

For binary discriminators that we will explain in the next Sections sections 4.1.2.1 to 4.1.2.5, we prepared datasets for each genre versus the other genres. For example, for Animation discrimination, we prepared a dataset containing 234 trailers where 117 of them belong to the Animation genre and other 117 others belong to any other genre. We kept these numbers equal for trailers to avoid class imbalance and to simplify the experiment process.

4.1.2.1 Animation Discriminator

In the Animation discrimination task, we set a frame rate as 4, which means we extracted 4 frames from each of the scenes. This rate was found successful for Animation discrimination task after experimenting frame rate values of 2,4,6,8 and 10 frames. Animated scenes do not have too many transitions in between frames whereas photo-graphic genres have more transitions. An optimal value of 4 for frame rate helped the model to discriminate the animated scenes from other genres. In this discriminator model, we used Intermediate Sectioned Frame Extraction with Equal Intervals (ISFE) algorithm to extract frames which were explained in Section 4.1.1.1.

To capture both spatial and temporal information, we used 3 blocks of time distributed CNN with 32, 64, and 128 units filters of 3 by 3; followed by a Gated Recurrent Unit (GRU) layer of 16 units (see Fig. 4.4). We also experimented with a higher number of GRU units which resulted in poor results. To prevent premature over-fitting, we accounted for l1 and l2 kernel regularizers, and we included a batch normalization layer with a momentum of 0.9 after each convolutional block and a dropout layer. For hidden units, we applied ReLU activation. For the final Dense layer, we used sigmoid activation since we give only one output that calculates the probability of the scene as a Animation class with values close to 1.0 or photo-realistic with values close to 0.0. We used Adam Optimizer with a 0.0001 learning rate, loss function as binary cross-entropy, and metric of evaluation as binary accuracy.

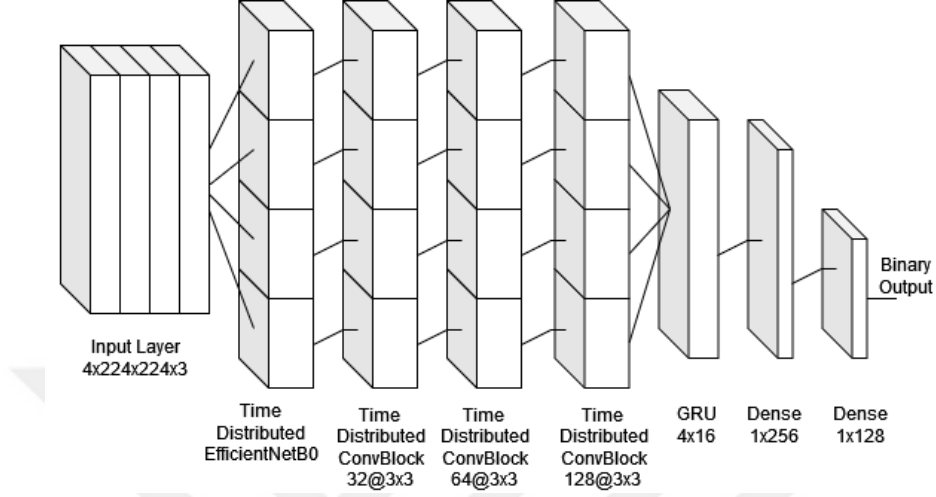


Figure 4.4: Animation Model Architecture

In order to accelerate the learning process, we used EfficientNet architectures, which are proved to be very shallow and fast, yet achieves high top-1-accuracy on the ImageNet dataset and a state-of-the-art accuracy on 5 other transfer learning datasets, with an order of magnitude fewer feature size [38]. The main reason to use this transfer learning network is to be explanatory for frames and efficient enough to execute in a Time Distributed fashion since we need to apply it to every time frame extracted from a scene. We also considered a few other transfer learning techniques, such as Residual Network (ResNet) and Visual Geometry Group Network (VGGNet). However, these models are not meant to do a profound search for objects in animated scenes and they demonstrated fast overfitting learning due to their depth, without providing a significant improvement on the validation set. Thus, fewer convolution layers are found to be helpful to discriminate Animations from other genres.

4.1.2.2 Comedy Discriminator

The Comedy genre covers any work in film or television whose general purpose is to create humor and intentional laughs for the audience [39]. For Comedy genre trailers, we expect to see laughing faces, colorful backgrounds, and actions of a

custard pie in the face, collapsing houses, a fall in the ocean, a loss of trousers or skirts, runaway crashing cars, people chases, etc. Therefore, we consider both spatial features and temporal features. Thus, unlike the Animation discriminator, we took spatial features into account as well. We observed different samples and 6 frames are optimal for frame_rate value when deciding on a comedy movie.

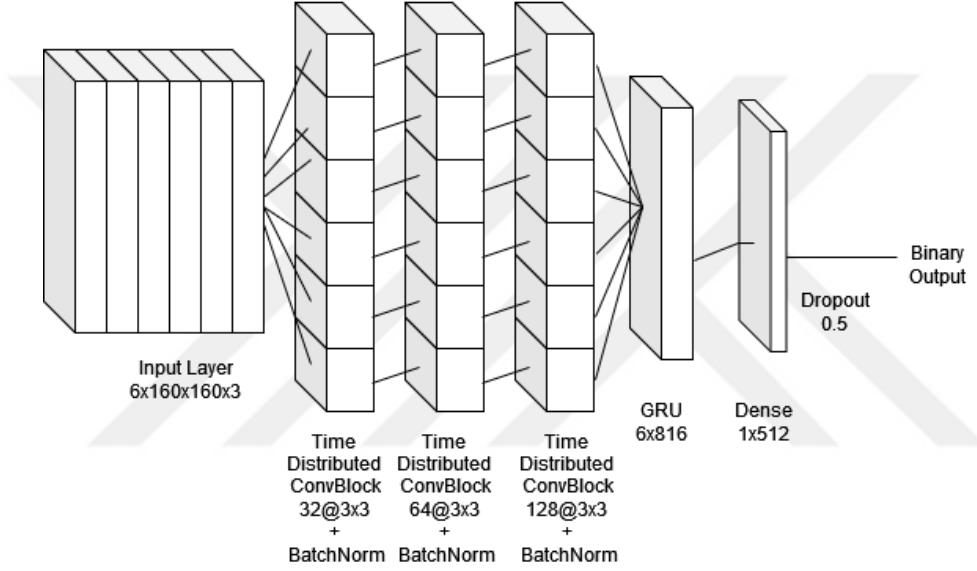


Figure 4.5: Comedy Model 1 Architecture

Since there are many samples for Comedy Label, we used the manual scene cleaned version of our dataset as explained in Chapter 3.3.3. We experimented with different numbers of hidden layers, hyper-parameters, and filters. In this section, we will present two of our best model architectures and parameters. In first architecture, we used time distributed convolutional blocks of 32, 64, and 128 units followed by GRU layer with 16 units, and finally flattened the GRU layer output to feed into 512 units of dense to produce a binary output as shown in Figure 4.5. In the second method, we used time distributed convolutional blocks of 32 and 64 units fed to two GRU layers with 16 and 8 units and concluded with a dense layer of 512 units.

Although we experimented with transfer learning in Comedy discriminator, it over-fitted fast on training data without significant improvement on the validation

data. Since we did not use a pre-trained model, we did not pre-process frame images with any methods. Convolutions through time were efficient enough to extract visual features of the frames.

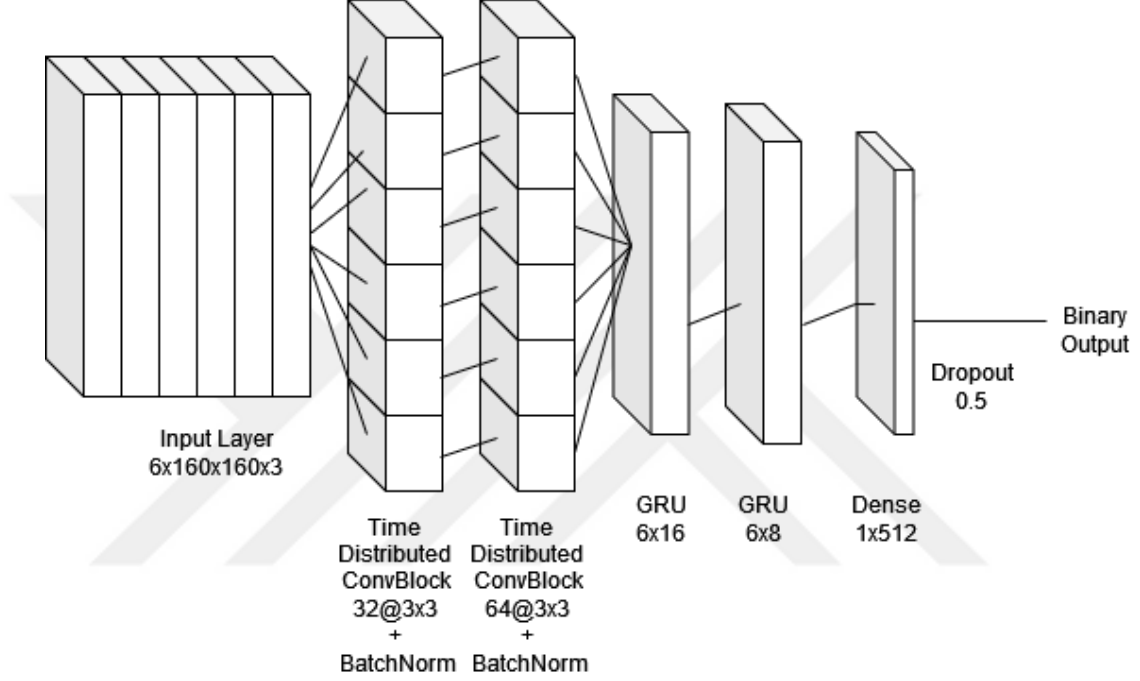


Figure 4.6: Comedy Model 2 Architecture

Moreover, we used Start to End Frame Extraction with Unequal Intervals (SEFE) from Section 4.1.1.2 for Comedy discrimination. This algorithm worked better for sudden movement changes to make the audience laugh in a comedy movie. Evaluation of models given in Figures 4.5 and 4.6 will be given in Section 5.4.1.

4.1.2.3 Horror Discriminator

A Horror film is a motion picture calculated to cause intense repugnance, fear, or dread [40]. Horror films include the theme of violence and terror, and they may be illustrations of disturbed, deformed, psychotic, or evil character. They may involve stories of terrifying monsters or mystery thrillers that use atmosphere

to build suspense. For the Horror genre, we did not have many trailer samples in manually cleaned data. Therefore, we used the original version created after the automated elimination model explained in Section 3.3.2. However, uncleaned scenes might cause poor results.

Some observed characteristics of horror movies include dark theme backgrounds, shocked or screaming human faces, and blood effects. Even though these characteristics make the horror genre distinguishable, there are scenes in horror genre trailers that do not give out information about the horror genre. Because there are scenes to create a mysterious environment and not every single scene in a horror movie is made to the audience scared, it is challenging to distinguish the genre of the trailer by looking at a single scene.

We used the ISFE algorithm for frame extraction in horror genre detection because in horror movies middle section of the scenes was more informative and there was less motion than in a comedy or action genre trailer. Our experiments proved that transfer learning works for the horror genre, however, deep networks such as Inception and ResNet were over-fitting in the 3rd epoch and were not informative. Thus, we used EfficientNetB7 with pre-trained ImageNet weights of the last 3 layers trainable, and without including the top layer. The model architecture is given in Figure 4.7.

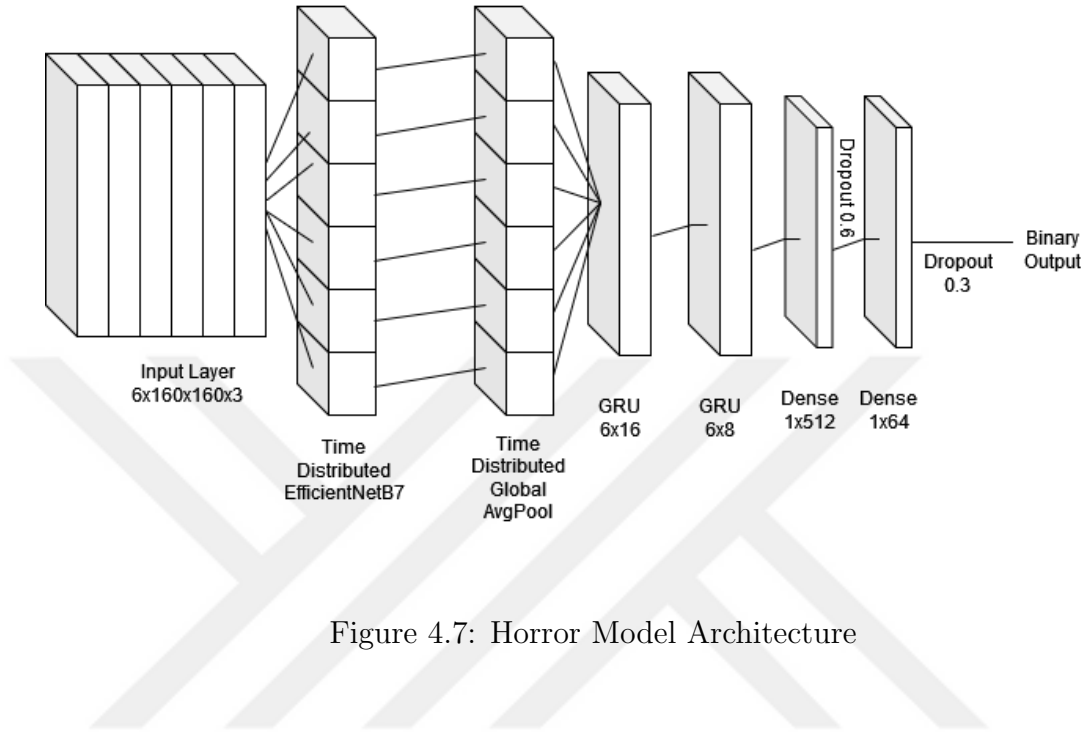


Figure 4.7: Horror Model Architecture

In Horror genre discriminator architecture we used fewer convolutions as they caused trouble trying to find objects in horror genre trailer frames. A characteristic of the horror genre is to create a mysterious environment by disguising objects with dark shadows or collisions. Fortunately, simpler features such as dark colors, and extreme close-up shots of human faces give away the genre when discriminating horror genre from others. Hence, we used more dense layers at the end of our model architecture. We also picked the frame rate as 6 frames as it was enough to capture motion in the horror genre.

4.1.2.4 Romance Discriminator

Romance films are love stories that center around two protagonists exploring some of the elements of love like relationships, sacrifice, marriage, obsession, or destruction [41]. The main difference between Drama and Romance genre movies is all Romance genre movies always ends with happy endings [41]. Therefore, it is challenging to distinguish between romance and drama genres just by looking at a trailer because trailers do not present the ending of the movie. As it will be

discussed later on in Section 5.4.3, these two genres appear very indistinguishable in their trailers. Therefore, we did not include a separate section for the Drama genre due to its similarity with the Romance genre.

Since we observed Romance trailers to have smoother actions, we used the ISFE frame extraction method. Moreover, Romance genre trailer scenes were longer than the other genres. Hence, even if we considered using lower frame_rate which believed to be enough the summarize the scene; we still used 6 frame_rate to get enough samples from the scene to draw the line between other genres and Romance.

Our architecture given in Figure 4.8 intended to learn spatial features rather than temporal features. We did not used transfer learning in this architecture, as the features learned from 3 convolutional blocks of 32, 64, and 128 units were informative. We added BatchNormalization layers at the end of each block to reduce over-fitting.

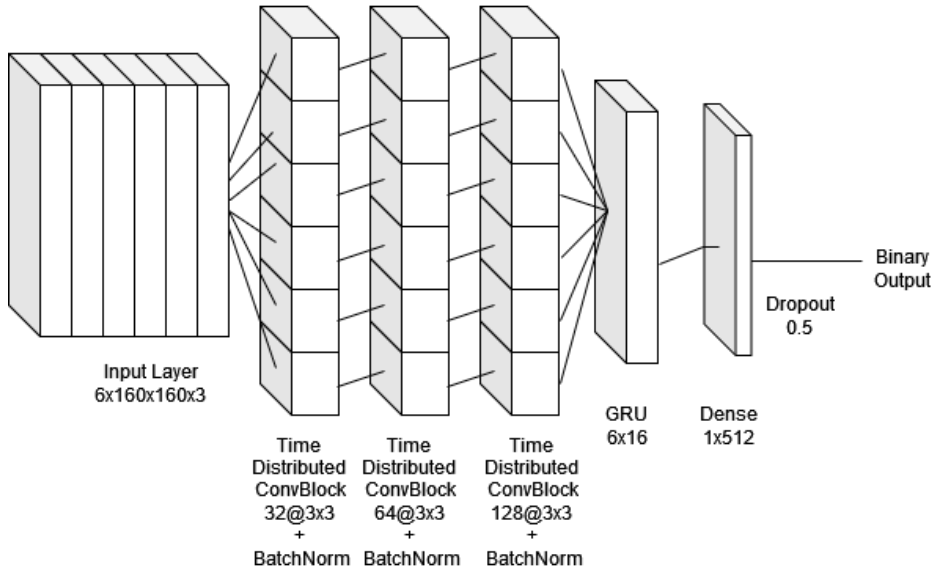


Figure 4.8: Romance Model Architecture

4.1.2.5 Action Discriminator

Action genre movies are fast-paced and includes a frequent action scenes such as fight scenes, chase scenes, and slow-motion shots etc. [42]. These high-octane films are more about the execution of the plot rather than the plot itself. Therefore, it is significant to describe the object-like features of Action genre films and their placement between each frame. According to Kendrick, the appearance of the action genre is more recent than the other genres [43]. In his words, “When the term ‘action’ was used as a generic descriptor prior to the 1970s, it was typically conjoined with ‘adventure’ to describe a range of films [43]”. Thus, after experimenting with the Action genre, we can also apply our knowledge to the task of describing genre features in adventure class.

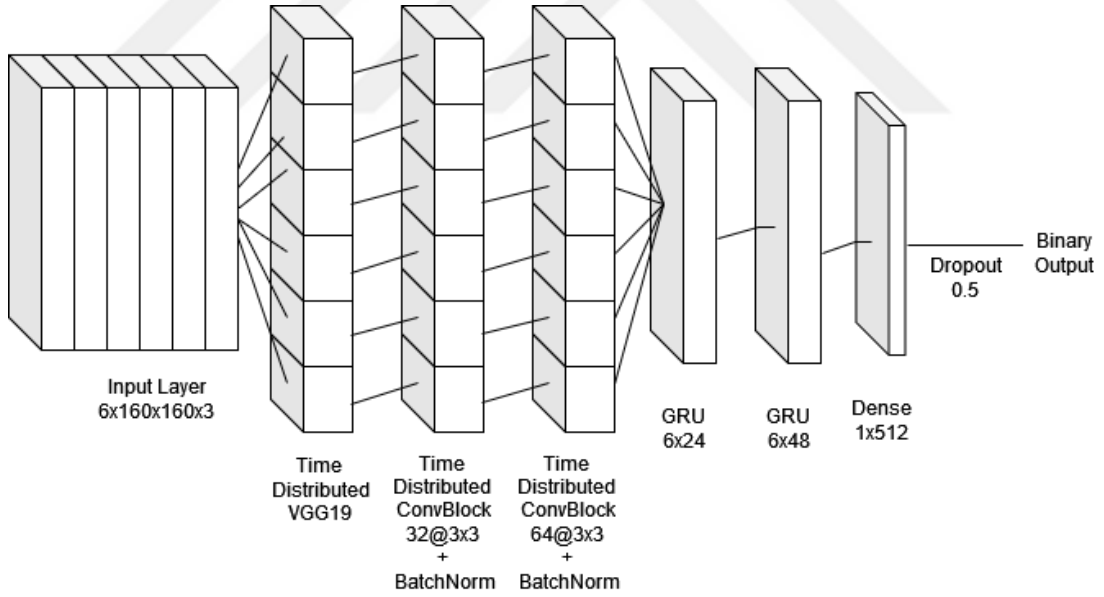


Figure 4.9: Action Model 1 Architecture

In the action discriminator, we mostly experimented with different transfer learning architectures because detecting the movement of certain objects such as guns, and knives in frames are significant. As it can be seen from Figures 4.9, 4.10 and 4.11; we experimented with EfficientNetB7, VGG19 and InceptionV3. We used an automatically cleaned version of our dataset to create equal subsets of binary labels with 0’s (not action genre) and 1’s (action genre).

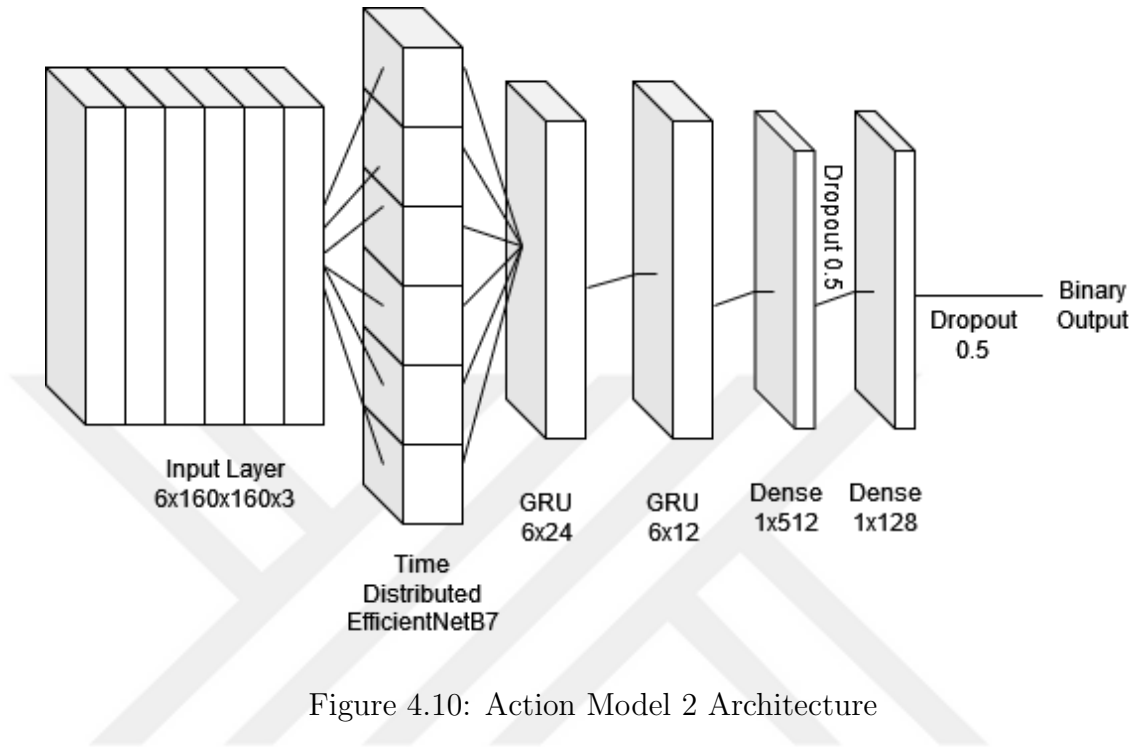


Figure 4.10: Action Model 2 Architecture

Since we required fast-paced transitions between frames, we used the SEFE algorithm. Although we experimented with values of frame rate bigger than 6, we observed no improvement of metrics, and in terms of efficiency, we preferred 6 as frame_rate. This value is also informative on more slow transitions in other genres, which might be the reason the value functioned well with our discriminator.

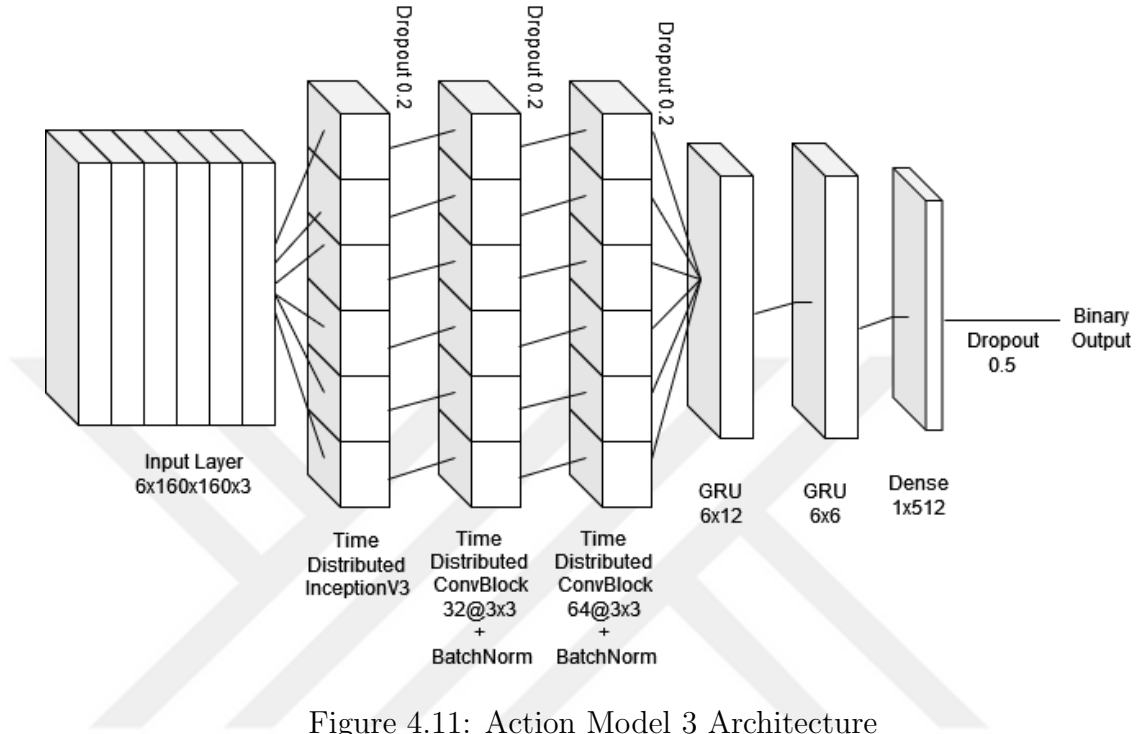


Figure 4.11: Action Model 3 Architecture

Noticeably, we used more GRU layers and units in these models compared to the romance or comedy genre. We experimented with quantity of GRU layers which works best with the corresponding transfer learning technique and concluded that InceptionNet requires more convolutional units than EfficientNet, and VGG19 features with higher units used in GRU layer improved learning performance.

4.1.2.6 Thriller Discriminator

The Thriller genre is a mix of mysterious environment, tension building, and generally builds anticipation into an exciting story [42]. Thrillers may often include a “ticking clock” sound, where the stakes are against a finite amount of time. According to Rankin’s research, the threat of death or capture is ever-present in the thriller genre and thrillers may include elements of science fiction, mystery, or horror. [44].

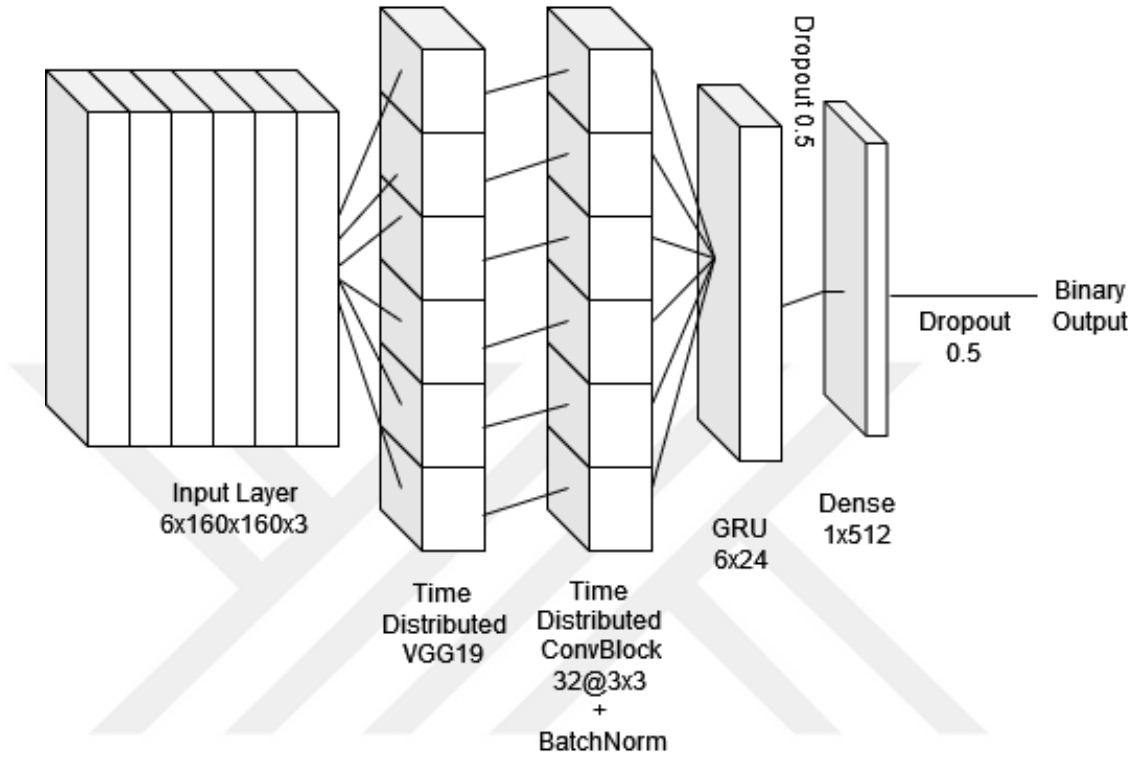


Figure 4.12: Thriller Model 1 Architecture

Although the thriller genre has many overlapping data samples with the horror genre, we aimed to distinguish these genres from each other. Our observations proved that usage of SEFE is adequate because thrillers have faster transitions than the Horror genre. To capture both movement and the differences of the objects between frames, we used a frame rate of 6 frames. We also experimented with VGG19 since the Action model functioned favorably with this transfer learning technique. Since we did not have many examples of the thriller genre in the manually cleaned dataset, we used the automatically cleaned dataset to prepare the Thriller vs other genres dataset, with equal samples from Thriller and non-Thriller genres.

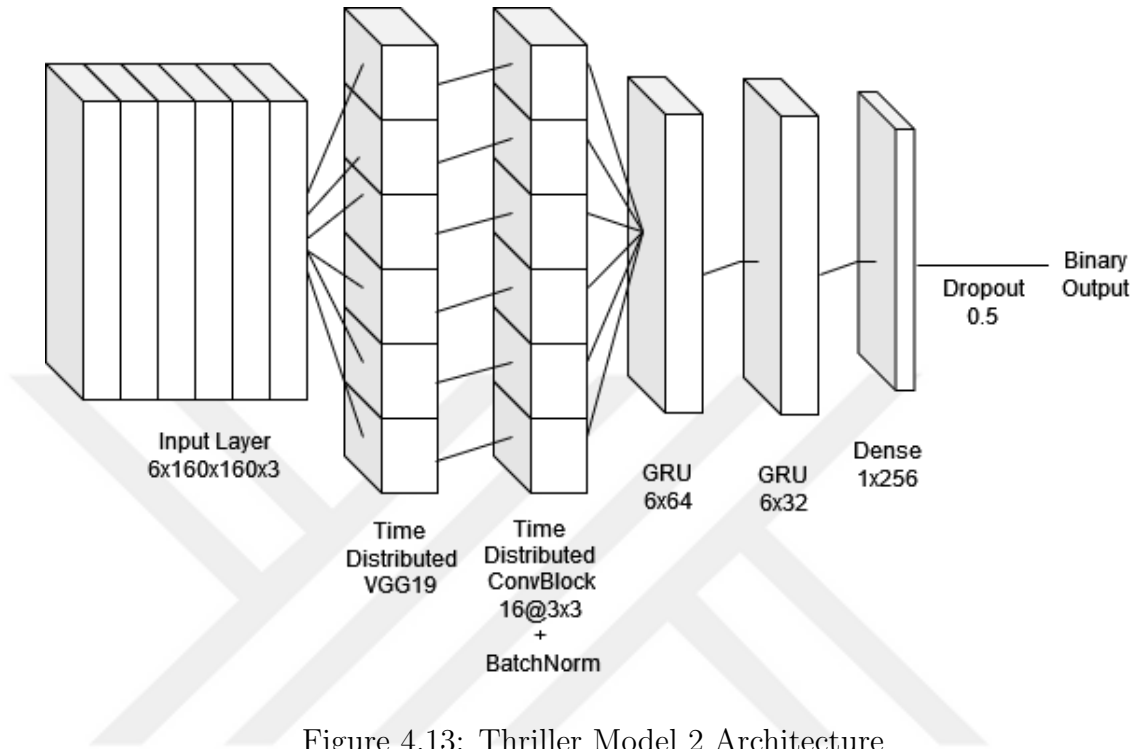


Figure 4.13: Thriller Model 2 Architecture

4.1.3 Multi-Label Prediction Model Architecture

In our multi-label prediction model, we discarded the Animation genre, as it is considered as a production method in which figures are manipulated to appear as moving images. We consulted Sami Arpa who is an expert in the movie producing area, to learn if the Animation genre is accounted as a movie genre [36]. Our interview concluded that Animation is a creative production type alongside narrative films, documentaries (which we already discarded), and multiple productions [36]. Moreover, we already built a successful model architecture to learn the animation genre. Therefore, we discarded the animation label from our dataset. Our final genres for the multi-label prediction model are: Action, Adventure, Comedy, Drama, Horror, Romance, Sci-Fi, Thriller, and Mystery. We removed other labels from our dataset. We removed the scenes from our dataset that does not belong to any of these genres. In the final database, we remained with 2904 samples from Drama, 2129 samples from Comedy, 1067 samples from Thriller, 1018 samples from Romance, 885 samples from Action, 605 samples from

Adventure, 597 samples from Horror, 476 samples from Sci-Fi, 339 samples from Mystery genres in the dataset created from the automatically cleaned dataset explained in Section 3.3.2.

For multi-label prediction models, we experimented with different transfer learning techniques, varying numbers of hidden layers, and different cost functions. Two different datasets we practiced are given in Sections 3.3.3 and 3.3.2. Our architectures are given with their evaluation scores in Table 5.4.2. Unlike a few binary model architectures we applied ISFE successful, we employed the SEFE algorithm for multi-label data generators because its adequate with most of the genres. We used sigmoid activation in the final layer as it is more compatible than the other activation functions for the multi-label classification task.

We created and experimented with 6 model architectures in which we used both manually cleaned versions and automatically cleaned versions of our dataset as explained in Section 3.1. Our model architectures are given in Table 5.4.2 with their relative evaluation results. T-Conv2D represents Time-Distributed Convolution layers which means that the convolutions are applied to each frame in the sequence of the visual data. We used *frame_rate* of 8 frames for these models. We incremented the frame rate compared to the binary model because multi-label model is more complicated and more features helped training performance.

4.2 Audio Training

Our two main challenges on audio training was the variable length of audio samples and scenes were partitioned according to the visual features. Therefore, we followed a few different methodologies to outcome these challenges.

In our first attempts, we experimented with training from variable length audio samples which failed to learn from audio data. In this experiment, we practiced length Mel-Frequency Cepstrum Coefficient (MFCC) features. These features are the presentation of the short-term power spectrum of a sound. There are a

few steps in extracting MFCC features : (1) windowing the signal, (2) applying the Discrete Fourier Transform (DFT), (3) taking the log of the magnitude, (4) warping the frequencies on a Mel scale, and (5) applying the inverse Discrete Cosine Transform (DCT) [45]. There were two most commonly used libraries in Python for extracting MFCC features. One provided by librosa² and other provided by python_speech_features³. We tried these libraries with parameters that should make outputs of both similar. However, we could not manage to achieve the same visualization of MFCC features. Due to the conflict between two open source libraries, we developed our MFCC feature extraction class. Although visualization of MFCC features seemed correct, once we fed these features to variable length architectures, learning performance was low and close to the random forest (randomly predicting the genres or biased towards imbalanced data).

In our successful attempt where we windowed audio segments and created an equal length of audio samples to extract audio level features and later trained a model. In the next section, we will explain our audio feature extraction method and audio model architecture.

4.2.1 Audio Feature Extraction

For audio features, we used Spectral Centroid (SC) feature extraction method. The Spectral Centroid (SC) is the center ‘gravity’ of the spectrum [46]. We used the Librosa library to extract spectral centroids for each frame which is a magnitude spectrogram that is normalized and treated as a distribution over frequency bins, from which the mean (centroid) is extracted per frame ⁴.

The individual centroid of a spectral the frame is defined as the average frequency weighted by amplitudes, divided by the sum of the amplitudes. More

²<https://librosa.org/doc/main/generated/librosa.feature.mfcc.html>

³<https://python-speech-features.readthedocs.io/en/latest/>

⁴https://librosa.org/doc/main/generated/librosa.feature.spectral_centroid.html

precisely, the centroid at frame t is defined as Equation 4.8:

$$C_t = \frac{\sum_{k=1}^{W_{f_l}} k X_t(k)}{\sum_{k=1}^{W_{f_l}} X_t(k)} \quad (4.8)$$

While using Spectral Centroid (SC) features researchers observed that higher values correspond to brighter sounds [46]. Because SC measures the spectral shape and position of the spectrum, a higher value of SC corresponds to more energy of the signal being concentrated within higher frequencies. Thus, these features can be used to classify movie trailers by capturing the background soundtracks with their frequencies and energy levels.

To construct our dataset, we used our cleaned data extracted from the elimination model explained in Section 3.3.2. Although our studies are focusing on variable-length audio samples, since we enough data samples, we extracted equal lengths of audio samples from each scene. We picked 4 seconds of windows to extract sound features as less than this value caused a major decrease in the feature size and it became challenging to classify the scenes shorter than 4 seconds just by audio features. We found this value by experimenting and listening to the sounds for less or more than 4 seconds, which proved that 4 seconds is the optimal value for our task. Thus, we removed data samples with lengths of less than 4 seconds.

For each of the scenes, we sectioned only the middle part of 4 seconds and we sampled 4 seconds of audio windows with a sample rate of 44100 Hz to simulate human hearing. To extract audio clips without tracing the videos from the beginning, we used the MoviePy library’s AudioFileClip module ⁵. A single data sample in audio training represents 4 seconds of audio taken from the middle of the scene. Thus, we need to access a random trailer’s a random scene’s audio. When we do not want to trace all the way to the beginning of each trailer to extract audio features from the other segments, it is called a random access. Although the Decord library also provides random access audio extractor, as well as the frame extractor, MoviePy’s audio extractor, attained better performance

⁵<https://zulko.github.io/moviepy/ref/AudioClip.html>

in terms of random access audio. After extracting audio waves, we pre-processed sound features to a separate file since the feature extraction process takes a long time. For each scene, we extracted exactly 345 SC features that represent audio of 120 frames.

Similar to the visual data separation, we separated trailers to be either in training data or validation data. All of the samples taken from a trailer are either in training or test sets. For the train split, we used 2800 trailers with 14621 scenes and for the validation split, we used 410 trailers with 2104 scenes. We also provided class weights based on our training data to prevent bias caused by the imbalanced data. Our data distribution by labels is given in Table 4.1.

Train Split			Validation / Test Split		
	Data Samples	Class Weights		Data Samples	Class Weights (not given to training)
Drama	6862	1.0	Drama	843	1.0
Thriller	4671	1.46906	Thriller	745	1.13154
Comedy	4027	1.70399	Comedy	669	1.26008
Mystery	3918	1.75140	Mystery	598	1.40969
Horror	3252	2.11008	Horror	408	2.06617
Romance	3115	2.20288	Romance	447	1.88590
Action	3098	2.21497	Action	484	1.74173
Adventure	2844	2.41279	Adventure	440	1.91590
Sci-Fi	2773	2.47457	Sci-Fi	353	2.38810

Table 4.1: Train Test Split Genre Distribution of Audio Training

4.2.2 Audio Model Architecture

Since we already extracted SC features, our model does not require a very *deep* neural network. We used Keras auto tuner to try different numbers of hidden layers, units of each hidden layer, and, learning rate. Keras tuner finds the hyper-parameters by experimenting with the given parameters to maximize the given cost or metric ⁶. Since our main goal is to increase precision, we directed our Keras tuner to maximize this metric. To provide more detail on why we emphasize precision metrics, we will explain our metrics in Section 5.1. The

⁶https://keras.io/keras_tuner/

Keras tuner converged the precision of the model with 5 sequential dense layers with units 416, 352, 32, 32, and 9 as can be seen in Figure 4.14.

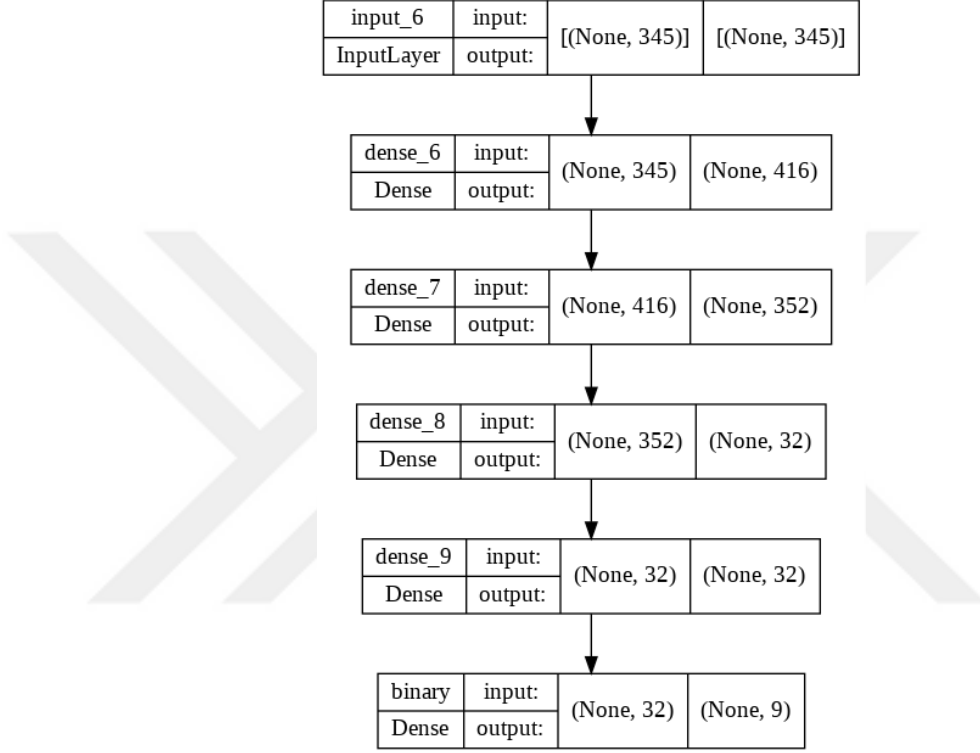


Figure 4.14: Audio Model Architecture

4.2.3 Multi-Modal Architecture

Multi-modal models are blended learning models that are designed to meet the needs of a variety of learners [47]. Human perception observes their surrounding with different types of modalities such as seeing colors and hearing sounds. To simulate human learning in machine learning tasks, researchers combine high-level embeddings from the different inputs of their modalities.

Working with video data, we are provided with both audio features and visual features. We proposed two different frameworks given in Sections 4.1.3 and 4.2.2, which we also handled working with multi-label outputs. In this section, we present a multi-modal architecture that combines multi-class audio and visual

inputs to output multi-label prediction results for 9 classes.

For a multi-modal network, we operated on automatically cleaned dataset from Section 3.3.2. Although this dataset did not give the best results in the visual model as can be seen from Table 5.4, we relied on previously learned weights of the manually cleaned dataset in the multi-modal network’s visual features part. We carefully selected the partition of the data in training and validation so that we do not include visual features of validation data on the previously trained weights of the visual branch. We discarded the last 3 layers of the Visual Model which are Dense(with 9 units), Dropout(with 0.5 dropout rate), and Dense(with 512 units). We fixed weights of the layers until the Dense(with 512 units) thus, only the Dense(with 1024 units) layer is trainable and other layers are not trained during the training of the multi-modal network.

For the audio branch of the multi-modal network, we fed 345 Spectral Centroid (SC) features explained in Section 4.2.1 and a portion of the dense net given in 4.2.2 with including Dense(with 416 units) and Dense(with 352 units). After concatenating visual and audio network outputs, we built a dense network with 864 inputs and 9 outputs. We used the Keras Tuner for the best hyper-parameters for the number of dense layers, and units of each dense layer at the end of the architecture. Our final architecture for the Multi-Modal Multi-Label Classification is given in Figure 4.1.3.

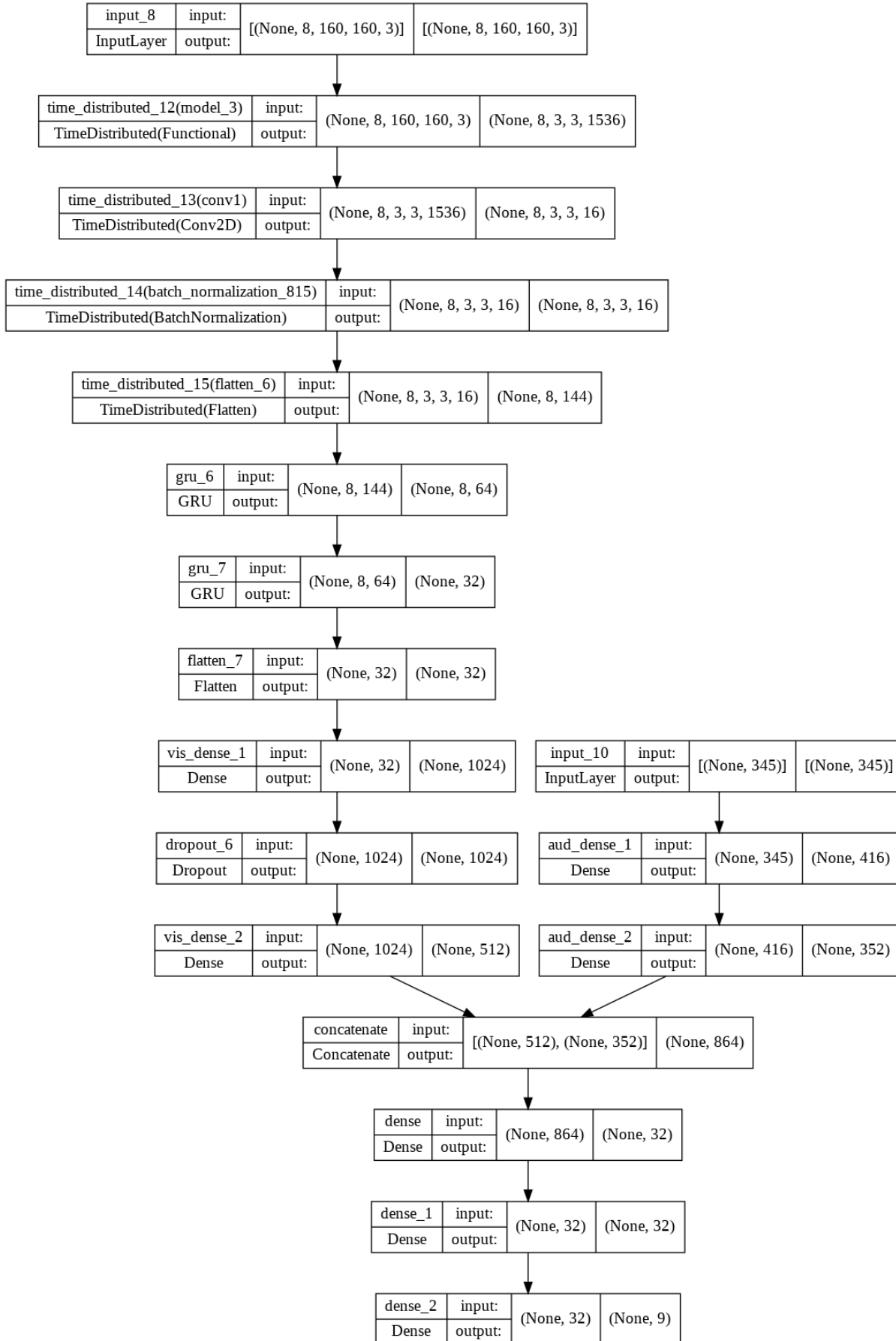


Figure 4.15: Multi-Modal Model Architecture

Chapter 5

Evaluation

5.1 Metrics and Cost Functions

In this thesis, we trained a deep learning network from multi - label inputs to output multi-label predictions. Multi-label classification (MLC) problem is considered as a complicated machine learning task since the more than one label might be assigned to each instance [48]. Difference between multi-class classification and multi-label classification is that, in multi-class problems the classes are mutually exclusive; whereas for multi-label classification problems each label represents a different classification task but the tasks are somehow related [49]. For example, classification of animals would be multi-class classification problem whereas an instance may belong to multi-label, such as our case, is named as multi-label classification. The real world problems are mostly multi-label classifications. For instance, to predict the set of birds species that are present in a environment, given a ten-second audio clip; given symptoms of a person to predict his illness; predicting music genres etc. [50].

Due to complexity of MLC, various measures have been developed from researchers. In this Section, we will explain our used metrics and cost (loss) functions. Determining these factors for multi-label input and multi-label prediction

is a requirement because unlike normal classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or “labels” [51].

Accuracy as a metric is used in many studies as an evaluation metric, because it usually refers to how often it achieves some goal. Accuracy measures only proportion of correctly measured to the total number of the data. However, for our challenging task of MLC accuracy fails to evaluate the model as we do not value only accuracy of getting the labels but also getting the negative predictions correctly in between these classes. Thus, our multi-label architectures require different techniques of evaluation.

5.1.1 Precision and Recall

Recall refers to the percentage of total relevant results correctly classified, and precision means the percentage of your results which are relevant. For a single class, the equation for precision and recall is given in Equation 5.1. Thus, precision is calculated by the proportion of correct predictions among all predictions of a certain class. Recall is the proportion of examples of a certain class that have been predicted by the model as belonging to that class.

$$\begin{aligned}
Precision &= \frac{1}{n} \sum_{i=1}^n \frac{|y_i \cap h(x_i)|}{|h(x_i)|} \\
Recall &= \frac{1}{n} \sum_{i=1}^n \frac{|y_i \cap h(x_i)|}{|y_i|}
\end{aligned}
\tag{5.1}$$

where n is the number of examples,

y_i is the target,

x_i is the i^{th} example,

$h(x_i)$ is the predicted labels for i^{th} example.

Undoubtedly, there is a trade-off between recall and precision metrics. If we maximize recall, more classes will be predicted which means that we will have both accurate results but also false positives. Since we are going to apply this system to a long movie as an end product, we require to have minimum amount of false positives. Thus, we have to maximize the proportion of true positives among all positive predictions, which is the precision. Precision is our main metric since we prioritize making most correct predictions even if it means we may miss some scene samples. We do not consider recall as a significant metric since we will have many samples that might be a true value for a genre, in an another section of the movie. Instead of finding all the true positive scene samples, we intended to avoid false positives.

For multi label classification tasks, recall and precision can be measured differently. The most straightforward way is to iterate precision metric calculation on every label. In each label iteration, current label prediction vs other labels predictions constructs values for True Negative, Positive or False Negative and Positive instances. Calculation of each metric through one vs all method and then taking the averages is called macro metric calculation. We will explain generalized versions of these metrics in Section 5.1.2

5.1.2 F1-Score

F1-score is a metric that combines both precision and recall of a classifier into a single metric by taking their harmonic mean. By calculating the F1-score, we can accomplish the balance between precision and recall. Since we already given equations for precision and recall in multi-label classification (Equation 5.1), f1-score is simply the mean of these mathematical expressions given in Equation 5.2.

$$F1score = \frac{2(P * R)}{(P + R)} \quad (5.2)$$

*where P is Precision given in Equation 5.1
and R is Recall given in Equation 5.1*

F1-measure effectively references the true positive predictions to the arithmetic mean of predicted positives and real positives [52]. For multi-label classification task, F1 score measures how well the system makes positive predictions that are correct proportion to the mean of real positive values. The F1 Score metric is a suitable measure for classification problems on imbalanced datasets. For a multi-class classification problem, the F1 score is calculated per class in a one-vs-rest manner. In this approach, we rate each class's success separately, as if there are distinct classifiers for each class.

To offer an overall F1-score, we average every classes F1-measure. When we have the arithmetic mean of the F1-scores, it is called macro-averaged F1-score. To account class weights while taking F1-measure, we used weighted-F1-score. For the micro-averaged F1-score, we first compute micro-averaged precision and micro-averaged recall over all the samples , and then combine the two. While calculating the micro versions of recall and precision, we take into account of every other class as negative and current class as positive. Thus, micro-averaged F1 score measures generalized versions of F1 measure and average is higher than the macro average since the F1 score of the majority class is the highest [53]. The Micro F1-score also represents the accuracy for the MLP.

In Equation 5.3 given below, we described the calculation of evaluation metrics with micro and macro averaging. $B(TP_j, FP_j, TN_j, FN_j)$ represent some specific binary classification metric [54] with given formulas in 5.1 and 5.2.

$$\begin{aligned}
B &\in \{Accuracy, Precision, Recall, F1score\} \\
B_{macro}(h) &= \frac{1}{q} B(TP_j, FP_j, TN_j, FN_j) \\
B_{micro}(h) &= B\left(\sum_{j=1}^q TP_j, \sum_{j=1}^q FP_j, \sum_{j=1}^q TN_j, \sum_{j=1}^q FN_j\right)
\end{aligned}
\tag{5.3}$$

where TN is the number of *True Negatives*
 FP is the number of *False Positives*
 TP is the number of *True Positives*
 FN is the number of *False Negatives*
 q is the number of labels (*movie genres*)

- The **macro-averaged** metrics (both precision and recall) are computed using the arithmetic mean of all the per-class metric scores [55]. Macro-average method can be advantageous for evaluating how the system performs overall across the sets of data.
- The **weighted-averaged** metric score is calculated by taking the mean of all per-class metric scores while considering each class's support [55]. Support refers to the number of actual occurrences of the class in the dataset. Thus, if we have imbalanced data it is preferable to evaluate the results with weighted-average scores.
- The **micro averaged** score computes a global average metric score by counting the sums of the True Positives (TP), False Negatives (FN), and False Positives (FP) [55].
- According to *keras*¹, these metrics are calculated globally, if we did not give top k value. Top k value is an int value specifying the top- k predictions

¹https://www.tensorflow.org/api_docs/python/tf/keras/metrics

to consider when calculating metrics. However, keras expects labels as either 0's and 1's. Thus, giving the results of prediction probabilities causes lower metric values even if a model produces correct outputs. Moreover, metrics in keras can be also considered by computing which predictions above the threshold and/or in the top- k highest predictions, and computing the fraction of them. Both of these translation methods were not suitable for our task since we have 9 classes and fractions are becoming smaller with n class or thresholding. Therefore, we developed our one-hot label translating system from the predicted probabilities which is explained in Section 5.2.

In the Table 5.2 we presented micro, macro, weighted and samples average for each of these evaluation metrics. In Table 5.4 we calculated F1 score, precision and recall as the arithmetic averages of the measurements for each class.

5.1.3 Hamming Loss

Gao et al. show that some recent multi-label learning approaches are inconsistent even for deterministic multi-label classification whereas hamming loss is reliable and consistent for the deterministic case [56].

As a case study, let's consider we have n classes in a mutually inclusive nature. If a system always labels all of the classes as 1's, it will hit accurate results in a binary view. However, a misclassification will not be punished even though the system has the bias towards labeling everything. This system should be punished for not being precise in its decisions. On another note, let's consider our system as it learns to predict every class as 0. Thus, whenever we have a real 0 in the dataset, it correctly labels. However, this system is actually not successful as it learns to predict nothing and takes no risks. Therefore, we can say that in multi-label classification, a misclassification is no longer a fully wrong or correct. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them or all of them. This is the case

when Hamming Loss is effective and reliable. Hamming loss is proved to balance precision, recall and accuracy in multi-label prediction [57].

In its simplest of terms, Hamming-Loss (HL) is the fraction of labels that are incorrectly predicted; i.e., the fraction of the wrong labels to the total number of labels [49].

$$HL = \frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} xor(y_{i,j}, z_{i,j}) \quad (5.4)$$

where $y_{i,j}$ is the target and $z_{i,j}$ is the prediction.

5.2 Maximum Gap Thresholding

Our classifiers for multi label prediction output a vector of per-class probabilities, and these probabilities can be converted to a binary vector by setting the values greater than a certain threshold to 1 and all other values to 0. One hot encoding means translating results of probabilities into 1's and 0's. It is an essential process of converting the categorical data variables to be provided to learning algorithms which in turn may suggest a better evaluation for the model. There are several methodologies proposed for this task. We used a type of one hot encoding in one of previous steps, data generator to map multi-label string list inputs to the 9 elements corresponding to the 9 genres, to a binary vector. However, this was a dictionary translated to an one-hot encoded vector since belonging to a genre by represented by 1 and not belonging to a genre shown by 0. On the other hand, our system gives output of 9 classification probabilities to each genre. Therefore, we need a translation algorithm one hot encoding method to evaluate our system to derive a confusion matrix.

In Section 5.3, we will also explain a survey conducted to examine human perception and we will evaluate our results compared to the human results. For this task, we considered a few different methods to translate both human perception and our system predictions to one hot encoding. Subsequently, we could derive

confusion matrices, and F1-score to compare the results.

The most straightforward method for translating the prediction results to one hot vectors is thresholding the output array with a meaningful value. For example, an output of array $[0.9, 0.2, 0.3, \dots]$ will be translated to $[1, 0, 0]$ when we given a one hot encoding with a threshold of 0.5. For this task, we experimented, and concluded that best results are achieved with threshold value equal to one divided by the number of classes. In the multi-modal case, we have 9 classes which means prediction results greater than 0.11 should be labeled as one and less than 0.11 should be labeled as 0.

Another commonly used method is to pick the best n variable probabilities. For example, an output of array $[0.9, 0.2, 0.3]$ will be translated to $[1, 1, 0]$ when we give an one hot encoding with $n=2$ best probabilities. From our experiments, we have seen that both of these methods are rather simple for our task and we came up with a new method to translate probabilities to one hot encoding.

On the other hand, both of these methods rely on certain numbers we calculate and some intuitions we make, thus they may not simulate the natural distribution of classes very well. Thus, we introduced our own technique for translating prediction results. In this section, we will explain a new one hot encoding method we proposed for evaluation of our model versus human perception.

Algorithm 1 Maximum Gap Thresholding Algorithm

```

1: procedure MAXIMUM GAP THRESHOLDING(input)
2:    $sorted\_input, sorted\_indices \leftarrow \text{SORT\_DESCENDING}(input)$ .
3:    $output[sorted\_indices[0]] \leftarrow 1$ .
4:   for  $i \leftarrow 1$  to  $\text{LENGTH}(input) - 1$  do
5:      $distances[i] \leftarrow sorted\_indices[i] - sorted\_indices[i + 1]$ .
6:    $max\_distance\_index \leftarrow \text{ARGMAX}(distances)$ .
7:   for  $i \leftarrow 1$  to  $\text{LENGTH}(input)$  do
8:     if  $i \leq max\_distance\_index$  then
9:        $output[sorted\_indices[i]] \leftarrow 1$ .
10:    else
11:       $output[sorted\_indices[i]] \leftarrow 0$ .

```

Maximum Gap Thresholding given in Algorithm 1, requires probability vector

to be in order. We also keep track of indices of sorted vector while doing the sorting. Later, we compute a distance vector which is distances between two adjacent vector elements. Thus, if the input vector to the algorithm is size n , we calculate a distance matrix size of $n-1$. This algorithm returns at least 1 class, which means that maximum prediction result is ensured to be encoded to 1. An example of execution of the MGT on a vector $[0.04, 0.08, 0.03, 0.22, 0.05, 0.26, 0.02, 0.01, 0.27, 0.02]$ is given in Table 5.1.

Original vector:	0.04	0.08	0.03	0.22	0.05	0.26	0.02	0.01	0.27	0.02
After sorting:	0.01	0.02	0.02	0.03	0.04	0.05	0.08	0.22	0.26	0.27
Gaps:	0	0.01	0	0.01	0.01	0.01	0.03	0.14	0.04	0.01
Max gap:	0.14									
Therefore, 1 if the value ≥ 0.14 0 otherwise										
Predicted labels:	0	0	0	1	0	1	0	0	1	0

Table 5.1: Maximum Gap Thresholding Executed on An Example

5.3 Survey on Human Perception

Since there is not an exact evaluation to compare our model classification results because we created our dataset, we conducted a survey on how humans classify the movie genres. The survey was conducted in Bilkent University, and 302 people contributed to answer to the survey. In this experiment, we desired to measure how successful university students are in predicting the genre of a scene by looking at a few frames taken from the scene (10 to be exact.). The genres of the trailer given to the survey attendants are Action, Thriller, Horror, Comedy, Adventure, Sci-fi, Romance, Drama and Mystery. Users was also offered “indecisive” or “not sure” options if they could not decide the genre of shown trailer images.

The form was constructed with the Google Forms², and we ensured to avoid age restriction content such as violation of child safety, nudity and sexually suggestive content, violent or graphic content. In this research we also collected the data for age group and genders (with prefer not to say option) to learn how well the participant of different groups was labelling the images. This survey was Institutional Review Board (IRB) approved by the senate in the on April 29, 2022 at Bilkent University. The name of the survey was “Participate in Research Survey: Movie Trailer Classification Performance of Human”. Students of the Bilkent University participated to the survey voluntarily.

For distribution of genres in the survey, we applied the same distribution of genres with our dataset. We gave 50 total movie trailers with each from different trailers. These trailer’s actual labels are 16 Comedy, 12 Action, 5 Adventure, 20 Drama, 6 Sci-Fi, 3 Romance, 5 Mystery, 11 Thriller and 7 Horror genres as shown in Table 5.2.

We picked most visually representative screens of the each genre. We picked these scenes from manually cleaned dataset explained in Section 3.3.3, and survey resulted in overall weighted average precision of 0.48 as and confusion matrices for each genre representing human evaluation is given in Figure 5.1. Moreover, we calculated the accuracy of the survey as 0.831 to include in the Table 5.6 for comparison.

Individual responses kept anonymous by assigning unique identification numbers to each participant and by associating their responses only to this number. Since our experiment is anonymous, we also asked questions to extract information about the attendant profile (Given in appendix A). An example question is given in Figure A.6.

²<https://www.google.com/forms/about/>

	precision	recall	f1-score	support
Horror	0.40	0.57	0.47	7
Thriller	0.30	0.27	0.29	11
Action	0.45	0.42	0.43	12
Comedy	0.73	0.50	0.59	16
Adventure	0.67	0.40	0.50	5
Sci-fi	0.50	0.50	0.50	6
Romance	0.33	0.33	0.33	3
Drama	0.45	0.50	0.48	20
Mystery	0.20	0.20	0.20	5
Overall	Overall Precision	Overall Recall	Overall f1-score	Overall Support
micro avg	0.46	0.44	0.45	85
macro avg	0.45	0.41	0.42	85
weighted avg	0.48	0.44	0.45	85
samples avg	0.56	0.52	0.49	85

Table 5.2: Human Prediction Scores

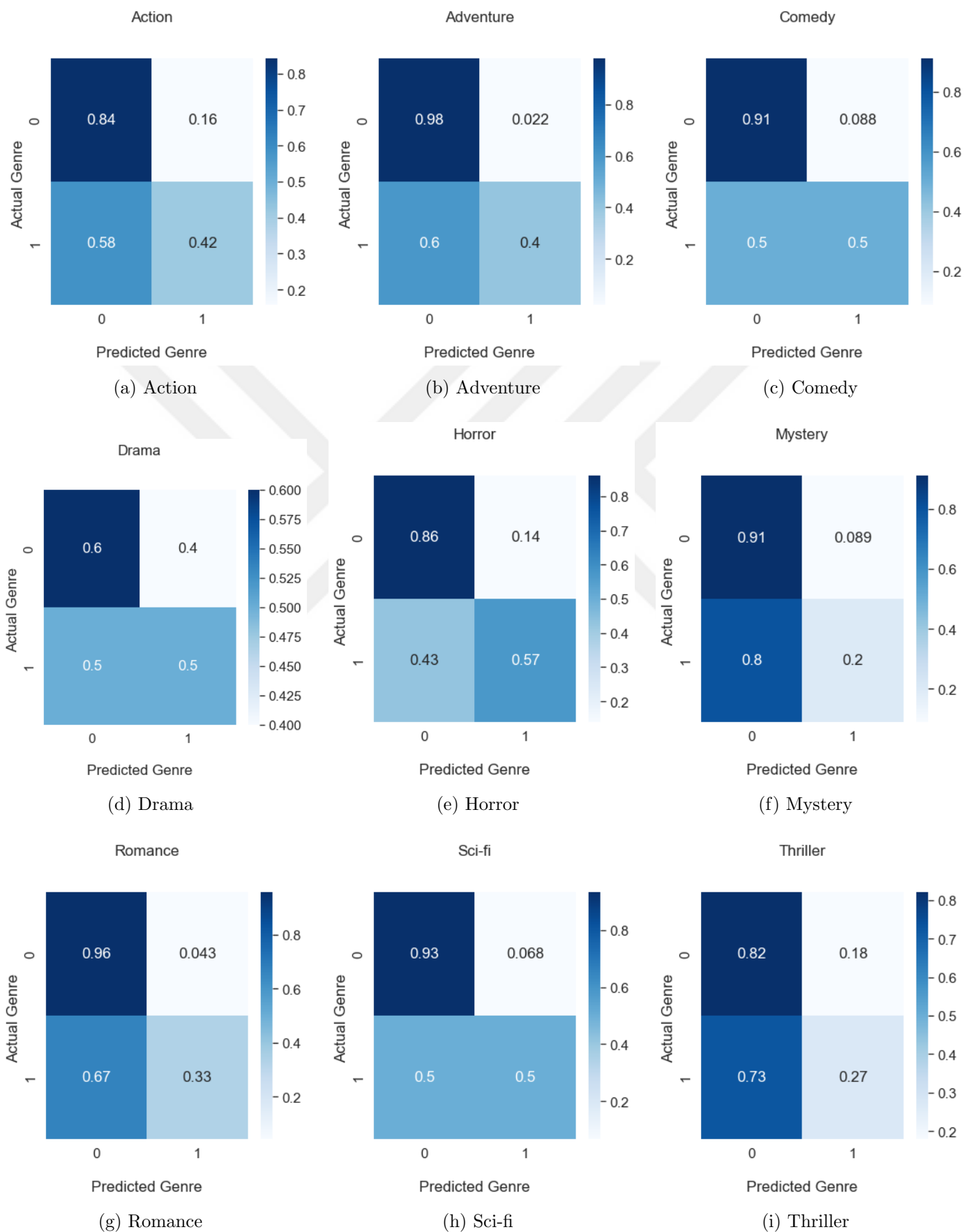


Figure 5.1: Human Prediction Results Confusion Matrices

5.4 Model Results

In this section, we will give results and evaluations of the model architectures given in Sections 4.1.2, 4.1.3 and 4.2.2. For binary models, the evaluation results are combined in Table 5.3. For multi-label cases, we first evaluated visual and audio model architectures separately, and then provided multi-modal architecture’s evaluation.

We also represented comparison of evaluation with the related work. Although we have not used the exact dataset given in the related work and created our own, some parts of our research are comparable. Since we created our dataset, our main evaluation element is the human perception survey given in Section 5.3 which we also conducted on our manually cleaned dataset.

5.4.1 Evaluation of Binary Models

In this section, we will evaluate each of the binary models that were built to experiment with different architectures. We combined and represented all of the binary model scores, data sample information, and the train validation split quantities on the Table 5.3. Since we achieved outstanding results with animation discriminator and animation is not considered as a movie genre, we later discarded the Animation genre on the multi-label discriminator.

Model Architecture	Dataset Cleaning Method	Frame Extraction Algorithm	Data Split (Trailer-wise)	Train/ Validation Split (Trailer-wise)	Train/ Validation Split (Scene-wise)	Transfer Learning	Val Binary Accuracy	Val Precision
Animation Model	automatically	ISFE	117 Animation 117 Others	200 train , 34 validation	1288 train , 289 validation	EfficientNetB0	0.9554	0.9324
Comedy Model 1	manually	SEFE	112 Comedy 112 Others	200 train, 24 validation	1833 train, 210 validation	-	0.6518	0.6753
Comedy Model 2	manually	SEFE	112 Comedy 112 Others	200 train, 24 validation	1833 train, 210 validation	-	0.6537	0.7278
Horror Model	automatically	ISFE	137 Horror 137 Others	200 train, 74 validation	1749 train, 738 validation	EfficientNetB7	0.6354	0.6932
Romance Model	automatically	ISFE	325 Romance, 325 Others	60 train, 50 validation	6905 train, 450 validation	-	0.6198	0.5902
Action Model 1	automatically	SEFE	750 Action, 750 Others	1200 train, 300 validation	13321 train, 2802 validation	VGG19	0.6464	0.5832
Action Model 2	automatically	SEFE	750 Action, 750 Others	1200 train, 300 validation	13321 train, 2802 validation	EfficientNetB7	0.6132	0.6009
Action Model 3	automatically	SEFE	750 Action, 750 Others	1200 train, 300 validation	13321 train, 2802 validation	InceptionV3	0.6212	0.6794
Thriller Model 1	automatically	SEFE	1067 Thriller, 1067 Others	1700 train, 434 validation	18304 train, 4340 validation	VGG19	0.6406	0.5642
Thriller Model 2	automatically	SEFE	1067 Thriller, 1067 Others	1700 train, 434 validation	18304 train, 4340 validation	VGG19	0.6847	0.6787

Table 5.3: Binary Models Evaluation

Evaluation of Animation Discriminator: In the training step, 1258 scenes extracted from 200 trailers are used to train the model and 189 scenes extracted from 34 trailers are used to validate it. To prevent the testing of the model with trailer scenes that are already in the training data, trailers are separated as training and validation beforehand.

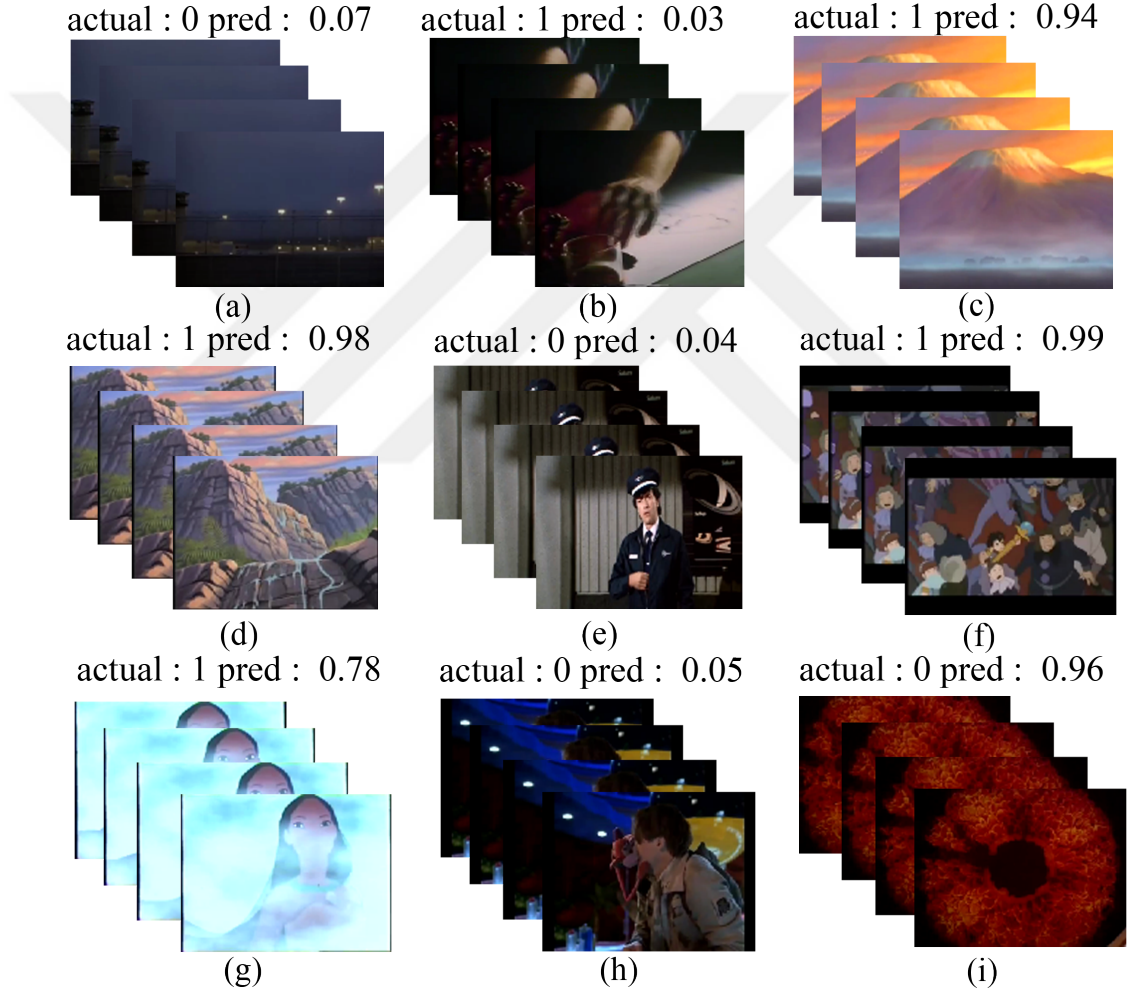


Figure 5.2: System Prediction Results

Some prediction examples of our final model are given in Fig. 5.2. We labeled the animated scenes as label 1 and the non-animated ones as 0. The prediction result of a scene is an approximation of the probability of being labeled as animation. Our architecture is successful in detecting complex and crowded scenes such as Fig. 5.2.f. Despite low background information given in some photo-realistic

scenes, the system could distinguish cartoon characters from photographic humans as it can be seen in Fig. 5.2.e.

On the other hand, there might be both animated scenes and photographic ones in the same trailer, and since we labeled each scene of a trailer with the same label as the trailer, the labels were not provided to the system concisely. Consider the example given in the validation set in Fig. 5.2.b, the scene that the trailer belongs to is a cartoon, but the scene itself is photographic. There are some scenes in the photo-graphic movie trailers that can resemble an animated scene like in Fig. 5.2.i. On the other hand, our model is successful in discriminating photo-graphic landscape in Fig. 5.2.a vs animated landscapes in Fig. 5.2.c-d, and could classify foggy scenes such as Fig. 5.2.g. Despite the challenging scenes we gave as examples, our model acquired **0.9552 binary accuracy** on our test set in 12 epochs which can be considered an outstanding result compared to the state-of-the-art given the related work in section 2.1.1.1. After 12 epochs, the over-fitting was observed and validation loss increased, although the training loss kept declining.

Evaluation of Comedy Discriminator: As it can be observed from Table 5.3, comedy model 2 acquired better result in both precision and accuracy. This proves that the Comedy genre can be distinguished from other genres with less convolutional blocks which means less spatial information was sufficient. We also observed that transfer learning techniques to catch spatial information in every frame were not beneficial to distinguish comedy genre. On the other hand, Comedy genre was more successfully classified in the second framework where there are more GRU units that highlight temporal information.

Evaluation of Horror Discriminator: For Horror genre, most fitting transfer learning technique was EfficientNetB7. We considered other transfer learning techniques which resulted in poor learning curves. Moreover, we observed that individual scenes of Horror genre trailers are hardly recognizable as the whole

trailer represents the genre more resolutely. We achieved one of the lowest accuracy scores compared to the other genres as can be observed from Table 5.3. Lack of Horror data samples in the manually cleaned dataset caused us to use the automatically cleaned. Pre-cleaned version of the horror model could achieve improved performance due to the removal of the non-informative scenes.

Evaluation of Romance Discriminator: Romance genre is a widely available genre in our dataset. However, this labels co-occurs with other genres in most of the data samples. Although our initial thoughts would be a transfer learning technique would helpful, our experiments concluded that Romance genre is better detected without transfer learning, considering that Romance genre can be cued with simple features such as color channels. The low accuracy we achieved might be caused by the removal of overlapped samples. For example, if a trailer belongs to both Romance and Comedy genres, we labeled it as 1, and if a trailer belongs to drama and comedy label we labeled it as 0. This caused loss of information as the model failed to determinate the Comedy genre labeled as 0.

Evaluation of Action Discriminator: The most challenging discriminator along the genres we experimented was the action discriminator. In order to benefit from the transfer learning techniques, we evaluated VGG19, EfficientNetB7 and InceptionV3. In comparison table given in 5.3, InceptionV3 acquired best precision score and VGG19 acquired best binary accuracy score. We believe that complexity of Action genre was caused by the shorter scene lengths and less information in the Action genre. The SEFE algorithm was more favorable than the ISFE algorithm to capture shorter changes in the action genre.

Evaluation of Thriller Discriminator: Thriller genre appears to have more distinct features than the other genres and our models could archive superior results than the other genre discriminators. Comparing two Thriller models, we can observe that modelw with more GRU units fit better to the Thriller features. For Thriller genre discriminator, temporal information highlighted whereas spatial

information remained less valuable.

5.4.2 Evaluation of Visual Only Multi-label Model

In Table 5.4 6 of our experimented models with respect to their validation split sample scores are displayed. In comparison, for multi label classification task, the higher number of GRU units achieved better results in terms of precision, and less convolutional layers after transfer learning helped learning curves to slowly learn the training split while improving on the validation splits.

Not surprisingly, cleaned version could achieve better results thanks to the removal of uninformative data samples as can be seen from Table 5.4. This also indicates that the human labeling can improve our performance as a future work.

Noticeably, transfer learning accelerated the training and especially Inception ResnetV2 was a good fit for our system. Our best model in terms of precision and recall given the manually cleaned dataset has acquired **0.6476** average precision and **0.4723** average F1 score as can be seen in Table 5.4. We also observed Time Distributed Convolution with 16 units archived improved results than two sequential Time Distributed Convolution with 8 units. Usage of hamming loss also boosted the precision compared to the binary loss.

Although we used scenes separately for our task which reduced the information size, we acquired better results compared to the state-of-the-art given the most relatable research by Wehrmann et al. [17]. In their research, they achieved 0.646 precision for 9 classes whereas we have reached 0.648 precision.

Even though we value F1 score and precision metrics of our models, we also observed that related work considered accuracy metrics. Thus, we included our accuracy scores in the Table 5.4. We also observed hamming loss acquired improved scores in terms of given metrics than the binary loss. As described in section 5.1.3, hamming loss has maintained balance between precision, recall, and binary accuracy which resulted in better F1 score.

Compared to the human perception survey on the 50 of the trailers with 50 total scenes in manually cleaned dataset that acquired 0.48 precision and 0.45 F1 score, we acquired 0.65 precision and 0.48 F1 score with our model architecture given 762 trailers with total of 6335 scenes (refer to the Table 3.1).



Model Architecture	Loss Used	Dataset Used	Transfer Learning	Val Precision	Val Recall	Val Binary Accuracy
T-Conv2D(32), T-MaxPool2D(2,2), T-Dropout(0.5), T-Conv2D(64), T-MaxPool2D(2,2), T-Dropout(0.5), T-Conv2D(128), T-MaxPool2D(2,2), T-Dropout(0.5) GRU(32), GRU(8), Dense(512), Dropout(0.5),	hamming loss	automatically	-	0.5254	0.1530	0.7907
T-Conv2D(32), T-Dropout(0.5), T-Conv2D(64), T-Dropout(0.5), T-Conv2D(128), T-Dropout(0.5), GRU(32), GRU(16), Dense(512), Dropout(0.5),	binary loss	automatically	Resnet	0.5137	0.0859	0.7890
T-Conv2D(8), T-Conv2D(8), GRU(64), GRU(32), Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5)	hamming loss	automatically	Inception ResnetV2	0.5393	0.1752	0.8031
T-Conv2D(8), T-Conv2D(16), GRU(64), Dropout(0.2), GRU(32), Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5)	binary loss	manually	-	0.5493	0.3640	0.8587
T-Conv2D(8), T-Conv2D(8), GRU(64), GRU(32), Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5)	binary loss	manually	Inception ResnetV2	0.6271	0.3719	0.8646
T-Conv2D(8), T-Conv2D(8), GRU(64), GRU(32), Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5)	hamming loss	manually	Inception ResnetV2	0.6020	0.2945	0.8555
T-Conv2D(16), GRU(64), GRU(32), Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5)	binary loss	manually	Inception ResnetV2	0.6476	0.3717	0.8651

Table 5.4: Visuals Models Evaluation

5.4.3 Evaluation of Audio Only Multi-label Model

Since we used keras tuner for audio training hyper-parameters on the pre-extracted SC features, we acquired an optimal evaluation score given these features. Given the data splits in Table 4.1 and model 4.2.2, we acquired 0.64327 precision and 0.7460 accuracy with 9 classes. Unfortunately, we acquired very low recall of 0.1019 which caused a low F1 score of 0.1759 as can be seen in Table 5.6.

Audio samples standalone could not achieve a good result in terms of recall and F1 score. One possible reason is that mapping audio signals to only 345 features could not make sense to the model. Another reason is since we partition videos with respect to their visual information, partitioning disturbs the continuity of the audio signals. Moreover, we extracted only the middle window of 4 seconds from each scene, which causes us to loss information of placed before and after those audio windows.

5.4.4 Evaluation of Multi Modal Multi-label Model

On Table 5.6 we gave the test binary accuracy, precision, recall and f1 scores calculated with globally calculating the metrics from the prediction probabilities with Keras's metric calculation methods. In Table 5.5 we show the one hot encoded versions of the predicted values calculated with the Maximum Gap Thresholding (MGT) explained in Section 5.2. The images given in Figure 5.3 provide the confusion matrices for each of the labels, after we translated prediction probabilities with MGT.

The improvement between multi-label visual or multi-label audio and the multi-modal architecture is significantly progressive. We have a few reflections about the advancement in learning.

- Since we used 4 seconds minimum of scenes for the audio feature extraction,

shorter scenes were removed. Thus, we provided more visually informative features to the visual part of the multi-modal architecture.

- Correlation between pre-learned audio and visual features extended the capabilities of decision making of our model.
- Because we partitioned scenes considering only visual information which caused audio continuity to be disturbed, once we combined audio and visual features it is more descriptive for video audio.
- Maximum Gap Thresholding (MGT) mapped prediction scores better than Keras’s k class prediction result metrics.
- MGT technique also accomplished robustness for the imbalanced classes.

	precision	recall	F1-score	support
Horror	0.82	0.83	0.82	811
Thriller	0.84	0.76	0.80	1289
Action	0.77	0.77	0.77	851
Comedy	0.81	0.73	0.77	915
Adventure	0.82	0.74	0.78	728
Sci-fi	0.76	0.77	0.77	599
Romance	0.80	0.64	0.71	658
Drama	0.81	0.89	0.85	1760
Mystery	0.84	0.89	0.87	971
Overall	Overall Precision	Overall Recall	Overall F1-score	Overall Support
micro avg	0.81	0.80	0.80	8582
macro avg	0.81	0.78	0.79	8582
weighted avg	0.81	0.80	0.80	8582
samples avg	0.80	0.78	0.77	8582

Table 5.5: Multimodal Prediction Scores

Observable from Table 5.4, our finest results were achieved with Multi-Modal with MGT applied metrics in terms of all metrics. Given the confusion matrices in Figures 5.3 and 5.1, we achieved outstanding results compared to the human labeling in the scene classification task.

In our research, we value a low rate of False Positives. Since there are many candidate scenes available in a movie that might represent a specific genre, we want to carefully select only the best scenes that characterize the given genre. For example, if we have 300 scenes given in a movie of 130 minutes, we want a comedy

genre trailer from the movie, we would only include 20 best-scoring probability results of belonging to the Comedy genre. Noticeably, from our results in the Table 5.3, we acquired low rates of False Positive (right upper corner) as we intended to achieve.

Author	Model	Dataset	Feature Size	#Classes	Accuracy	Precision	Recall	F1-Score
Shambharkar et al.[18]	3D CNN	LMTD-4	64 frames of size 224x224x3	4	0.821	N\A	N\A	N\A
Wehrmann et al.[17]	Convolution-Through-Time	LMTD-4	10 keyframes of size 224x224x3	9	0.742	0.663	N\A	N\A
Huang et al. [16]	SVMs	Apple Trailers, IMDB	277 Visual + Audio features extracted	7	0.916	N\A	N\A	N\A
Zhou et al. [14]	chi-square distance K- Nearest Search	IMDB, Apple Trailers, AllTrailers.net, Movie-List.com	Centrist Features 320x320	4	0.747	N\A	N\A	N\A
Human Perception Survey Given in Section 5.3	-	50 Trailers from Manually Cleaned Dataset 3.3.3	10 Frames of 224x224x3	9	0.831	0.48	0.44	0.45
Ours (Visual Only)	Time Distributed Convolutions and GRU's	Automatically Cleaned Dataset sec.3.3.2	8 Frames of size 160x160x3	9	0.8587	0.5393	0.1742	0.131
Ours (Visual Only)	Time Distributed Convolutions and GRU's	Manually Cleaned Dataset sec.3.3.3	8 Frames of size 160x160x3	9	0.865	0.6476	0.3717	0.4723
Ours (Audio Only)	Dense net	Automatically Cleaned Dataset sec. 3.3.1	345 Spectral Centroid (4 seconds of audio)	9	0.7460	0.6432	0.1019	0.1759
Ours (Multi-Modal) + Keras Metrics	Multi-Modal	Automatically Cleaned Dataset sec. 3.3.1	Audio Features and Visual Features	9	0.7904	0.6042	0.5329	0.5635
Ours (Multi-Modal) + MGT Applied Metrics	Multi-Modal	Automatically Cleaned Dataset sec. 3.3.1	Audio Features and Visual Features	9	0.80	0.81	0.80	0.79

Table 5.6: All Models Evaluation and Comparison

Comparing our visual model with multi-modal, we achieved lower accuracy. This is due to values being less sure of their prediction whilst multi-modal predictions are less close to the 0 values when the true values are 0. This is actually a better performance because our multi-modal takes risks in terms of probabilities of 0. In the visual model, predicting 0's with values closer to 0 have increased the accuracy; however, as can be seen from recall and precision scores, it decreased the fraction of truly predicted values of negatives and positives. As we discussed earlier in Section 5.1, accuracy is not a successful metric in determining multi-label classification performance.

Our confusion matrices in Figure 5.3 show that the most problematic genres are Drama and Romance. Our observations are that these genres are mostly confused with each other. As explained in Section 4.1.2.4, one main difference between romance and drama movies is that the romance movies always end with a happy ending. However, in movie trailers, the ending is not given to the audience to maintain excitement and curiosity. We believe that this confusion is due to the thin line between Romance and Drama which is not clued in trailers.

In terms of scenes of Horror, Mystery, Adventure, and Thriller genres, in which humans could also achieve better predictions than the other genres; we achieved exceptional scores in our system, as well. From our experiments with the binary models and only visual models, we believe that these genres are more accurately predictable with the addition of audio information.

Considering that only Wehrmann et al. accessed 9 class multi-label classification which is the only similar problem to our research we found, we acquired better macro and micro scores in terms of precision, recall, and F1 scores as can be seen in Table 5.6. Their micro precision for all of the genres with their best model is 0.651 and macro precision is 0.663 whereas ours are both 0.81. We believe that we achieved better results, because we enlarged our dataset with the scene extraction method. Moreover, Wehrmann et al. did not use audio features in their research which we believe are highly influential on trailer genre classification.



Figure 5.3: Multi-Modal Prediction Results Confusion Matrices

Chapter 6

Conclusion

Trailer making process is costly and time-consuming. There have been a few attempts to automatize parts of the trailer creation process. Nevertheless, the process of trailer making relies on the creativity of producers, and fully-automatically created trailers could not achieve the attractiveness of human creations. Moreover, the movie industry now relies on at-home streaming services which expose advertisements privately. Although there are many personalized advertisements provided on many platforms now, genre interest group-based trailers have not been considered in the literature. It is believed that the creation of trailers for the same movie for different genre lovers can improve the experience of the investors and audience.

In conclusion, we accomplished the task of multi-modal multi-label classification on video-based data. In order to build our visual system, we experimented on hyper-parameters, the number of layers, units of layers, and transfer learning techniques with binary classification tasks. Later, we combined this parameter knowledge on several different multi-label classification architectures. For the audio part of the system, we extracted Spectral Centroid features. To combine audio and visual features we built dense architecture on top of learned weights from these architectures. For the genre classification task for automated trailer

classification with respect to genre, we valued F1-score and precision. We presented a new methodology to translate prediction results to the one-hot encoding prediction results, namely Maximum Gap Thresholding. Our final system achieved improved results compared to the literature.

6.1 Future Work

- The current scene extractor we used required many cleaning steps and was not suitable for trailer scenes. Thus, the scene extractor by PySceneDetect can be modified or a new scene extractor can be developed for the short transitions of the trailers. Since this part of the system involved signal processing rather than a computer vision task, we had to develop computer vision related solutions such as elimination model or manual scene cleaning interface.
- Continuing with manual scene cleaning given more time can improve the performance of our system further.
- HSV Color Space instead of RGB can improve the results of the visual part of our system.
- More sound Features such as Spectral Bandwidth, Spectral Contrast, Spectral Rolloff, Spectral Spread, etc. can be added and experimented with our audio part of the system.
- Windowing technique extended to the whole audio instead of middle sections and variable-length training with usage of Recurrent Neural Networks (RNN) or Long-Short Term Memory (LSTM) may improve the performance of learning in the audio part of the system.
- Movie scenes can be long to directly add to the trailers. Thus, a method such as anomaly detection can be used on extracted scenes to find the most exciting scenes.

- In our thesis, we stated the requirement of creative editorial post-production on the scenes we extracted from the movies. An automatized post-production process can be experimented with the direction given by trailer editors. This can be also applied to the sound and special effects. This would allow to fully automatize the trailer creation process.



Bibliography

- [1] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, pp. 19:1–19:19, 2015.
- [2] J. G. Navarro, “Watching movies in the theater vs. via a streaming service u.s. 2018-2020,” 2020.
- [3] O. Rehusevych and T. Firman, “movie2trailer: Unsupervised trailer generation using anomaly detection,” 03 2020.
- [4] S. Karray and L. Debernitz, “The effectiveness of movie trailer advertising,” *International Journal of Advertising*, vol. 36, no. 2, pp. 368–392, 2017.
- [5] A. Joshi and D. Hanssens, “Movie advertising and the stock market valuation of studios: A case of ”great expectations?”,” *Marketing Science*, vol. 28, pp. 239–250, 03 2009.
- [6] M. Malheiros, C. Jennett, S. Patel, S. Brostoff, and M. A. Sasse, “Too close for comfort: A study of the effectiveness and acceptability of rich-media personalized advertising,” (New York, NY, USA), Association for Computing Machinery, 2012.
- [7] S. Godbole, “I believe netflix needs social features—and users agree!,” <https://medium.com/>, 2019.
- [8] G. Mujtaba and E.-S. Ryu, “Client-driven personalized trailer framework using thumbnail containers,” *IEEE Access*, vol. PP, pp. 1–1, 03 2020.
- [9] J. R. Smith, D. Joshi, B. Huet, W. Hsu, and J. Cota, “Harnessing a.i. for augmenting creativity: Application to movie trailer creation,” in *Proceedings*

- of the 25th ACM International Conference on Multimedia, MM '17, (New York, NY, USA), p. 1799–1808, Association for Computing Machinery, 2017.
- [10] I. Türköz and H. A. Güvenir, “Detection of animated scenes among movie trailers,” *18th International Conference on Machine Learning and Data Mining*, pp. 179–190, 2022.
 - [11] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, “Learning multi-label scene classification,” *Pattern Recognit.*, vol. 37, pp. 1757–1771, 2004.
 - [12] B. A. Halim and T. Faiza, “Shot boundary detection: Fundamental concepts and survey,” in *CITSC*, 2019.
 - [13] D. Guru, M. Suhil, and P. Lolika, *A Novel Approach for Shot Boundary Detection in Videos*, vol. 213, pp. 209–220. 01 2013.
 - [14] H. Zhou, T. Hermans, A. Karandikar, and J. Rehg, “Movie genre classification via scene categorization,” pp. 747–750, 10 2010.
 - [15] G. Simões, J. Wehrmann, R. Barros, and D. Ruiz, “Movie genre classification with convolutional neural networks,” pp. 259–266, 07 2016.
 - [16] Y.-F. Huang and S.-H. Wang, “Movie genre classification using svm with audio and video features,” 12 2012.
 - [17] J. Wehrmann, “Movie genre classification: A multi-label approach based on convolutions through time,” *Applied Soft Computing*, vol. 61, 08 2017.
 - [18] P. G. Shambharkar, P. Thakur, S. Imadoddin, S. Chauhan, and M. N. Doja, “Genre classification of movie trailers using 3d convolutional neural networks,” in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 850–858, 2020.
 - [19] H. Farid and M. J. Bravo, “Perceptual discrimination of computer generated and photographic faces,” *Digital Investigation*, vol. 8, no. 3, pp. 226–235, 2012.

- [20] M. Roach, J. Mason, and M. Pawlewski, "Motion-based classification of cartoons," in *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, pp. 146–149, 2001.
- [21] T. Ianeva, A. de Vries, and H. Rohrig, "Detecting cartoons: a case study in automatic video-genre classification," in *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, vol. 1, pp. I–449, 2003.
- [22] R. Glasberg, K. Elazouzi, and T. Sikora, "Video-genre-classification: recognizing cartoons in real-time using visual-descriptors and a multilayer-percetpron," in *The 7th International Conference on Advanced Communication Technology, 2005, ICACT 2005.*, vol. 2, pp. 1121–1124, 2005.
- [23] T.-T. Ng and S.-F. Chang, "An online system for classifying computer graphics images from natural photographs," in *Security, Steganography, and Watermarking of Multimedia Contents VIII* (I. Delp, Edward J. and P. W. Wong, eds.), vol. 6072 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 397–405, Feb. 2006.
- [24] W. Chen, Y. Shi, and G. Xuan, "Identifying computer graphics using hsv color model and statistical moments of characteristic functions," pp. 1123 – 1126, 08 2007.
- [25] F. Chen, M. Lai, and Z. Ye, "Detecting cartoons: Automatic video genre classification," in *2010 International Conference on Management and Service Science*, pp. 1–4, 2010.
- [26] G. Sankar, H. V. Zhao, and Y.-H. Yang, "Feature based classification of computer graphics and real images," *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1513–1516, 2009.
- [27] B. Ionescu and P. Lambert, "Classification of animated video genre using color and temporal information," 2013.

- [28] M. Montagnuolo and A. Messina, “Automatic genre classification of tv programmes using gaussian mixture models and neural networks,” in *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*, pp. 99–103, 2007.
- [29] W. Quan, K. Wang, D.-M. Yan, and X. Zhang, “Distinguishing between natural and computer-generated images using convolutional neural networks,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2772–2787, 2018.
- [30] R. Zhang, W. Quan, L. Fan, L.-m. HU, and D. Yan, “Distinguishing computer-generated images from natural images using channel and pixel correlation,” *Journal of Computer Science and Technology*, vol. 35, 02 2020.
- [31] G. Irie, T. Satou, A. Kojima, T. Yamasaki, and K. Aizawa, “Automatic trailer generation,” pp. 839–842, 10 2010.
- [32] H. Xu, Y. Zhen, and H. Zha, “Trailer generation via a point process-based visual attractiveness model,” in *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15*, p. 2198–2204, AAAI Press, 2015.
- [33] S. W. Ji and D. Waterman, “Production technology and trends in movie content: An empirical study,” 01 2011.
- [34] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, dec 2015.
- [35] “Interlacing,” 02 2014. Archived from the original on 2014-02-22.
- [36] S. Arpa. Personal Interview, 02 2022.
- [37] L. Evan, “Decord library on github.” <https://github.com/dmlc/decord>, 2021.
- [38] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” 05 2019.
- [39] J. Hellerman, “Mastering the comedy genre in movies and tv [w/ examples].” nofilmschool.com, 2022.

- [40] J. Luebering, “Horror film.” [britannica.com](https://www.britannica.com), 2022.
- [41] “What is a romance novel? learn about the history and characteristics of romance novels.” [masterclass.com](https://www.masterclass.com), 11 2020.
- [42] “How to identify movie genres: Beginner’s guide to 13 film genres.” [masterclass.com](https://www.masterclass.com), 08 2022.
- [43] J. Kendrick, “Action cinema.” [oxfordbibliographies.com](https://www.oxfordbibliographies.com), 01 2015.
- [44] A. Rankin, “What are the characteristics of the thriller genre?.” [language-humanities.org](https://www.language-humanities.org), 07 2022.
- [45] J. W. Picone, “Signal modeling techniques in speech recognition,” *Proc. IEEE*, vol. 81, pp. 1215–1247, 1993.
- [46] T. Giannakopoulos and A. Pirkakis, *Introduction to Audio Analysis: A MATLAB Approach*. USA: Academic Press, Inc., 1st ed., 2014.
- [47] A. Picciano, “Blending with purpose: The multimodal model,” *JALN*, vol. 13, pp. 11–13, 04 2009.
- [48] G. Wu and J. Zhu, “Multi-label classification: do hamming loss and subset accuracy really conflict with each other?,” *CoRR*, vol. abs/2011.07805, 2020.
- [49] K. Nooney, “Deep dive into multi-label classification..! (with detailed case study).” towardsdatascience.com, 2018.
- [50] J. M. Moyano, “Multi-label classification dataset repository,” 2022.
- [51] J. Brownlee, “Multi-label classification with deep learning,” 08 2020.
- [52] D. M. W. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” *ArXiv*, vol. abs/2010.16061, 2020.
- [53] B. Shmueli, “Multi-class metrics made simple, part ii: the f1-score.” towardsdatascience.com, 2019.

- [54] M.-L. Zhang and Z.-H. Zhou, “A review on multi-label learning algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.
- [55] K. Leung, “Micro, macro weighted averages of f1 score, clearly explained.” [towardsdatascience.com](https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained/), 2022.
- [56] W. Gao and Z.-H. Zhou, “On the consistency of multi-label learning,” in *Proceedings of the 24th Annual Conference on Learning Theory* (S. M. Kakade and U. von Luxburg, eds.), vol. 19 of *Proceedings of Machine Learning Research*, (Budapest, Hungary), pp. 341–358, PMLR, 09–11 Jun 2011.
- [57] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *International Journal of Data Warehousing and Mining*, vol. 3, pp. 1–13, 09 2009.

Appendix A

Survey

What is your age group?

302 responses

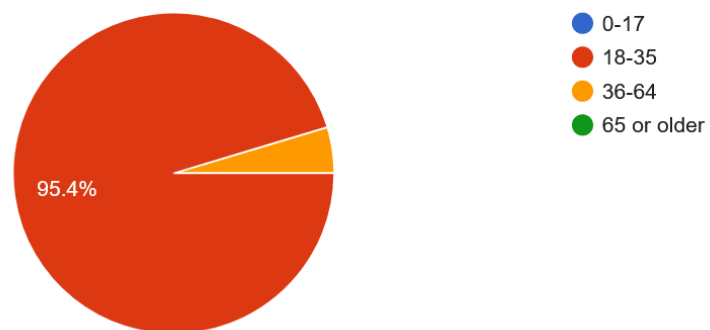


Figure A.1: Survey results: Answer chart for question "What is your age group?"

What is your gender?

302 responses

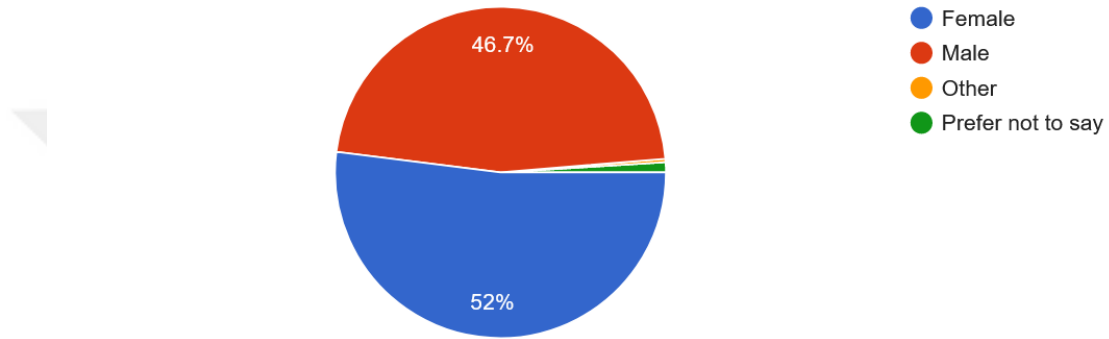


Figure A.2: Survey results: Answer chart for question "What is your gender?"

How frequently do you watch movies?

302 responses

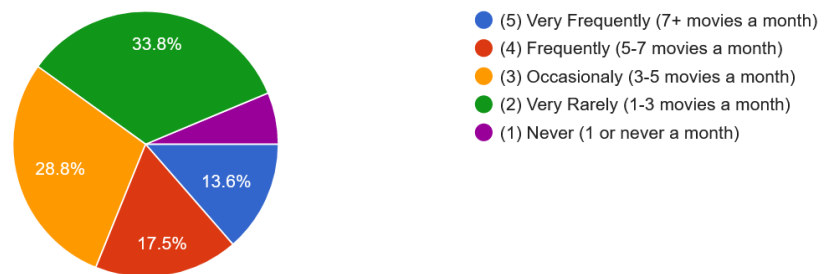


Figure A.3: Survey results: Answer chart for question "How frequently do you watch movies?"

Which of the following movie genres are your favorite?

302 responses

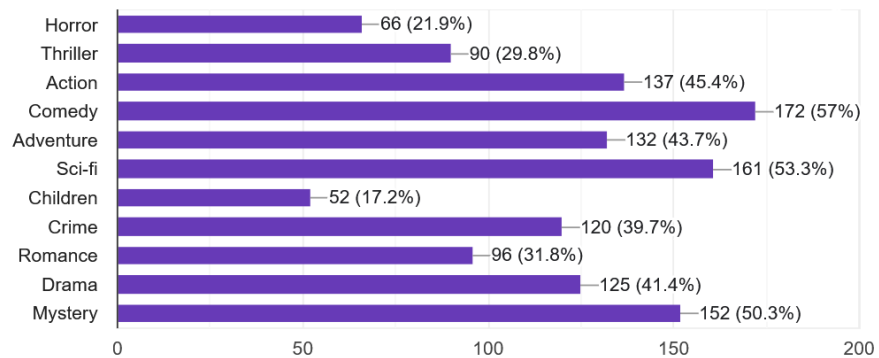


Figure A.4: Survey results: Answer chart for question "Which of the following movie genres are your favorite?"

Which of the following movie genres you NEVER prefer to watch?

302 responses

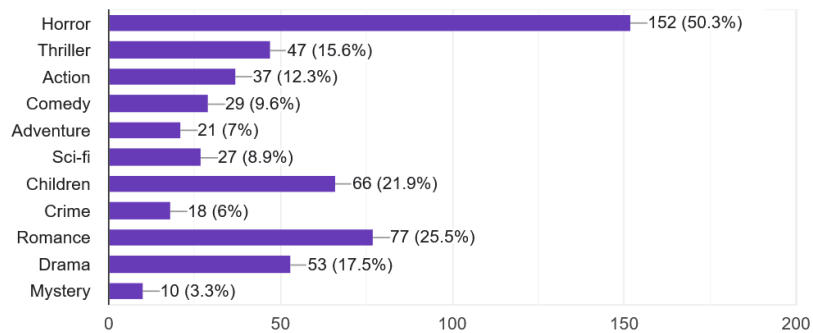


Figure A.5: Survey results: Answer chart for question "Which of the following movie genres you NEVER prefer to watch?"

Which genre does this part of movie might belong to? *

(* currently , 6, 7th scene is shown *)



- ☐ Horror
- ☐ Thriller
- ☐ Action
- ☐ Comedy
- ☐ Adventure
- ☐ Sci-fi
- ☐ Romance
- ☐ Drama
- ☐ Mystery
- ☐ Indecisive (Not sure)

Figure A.6: Example of a Question in Survey