Master Thesis

# "Seeing, Hearing, and Reading": A Deep Intermodal Fusion Learning Network for Content-Based Video Recommendations

Juliette J V P Maes

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Data Science for Decision Making
at the Department of Advanced Computing Sciences
of the Maastricht University

**Thesis Committee:**

Dr. T. D. Rienstra
Dr. F. Barile
Ward Pennink

Maastricht University
Faculty of Science and Engineering
Department of Advanced Computing Sciences

July 1st, 2025

**Abstract**

In an era of overwhelming video content, recommender systems are essential for discovery. However, the prevalent collaborative filtering approach suffers from the "cold-start" problem (the inability to recommend new items that lack user interaction history), failing to recommend new items. This thesis tackles this issue by developing and evaluating a purely content-based video recommendation system that understands an item's intrinsic properties from its visual, aural, and textual modalities.

This research follows a two-pronged methodology. First, a comprehensive baseline system was constructed using state-of-the-art feature extractors (DINOv2, VGGish, BERT) and a fixed cosine similarity metric. Rigorous ablation studies on this baseline revealed a clear modality hierarchy: textual features (especially genre) are dominant, audio features provide valuable secondary signals, and naively concatenated visual features introduce more noise than signal. Feature selection was proven to be paramount, reducing feature dimensionality by 98.1% while improving ranking performance, resulting in a final nDCG@10 of 0.0502.

Second, motivated by the baseline's inability to handle visual noise, an end-to-end (E2E) deep metric learning network was developed using a Siamese architecture. This model was trained to learn a bespoke similarity function directly from multimodal data. The results revealed a crucial "classification-ranking gap": while the E2E model excelled in its primary classification task (AUC 0.92), its performance on the global ranking task deteriorated with training.

The primary conclusion is that for a general-purpose ranking, a well-curated, feature-selected baseline outperforms a more complex E2E model. However, the end-to-end model's strength lies not in ranking but in its ability to function as a powerful thematic filter or candidate generator. This work presents a robust, reproducible pipeline for multimodal content-based recommendation and provides critical insights into the trade-offs between classification and ranking in learned similarity metrics, positioning such E2E models as a vital first stage in a hybrid recommendation system.

# Acknowledgments

This thesis is the culmination of a journey that would not have been possible without the support, guidance, and encouragement of many individuals. I would like to take this opportunity to express my deepest gratitude to all of them.

First and foremost, I wish to thank my primary supervisor, Dr. Tjitze Rienstra. His consistent guidance and unwavering trust in my abilities were instrumental in shaping this project. Thank you for always pushing me to refine my ideas and for helping me transform my work into a final product I can be proud of.

I would also like to extend my sincere thanks to my thesis committee. To Dr. Francesco Barile, thank you for your valuable time and for helping steer the project towards a strong research topic during the initial proposal phase. A special and profound thank you goes to Ward Pennink, who served brilliantly in the dual role of committee member and company supervisor. Our weekly discussions were the cornerstone of my progress. Ward, your ability to provide clear direction when my thoughts were scattered was invaluable, and your detailed feedback on my drafts significantly elevated the quality of this thesis. Your encouragement and mentorship were key to making this a success, for which I am truly grateful.

This research would not have been possible without the support of Media Distillery. I am grateful for the opportunity to conduct my thesis in such a dynamic and supportive industrial environment, as well as for the resources provided. The company trip offered a perfect, timely opportunity to restructure my thoughts, and I would also like to specifically thank Martin Prins for his insightful tips.

On a personal level, I am indebted to those who provided the emotional and intellectual backbone for this endeavour. My heartfelt thanks go to Julien Bol for his constant encouragement, for believing in me, and for providing invaluable feedback. To my partner, Lauraline Meyer, thank you for the late nights, the unwavering emotional support through the tough times, and your creative talent in transforming my rough drawings into clean, professional figures.

Finally, I would like to dedicate my deepest gratitude to my family for their unwavering support, love, and pride. I am especially thankful to my sister, Marie Maes, for her incredible dedication to reading my drafts and offering feedback on an almost daily basis. And to my father, Luc Pierre Maes, thank you for being my go-to technical expert, for lending me a second pair of eyes (because mine were clearly lying to me), and for providing tons of support and invaluable feedback on the final draft. This work is as much a reflection of your support as it is of my own efforts.

# Contents

**List of Abbreviations** **ix**

**1 Introduction** **1**
1.1 Background and Motivation . . . 1
1.1.1 Information Overload in the Digital Age . . . 1
1.1.2 The Dominant Paradigm and Its Critical Flaw . . . 1
1.1.3 The Alternative Solution: Content-Based Recommendation . . . 2
1.2 The New Challenge: The Multimodal Nature of Video Content . . . 2
1.3 Research Approaches and Contributions . . . 3
1.3.1 The Two-Pronged Methodological Framework . . . 3
1.4 Research Questions . . . 3
1.4.1 Primary Research Contributions . . . 4
1.5 Thesis Structure . . . 4

**2 Related Work** **6**
2.1 Recommender Systems and Content-Based Approaches . . . 6
2.2 Video Content Representation: A Multimodal Perspective . . . 7
2.2.1 Visual Modality . . . 7
2.2.2 Aural Modality . . . 7
2.2.3 Textual Modality . . . 8
2.2.4 Multimodal Integration and Fusion . . . 8
2.3 Deep Learning and Multimodal Learning in CBRS . . . 9
2.3.1 End-to-End Trainable Models . . . 9
2.4 Explainability in CBRS . . . 10
2.5 Feature Selection and Dimensionality Reduction . . . 10
2.6 Cold-Start and Beyond-Accuracy Challenges . . . 11
2.7 Siamese Neural Networks in Content-Based RS . . . 11
2.7.1 SNN Architectures and Training Strategies . . . 12
2.7.2 Integration in Recommender Systems . . . 12
2.7.3 Applications and Empirical results . . . 12
2.8 Evaluation Methodologies and Metrics . . . 13
2.9 Challenges and Future Directions . . . 13
2.10 Summary and Research Gaps . . . 14

**3 Methodology: A Baseline Multimodal Recommender** **16**
3.1 Dataset Preparation . . . 16
3.1.1 Data Sourcing and Integration . . . 17
3.1.2 Data Filtering and Final Corpus Composition . . . 17

3.1.3 Final Dataset Characteristics and Evaluation Data . . . 18
3.2 Feature Extraction . . . 18
3.2.1 Visual Feature Extraction . . . 19
3.2.2 Aural Feature Extraction . . . 27
3.2.3 Textual Feature Extraction . . . 28
3.3 Feature Fusion . . . 29
3.3.1 Preprocessing and Standardisation . . . 29
3.3.2 Early Fusion Architecture . . . 30
3.4 Feature Selection . . . 31
3.4.1 Data Preparation for Feature Selection . . . 31
3.4.2 Feature Selection Pipeline . . . 31
3.4.3 Combined Selection Strategy . . . 32
3.5 Similarity Computation and Recommendation . . . 32
3.5.1 Similarity Computation . . . 32
3.5.2 Candidate Filtering and Recommendation Generation . . . 33
3.6 Evaluation . . . 34
3.6.1 Experimental Setup . . . 34
3.6.2 Evaluation Metrics . . . 34
3.6.3 Ranking-Based Metrics . . . 35
3.6.4 Evaluation Protocol Summary . . . 35

**4 Methodology: End-to-End Similarity Learning 37**
4.1 Motivation for a Learned Approach . . . 37
4.2 Siamese Neural Networks: Exploratory Investigation . . . 38
4.2.1 Rationale and Objectives . . . 38
4.2.2 Implementation Focus Areas . . . 38
4.2.3 Framework Selection and Technical Decisions . . . 38
4.3 Training Data Generation for End-to-End Learning . . . 39
4.3.1 Binarisation of User Ratings and User Filtering . . . 39
4.3.2 Evolution of Pair Generation Strategies . . . 40
4.4 End-to-End Model Architecture . . . 42
4.4.1 Overall Design Philosophy: The Twin-Tower Mode . . . 42
4.4.2 Multimodal Feature Processing Modules . . . 43
4.4.3 Feature Fusion and Embedding Generation . . . 44
4.5 Experimental Design: A 2x2 Factorial Approach . . . 44
4.5.1 Pairwise Contrastive Learning Path . . . 44
4.5.2 Triplet Learning Path with Semi-Hard Negative Mining . . . 46
4.6 Implementation Considerations . . . 46
4.6.1 Computational Efficiency . . . 46
4.6.2 Scalability and Extensibility . . . 48
4.7 Evaluation Protocol for End-to-End Models . . . 48
4.7.1 Quantitative Ranking Evaluation . . . 48
4.7.2 Qualitative and Diagnostic Analysis . . . 48

**5 Implementation and Experimental Setup 50**
5.1 Technology Stack . . . 50
5.2 Hardware and Computational Environment . . . 50
5.3 Model Implementation . . . 51
5.3.1 Baseline Pipeline Implementation . . . 51

5.3.2 End-to-End Architecture and Training . . . 51
5.4 Per-Epoch Evaluation Pipeline . . . 54

**6 Results and Analysis** **55**
6.1 Performance of the Baseline Recommender System . . . 55
6.1.1 Baseline System Configuration . . . 55
6.1.2 Quantitative Results . . . 56
6.2 Ablation Study: Quantifying Modality Contributions . . . 56
6.2.1 Key Findings . . . 56
6.2.2 Follow-Up Investigation of Visual Features . . . 58
6.3 Feature Selection: Improving Efficiency and Performance . . . 59
6.3.1 Feature Reduction and Performance Impact . . . 59
6.3.2 Modality-Specific Importance . . . 60
6.3.3 Validating the Selection Strategy . . . 64
6.3.4 Performance Impact of Feature Selection . . . 65
6.3.5 Feature Selection Insights for End-to-End Model Design . . . 66
6.4 Domain-Specific Analysis: The Power of Context . . . 66
6.5 Performance Analysis of the End-to-End Models . . . 67
6.5.1 The Ranking Performance Anomaly . . . 67
6.5.2 Evaluating the Primary Training Objective . . . 70
6.5.3 Interpreting E2E Model Performance: Classification vs. Ranking . . . 74
6.6 Qualitative Analysis: A Case Study on Recommendation Content . . . 75
6.6.1 Case Study: Toy Story . . . 75
6.6.2 Case Study: Fight Club . . . 76
6.6.3 Overall Interpretation of the Case Study . . . 77
6.7 Summary of Research Progression and Key Findings . . . 78

**7 Discussion** **81**
7.1 RQ1: The Hierarchy of Modality Contribution . . . 81
7.2 RQ2: Learned vs. Traditional Similarity Metrics . . . 82
7.3 RQ3: Strengths and Limitations of a Purely Content-Based Approach . . . 83
7.4 Limitations of the Study . . . 84

**8 Conclusion** **85**
8.1 Future Work: An actionable Roadmap . . . 86

**A Technical Specifications and Environment** **88**
A.1 Hardware and Cloud Platform . . . 88
A.2 Software and Libraries . . . 88
A.2.1 Core Environment . . . 88
A.2.2 Machine Learning and Data Processing . . . 89
A.2.3 Multimedia Feature Extraction . . . 89

**B Data Exploration and Validation** **90**
B.1 User Rating Analysis . . . 90
B.2 Dataset Bias and Representativeness Analysis . . . 91
B.2.1 Genre and Language Distribution . . . 91
B.3 Dataset Representativeness and Bias Analysis . . . 91
B.3.1 Data Quality Issues Breakdown . . . 92
B.3.2 Statistical Validation Results . . . 93

B.3.3 Conclusion of Validation . . . . . 94

**C Additional Experimental Results** **95**
C.1 Iterative Enhancement: Feature Engineering and Fusion Methods . . . . . 95
C.1.1 Alternative Fusion and Enhancement Methods . . . . . 95
C.1.2 Dimensionality Reduction Techniques . . . . . 96

**D End-to-End Model Implementation Details** **97**
D.1 Modality-Specific Processing Modules . . . . . 97
D.1.1 Visual Processing Module . . . . . 97
D.1.2 Audio Processing Module . . . . . 97
D.1.3 Textual Processing Module . . . . . 98
D.2 Fusion Network Architectures . . . . . 98
D.2.1 HybridFusionNetwork (Used for Pair-Cosine) . . . . . 98
D.2.2 SimpleFusionNetwork (Used for Pair-Euclidean & Triplet-Cosine) . . . . . 98
D.2.3 UltraSimpleFusionNetwork (Used for Triplet-Euclidean) . . . . . 99
D.3 Model Architecture Summary . . . . . 99

**E Core End-to-End Model Implementation** **100**
E.1 Siamese Network Architectures . . . . . 100
E.1.1 Pair-Based Model . . . . . 100
E.1.2 Triplet Based Model . . . . . 102
E.2 Multimodal Fusion Network . . . . . 103
E.2.1 SimpleFusionNetwork . . . . . 104
E.2.2 UltraSimpleFusionNetwork . . . . . 105
E.2.3 HybridFusionNetwork . . . . . 106
E.3 Custom Loss Functions . . . . . 108
E.3.1 Contrastive Loss . . . . . 108
E.3.2 Triplet Loss . . . . . 109

**F Hyperparameters Configuration** **111**
F.1 Learning Rate Strategy . . . . . 111
F.2 Optimisation and Regularisation . . . . . 111
F.3 Loss Configuration . . . . . 111
F.4 Training Dynamics . . . . . 112

# List of Figures


3.1 Data Integration Pipeline, Combining MovieLens, OMDb, and YouTube Sources. 17
3.2 Examples of Problematic Content Frames . . . . . 20
3.3 Segment-Based Contamination Challenge . . . . . 21
3.4 Temporal Representation Bias Challenge . . . . . 22
3.5 Frame Filtering Problems . . . . . 23
3.6 The Two-Pass Sampling Solution . . . . . 24
3.7 Two-Pass Sampling Strategy Comparison . . . . . 25
3.8 Methodology Flowchart . . . . . 36

4.1 Twin Networks Architecture . . . . . 38
4.2 Analysis of Rating Threshold for Creating Training Pairs. . . . . 40
4.3 Twin-Tower Architecture with Multimodal Processing . . . . . 42
4.4 Experimental Design Matrix . . . . . 45
4.5 Semi-Hard Negative Mining in Embedding Space . . . . . 47

6.1 Percentage Change in Key Performance Metrics After Removing Each Modality. 57
6.2 Performance of Precision@K (left) and Hit Rate@K (right) Across the Ablation Study Configurations. . . . . 58
6.3 Impact of Feature Selection on Dimensionality and Performance. . . . . 59
6.4 Comparison of Feature Importance Scores (MI vs. RFFI) for Textual Features. . 61
6.5 Comparison of Feature Importance Scores (MI vs. RFFI) for Audio Features. . . 62
6.6 Comparison of Feature Importance Scores (MI vs. RFFI) for Visual Features. . . 63
6.7 Cumulative Mutual Information for Visual Features . . . . . 64
6.8 Comparison of Normalised Feature Importance Scores from Mutual Information and Random Forest for the Textual Modality. . . . . 65
6.9 Performance Comparison Between the Full Dataset and a Curated Domain-Specific Subset. . . . . 67
6.10 Comparison of Top-K Ranking Performance for the Four E2E Model Variants Across Training Epochs. . . . . 69
6.11 Diagnostic Analysis of the E2E-Pair-Cosine Model. . . . . 71
6.12 Diagnostic Analysis of the E2E-Pair-Euclidean Model . . . . . 72
6.13 Diagnostic Analysis of the E2E-Triplet-Cosine Model. . . . . 73
6.14 Diagnostic Analysis of the E2E-Triplet-Euclidean Model. . . . . 74
6.15 Evolution of Key Performance Metrics Throughout the Research Process. . . . . 79

B.1 Distribution of Raw User Ratings (0.5-5.0) in the MovieLens 20M Dataset for the Movies Included in the Final Corpus. . . . . 90

B.2 Correlation Between MovieLens Ratings (Scaled to 1-10) and Corresponding IMDb Ratings. . . . . 91
B.3 Distribution of the Standard Deviation of Ratings for Each User. . . . . 92
B.4 Genre Distribution in the Final Multimodal Dataset. . . . . 93
B.5 Primary Language Distribution in the Final Dataset. . . . . 94

# List of Tables


3.1 Dataset Composition Breakdown Showing Retention Rates at Each Processing Stage. . . . . 17
3.2 Summary Statistics of the Final Multimodal Dataset . . . . . 18

5.1 Architectural and Normalisation Strategy for End-to-End Models . . . . . 53

6.1 Baseline System Performance Metrics . . . . . 56
6.2 Ablation Study: Impact of Modality Removal on Performance . . . . . 57
6.3 Summary of Feature Selection by Modality . . . . . 60
6.4 Feature Selection Impact on Performance . . . . . 65
6.5 Performance on Domain-Specific Dataset (400 movies) . . . . . 66
6.6 Qualitative Comparison of Top Recommendations for Toy Story (1995) . . . . . 76
6.7 Qualitative Comparison of Top Recommendations for Fight Club (1999) . . . . . 77
6.8 Comprehensive Performance Comparison of Key Models . . . . . 78

B.1 Breakdown of Data Quality Issues and Their Impact on Dataset Composition . . 94

C.1 Enhanced Fusion Methods Performance . . . . . 95
C.2 Impact of Dimensionality Reduction on Performance . . . . . 96

D.1 Trainable Parameter Summary of the 'E2E-Triplet-Cosine' Model Architecture . 99

F.1 Final Hyperparameters for End-to-End Model Variants . . . . . 113

# List of Abbreviations

| | |
|---|---|
| **AUC** | Area Under the Curve |
| **AWS** | Amazon Web Services |
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **CBRS** | Content-Based Recommender System |
| **CF** | Collaborative Filtering |
| **CFS** | Correlation-Based Feature Selection |
| **CNN** | Convolutional Neural Network |
| **CTR** | Click-Through Rate |
| **DINOv2** | Data-efficient Image Network v2 |
| **E2E** | End-to-End |
| **FiLM** | Feature-wise Linear Modulation |
| **FS** | Feature Selection |
| **HPSS** | Harmonic-Percussive Source Separation |
| **IG** | Information Gain |
| **IMDb** | Internet Movie Database |
| **k-NN** | k-Nearest Neighbours |
| **MI** | Mutual Information |
| **MLP** | Multi-Layer Perceptron |
| **MRR** | Mean Reciprocal Rank |
| **MFCC** | Mel-Frequency Cepstral Coefficients |
| **nDCG** | Normalized Discounted Cumulative Gain |
| **NLTK** | Natural Language Toolkit |
| **NN** | Neural Network |

| | |
|---|---|
| **OMDb** | Open Movie Database |
| **PCA** | Principal Component Analysis |
| **PC** | Pair-Cosine |
| **PE** | Pair-Euclidean |
| **RFFI** | Random Forest Feature Importance |
| **RNN** | Recurrent Neural Networks |
| **ROC** | Receiver Operating Characteristic |
| **RS** | Recommender System |
| **SNN** | Siamese Neural Network |
| **SSIM** | Structural Similarity Index |
| **TC** | Triplet-Cosine |
| **TE** | Triplet-Euclidean |
| **TF-IDF** | Term Frequency-Inverse Document Frequency |
| **VGGish** | Visual Geometry Group-inspired audio classification model |
| **VSM** | Vector Space Model |
| **Wav2Vec2** | Waveform-to-Vector v2 |
| **XAI** | eXplainable Artificial Intelligence |
| **YOLO** | You Only Look Once |

# Chapter 1

# Introduction

## 1.1 Background and Motivation

### 1.1.1 Information Overload in the Digital Age

Every minute, sixty hours of new video content are uploaded to YouTube [13]. By the time you finish reading this sentence, thousands of new films, series, and videos will have joined the billions already online. We live in an age where the problem is no longer finding something to watch; it is choosing what to watch from an endless sea of options.

This digital revolution has created a paradox. More choice should mean greater satisfaction, yet audiences often feel overwhelmed and struggle to discover content they genuinely enjoy. What is the solution to this problem? Intelligent recommendation systems.

These digital curators have become so essential that two-thirds of Netflix's streaming comes from algorithmic suggestions [15]. Without them, navigating today's vast content libraries would be nearly impossible.

Recommendation technologies now power entertainment, education, and news platforms worldwide [3], quietly shaping what millions of people watch every day. They have become the invisible hand guiding our digital media consumption.

### 1.1.2 The Dominant Paradigm and Its Critical Flaw

The prevailing approach to content recommendation has been collaborative filtering (CF), a technique that leverages user-item interaction patterns to generate suggestions. Operating on the principle that "people who liked X also liked Y", collaborative filtering has demonstrated remarkable success in scenarios with abundant user data and established interaction histories. This approach capitalises on the wisdom of crowds, identifying latent preference patterns across user populations to make personalised recommendations.

However, CF suffers from a fundamental limitation: the cold-start problem. This challenge manifests in two critical scenarios. First, new items entering the system, regardless of their intrinsic quality or relevance, remain invisible to recommendation algorithms until sufficient user interactions accumulate. Second, new users with sparse or non-existent interaction histories receive poor-quality recommendations, often leading to suboptimal initial platform experiences. This creates what might be termed a "visibility barrier" for new and niche content, potentially stifling discovery and reducing platform diversity while disadvantaging both emerging content creators and users seeking novel experiences.

### 1.1.3 The Alternative Solution: Content-Based Recommendation

Content-based recommender systems (CBRS) offer a compelling alternative that directly addresses the cold-start problem. Rather than relying on user behaviour patterns, content-based approaches analyse the intrinsic features of items themselves, operating on the principle that "if you liked this item, you will likely appreciate other items with similar characteristics". For traditional media, such as films, these characteristics have historically included metadata elements, including genre classifications, directorial credits, and cast information.

This approach provides immediate utility for newly introduced content, as recommendations can be generated based solely on content features without requiring user interaction data. Furthermore, content-based systems offer enhanced transparency and explainability as recommendations can be directly attributed to specific content characteristics, fostering user trust and enabling more informed content discovery decisions. However, a significant challenge in content-based research is the scarcity of large-scale, publicly available datasets with explicit, human-annotated similarity labels. Consequently, researchers must often rely on 'weak supervisory signals' derived from user interaction data, such as co-ratings, to infer content similarity. As demonstrated in this work, such signals contain inherent noise, making the development of robust data processing and learning strategies paramount.

## 1.2 The New Challenge: The Multimodal Nature of Video Content

While content-based recommendation presents an elegant solution to the cold-start problem, applying it to video introduces a formidable set of challenges. The true "content" of a film is not a simple set of keywords; it is a complex synthesis of sensory and narrative information. This thesis argues that to truly recommend video, a system must grapple with unique difficulties posed by three distinct modalities:

**Seeing: The Challenge of Visual Noise and Ambiguity.** The visual stream offers a fire-hose of data, from cinematography and colour palettes to scene dynamics. The primary challenge is not just extracting these features but shifting through immense perceptual noise (studio logos, credit sequences, and generic filler frames). More profoundly, how can a system learn the difference between a visually *bright* film and a thematically *light* one? This requires moving beyond simple pixel analysis to a semantic understanding of visual language.

**Hearing: The Challenge of Differentiating Signals.** The audio modality is a heterogeneous mix of dialogue, musical tone, and ambient sound. A system must learn to differentiate these signals to recognise that the frantic score of an action sequence and the quiet dialogue of a drama requires entirely different interpretations. Failing to do so risks conflating films with similar energy but opposite intent.

**Reading: The Challenge of Transcending a Single Label** The textual modality, which includes plot summaries and metadata such as genre, is often the most powerful predictor. However, this strength is also a weakness. Relying too heavily on a simple "genre" tag creates a coarse system that cannot capture intra-genre nuance. The challenge is to force the model to read deeper, to understand the narrative themes and stylistic choices that make one "thriller" fundamentally different from another.

Ultimately, the greater challenge lies in fusion. How can a system intelligently weigh these disparate signals? How should it handle a conflict between a positive textual cue (shared genre) and a negative visual one (very different cinematography)? This requires learning the very geometry of similarity in a high-dimensional space, a central theme that this thesis will demonstrate is critical for success.

This thesis argues that to capture a video's essence for recommendation purposes truly, a system must effectively understand and integrate all three modalities, seeing, hearing, and reading, to create a comprehensive representation of content that transcends only metadata-based approaches.

## 1.3 Research Approaches and Contributions

### 1.3.1 The Two-Pronged Methodological Framework

This research employs a two-pass approach designed to both establish empirical baselines and advance the state-of-the-art in multimodal content-based recommendation.

**Establishing a Comprehensive Baseline:** The investigation begins by constructing a traditional content-based recommender system that serves as both a performance benchmark and an analytical tool. This baseline system utilises state-of-the-art feature extractors, including DINOv2 for visual analysis, VGGish for audio processing, and BERT for textual understanding, which are combined through simple concatenation and evaluated using cosine similarity metrics. This approach enables a controlled analysis of each modality's individual contribution while providing a rigorous foundation for comparison with more sophisticated approaches.

**Developing an Advanced End-to-End Solution:** Motivated by the limitations of the baseline in handling high-dimensional, multimodal feature spaces, the research subsequently develops an advanced deep metric learning architecture using Siamese Neural Networks (SNN). This approach, which can be considered end-to-end at the feature level, learns optimal similarity functions directly from data, thereby overcoming the noise and complexity inherent in high-dimensional multimodal features that fixed similarity metrics cannot adequately address. Crucially, it is designed to test the hypothesis that a learned model can extract meaningful signals from the very visual features that were found to be problematic for the simple baseline approach.

## 1.4 Research Questions

To guide this investigation systematically, the following research questions are posed:

1. **Which multimodal content features (visual, audio, textual) contribute most to determining program similarity in a content-based video recommendation system?**
2. **How do learned representations from a Siamese Neural Network compare to traditional, feature-engineered representations when used for similarity-based ranking?**

3. **Based on the literature and the challenges addressed in this work, what are the primary strengths and limitations of a purely content-based approach for video recommendation?**

The questions collectively address the fundamental challenges of multimodal feature integration, similarity metric optimisation, and system-level performance evaluation that define the current frontier in content-based video recommendation research.

### 1.4.1 Primary Research Contributions

To address these research questions, this thesis makes four significant contributions to the field of content-based video recommendation:

1. **Systematic Multimodal Dataset Creation:** The research presents a comprehensive and reproducible pipeline for creating large-scale multimodal datasets, demonstrated through the construction of a 13,920-movie corpus that enriches the established MovieLens 20M dataset [9] with visual, audio, and textual features extracted from movie trailers and metadata sources.

2. **Empirical Modality Analysis:** Through rigorous ablation studies, this work quantifies the individual and combined contributions of all three modalities to recommendation performance. The analysis revealed a clear hierarchy of modality importance.

3. **End-to-End Similarity learning Architecture:** The thesis designs, implements, and evaluates Siamese Neural Network architectures for multimodal similarity learning, providing direct empirical comparison against traditional fixed-metrics approaches and demonstrating the potential for learned similarity functions in high-dimensional feature spaces.

4. **Feature Selection Optimisation:** The research conducts comprehensive feature selection analysis, achieving significant feature reduction while maintaining performance.

5. **Analysis of the Classification-Ranking Gap:** The thesis provides a crucial diagnostic analysis that reveals the divergence between optimising for a classification task and a ranking task in Siamese Networks, reframing their role as powerful candidate generators rather than direct rankers.

## 1.5 Thesis Structure

This thesis is organised into eight chapters. Chapter 2 reviews the existing literature on recommender systems, focusing on content-based approaches, multimodal feature extraction, and similarity learning methodologies. Chapter 3 establishes the baseline methodology through multimodal dataset construction, feature extraction techniques, and traditional similarity computation. Chapter 4 introduces the end-to-end SNN architecture, providing theoretical justification, implementation details, and training procedures for learned similarity metrics in high-dimensional multimodal spaces. Chapter 5 outlines the comprehensive implementation framework, including system architecture decisions, computational requirements, and reproducibility considerations. Chapter 6 presents empirical results, including baseline performance analysis, ablation studies that quantify the contributions of each modality, feature selection outcomes, and a comparative evaluation of traditional versus learned similarity approaches. Chapter 7 discusses the broader implications of the findings, contextualising results within the existing literature while identifying limitations and future research directions. Chapter 8 concludes with a synthesis of contributions,

practical implications for content-based recommendation systems, and recommendations for future research in multimodal content analysis.

# Chapter 2

# Related Work

This chapter reviews the foundational and state-of-the-art literature that underpins this research. It begins by exploring the evolution of recommender systems, with a specific emphasis on content-based approaches. It then examines multimodal feature extraction and advanced neural architectures for similarity learning. The review progresses from fundamental concepts to cutting-edge techniques, highlighting the challenges and solutions in modern movie and video recommendation systems.

## 2.1 Recommender Systems and Content-Based Approaches

Recommender systems (RS) serve as essential tools for filtering large information sceneries to guide users to their preferential content [14], addressing the challenge of overwhelming today's digital landscape [22]. These systems are supporting decision-making scenarios by suggesting interesting items to users [15].

Among RS paradigms, including collaborative filtering and content-based recommender systems, the latter recommend items by analysing the similarity with other items previously liked by the user [14]. Content-based filtering, also known as content-based RS, aims to recommend items similar to those a user has previously liked [4, 10, 15]. This approach analyses the characteristics and features. It then matches these with a user's profile, which captures their preferences and interests [4, 10]. In other words, the underlying principle is that items with similar attributes or characteristics will be given a similar rating [26].

Collaborative filtering relies on user-item interaction histories and matches users with others who share similar tastes [18]. In contrast, CBRS relies exclusively on the item's attributes to make a recommendation. This fundamental difference makes CBRS especially advantageous and effective for tackling the cold-start problem [2, 13]. The cold-start problem represents a fundamental challenge when CF is used, arising when recommending new items or new users, as there is insufficient historical interaction data available to build profiles or identify similar users or items [13, 18, 22]. In music streaming, CF requires artists to already have a significant number of followers for a fair chance of being recommended. This creates an imbalance where only a small percentage of artists receive the vast majority of streams [18].

On the other hand, content-based methods are particularly well-suited to face the new item cold-start problem as they rely on item characteristics and features rather than user behaviour [2, 13]. Indeed, when no user interaction data exists, recommendations can still be generated based exclusively on item attributes. Some studies emphasise the role of intrinsic features, especially visual style and deep features, as a mitigating solution for the obstacle [13]. Using

implicit or intrinsic features offers a promising approach to this issue. These features include low-level or stylistic visual features when explicit metadata is scarce or unavailable for new content [13].

## 2.2 Video Content Representation: A Multimodal Perspective

Videos, and more specifically, movies, are inherently multimodal. They combine temporal and spatial information across multiple sensory streams [15, 24]. Video content encompasses aural (audio information), visual (video frames), and textual (description and metadata) modalities [2,15,19]. Therefore, effective CBRS must extract and fuse features from all modalities to capture the rich semantic content of videos.

### 2.2.1 Visual Modality

The analysis of the visual modality is crucial for video recommendation [19]. Traditional methods extract low-level stylistic features (e.g., colour, brightness, edges, or motion) [5, 13]. Early approaches may have relied on metadata only, but advances in computer vision and video analysis have allowed for the extraction of features directly from visual frames [19]. These visual features range from low-level stylistic features like lighting and camera motion [5, 13] to high-level visual appearance and motion information captured by deep learning models [2].

Stylistic features, grounded in Applied Media Theory, are crucial elements. They convey emotions and feelings. This can lead to the evaluation and recommendation of new artistic work based on its intrinsic features [5, 13]. Computer vision techniques are then employed to extract these visual-semantic attributes from images and videos [3].

State-of-the-art methods now utilise deep Convolutional Neural Networks (CNNs) or vision transformers to achieve rich semantic embeddings from video frames [2,3]. DINOv2[1] represents a significant advancement in learning robust, general-purpose visual features, using self-supervised pre-training on large, curated image dataset [16]. DINOv2 utilises Vision Transformers (ViT) and related self-supervised objectives to prepare effective visual embeddings [16]. These embeddings work "out-of-the-box" for various computer vision tasks without fine-tuning [16]. Those features have shown strong and valid performance on tasks that include image classification, object recognition, semantic segmentation, and depth estimation [16]. The ability of DINOv2 features to capture semantic meaning and structural information [16] makes them crucially relevant for the representation of the visual content of a movie in a content-based recommendation system.

### 2.2.2 Aural Modality

The aural modality provides crucial context that vision alone cannot capture, including mood, scene dynamics, and narrative tone. Traditional audio features, such as Mel-frequency Cepstral Coefficients (MFCCs), provide a foundational representation of timbral characteristics. However, modern approaches increasingly rely on deep learning models pre-trained on vast audio datasets. Models like VGGish[2], trained on YouTube-8M, are adept at extracting high-level semantic audio embeddings capable of distinguishing between speech, music, and ambient sounds. When

[1] https://dinov2.metademolab.com

[2] https://docs.pytorch.org/audio/master/generated/torchaudio.prototype.pipelines.VGGISH.html

combined with vision, these rich audio features contribute to a more comprehensive narrative understanding [2].

### 2.2.3 Textual Modality

Historically, textual descriptors, like plot summaries or metadata (e.g., genre, actor), were the most commonly used textual features in RS [2, 5, 19]. Specifically for movie recommendations, CBRS can utilise features extracted from item descriptions or metadata, such as actors and directors, although this often requires domain knowledge [4]. Traditional methods that can represent this kind of text often involve techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) [3, 14] or bag-of-words methods [5, 14].

For most text-based content, standard approaches involve selecting single words from documents as descriptors [26]. TF-IDF techniques assign weights to these words, reflecting their importance within documents [4, 15, 26]. TF-IDF style measures were first observed from empirical studies for combining term weighting factors, with theoretical arguments for IDF linking it to probabilistic models of information retrieval [21]. IDF is presented as a simplified form of probabilistic relevance weighting, applicable when detailed relevance information is limited [21]. These terms can then be used to represent documents as vectors in a multidimensional space, often using the Vector Space Model (VSM) [4,10,26]. Cosine similarity is widely used to measure and determine the closeness between documents represented as vectors in VSM [4].

Notwithstanding, these approaches fall short of capturing semantic subtlety. There has been a genuine need to move beyond simple syntactic evidence from keywords. Modern approaches require more complex "*semantic analysis*" of contextual content in order to gain a deeper understanding of user interests [14]. This has led to the inclusion of domain semantics through semantic lexicons or ontologies [3, 14, 15]. Recent studies have integrated semantic-aware text representations by incorporating world knowledge from sources like Wikipedia and ontologies [3, 14, 19].

More modern research is exploring the use of contextual word representations due to their enhanced ability to model the precise meanings of words and sentences [19]. Additionally, modern research is also exploring the use of large transformer-based models to generate contextual word representations. Architectures like BERT [7] have shown an enhanced ability to model the precise meanings of words and sentences, representing a significant improvement over traditional methods. BERT's pre-training procedure utilises large corpora, including BooksCorpus and English Wikipedia. The procedure focuses on extracting long, contiguous sequences from document-level text [7].

### 2.2.4 Multimodal Integration and Fusion

Research in video recommender systems often focuses on methods for feature extraction and how to integrate multimedia data [15]. Manually extracted multimodal features, such as low-level visual features from trailers or automatically extracted high-level semantic concepts (e.g., genres and actors), can contribute to providing accurate recommendations [15]. In most multimodal systems, the feature extraction pipeline typically consists of segmentation, preprocessing, feature encoding, dimensionality reduction, and feature fusion [15, 19].

One of the most important steps is fusion, which can either be either early, late or hybrid [2]. Early fusion strategies concatenate features before processing. Late fusion processes each modality independently before merging decisions. Hybrid fusion combines both, allowing the benefits of each strategy. The combination of features through a multimodal approach has shown potential for improving accuracy [15]. However, a simple feature combination does not

guarantee improved accuracy. Lack of correlation or semantic dissimilarity between different feature types can sometimes reduce quality [15].

Both manually created and automatically extracted features from visual, aural, and textual modalities can be combined. These combined attributes are then mapped to textual descriptions for compatibility with textual recommendation methods [15].

The foundation of multimodal feature fusion has paved the way for more sophisticated approaches. Recent research in deep learning have revolutionised how these multimodal features are extracted, learned and integrated in CBRS.

## 2.3 Deep Learning and Multimodal Learning in CBRS

Recently, deep learning has emerged in various technologies, and in the past few years, it has revolutionised feature extraction for multimedia RS. Recent studies have increasingly leveraged deep neural networks for feature extraction and representation learning from video content [2, 17]. Convolutional neural networks are widely used both for visual and audio inputs. They process different aspects like frame-based features or Mel-Spectrograms [17]. Multimodal learning architectures integrate these features to model semantic alignment across modalities [2, 3].

Beyond independent, single-modality feature extraction, multimodal learning, which explores visual, audio, and text features, is a significant area of research [2, 3, 15, 17, 19]. A noteworthy approach is multimodal topic learning. It generates compact, topic-based features from images, tags, and titles, thereby effectively reducing dimensionality while preserving semantics [17]. This algorithm generates low-dimensional semantic topic features that can be used rather than high-dimensional visual features like CNN features. This approach drastically improves computational efficiency and therefore enhances various RS modules, including top-k candidate generation and re-ranking, by facilitating preference scope determination [17].

### 2.3.1 End-to-End Trainable Models

Traditional recommender systems often rely heavily on ID features (i.e., unique identifiers for users and items), which lack inherent semantic meaning, thus facing challenges like cold-start problems and limited generalisation [6]. Modelling pre-extracted content features can help but may be suboptimal. It occurs due to discrepancies between training tasks and model parameters [6]. End-to-end training is presented as a promising solution to these issues [6].

Deng et al. [6] propose EM3 (End-to-end training of Multimodal Model and ranking Model), an industrial multimodal recommendation framework. This system uses multimodal information to enable personalised ranking tasks. It directly trains the core modules within the multimodal model architecture [6]. The goal is to obtain more task-oriented content features while minimising resource consumption. EM3 includes a Fusion-Q-Former module to fuse different modalities and generate fixed-length, robust multimodal embeddings [6]. It also incorporates techniques like Low-Rank Adaptation (LoRA) for user sequential modelling to alleviate resource consumption [6]. Experiments on industrial systems and public datasets reportedly show significant improvements with EM3 [6]. This research opens up avenues for investigating the use of end-to-end trainable models in multimedia content-based recommender systems.

While these advanced deep-learning approaches have significantly improved recommendation performance, they introduce a critical challenge: reduced interpretability. Undoubtedly, CBRS systems will become more sophisticated through multimodal fusion and end-to-end training. Therefore, understanding why specific recommendations are made becomes increasingly complex. This leads to concerns about user trust and system transparency.

This trade-off between performance and transparency leads directly to the growing field of explainability in recommender systems.

## 2.4 Explainability in CBRS

The transition from traditional content-based methods to sophisticated deep learning architectures creates a fundamental trade-off between performance and interpretability. While models like DINOv2 for visual features and BERT for textual content have significantly improved the representative power of a CBRS, they often operate as "black boxes", making their internal decision-making opaque [5]. This lack of transparency presents challenges for establishing user trust, debugging systems, and identifying potential biases.

In the context of multimodal movie recommender, this challenge is amplified. It is not enough to know that the system works; one must also understand how it works. It is crucial to know *why* it works. Are recommendations driven by cinematographic style, acoustic patterns, or narrative themes? Answering this question is key to diagnosing model weaknesses and making targeted improvements. For example, if a model is found to be ignoring a valuable modality, its architecture can be redesigned to incorporate it.

The field of eXplainable AI (XAI) has produced powerful model-agnostic techniques to address this opacity. Methods like LIME (Local Interpretable Model-gnostic Explanations) and SHAP (SHapley Additive exPlanations) excel at providing local explanations, attributing an individual prediction to the contribution of each input feature [20, 25]. Such methods could, for example, explain that a specific action movie was recommended due to shared audio-visual suspense cues with a previously liked film.

The implementation of local, per-recommendation XAI techniques is beyond the scope of this thesis. However, the underlying principle of model interrogation remains critical. An alternative and complementary approach is to conduct a global feature-level analysis to understand which content characteristics a model finds most predictive overall. Investigating the relative importance of multimodal features (from high-level genre tags to low-level visual textures) is thus a crucial step in diagnosing and improving complex "black-box" recommender systems. This analytical perspective is central to the methodology adopted in this work.

## 2.5 Feature Selection and Dimensionality Reduction

Beyond the explainability challenges discussed above, dealing with multiple modalities (visual, audio, side information) and thus potentially high-dimensional feature spaces, as can arise from methods like TF-IDF or advanced visual models (e.g., DINOv2, ResNet), makes Feature Selection (FS) an important consideration [12]. Interestingly, explainability methods can inform feature selection by revealing which features contribute most to recommendation decisions.

Khalid et al. [12] define feature selection (FS) as a dimensionality reduction technique that involves choosing a subset of original features. This selection aims to improve performance and efficiency. Its goal is also to comprehensibility identify features containing the most relevant information. Features can be categorised as relevant, irrelevant, or redundant [12], and effectively addressing both irrelevant and redundant features is crucial [12].

Feature selection is the process of evaluating and choosing the most informative features to enhance model performance and efficiency [8]. For content-based systems dealing with textual features, metrics can be used to evaluate the relationship between each term and specific categories [26]. FS methods are categorised into filters, wrappers, and embedded or hybrid approaches. Each approach involves trade-offs in computational costs and reliability [12]. Common

metrics and algorithms include the Chi-square ($\chi^2$), Information Gain (IG), Correlation-Based Feature Selection (CFS), and Mutual Information (MI) [12].

Feature selection is also considered important in general multimodal signal processing, as using only relevant information can improve results, build more accurate models with less data, and reduce processing time [8]. There is no one best method for all applications [12].

## 2.6 Cold-Start and Beyond-Accuracy Challenges

As previously briefly mentioned, one significant advantage of CBRS is its robustness to the cold-start problem, which occurs for both new users and new items. In such cases, no data is currently available, and therefore, recommending similar items is challenging [2,13]. Deep learning features extracted from video content (visual, aural, and motion) have been investigated for their potential in enhancing recommendations for newly added videos [2]. Addressing this cold-start issue is crucial for providing exposure to new content and, therefore, ensuring a diverse recommendation landscape [18, 22].

Beyond explicit textual and visual content, structured metadata or "side information" can be valuable [11]. Research highlights the importance of incorporating this "side information" such as categories, genre, or production year. These attributes closely relate to an item's peak popularity and duration [11]. This type of information can be integrated into recommendation models, particularly for cold-start items, where data is scarce [11].

Although CBRS seems to be the solution to one of the main RS issues, it still faces broader challenges. The first one is the bias and fairness on specific content types that may dominate in recommendations. Second, explainability is crucial because deep models lack transparency in decision-making [5]. The third challenge encompasses user engagement and trust, where segment-level preference understanding, novelty, diversity, and explainability are emerging as crucial evaluation dimensions [15].

While content-based video recommendation has seen significant progress, ongoing research continues to explore areas for improvement. These include enhancing multimodal content representation [15], exploring the contribution of diverse deep learning features and their fusion methods [19], integrating affective signals [15], developing real-time recommendation approaches for dynamic content like live streams [15], and improving the understanding of user feedback on specific segments of longer videos [15]. Furthermore, challenges such as addressing biases, improving transparency and explainability in systems that leverage deep neural networks, and exploring beyond-accuracy metrics like perceived novelty and diversity remain active areas of research [5, 19].

## 2.7 Siamese Neural Networks in Content-Based RS

To move beyond traditional similarity metrics, such as cosine or Euclidean distance, recent work has explored Siamese Neural Networks (SNNs). SNNs are used to learn similarity functions in the embedding space [18, 22]. These architectures utilise twin subnetworks with shared weights to process input pairs (positive or negative) or triplets (anchor, positive, and negative). They map those inputs to a shared vector space where similar items are positioned closer together, and dissimilar items are positioned farther apart.

Siamese Neural Networks are a subtype of Artificial Neural Networks explicitly designed for learning a similarity metric between two inputs [22]. First proposed in 1993 for signature verification, SNNs consist of at least two identical neural networks that share weights [22]. The core idea involves two inputs through identical networks to produce embedding vectors. Once the

twin networks produce these embedding vectors, fixed-distance metrics like cosine or Euclidean distance are used to quantify their similarity [17, 22]. The network trains to reduce the distance between similar item embeddings while increasing the distance between dissimilar items [22].

### 2.7.1 SNN Architectures and Training Strategies

As mentioned earlier, SNNs can be configured with different numbers of input parameters, such as pairs or triplets [22]. Pair-based SNNs take two elements as input and learn to adjust the embeddings based on their similarity through loss functions [22]. Commonly used loss functions include Binary Cross Entropy (BCE) and Contrastive loss [22]. Elements can be positive pairs, where both are similar (e.g., one liked movie and another liked movie by one same user) or negative pairs, in which both are different (e.g., one liked movie and one disliked movie by one same user). Triplet-based SNNs comprise three elements: an anchor, a positive example (similar to the anchor), and a negative example (dissimilar to the anchor) [22]. These systems use Triplet Loss to ensure the anchor remains closer to the positive example. The loss function maintains a specific margin from the negative example [22].

Various types of feed-forward networks can be used as the identical 'twin' networks within the SNN architecture, including Multilayer Perceptrons (MLPs), CNNs, and Recurrent Neural Networks (RNNs) [22]. CNNs are frequently used for image-like inputs as they are adequate for extracting features from grid-like data [18, 22]. Mel-spectrograms, which are bitmap representations of audio signals, serve as standard CNN inputs within SNNs for music similarity tasks [18].

### 2.7.2 Integration in Recommender Systems

The integration of SNNs into RSs can be done in different ways [22]. SNNs integrate into RSs through two main approaches. First, they can be used for prediction, where SNN output directly calculates the similarity score used for ranking items [22]. Second, they can be used for feature extraction, where the SNN embedding vectors serve as intermediate data for larger models like clustering or learning-to-rank approaches [22].

SNNs appear suitable for CBRS because they operate without requiring any user history, making them cold-start friendly. They learn task-specific embeddings based on item content. Additionally, they adapt to pairwise or triplet training using user-annotated data or implicit similarity data, such as genre features. The ability of SNNs to learn similarity based purely on content makes them a viable solution for the item cold-start problem in CBRS [18], as well as for new users since no data is mandatorily required. By training an SNN on item content (e.g., audio features for music, visual/audio features for video) to learn which items are similar, a system can recommend new, unseen items by finding those whose content is similar to items the user already likes, avoiding the need for interaction data [18].

### 2.7.3 Applications and Empirical results

Research into applying SNNs for content-based recommendation to address the cold-start problem has been explored, mainly in domains like music [18]. This study by Pulis et al. proposes using SNNs to determine music similarity based solely on audio content, converting audio clips into Mel-Spectrograms as input to an SNN composed of two identical CNNs [18]. The output of each CNN forms an embedding vector, and their Euclidean distance is used to determine similarity [18].

An evaluation through a blind-test survey showed that participants gave higher ratings to recommendations from the SNN-based system than to some naive genre-based baseline [18]. The

findings showed that the proposed content-based SNN approach was practical in recommending songs with low listen counts. Specifically, 55% of recommended songs had fewer than 1500 listens. This proves the system's capability to provide fair exposure to lesser-known artists and address the cold-start issue using musical content alone [18].

The application of SNNs in recommender systems spans various domains, including e-commerce, fashion, film, job seeking, music, news, and tourism [22]. Although audio and image data are primarily used as input [22], video data has also been utilised in SNN-based recommendation approaches, albeit less frequently in the surveyed literature [22]. This suggests that SNNs can be adapted to process the content of video items for similarity computation within video CBRS.

## 2.8 Evaluation Methodologies and Metrics

Evaluating the recommender system's effectiveness is crucial and can involve offline experiments, user studies, and online trials [23]. When evaluating, it is essential to consider factors such as generalisation and statistical significance of results [23]. The evaluation extends beyond traditional accuracy metrics, especially given the opacity of modern deep learning approaches. Recent discussions in the RS community highlight beyond-accuracy dimensions, including novelty, diversity, explainability, and fairness, which are crucial for ensuring systems provide value beyond simple prediction accuracy [22].

When it comes to evaluating SNN-based RSs, it often involves offline methodologies that use collected data to simulate user behaviour [22]. Although various metrics are used, there is a tendency to focus on accuracy-related metrics, such as Precision@K, Recall@K, and AUC [22]. To measure performance properties, specific metrics are used for different purposes [23]. For content-based systems, accuracy metrics such as ROC, PR, F1, and Kappa are used for semantics-driven systems [3]. Other metrics, like MAP, Hit Rate, and nDCG, are also commonly employed [2].

Extensive evaluations have been conducted on datasets such as Douban Movies and Amazon datasets. Results are reported using multiple metrics: HR@10 (Hit Rate at 10), nDCG@10 (Normalised Discounted Cumulative Gain at 10), MRR@10 (Mean Reciprocal Rank at 10), Precision, and Recall [11]. These different metrics are standardly used to assess the quality of recommended lists' rankings. For instance, a Content-based Journals & Conferences Recommender System achieved 61.37% accuracy. It suggested venues in about 5 seconds on average [26] and provided both Top-1 and Top-3 recommendations [26].

Recent discussions in the RS community highlight beyond-accuracy evaluation dimensions. These include novelty, diversity, serendipity, coverage, and fairness. Such metrics are particularly important to avoid popularity biases [22]. These beyond-accuracy metrics are crucial for ensuring that recommendation systems provide value that is above simple prediction accuracy.

## 2.9 Challenges and Future Directions

Despite the real potential of SNNs and multimodal CBRS, several challenges remain for real-world applications. These include adapting the latest SNN techniques and custom loss functions. Additional challenges involve extending applications beyond simple prediction to different domains and improving research reproducibility [22]. Additional challenges involve adapting the latest SNN architectures and training strategies to new domains, as the optimal choice of loss function, network depth, and data sampling method often depends on the specific problem context [22].

The ongoing research landscape continues to explore several key areas. These include enhancing multimodal content representation and developing more sophisticated fusion techniques. Additional areas involve integrating affective computing signals, creating real-time recommendation

approaches for dynamic content, and improving segment-level user-preference understanding [15]. Moreover, addressing algorithmic bias, improving model explainability, and developing comprehensive evaluation frameworks that consider beyond-accuracy metrics remain critical challenges for the field [5, 19].

In summary, SNNs provide a powerful method for learning content-based similarity, making them a valuable tool for addressing the cold-start problem in content-based recommender systems. This enables recommendations to be made based on item features rather than solely on historical interactions. While their application in music recommendation using audio features has shown promising results in recommending less popular content, their use in video CBRS represents an area with significant potential for further exploration and development, particularly concerning specific network architectures, loss functions, and comprehensive evaluation metric that encompass both accuracy and beyond-accuracy metrics.

## 2.10 Summary and Research Gaps

This literature review reveals a rapidly evolving landscape in content-based movie recommender systems. Major advances in multimodal feature extraction, deep learning architectures, and similarity learning approaches characterise this. However, several critical research gaps and opportunities emerge.

The field has progressed from simple metadata-based approaches to sophisticated multimodal systems which integrate visual, audio, and textual features. Deep learning has revolutionised feature extraction. Moreover, Siamese Neural Networks have shown promise in learning content-based similarity that does not require interaction history.

Research must, therefore, focus on these research gaps:

1. **Limited Multimodal Integration in Video CBRS**

   While multimodal approaches exist, only a few studies integrate visual, audio, and textual features in the specific case of movie recommendation. Most research focuses on single modalities or simple concatenation strategies. Therefore, sophisticated fusion techniques and multimodal integration remain unexplored in the movie domain.

2. **Insufficient Exploration of SNNs for Movie Content**

   Although Siamese Neural Networks have shown success in music recommendation, their application to movie content remains vastly unexplored. The multimodal nature of movies presents unique opportunities for SNN architectures that have not yet been investigated.

3. **Explainability-Performance Trade-Off**

   The transition to deep learning approaches has created a fundamental tension between the accuracy of the recommendations and the model's interpretability. Current explainability methods like SHAP and LIME have been primarily evaluated on traditional features. Their effectiveness on deep, multimodal movie features remains under-investigated.

4. **Inadequate Cold-Start Solutions for Multimodal Content**

   Although content-based approaches theoretically address the cold-start problem, few studies have evaluated multimodal feature combinations for new item recommendations. Limited research compares these approaches to traditional metadata-based methods.

5. **Limited Beyond-Accuracy Evaluation**

Most evaluations focus on accuracy metrics. Usually, in movie recommendation contexts, insufficient attention is given to diversity, novelty, fairness, and user satisfaction. This gap is particularly problematic given the entertainment nature of movie consumption.

6. **Feature Selection for High-Dimensional Multimodal Spaces**

   Combining features from the three modalities creates high-dimensional spaces. Limited research addresses optimal FS strategies. These must balance computational efficiency, recommendation quality, and explainability.

7. **Real-World Scalability and Efficiency**

   Most proposed approaches lack large-scale evaluation on real-world datasets. Limited attention is given to computational efficiency considerations. The gap between research prototypes and production-ready systems remains significant.

These gaps present several concrete research opportunities: **(1)** developing novel multimodal fusion architectures specifically for movie content, **(2)** adapting and evaluating SNN approaches for video recommendation, **(3)** creating explainability frameworks tailored to multimodal movie features, **(4)** establishing comprehensive evaluation protocols that include beyond-accuracy metrics, and **(5)** investigating the optimal balance between feature dimensionality, computational efficiency, and recommendation quality.

# Chapter 3

# Methodology: A Baseline Multimodal Recommender

This thesis addresses the identified research gaps by developing and evaluating a comprehensive multimodal content-based recommendation system. The research follows a two-pronged approach, which is detailed in this and the subsequent chapter.

First, this chapter establishes the complete methodology for a feature-engineered baseline recommender.This process begins with data preparation and the extraction of a wide array of features from visual, audio, and textual modalities. It then details the pipeline for fusing these features, selecting the most informative subset, and finally computing similarity scores using a fixed metric (cosine similarity). This baseline serves as a crucial performance benchmark against which more advanced methods can be judged.

Subsequently, Chapter 4 will detail the development of an end-to-end trainable Siamese Neural Network designed to learn a more nuanced similarity representation directly from data. Through this dual approach, this research investigates the interplay between feature engineering, modality fusion, and similarity learning to advance the state-of-the-art in content-based video recommendation.

All computational work was implemented in Python, using both local development environments (Poetry dependency management and Jupyter Notebooks) and cloud-based machine learning platforms for computationally intensive tasks. Specific technical environment details and computational requirements are documented in **Appendix A** for reproducibility purposes.

## 3.1 Dataset Preparation

The foundation of this research is a multimodal dataset built upon the widely adopted MovieLens 20M dataset [9]. Creating a high-quality corpus suitable for multimodal analysis requires a multi-stage process. This process involved data sourcing, integration, and rigorous filtering. This rich dataset serves a dual purpose: the raw multimodal content (trailers, metadata) is used to build content-based item profiles. At the same time, the user ratings provide the "ground truth" for evaluating whether the system's content-based recommendations align with human preference patterns.

### 3.1.1 Data Sourcing and Integration

Three primary data sources were integrated to build rich, multimodal content profiles. The data integration process began with the MovieLens 20M dataset [9], which served as the cornerstone of the project. This dataset provides a high-quality index of 27,278 movies, cross-platform identifiers (links.csv), and over 20 million user ratings (ratings.csv) reserved for the evaluation phase.

Second, textual metadata was enriched using the IMDb[1] API and the OMDb API[2]. Each MovieLens entry was programmatically linked to its OMDb record to retrieve structured plot summaries and other essential metadata.

Third, audiovisual content was acquired in the form of movie trailers, following established practice in video recommendation [2, 5]. Trailer videos were sourced from the MovieLens 20M YouTube Trailer Dataset [1], which provides validated YouTube video identifiers for 94% of the MovieLens collection. The complete data integration pipeline is illustrated in Figure 3.1.

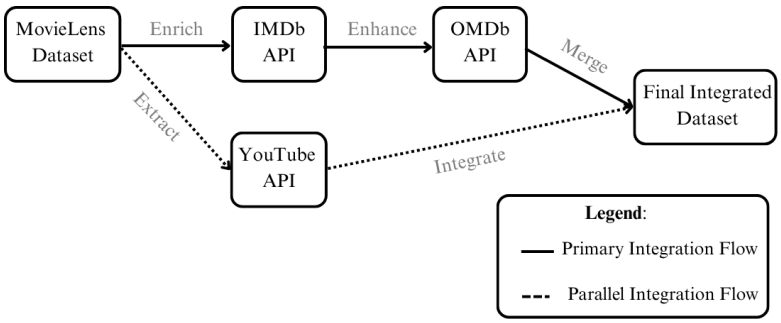



Figure 3.1: Data Integration Pipeline, Combining MovieLens, OMDb, and YouTube Sources.

### 3.1.2 Data Filtering and Final Corpus Composition

To ensure the quality and completeness required for multimodal analysis, a rigorous filtering pipeline was applied. Movies were retained only if they possessed all required data: essential textual metadata (title, plot), a valid and accessible trailer video, and cross-platform identifiers. The filtering resulted in a final set of 13,920 movies. As detailed in Table 3.1, this represents a 49.0% reduction from the original dataset, primarily due to the lack of available trailers and complete metadata.

Table 3.1: Dataset Composition Breakdown Showing Retention Rates at Each Processing Stage.

| **Processing Stage** | **Count** | **Cumulative Loss** |
|---|---|---|
| Initial MovieLens dataset | 27,278 | 0.0% |
| Movies with valid YouTube links | 19,265 | 29.4% |
| Successfully retrieved videos | 14,942 | 45.2% |
| Final corpus with complete metadata | **13,920** | **49.0%** |

Given the significant data reduction, a comprehensive validation analysis was conducted to assess potential selection biases. The analysis, detailed in **Appendix B**, confirmed that the

[1] `https://developer.imdb.com`
[2] `https://www.omdbapi.com`

final corpus remains a representative sample of the original MovieLens dataset in terms of genre, temporal, and user rating distributions.

### 3.1.3 Final Dataset Characteristics and Evaluation Data

The processed user ratings corresponding to the 13,920 movies are essential for evaluation. Following standard practice [14], 5-star ratings were converted into binary relevance labels: ratings $\geq 3.5$ were classified as positive interactions (liked), and ratings $< 3.5$ were considered negative interactions. The binarisation aligns with the ranking-based evaluation metrics used later in this thesis.

Table 3.2 summarises the key characteristics of the final multimodal dataset, which serves as the foundation for subsequent feature extraction and model development.

Table 3.2: Summary Statistics of the Final Multimodal Dataset

| **Characteristic** | **Value** |
|---|---|
| Total movies | 13,920 |
| Temporal coverage | 1894–2014 |
| Average trailer duration | 2.3 minutes |
| Movies with plot summaries | 13920 (100.0%) |
| Total unique languages | 159 |
| Movies with English | 11088 (79.7%) |
| Multilingual movies (movies with multiple languages) | 4484 |
| Unique genres represented | 20 |
| Average genres per movie | 2.1 |
| **Temporal Distribution** | |
| Pre-1950 | 941 (6.8%) |
| 1950-1979 | 2204 (15.8%) |
| 1980-1999 | 4021 (28.9%) |
| 2000-2015 | 6753 (48.5%) |
| **Rating Statistics** | |
| Total ratings (binarised) | 16,252,536 |
| Positive interactions ($\geq 3.5$) | 9,986,983 (61.4%) |
| Negative interactions ($< 3.5$) | 6,265,553 (38.6%) |
| Average ratings per movie | 1172.8 |

## 3.2 Feature Extraction

Following dataset preparation, comprehensive feature extraction was performed across three modalities to capture diverse content characteristics. This research employs a comprehensive multimodal feature extraction pipeline. The pipeline generates rich, content-based representations for each movie trailer. The methodology involves three distinct modalities: visual, audio, and textual. Each captures different semantic and stylistic dimensions of the video content. The objective of this process is to create high-quality, unified feature vectors suitable for subsequent learning tasks.

The feature extraction process was implemented on AWS SageMaker notebooks, leveraging parallel processing capabilities to manage computational load efficiency. All 13920 raw trailer

video files were sourced from an AWS S3 bucket. The subsequent extracted features were stored back to dedicated S3 prefixes for each modality. The distributed architecture enabled scalable processing of the entire dataset while maintaining data organisation and accessibility.

### 3.2.1 Visual Feature Extraction

Visual features characterise the comprehensive visual content of movie trailers, including scene composition, object presence, motion dynamics, and overall visual aesthetics. The extraction process involved multiple stages, from intelligent frame sampling to high-level semantic embedding generation.

#### Frame Sampling and Preprocessing

The visual feature extraction pipeline began with a sophisticated frame sampling strategy to ensure representative coverage of trailer content while filtering out non-content frames. Each trailer was processed to extract a fixed number of 120 frames, uniformly distributed across the video's duration. However, preliminary analysis revealed that simple uniform sampling was insufficient, as it was susceptible to contamination from non-content segments such as studio logos, rating cards, and end-of-trailer credit sequences.

To address the problem of content contamination, an intelligent filtering process was developed. The algorithm analysed frames from the first and last 15% of each trailer, where non-content segments are most common. The algorithm used a combination of metrics, including brightness thresholds (to detect black/white frames), text density from OCR (EasyOCR), and edge complexity, to identify "bad" frames.


Figure 3.2: **Examples of Problematic Content Frames Identified by the Filtering Algorithm.** The algorithm effectively detects various types of contamination that can introduce noise into the feature space, including (a) Walt Disney Pictures studio logo, (b) PIXAR studio logo, and (c) coming to theaters textual overlay.

The initial implementation of this filter highlighted two practical challenges that necessitated a more advanced, two-pass approach, as demonstrated using the *Toy Story* trailer.

- **Challenge 1: Segment-Based Contamination** The initial filter identified individual "bad" frames. However, non-content sequences, such as studio logos, typically span multiple consecutive frames, as depicted in Figure 3.3. These frames are nearly identical.

Removing only the single frame that triggered the filter would leave many contaminated frames in the dataset. The solution was to enhance the algorithm to identify the start and end timestamps of these non-content segments and exclude the entire temporal block from consideration.

- **Challenge 2: Temporal Representation Bias** This segment-based filtering introduced a new potential problem: temporal bias. After removing non-content segments, simple uniform sampling on the remaining frames could result in a frame set disproportionately drawn from one section of the trailer. Such bias could lead to the final sampled frame corresponding to a moment only partway through the video, completely missing content from the latter half, as in Figure 3.4.

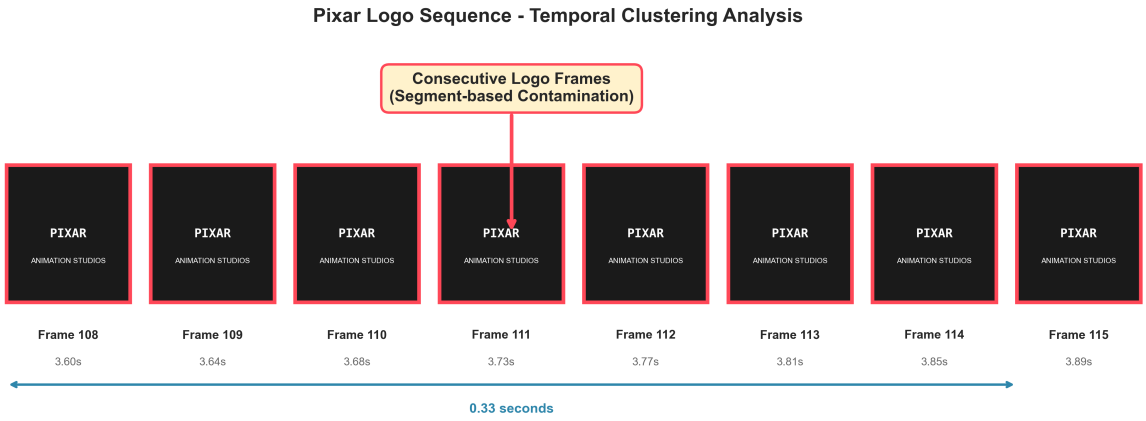



Figure 3.3: **Segment-Based Contamination Challenge.** Non-content often appears in consecutive sequences. Removing only a single detected frame is insufficient, as it leaves other nearly identical noisy frames in the dataset.

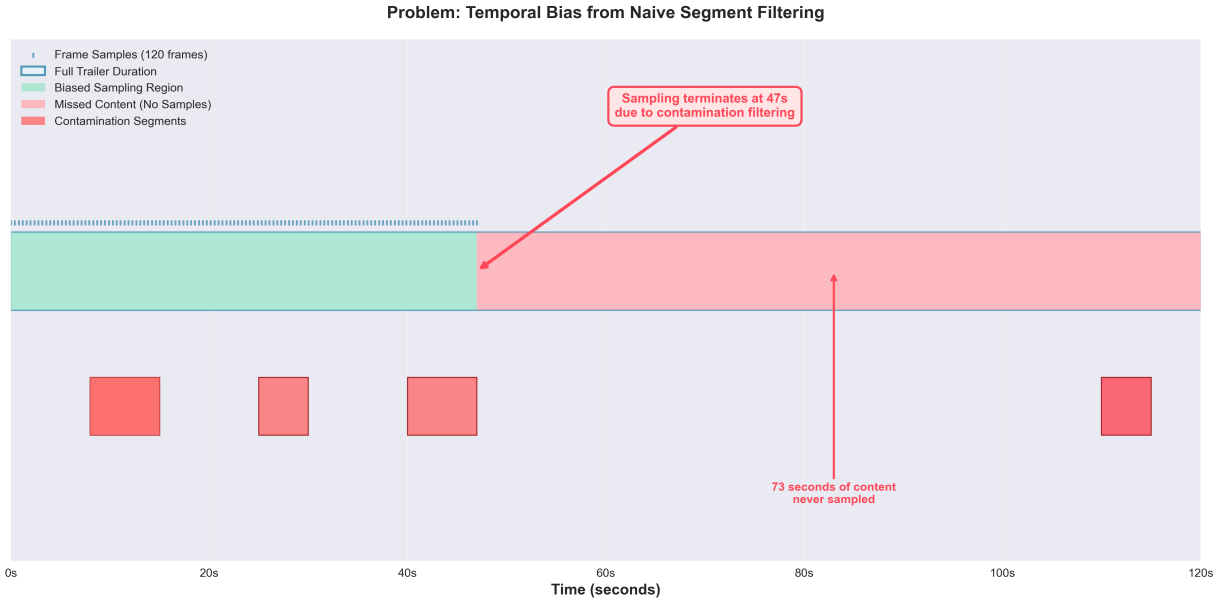
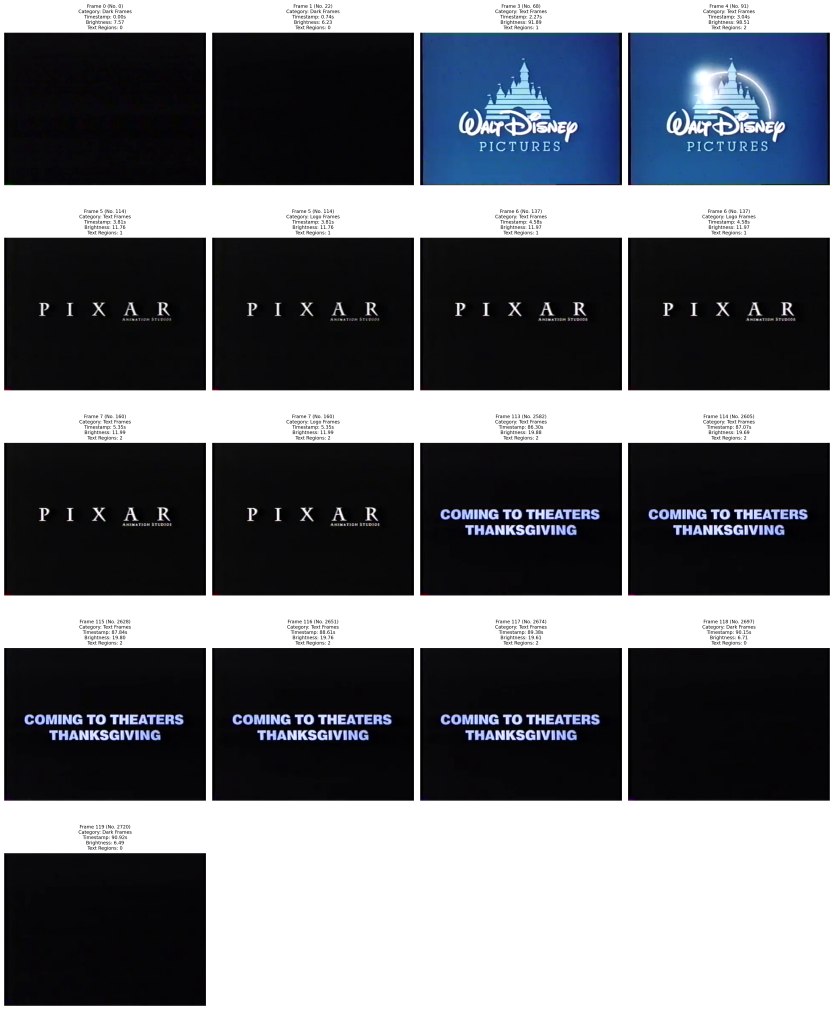



Figure 3.4: **Temporal Representation Bias Challenge.** After removing non-content segments, a naive uniform sampling of the remaining frames can result in a biased selection, failing to capture content from the trailer's entire duration. Here, sampling terminates at 47s, missing 73s of content.

To illustrate these challenges more clearly, Figure 3.5 demonstrates the problems with the initial frame filtering approach. The algorithm successfully identifies individual non-content frames, such as studio logos and release information cards. However, problematic content often appears in segments of multiple consecutive frames, making single-frame removal insufficient.




(a) Initial algorithm detects individual problematic frames


(b) Non-content segments span multiple consecutive frames

Figure 3.5: **Problems with Initial Frame Filtering Approach.** *(a)* The algorithm successfully identifies individual non-content frames, such as studio logos and release information cards. *(b)* However, problematic content often appears in segments of multiple consecutive frames, making single-frame removal insufficient.

To mitigate both challenges, the methodology adopted a final **two-pass sampling strategy**. In the first pass, all non-content segments are identified and masked. In the second pass, a new uniform sampling of 120 frames is performed exclusively on the remaining validated content segments. This robust, two-pass approach ensures high-quality frames while maintaining representative temporal coverage across the trailer's meaningful content.

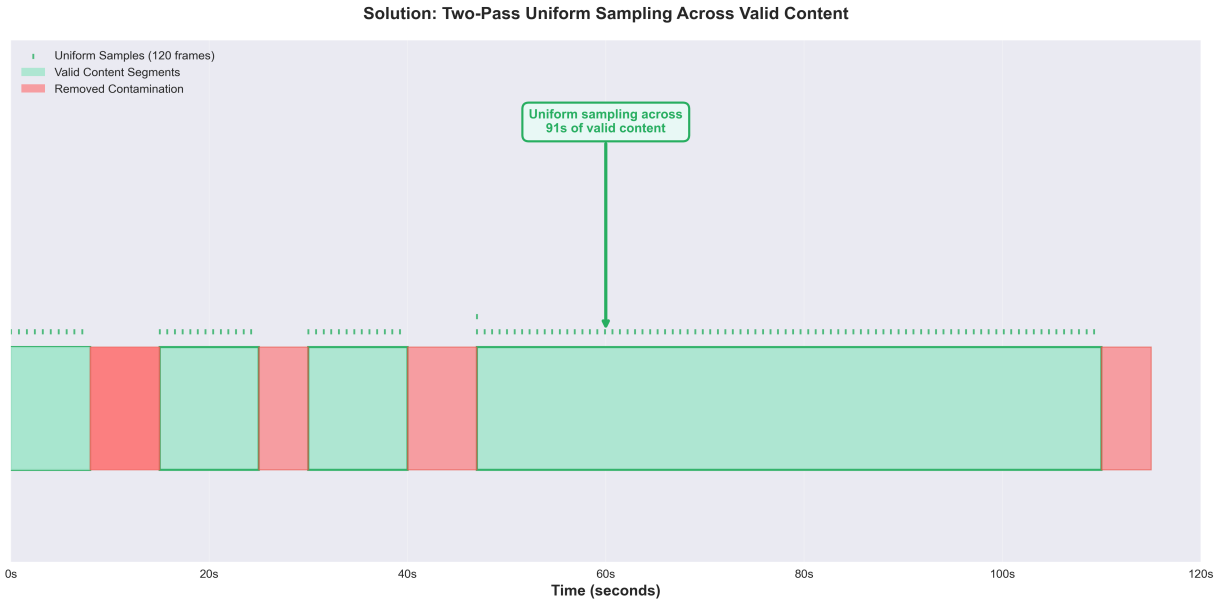
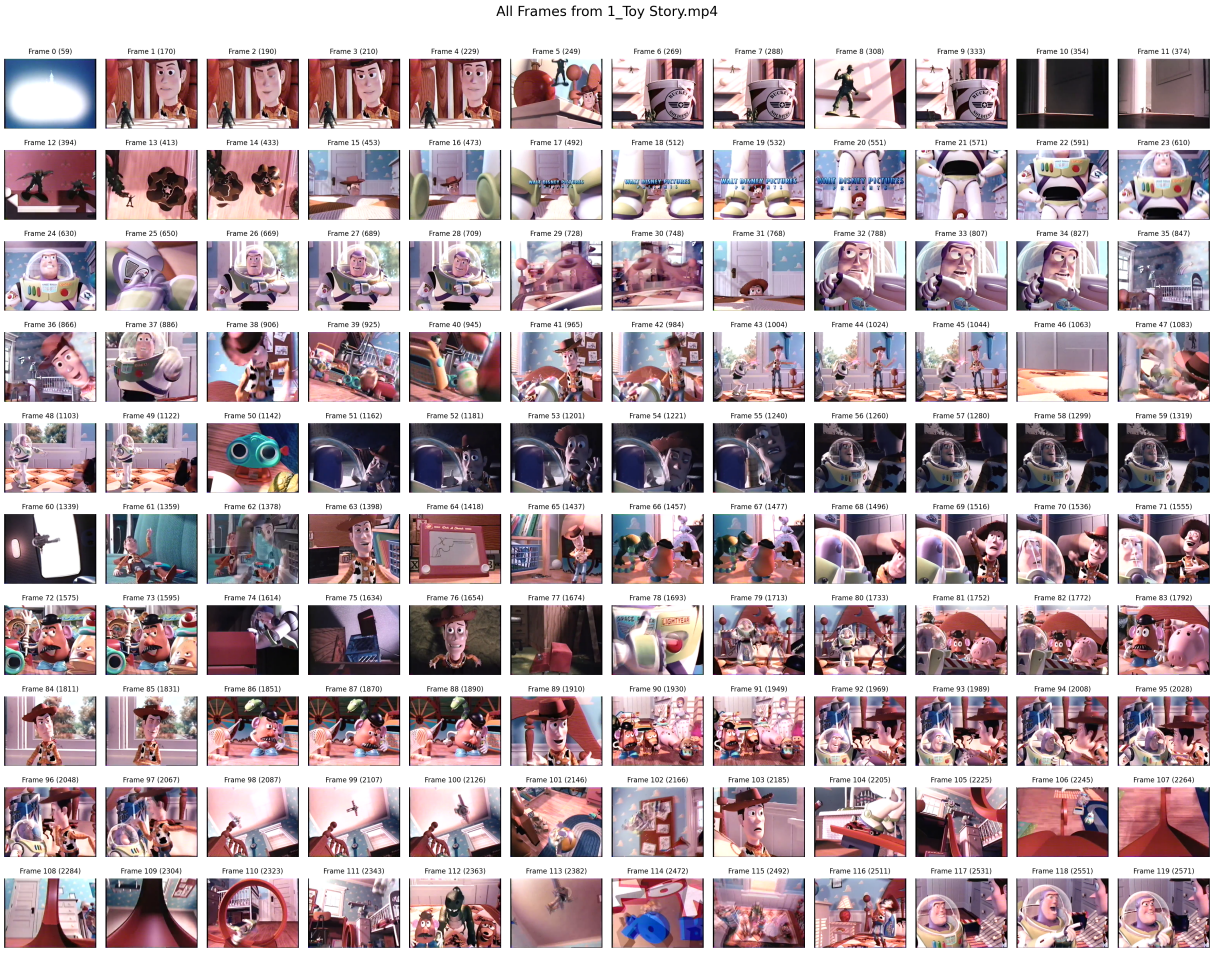



Figure 3.6: **The Implemented Two-Pass Sampling Solution.** This strategy first identifies and masks all contaminated segments. In a second pass, it performs a new uniform sampling exclusively across the remaining, disjoint regions of valid content. This ensures the final frame set is both high-quality and temporally representative of the entire trailer.

Figure 3.7 demonstrates the effectiveness of this two-pass sampling strategy. The comparison shows how naive filtering creates temporal bias, with the final sampled frame occurring only 47 seconds into the trailer. In contrast, the final two-pass approach ensures uniform sampling across the entire duration of meaningful content.


(a) Temporal bias: sampling ends at 47 seconds




(b) Final two-pass sampling: complete temporal coverage

Figure 3.7: **Two-Pass Sampling Strategy and Results.** *(a)* Naive filtering creates temporal bias, with the final sampled frame occurring only 47 seconds into the trailer. *(b)* The final two-pass approach ensures uniform sampling across the entire duration of meaningful content.

The final output of this two-pass sampling process is demonstrated in Figure 3.7b, which shows a representative subset of the final 120 frames for the *Toy Story* trailer. The result is a set of high-quality, visually diverse frames that are uniformly distributed across the trailer's meaningful content, providing a robust foundation for feature extraction.

### Semantic and Low-Level Feature Extraction

From this curated set of 120 frames, both high-level semantic and low-level traditional features were extracted.

**Semantic Visual Embeddings** High-level semantic visual features were extracted using DINOv2 [16]. The model was employed without fine-tuning to preserve its general-purpose semantic representation capabilities.

Each sampled frame was processed through the DINOv2 base model ('facebook/dinov2-base'[3]) to obtain 768-dimensional feature vectors. Two types of visual representation were computed. First, frame-level embeddings for potential temporal analysis. Additionally, aggregated trailer-level embeddings created by averaging frame-level features. The dual approach enables both detailed temporal analysis and efficient similarity computation at the trailer level.

To complement the high-level semantic understanding provided by DINOv2 embeddings, traditional computer vision techniques were used to capture the fine-grained visual characteristics that deep learning models might overlook.

**Low-Level Visual Features** Complementing the high-level semantic embeddings, an extensive set of traditional computer vision features were extracted to capture low-level visual characteristics. The methodology computed these features for each sampled frame and subsequently aggregated them to provide trailer-level descriptors.

The extracted low-level features included:

- **Colour Intensity Features:** 32-bin RGB histograms, colour channel statistics (mean and standard deviation), HSV colour space representations, and dominant colour extraction using k-means clustering on pixel samples.
- **Texture Analysis:** Haralick-inspired texture features computed from greyscale histograms, Local Binary Pattern-like metrics, and gradient-based descriptors, including Sobel gradient magnitude statistics.
- **Spatial Features:** Edge density computed using Canny edge detection, brightness and contrast measurement, image sharpness assessed via Laplacian variance, and compositional features such as rule-of-thirds intersection point intensities and symmetry scores.
- **Temporal Dynamics:** Frame-to-frame motion analysis using histogram comparisons, pixel-wise differences, and Structural Similarity Index (SSIM) calculations to detect scene changes and estimate motion intensity.

**Scene Composition and Object Analysis** To complete the visual feature set, advanced scene understanding was done. Analysis was achieved through object detection and shot composition analysis. The YOLOv8n[4] model was employed for real-time object detection. The model

[3]https://dinov2.metademolab.com
[4]https://docs.ultralytics.com/models/yolov8/

provided categorised object presence information, including people, vehicles, animals, and other semantic categories.

Shot type classification was performed by analysing the proportion of detected faces and edge density patterns. The algorithm estimated ratios for close-ups, medium shots, and wide shots, providing insights into cinematographic style and content focus. Scene change detection used multiple complementary approaches. These included pixel-wise absolute differences and SSIM-based structural analysis. The algorithm applied a temporal constraint to ensure a minimum one-second separation between detected scene changes, thereby avoiding false positives from minor variations.

While visual features capture the aesthetic and compositional elements of movie trailers, the audio modality provides equally important information about music style, dialogue patterns, and sound design. Audio features complement visual representations by encoding temporal dynamics and emotional cues that may not be apparent from the visual content alone.

### 3.2.2 Aural Feature Extraction

Audio features capture the acoustic characteristics of movie trailers, including musical content, speech patterns, sound effects, and overall auditory ambience. The extraction process combined traditional signal processing with modern deep learning approaches.

#### Audio Processing and Traditional Acoustic Features

Audio tracks were extracted from video files using the Librosa Python library, preserving original sampling rates to maintain signal fidelity. For compatibility with deep learning models, audio was later resampled to 16 kHz mono format.

A comprehensive set of acoustic descriptors was computed using Librosa, covering temporal, spectral, and rhythmic characteristics.

The comprehensive acoustic descriptor set included the following:

- **Temporal Features:** Root Mean Square (RMS) energy statistics capturing loudness and dynamic range, zero-crossing rate measurements indicating signal noisiness and percussive content.
- **Spectral Analysis:** Mel-frequency Cepstral Coefficients (MFCC) for timbral characterisation, spectral centroid reflecting the audio "brightness", spectral bandwidth and roll-of measurements, and spectral contrast features distinguishing harmonic from noisy content.
- **Harmonic and Rhythmic Features:** Chroma features representing the twelve distinct pitch classes for harmonic analysis, tempo estimation and beat detection for rhythmic characterisation, and onset detection for identifying sound even boundaries.
- **Advanced Processing:** Harmonic-percussive source separation (HPSS) to analyse harmonic versus percussive energy distribution, providing insights into musical versus sound effect content.

Traditional acoustic features provide interpretable audio characteristics. However, modern deep learning approaches can capture more complex patterns and relationships within audio content.

#### Deep Audio Embeddings

Two complementary deep learning models were employed to extract high-level audio representations.

**VGGish Embeddings** were extracted from the VGGish model, which is pre-trained on the YouTube-8M dataset. The model was used to capture general acoustic scene characteristics. Audio was processed in segments. The resulting 128-dimensional embeddings were averaged across the entire trailer duration to produce a single trailer-level representation.

**Wav2vec2 Embeddings** were extracted from the 'facebook/wav2vec2-base-960h'[5] model. The model provides speech-focused representations, which are particularly valuable for capturing the characteristics of dialogue and narration. The model's last hidden state was mean-pooled to generate 768-dimensional embeddings, with input limited to 30 seconds to manage computational requirements.

#### Speech and Music Analysis

Finally, the methodology implemented a heuristic-based approach to estimate the relative proportions of speech and music content within each trailer. The algorithm segmented audio into 500 ms frames, analysing zero-crossing rates, spectral centroids, and chroma variance to classify frame content. The analysis provided valuable metadata for understanding trailer composition and potential genre characteristics.

Beyond the audiovisual content of trailers, textual metadata provides rich semantic information about movie themes, genres, and narrative elements. While audio and visual features capture stylistic and aesthetic properties, textual features encode explicit semantic content and categorical information that are essential for content-based recommendations.

### 3.2.3 Textual Feature Extraction

Textual features were derived from movie metadata, including titles, plot summaries, genre classification, and release information. The extraction process combined traditional natural language processing techniques with modern transformer-based embeddings.

#### Data Sources and Preprocessing

Textual features were extracted from cleaned movie metadata, which included titles, plots, genres, and release years, supplemented by IMDb integration for additional language and production information. Natural language processing used NLTK for tokenisation and stop-word identification, while spaCy's multilingual model provided named entity recognition capabilities across different languages.

#### Statistical and Linguistic Features

Basic textual statistics were computed to capture document characteristics. The statistical features include text length measurement for both titles and plot summaries, word count analysis, and calculations of stop-word ratios. These features provide insights into content complexity and writing style. Language analysis included primary language identification through IMDb integration, with subsequent encoding for numerical processing. Multi-hot encoding was applied to present the full spectrum of languages associated with each movie.

[5] https://huggingface.co/facebook/wav2vec2-base-960h

#### Named Entity Recognition and Traditional Embeddings

Semantic content analysis was performed through named entity recognition. It involves extraction and categorising mentions of persons, organisations, and locations from plot summaries. The approach captures thematic elements and provides structured representations of narrative content.

TF-IDF vectorisation was applied separately to movie titles and plot summaries. The TF-IDF implementation employed preprocessing steps, including lowercasing, stop-word removal, and punctuation filtering. Feature dimensionality was controlled through maximum feature limits (50 for titles, 100 for plots) to balance representation quality with computational efficiency.

#### Contextual Text Embeddings

Modern transformer-based embeddings were generated using the BERT model to capture semantic meaning and contextual relationships. Both titles and plot summaries were processed through the model. The last two hidden states were mean pooled to produce 768-dimensional dense vectors. The maximum sequence length was set to 128 tokens to strike a balance between computational requirements and content coverage.

The comprehensive multimodal feature extraction pipeline successfully captured diverse content characteristics across visual, audio, and textual modalities, providing a robust foundation for the subsequent feature fusion and similarity learning components of the recommendation system.

The comprehensive feature extraction process yields rich but heterogeneous representations across visual, audio, and textual modalities. Each modality contributes unique insights: visual features capture aesthetic and compositional elements, audio features encode temporal dynamics and musical characteristics, and textual features provide explicit semantic content. However, these modalities exist in different feature spaces with varying dimensions and scales. To leverage their complementary nature for recommendation, these diverse representations must be integrated into unified feature vectors through systematic feature fusion techniques.

## 3.3 Feature Fusion

The feature extraction process yields rich but heterogeneous representations across the three modalities. To leverage their complementary nature for recommendation, these diverse representations must be integrated into a unified feature vector. This chapter focuses on creating a static, feature-engineered representation for the baseline model. This involves a pipeline of preprocessing, fusion (concatenation), and, eventually, feature selection to create a final, optimised vector. This static vector is then used with a fixed similarity metric. This stands in contrast to the end-to-end approach in Chapter 4, where the fusion process itself is a learnable component of the neural network.

### 3.3.1 Preprocessing and Standardisation

A critical step before fusion is the standardisation of heterogeneous features. This involves two main challenges: normalising features with variable lengths and harmonising features with different value scales.

#### Variable-Length Feature Normalisation

A critical challenge in multimodal fusion involves standardising features with inherently variable dimensions. Scene change detection features, for instance, produced variable-length arrays depending on trailer content. The variable dimensions necessitated specialised normalisation approaches.

The methodology implemented and evaluated three standardisation methods:

- **Truncation/Padding Strategy:** Variable-length feature vectors were adjusted to a fixed target dimension through truncation of excess elements or zero-padding for insufficient elements.
- **Interpolation Method:** Linear interpolation was employed to resample variable-length arrays to target dimensions, preserving temporal distribution characteristics while ensuring dimensional consistency.
- **Statistical Aggregation:** Rather than preserving raw temporal sequences, statistical summaries were computed, including count, mean, standard deviation, minimum, and maximum values of change points.

The interpolation method demonstrated superior performance by preserving temporal patterns while ensuring dimensional consistency and was adopted as the primary standardisation approach.

Beyond dimensional consistency, all features require scale normalisation.

#### Global Feature Normalisation

All feature vectors underwent normalisation to prevent scale-based bias during fusion, which could arise when one modality contains significantly denser or larger-magnitude features than others. Three normalisation strategies were implemented: z-score standardisation, min-max scaling, and robust scaling. The latter is median-based scaling using interquartile range. It provides resilience against outliers.

Z-score standardisation was selected as the primary approach due to its effectiveness with predominantly Gaussian-distributed deep learning embeddings and its ability to preserve relative feature relationships.

### 3.3.2 Early Fusion Architecture

The primary approach combines all modalities through concatenation.

#### Concatenation-Based Fusion

The primary fusion strategy employed early fusion through feature concatenation. It combines normalised vectors from all modalities into a unified representation:

$$\mathbf{f}_{\text{early}} = [\mathbf{f}_{\text{visual}} \parallel \mathbf{f}_{\text{audio}} \parallel \mathbf{f}_{\text{text}}] \tag{3.1}$$

The concatenation approach yields fixed-size multimodal embedding vectors. The concatenated vectors capture visual aesthetics, acoustic patterns, and semantic narrative content in a unified representation suitable for downstream similarity computation and item recommendation.

**Alternative Fusion Methods**

While concatenation formed the primary fusion approach, the research explored additional strategies to provide flexibility for handling missing modalities:
**Weighted Sum Fusion:** Modality-specific weights were applied to emphasise certain feature types based on preliminary performance analysis. Weights were automatically adjusted when specific modalities were unavailable. The weighting approach ensured normalised contribution across available features.
**Average Fusion:** Equal weighting of available modalities, proving a balanced representation when modality importance was uncertain.

These alternative approaches enabled comparative analysis of fusion strategy effectiveness and provided robustness when dealing with incomplete multimodal data.

The feature fusion process successfully combines multimodal representations into unified vectors, but this concatenation approach can result in high-dimensional feature spaces containing redundant or noisy information. Not all extracted features contribute equally to recommendation quality, and some may introduce computational overhead without improving performance. Therefore, a systematic feature selection process is crucial for identifying the most informative features while reducing dimensionality and enhancing efficiency.

## 3.4 Feature Selection

Following the feature fusion stage, the resulting high-dimensional vector space (initially containing 2,304 features) contains potentially redundant and noisy information. To address this, a systematic feature selection pipeline was designed to identify the most predictive subset of features, with the dual goals of improving model efficiency and interpretability.

### 3.4.1 Data Preparation for Feature Selection

The feature selection process began with the systematic preparation of both feature and rating datasets to ensure compatibility and quality. The fused multimodal feature sets were loaded, and array-like features stored as strings were expanded into individual numerical columns. The MovieLens ratings dataset was then integrated to serve as the target variable for supervised selection methods.

### 3.4.2 Feature Selection Pipeline

The selection process employed a sequential, three-stage pipeline, moving from broad, unsupervised filtering to fine-grained, supervised selection.

**Stage 1: Unsupervised Pre-Filtering.** This initial stage aimed to remove uninformative and redundant features without considering the target variable.

- **Variance Thresholding:** A threshold of 0.01 was applied to remove features with minimal variance across all samples.
- **Correlation-Based Removal:** A Pearson correlation matrix was computed, and for any pair of features with an absolute correlation coefficient exceeding 0.85, one feature was removed to reduce collinearity.

**Stage 2: Supervised Importance Scoring.** After pre-filtering, two complementary supervised methods were used in parallel to score the remaining features based on their relevance to user ratings.

- **Mutual Information:** MI was used to measure the statistical dependence between each feature and the target ratings, capturing both linear and nonlinear relationships.
- **Random Forest Feature Importance:** A Random Forest regressor was trained to predict ratings from the features, and importance scores were extracted based on the mean decrease in impurity.
- **Principal Component Analysis (PCA):** For high-dimensional deep learning embeddings (DINOv2, VGGish, BERT), PCA was applied after initial filtering to transform the correlated features into a set of orthogonal principal components. This step was performed prior to importance scoring to provide the MI and RFFI algorithms with decorrelated, information-rich inputs. The feature importance was then measured on these principal components (e.g., `visual_dinov2_pca_4`).

**Stage 3: Threshold-Based Selection.** Rather than selecting a fixed number of features, a data-driven approach based on cumulative importance was adopted. For each scoring method (MI and RFFI), features were ranked, and a threshold was applied to retain the top features that collectively accounted for a predefined percentage (e.g., 90%) of the total importance score.

### 3.4.3 Combined Selection Strategy

To leverage the complementary strengths of the two supervised methods, the final feature set for each modality was formed as the union of the features selected by MI and those selected by RFFI, as expressed by the formula: $F_{\text{final}} = F_{\text{MI}} \cup F_{\text{RFFI}}$. The union strategy ensures that any feature deemed important by at least one robust method is retained, creating a final feature set that is both comprehensive and powerful.

## 3.5 Similarity Computation and Recommendation

Unified feature representations enabled the generation of similarity-based recommendations. After obtaining unified multimodal feature representations through feature fusion, the next critical step involves computing similarities between movie trailers and generating recommendations based on these similarities. This section outlines the computational methodology for transforming fused feature vectors into actionable similarity scores and recommendation lists.

### 3.5.1 Similarity Computation

The similarity computation process transforms high-dimensional fused feature vectors into interpretable similarity scores between movie pairs, forming the foundation for content-based recommendations.

#### Input Data and Preprocessing

The primary input for similarity computation consists of the fused feature vectors obtained from the previous feature fusion stage. The system stored these features as NumPy arrays where each row represents the d-dimensional fused feature vector for a single movie derived from its trailer

and associated metadata. Alongside is the corresponding list of trailer identifiers. For datasets containing N trailers, this results in an N x d matrix where d represents the dimensionality of the fused feature space.

#### Similarity Metrics

To quantify the similarity between two movie ($i$, $j$) feature vectors $\mathbf{e}_i$ and $\mathbf{e}_j$, several distance functions were evaluated:

**Cosine Similarity:** It is the primary similarity metric employed. It measures the cosine of the angle between two feature vectors:

$$\text{sim}_{\cos}(\mathbf{e}_i, \mathbf{e}_j) = \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\|\|\mathbf{e}_j\|} \tag{3.2}$$

Cosine similarity was selected as the default metric due to its scale-invariant properties and robustness to differences in vector magnitudes. This property makes cosine similarity particularly suitable for the high-dimensional and often sparse feature representations common in multimedia content analysis.

**Euclidean Distance:** As an alternative distance measure, Euclidean distance was also considered:

$$\text{dist}_{\text{euc}}(\mathbf{e}_i, \mathbf{e}_j) = \|\mathbf{e}_i - \mathbf{e}_j\|^2 \tag{3.3}$$

However, cosine similarity demonstrated superior performance in preliminary experiments, and the methodology, therefore, adopted this metric.

#### Similarity Matrix Computation

The computation of pairwise similarities between all trailers results in an N x N similarity matrix S. The element $S(i, j)$ represents the similarity between trailer $i$ and trailer $j$. For the complete dataset of 13,920 trailers, the computation involves computing approximately 193 million pairwise similarities.

To manage computational complexity, a batched processing approach was implemented for datasets exceeding 5,000 items. The computation divides trailers into batches of 1,000, computing similarities between each batch and all other trailers while storing only the top-K similarities to reduce memory requirements.

### 3.5.2 Candidate Filtering and Recommendation Generation

The recommendation generation employs a two-stage approach: candidate filtering through k-nearest neighbours (k-NN) selection followed by ranking-based recommendation output.

#### Candidate Selection

For each trailer i, the k-NN process identifies the top-K most similar trailers based on cosine similarity scores. Self-similarity scores are excluded by setting them to -1, preventing self-recommendations. All remaining trailers are ranked by similarity scores in descending order, with the top-K forming the candidate set $C_i$.

#### Recommendation Ranking

The final recommendation generation selects the top-K items from each candidate set based on similarity scores, sorted in descending order. The output includes both trailer identifiers and their corresponding similarity scores, stored in both pickle and CSV formats for subsequent evaluation and analysis.

For robust evaluation and analysis, the generated top-K recommendation lists were stored in both a dictionary-based pickle format for programmatic access and a human-readable CSV format for easy reference.

The recommendation generation process produces similarity-based movie suggestions, but the effectiveness of these recommendations must be rigorously assessed. Evaluation requires comparing system outputs against known user preferences to determine whether content-based similarities align with actual user behaviour. This section outlines a comprehensive evaluation framework designed to assess the quality of recommendations from multiple perspectives.

## 3.6 Evaluation

The evaluation framework assesses the content-based movie recommendation system from multiple perspectives, incorporating both traditional metrics based on user preferences and content-similarity metrics derived from feature embeddings. The evaluation process provides a comprehensive assessment of the quality, relevance, and coherence of content in recommendations.

### 3.6.1 Experimental Setup

#### Ground Truth Dataset Construction

The evaluation dataset was constructed from MovieLens ratings, filtered to align with the 13,920 processed movie trailers. Original ratings (scale 0.5-5.0) were converted to binary relevance scores, where ratings of 3.5 or higher were considered positive interactions. The filtering process yielded approximately 16.2 million relevant ratings from users who had at least five positive interactions, ensuring a meaningful evaluation. The processed data was organised into two key structures: user_items (mapping user IDs to positively rated movie sets) and user_item_ratings (mapping user IDs to all rated movies with scores).

#### ID Mapping and Consistency

Consistent mappings between trailer IDs and MovieLens movie IDs were established through multiple strategies: direct string matching, numerical component extraction, and substring matching. Both forward and reverse mappings ensure bidirectional consistency during evaluation.

### 3.6.2 Evaluation Metrics

The evaluation employs an item-based perspective where, for each target movie that users have rated positively, the system's recommendations are evaluated against other movies that the same users also rated positively.

### 3.6.3 Ranking-Based Metrics

**Precision@K** measures the proportion of recommended items that are relevant to the user:

$$\text{Precision@K} = \frac{|\{\text{Recommended items @K}\} \cap \{\text{Relevant items}\}|}{K} \tag{3.4}$$

**Recall@K** [6] measures the proportion of all relevant items that are captured within the top-K recommendations:

$$\text{Recall@K} = \frac{|\{\text{Recommended items @K}\} \cap \{\text{Relevant items}\}|}{|\{\text{Relevant items}\}|} \tag{3.5}$$

**Hit Rate@K** indicates whether at least one relevant item appears in the top-K recommendations:

$$\text{Hit Rate@K} = \begin{cases} 1 & \text{if } |\{\text{Recommended items @K}\} \cap \{\text{Relevant items}\}| \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

**nDCG@K** Normalised Discounted Cumulative Gain considers both relevance and ranking position, computed as:

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}$$

Where:

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

And $\text{rel}_i$ is the relevance score (actual rating) of the item at rank i. IDCG@K represents the DCG@K for an ideal ranking sorted by true ratings.

**MRR@K** Mean Reciprocal Rank measures the quality of ranking by computing the reciprocal of the rank of the first relevant item:

$$\text{MRR@K} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\text{rank}_q}$$

Where Q is the set of queries and $\text{rank}_q$ is the rank of the first relevant item for query q.

### 3.6.4 Evaluation Protocol Summary

The comprehensive evaluation framework combines ranking-based metrics (Precision@K, Recall@K, Hit Rate@K, nDCG@K, MRR@K) to assess recommendation quality from multiple perspectives. All metrics are computed at $K \in \{1, 3, 5, 10\}$ to evaluate both immediate relevance and broader recommendation coverage. The item-based evaluation approach ensures that content similarity is validated against actual user preference patterns, providing a robust assessment of the multimodal feature extraction and fusion methodology.

[6] Recall@K can be a less informative metric in large-catalogue recommendation scenarios, as the denominator (the total number of relevant items for a user) can be very large, making high scores difficult to achieve. Therefore, this thesis places greater emphasis on Precision, Hit Rate, and ranking-aware metrics like nDCG and MRR.

This chapter has presented a comprehensive methodology for developing a content-based movie recommendation system through multimodal feature extraction and similarity learning. The approach addressed each component of the recommendation pipeline, from dataset preparation to evaluation. Starting with the MovieLens 20M dataset, the methodology establishes a robust multimodal dataset of 13,920 movies with complete visual, audio and textual content. The feature extraction process captures complementary characteristics across three modalities: visual features, encoding aesthetic and compositional elements; audio features, representing temporal dynamics and acoustic patterns; and textual features, providing explicit semantic content. Through feature fusion and selection, these heterogeneous representations are transformed into unified, optimised feature vectors. The methodology culminates in a framework for generating and evaluating content-based recommendations using cosine similarity, providing a solid foundation for testing whether multimodal content analysis can effectively capture user preferences.

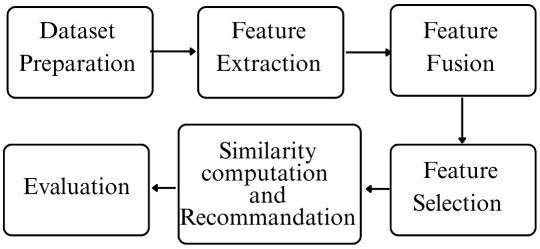



Figure 3.8: Methodology Flowchart

# Chapter 4

# Methodology: End-to-End Similarity Learning

## 4.1 Motivation for a Learned Approach

While the baseline system described in Chapter 3 provides a functional content-based recommender, its reliance on fixed similarity metrics, such as cosine similarity, can present fundamental limitations and potentially lead to suboptimal results. Such metrics treat all features as equally important and are incapable of learning nuanced, content-dependent relationships between content elements. Fixed metrics assume that the mathematical formulation of similarity (e.g., dot product in high-dimensional space) accurately captures human perceptions of content similarity. However, this is often not the case for complex multimedia content.

The ablation study of the baseline model revealed a critical insight: naively concatenating visual features introduced more noise than signal. This finding directly motivated the design of the end-to-end model architecture, which is not merely a feature-fusion mechanism, but a similarity-learning framework designed to autonomously discover the most salient features and suppress the noise inherent in high-dimensional visual data.

To overcome these limitations, a more sophisticated approach is required, one that can learn a similarity function directly from user interactions. This motivated the development of an end-to-end trainable model. The core of this end-to-end approach is the learning of a bespoke embedding space. The Siamese network learns a powerful, non-linear transformation that maps the raw, high-dimensional multimodal features into a new, low-dimensional space. In these spaces, spatial proximity (as measured by a simple metric, such as cosine or Euclidean distance) is trained to directly correspond to the notion of content similarity derived from user interaction data. In essence, the network does not learn a new distance formula; it learns an optimal representation of the items themselves, making a simple distance metric effective.

This learned approach has the potential to automatically weigh the most salient features, discover complex cross-modal interactions, and ultimately produce a more semantically meaningful representation of content for recommendation.

It is important to note that while this approach is end-to-end in the sense that it learns directly from high-level features to a final similarity score, the core visual and textual backbones (DINOv2, BERT) remain largely frozen. This design choice is a pragmatic trade-off, allowing the system to leverage the powerful, general-purpose representations of these large, pre-trained models while focusing the learning process on task-specific fusion and similarity functions, a

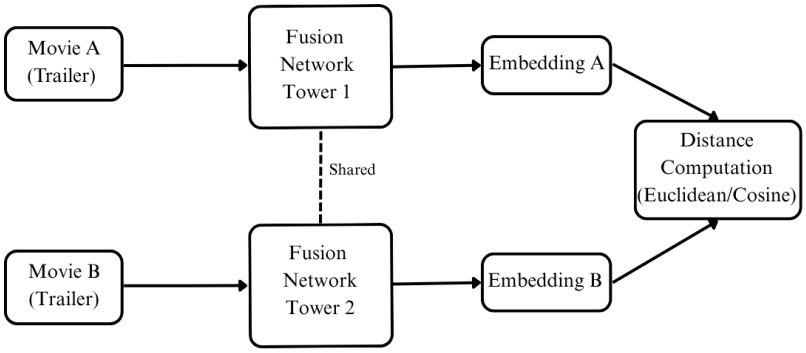



Figure 4.1: Twin Networks Architecture

common and effective strategy in applied deep learning.

## 4.2 Siamese Neural Networks: Exploratory Investigation

### 4.2.1 Rationale and Objectives

As a preliminary step towards building the final architecture, an exploratory investigation into similarity learning using SNN was conducted. The primary objective of this investigation was not to produce a final, optimised model but rather to serve as proof-of-concept and to resolve key implementation decisions that would inform the design of the end-to-end system. The twin-tower architecture, a hallmark of SNNs (Figure 4.1), was chosen as the structural foundation for this exploration.

### 4.2.2 Implementation Focus Areas

The exploratory SNN investigation focused on critical key aspects:

- **Prototyping the Data Pipeline:** Establishing the initial logic for generating positive and negative training pairs from the MovieLens rating data. This involved developing algorithms to extract meaningful user preferences and convert explicit ratings into implicit similarity signals.
- **Validating the Learning Framework:** Testing the basic training loop of a twin-tower SNN architecture to ensure gradients could be backpropagated correctly through fused multimodal feature vectors. This validation was crucial to confirm that the networks could effectively learn from heterogeneous feature representations.
- **Evaluating Deep Learning Frameworks:** Implementing initial prototypes in both TensorFlow and Pytorch to assess their respective suitability for the custom data loaders and model architecture required for this project. Key considerations included flexibility with the existing feature extraction pipeline.

### 4.2.3 Framework Selection and Technical Decisions

Through a systematic comparison of implementation approaches, PyTorch was selected as the primary framework for its flexible and intuitive interface for building custom neural network

modules. The decision was particularly influenced by PyTorch's dynamic computation graph, which facilitated the debugging of the complex multimodal fusion components and its seamless integration with the custom data loading requirements of the similarity learning task.

This exploratory work, while not yielding stand-alone performance results, proved invaluable in establishing the foundational architecture and confirming the viability of the similarity learning approach. The insights gained from this phase directly informed the design principles and implementation strategies employed in the final end-to-end model.

## 4.3 Training Data Generation for End-to-End Learning

The efficacy of a learned similarity model is fundamentally dependent on the quality of its training data. The primary challenge lies in translating raw user interaction data (1-to-5 ratings) into a supervised learning problem for an SNN. This process involved defining what constitutes a "similar" (positive) and "dissimilar" (negative) pair of movies. This was an iterative process, with each strategy designed to address fundamental limitations discovered in the previous one.

### 4.3.1 Binarisation of User Ratings and User Filtering

The initial step in creating supervision signals was the binarisation of the MovieLens 1-to-5 star ratings into implicit "liked" and "disliked" feedback. A systematic analysis was conducted to find the optimal threshold that would maximise the number of users contributing meaningful training data. A "valid user" was defined as a user having a sufficient number of both liked and disliked items. This is a crucial prerequisite for generating both positive and negative training examples for contrastive learning.

As shown in Figure 4.2, a threshold of 4.0 was selected, which is stricter than 3.5 used for the baseline evaluation ground truth. This choice was made to provide the learning algorithm with a cleaner, less ambiguous supervision signal by defining "liked" items with higher confidence. Indeed, this threshold offered the best trade-off, retaining 99.58% of the user base while ensuring a sufficient population of "valid users" capable of providing diverse preference signals. Thresholds lower than 3.5 resulted in a sharp drop-off in valid users, while a higher threshold of 4.5 began to exclude users unnecessarily. Consequently, a rating of 4.0 or higher was established as a "liked" signal for all subsequent data generation strategies.

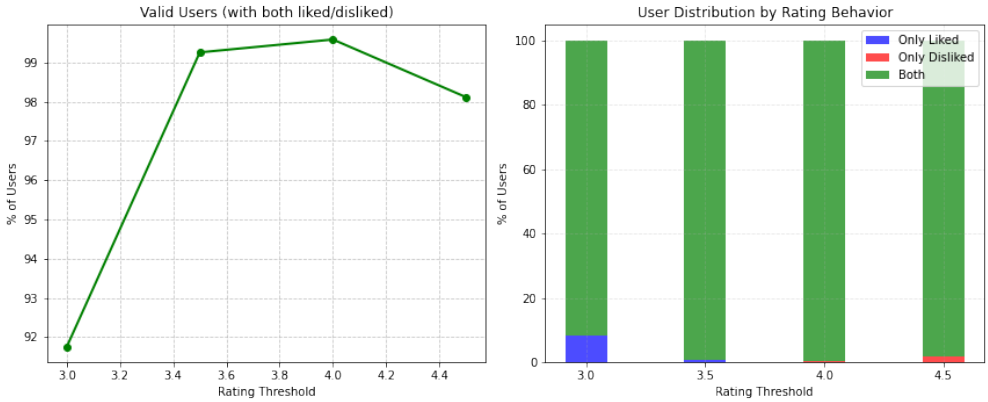



Figure 4.2: Analysis of Rating Threshold for Creating Training Pairs.
The left plot shows that a threshold of 4.0 maximises the percentage of "valid users" (users with both liked and disliked items). The right plot shows the distribution of user types at each threshold.

### 4.3.2 Evolution of Pair Generation Strategies

The development of an effective pair-generation strategy required multiple iterations. Each additional improvement was made to address fundamental limitations discovered in the previous approach. This iterative refinement process is detailed below to provide insights into the challenges inherent in similarity learning for CBRS.

#### Strategy 1: Genre-Based Heuristic Approach

The initial strategy combined collaborative filtering signals from user ratings with genre overlap as a content-based heuristic. The generation of positive pairs respected two rules: both movies must be co-rated highly ($\geq 4$) by the same user, and they must share at least one genre. For the negative pairs, both movies were randomly selected with no shared genres.

This approach assumed that genre overlap could serve as a reliable proxy for content similarity, providing an intuitive foundation for similarity learning. However, training revealed that this strategy introduced a severe confounding variable, effectively creating a data leakage scenario. The model was given a feature (genre) that was also used to construct its supervisory signal. Unsurprisingly, it achieved near-perfect training metrics within a few epochs by discovering a trivial classification rule: if movies share genres, they are similar; otherwise, they are dissimilar. This shortcut prevented the network from processing visual, audio, or textual content, thus resulting in complete generalisation failure.

#### Strategy 2: Pure Collaborative Filtering Approach

To eliminate genre-based shortcuts, the second strategy relied exclusively on user behaviour. The positive pairs were generated as both movies co-rated highly ($\geq 4$) by the same user, regardless of genre. The negative pairs were the hard negative pairings of a user's liked movie (rated $\geq 4$) with their disliked movie (rated $\leq 3$).

This approach forced the model to learn fine-grained content distinctions by creating challenging negative examples where movies might share high-level categorical features but differ in user preference. However, qualitative analysis revealed that pure collaborative signals could

introduce semantic noise. Users occasionally co-rate semantically disparate films highly (e.g., action blockbusters and animated children's films). While recommending a children's film to an action fan might be a valid recommendation for that user, who has diverse tastes, it provides a noisy and often contradictory signal for a model attempting to learn content-based similarity. Since the explicit goal of this research is to model the intrinsic properties of the content, these noisy pairs were deemed counterproductive for the learning task.

**Strategy 3: Hybrid Content-Collaborative Approach**

The final strategy combined user preference signals with a quantitative measure of content similarity, specifically the Jaccard similarity. Jaccard similarity is a measure of set overlap, defined as the size of the intersection divided by the size of the union of two sets. Positive pairs are created such that movies are co-rated highly ($\geq 4$) by the same user AND exhibit a genre Jaccard similarity $\geq 0.25$. Negative pairs consist of movies where one movie is liked and another disliked by the same user AND exhibit a Jaccard similarity $< 0.1$.

**Jaccard Similarity Calculations:** For movies with genre sets $G_1$ and $G_2$, the Jaccard similarity is computed as:

$$J(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \tag{4.1}$$

- **Positive Pair Example:** "The Dark Knight" (Action, Crime, Drama) and "Heat"(Action, Crime, Thriller) have $J = 2/4 = 0.5$, qualifying as a positive pair if co-rated highly.
- **Negative Pair Example:** "The Dark Knight" (Action, Crime, Drama) and "Finding Nemo"(Animation, Adventure, Comedy) have $J = 0/6 = 0.0$, qualifying as a negative pair with disagreeing ratings.

This hybrid approach has many advantages. First, it filters out noisy collaborative signals while preserving authentic user preferences. Second, it ensures thematic coherence in positive pairs without creating trivial classification tasks. Third, it provides unambiguous negative examples with explicit content and preference distinctions. Last, it maintains sufficient complexity to require learning from multimodal content features.

The final set of pairs was then split into training (70%), validation (15%), and test (15%) sets. The split was performed by first combining all positive and negative pairs into a single collection, which was then randomly shuffled and partitioned into three sets. It is important to note that this is a random split at the pair level, not the user level. While a user-level split is the standard for preventing any data leakage in user-centric recommendation tasks, this research has a different primary objective: learning a universal, content-based similarity metric independent of any single user's preferences. The set of generated pairs represents a valuable, and computationally expensive, collection of "ground truth" similarity signals. A pair-level split was therefore chosen to maximise the utilization of this data for training the similarity function itself. It is acknowledged that this introduces a minor risk of the model learning latent user-level taste patterns, a trade-off made to prioritise the learning of a robust content-to-content similarity space. The potential for minor data leakage is acknowledged as a limitation of this study in Section 7.4.

Initial data generation revealed significant popularity bias. Training pairs were heavily skewed toward older, more frequently-rated movies. This bias risked producing a model that performs well on popular items but poorly on the long tail of the catalogue.

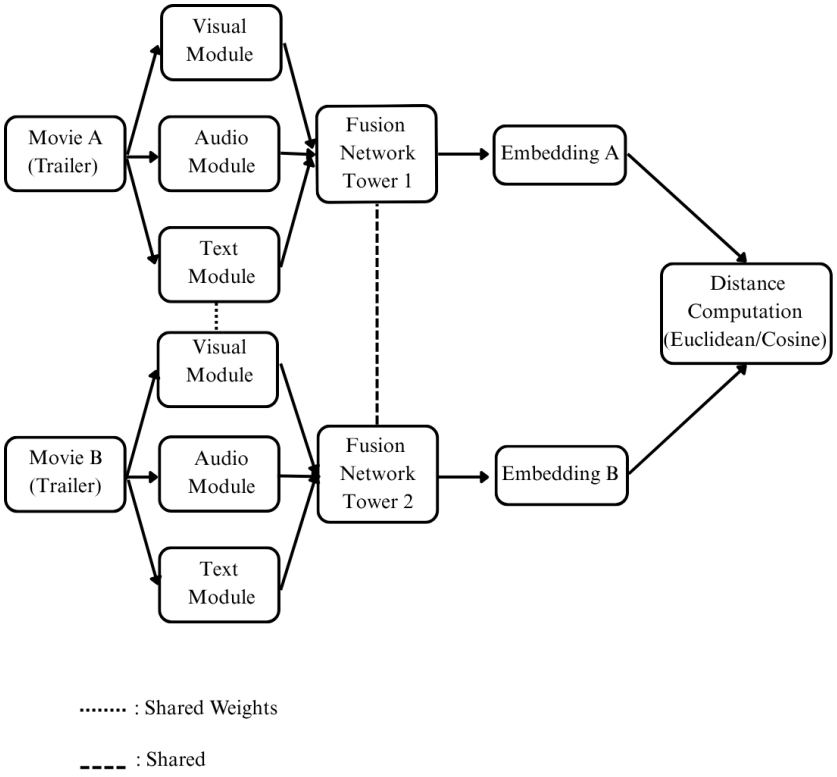



Figure 4.3: Twin-Tower Architecture with Multimodal Processing

To address this limitation, a stratified sampling approach was implemented based on movie ID ranges, which correlate with release years and popularity. The movie catalogue was split into three strata: low, medium, and high IDs. Balanced quotas were enforced to ensure adequate representation across all segments. This approach produced a more evenly distributed dataset, forcing the model to learn similarity patterns across the entire movie catalogue rather than optimising primarily for popular items.

## 4.4 End-to-End Model Architecture

To address the limitations of the fixed-metric baseline, an end-to-end trainable model was developed using a multimodal Siamese Neural Network architecture. This approach allows the model to learn a bespoke similarity function directly from data, jointly optimising the feature processing, fusion, and distance-metric space.

### 4.4.1 Overall Design Philosophy: The Twin-Tower Mode

The model employs a "twin-tower" structure, a hallmark of Siamese networks. Two identical processing pipelines (the "towers") operate in parallel, each taking one movie from a pair as input. Both towers share the exact weights, ensuring that identical inputs produce identical outputs. The outputs of the towers are two embedding vectors in a shared, low-dimensional space.

### 4.4.2 Multimodal Feature Processing Modules

Each tower is composed of three specialised sub-modules to process the visual, aural, and textual modalities of a movie trailer. These modules are similar to those used in the baseline system but are now integrated into the end-to-end training framework.This architecture allows for their parameters to be fine-tuned. However, in this implementation, the core backbones are kept frozen to leverage their powerful pre-trained representations, while newly added fusion and projection layers are trained from scratch.

#### Visual Processing Module

This module implements a hybrid approach combining semantic and low-level visual analysis. Pre-computed DINOv2 embeddings (384 dimensions) provide a high-level semantic understanding of video frames, capturing objects, scenes, and compositional elements. These embeddings are pre-extracted and cached to optimise processing speed during training.

The semantic features are supplemented by dynamically computed low-level features, including brightness, contrast, edge density, and colour histograms. These features capture cinematographic style and visual aesthetics that may be crucial for content similarity but are not captured by high-level semantic embeddings.

The combined visual features are processed through a dedicated Multi-Layer Perceptron (MLP) that learns optimal weighting and projections. This MLP consists of three connected layers with ReLU activations and dropout regularisation, transforming the concatenated visual features into a standardised representation suitable for fusion.

#### Audio Processing Module

Similar to the visual module, this module employs a dual-pathway approach to capture both general acoustic characteristics and task-specific patterns. Pre-computed VGGish embeddings (128 dimensions) provide representations trained on general audio classification tasks, capturing instrument types, vocal characteristics, and ambient sounds. These are also pre-extracted for processing efficiency.

Simultaneously, a custom convolutional neural network processes Mel-spectrograms generated from raw audio waveforms. This CNN consists of three convolutional layers followed by global average pooling, learning trailer-specific audio patterns that general-purpose embeddings may not capture. The architecture is designed to identify the rhythmic patterns, dynamic ranges, and frequency distributions characteristic of different movie genres and styles.

#### Text Processing Module

It is designed to understand narrative content from plot summaries and titles. A critical and intentional design choice in this end-to-end architecture was the explicit exclusion of genre as a feature input. While the baseline model and feature selection analysis confirmed that genre is the single most powerful predictor, it was omitted here to force the model to learn a deeper, more nuanced form of similarity directly from the semantic and stylistic content of the movies themselves, rather than relying on a high-level categorical label. This approach directly tests the hypothesis that a multimodal network can infer thematic closeness from raw content, moving beyond simple genre matching.

To achieve this, the module integrates semantic and keyword-based text analysis to understand narrative content. A pre-trained BERT model generates contextualised embeddings from movie titles and plot summaries, capturing semantic relationships and narrative themes. The

BERT embeddings were complemented by TF-IDF vectorisation of plot text, which helps preserve important keywords and phrases that may be diluted in dense semantic representations.

Additional categorical features, including language and release year, are encoded and incorporated to provide temporal and linguistic context that may influence content similarity patterns.

### 4.4.3 Feature Fusion and Embedding Generation

The features extracted from each modality are concatenated into a unified high-dimensional vector that preserves the distinct contributions of visual, audio, and textual information. This concatenated representation is processed through a fusion network. The complexity of this fusion network was a key variable in the experimental design, adapted to the stability requirements of different training configurations. While more complex architectures, such as cross-attention, were explored, simpler MLP-based networks proved more robust, particularly for the more challenging Triplet Loss training paths. The specific fusion network used for each of the four experimental models is detailed in **Appendix D**.

The fusion network, an MLP with dropout regularisation, is designed to learn the optimal combination of these multimodal signals. By processing the concatenated vector through its layers, it implicitly learns to weigh the relative importance of different modalities, reduce dimensionality, and project the content into a final, semantically rich 128-dimensional embedding space.

These embeddings undergo L2-normalisation in experiments using cosine distance, constraining them to the surface unit hypersphere. This normalisation provides several advantages: it stabilises training by preventing embedding magnitude from growing unbounded; it enables direct use of cosine similarity as a distance metric; and it facilitates comparison with baseline methods. For Euclidean distance experiments, this normalisation step is omitted to allow the model to learn both directional and magnitude information.

## 4.5 Experimental Design: A 2x2 Factorial Approach

To comprehensively investigate which similarity learning strategy is most effective, a $2 \times 2$ factorial experimental design was employed. This approach systematically combines two types of loss functions with two types of distance metrics, resulting in four distinct end-to-end models:

1. **Pairwise Contrastive Loss with Euclidean Distance**
2. **Pairwise Contrastive Loss with Cosine Distance**
3. **Triplet Loss with Euclidean Distance**
4. **Triplet Loss with Cosine Distance**

This design allows for direct comparison of not only the loss functions (Pairwise vs. Triplet) but also the underlying geometric assumptions of the embedding space (Euclidean vs. Cosine).

### 4.5.1 Pairwise Contrastive Learning Path

The first learning path uses a pairwise approach, where the model is presented with pairs of movies labelled as either "similar" (1) or dissimilar" (0).

**2×2 Factorial Experimental Design**

Distance Metric

| Loss Function | Euclidean | Cosine |
|---|---|---|
| **Contrastive** | **Model 1**<br>Contrastive Loss<br>Euclidean Distance | **Model 2**<br>Contrastive Loss<br>Cosine Distance |
| **Triplet** | **Model 3**<br>Triplet Loss<br>Euclidean Distance | **Model 4**<br>Triplet Loss<br>Cosine Distance |

Figure 4.4: Experimental Design Matrix

### Training Data and Loss Function

The training data consists of positive and negative pairs generated using the hybrid content-collaborative approach described earlier.

The model is trained with a Contrastive Loss function, which aims to pull similar items together in the embedding space while pushing dissimilar items apart by at least a predefined margin, m:

$$\mathcal{L}_{contrastive} = \frac{1}{2N} \sum i = 1^{N} \left[ y_i \cdot d_i^2 + (1 - y_i) \cdot \max(0, m - d_i)^2 \right] \tag{4.2}$$

Where:

- $d_i$ represents the Euclidean distance between the $i$-th pair of embeddings
- $y_i$ is the similarity label for the $i$-th pair (1 for similar, 0 for dissimilar)
- $m$ defines the minimum separation distance (margin) for negative pairs
- $N$ is the batch size

### Distance Metric Variants

- **Cosine Distance Implementation:** The final embeddings from the fusion network are L2-normalised, constraining them to the surface of the unit hypersphere. The distance is calculated as $d = 1 - cosine\_similarity(emb_1, emb_2)$. A margin of 0.8 was used, which proved efficient in initial tests.[1]

[1]For L2-normalised vectors, cosine similarity is in [-1,1]. The cosine distance (1-sim) is therefore bounded in [0,2]. A smaller margin, like 0.8, is effective in this constrained space.

- **Euclidean Distance Implementation:** The final L2-normalisation step in the fusion network is removed, allowing the model to learn both angular and magnitude information. The distance is the standard L2-norm: $d = ||emb_1, emb_2||_2$. A margin of 1.2 was employed for this variant.[2]

### 4.5.2 Triplet Learning Path with Semi-Hard Negative Mining

#### Training Data and Online Mining Strategy

Rather than pre-generating triplets offline, this approach employs an online semi-hard negative mining strategy during training. The dataloader provides pairs of (Anchor, Positive) items based on the same hybrid content-collaborative strategy used for pairwise learning. For each anchor-positive pair in a batch, the training loop intelligently selects a "semi-hard" negative from within the same batch.

A semi-hard negative is defined as an item that is farther from the anchor than the positive example but whose distance from the anchor is still within the loss margin. Formally, for an item n such that $d(a, p) < d(a, n) < d(a, p) + margin$. This selection strategy forces the model to learn fine-grained distinctions by focusing on challenging but learnable examples, avoiding both trivially easy negatives (which provide no learning signal) and impossibly hard negatives (which can destabilise training).

#### Triplet Generation Process

For experiments utilising the triplet loss, triplets of the form (anchor, positive, negative) are constructed dynamically during training. Given a positive pair (anchor, positive) from the dataloader, negative items are sampled from the current batch, ensuring thematic dissimilarity (Jaccard similarity $< 0.1$) with the anchor-positive pair when possible. This structure enables learning through relative distance optimisation, where the model learns to position similar items closer than dissimilar ones in the embedding space.

#### Loss Function: Triplet Margin Loss

The model is trained with a Triplet Margin Loss that ensures the distance between the anchor (a) and positive (p) is smaller than the distance between the anchor and the negative (n) by a margin, m:

$$L(a, p, n) = \max(0, d(a, p) - d(a, n) + m)$$

This loss formulation encourages the formation of clusters in the embedding space, where similar content gravitates toward each other while maintaining separation from dissimilar content.

## 4.6 Implementation Considerations

### 4.6.1 Computational Efficiency

The end-to-end architecture was designed with computational efficiency in mind. In particular for the feature extraction components that must process large volumes of multimedia content.

[2]This distance is unbounded, $[0, \infty)$. The magnitude of the embeddings can increase, resulting in a significantly larger distance between points. A larger margin, such as 1.2, is required to create a meaningful separation between negative air in this unconstrained space.

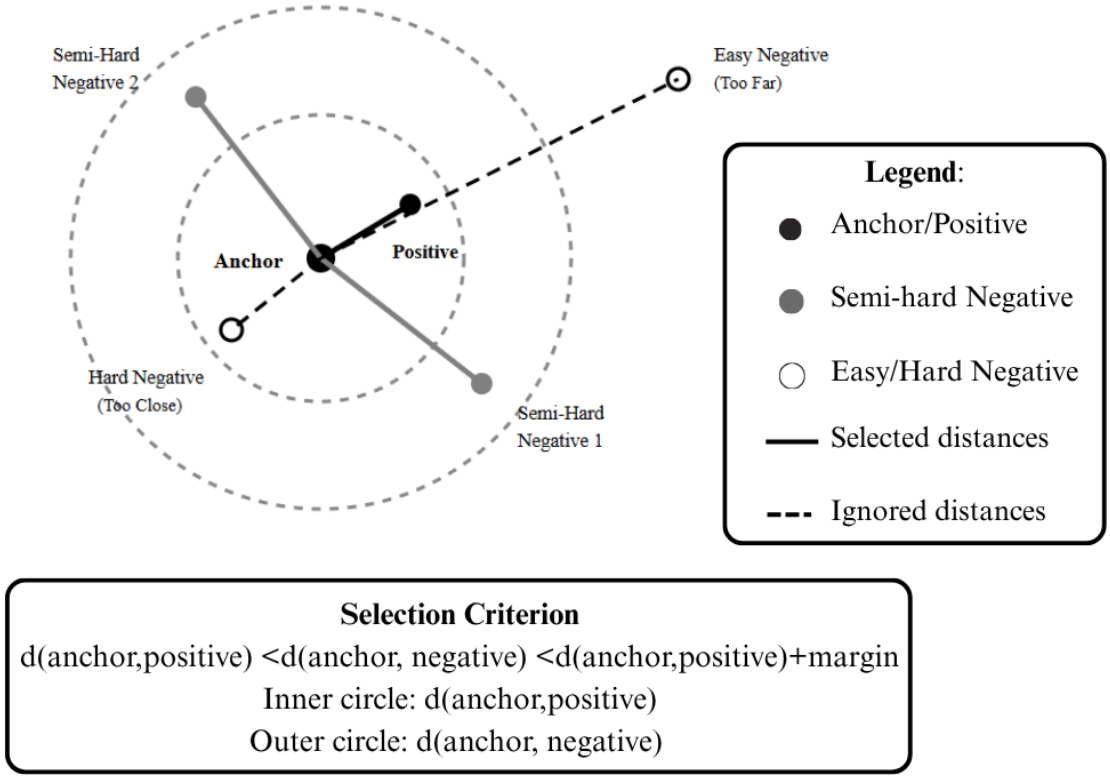



Figure 4.5: Semi-Hard Negative Mining in Embedding Space

Pre-computed embeddings (DINOv2, VGGish) are cached and reused across training epochs, while computationally intensive operations such as Mel-Spectrogram generation are optimised using vectorised implementations.

The twin-tower architecture enables efficient batch processing by computing embeddings for multiple movie pairs simultaneously. This can be achieved by taking advantage of modern GPU architectures optimised for parallel processing. Memory usage is further optimised through gradient check-pointing and mixed-precision training where appropriate.

### 4.6.2 Scalability and Extensibility

The modular design facilitates easy extension to additional content modalities or alternative feature extraction approaches. Each processing module operates independently, allowing for individual components to be updated or replaced without requiring architectural changes to the entire system.

The fusion network's learnable weighting mechanism automatically adapts to the inclusion of new modalities. This makes the architecture extensible to future enhancements such as subtitle analysis, metadata features, or advanced visual processing techniques.

## 4.7 Evaluation Protocol for End-to-End Models

Once the four end-to-end models were trained using their respective loss functions, a unified protocol was required to evaluate their performance in a recommendation task. This protocol combines quantitative raking metrics with qualitative diagnostics to provide a comprehensive understanding of each model's behaviour.

### 4.7.1 Quantitative Ranking Evaluation

To ensure a fair comparison, a standardised evaluation process was applied to all models. This evaluation used the same logic as the baseline evaluation protocol.

First, an embedding index was created for each model. The entire corpus of 13,920 movies was passed through the trained model's "tower" to generate a 128-dimensional embedding for each movie.

Second, for generating recommendations, cosine similarity was used as the universal ranking metric. This was applied to all models, regardless of the distance metric used during training (Euclidean or Cosine), to provide a consistent and normalised measure of similarity for the final ranking task. This provides a consistent and normalised measure of similarity for evaluation across all four models, allowing for a fair comparison of how well each learned representation captures semantic closeness.

The final recommendation lists were then evaluated against the ground-truth user data using the same ranking-based metrics (Precision@K, nDCG@K, etc.) outlined in Chapter 3, Section **??**. This ensures a direct and fair comparison between the baseline model and all four end-to-end variants.

### 4.7.2 Qualitative and Diagnostic Analysis

To understand why the models perform as they do, several diagnostic analyses were planned:

- **Pairwise Classification Metrics:** For the two models trained with a pairwise contrastive loss, their performance as binary classifiers was assessed by generating a confusion matrix

and calculating metrics such as accuracy, precision, recall, and F1-score on the validation set. This will reveal any systematic biases in classifying pairs as "similar" or "dissimilar."

- **Embedding Space Visualisation :** To visualise the quality of the learned metric space, the distribution of distances for known positive and known negative pairs was plotted as histograms. A well-trained model should exhibit a clear separation between these two distributions.
- **Case Study Analysis:** To intuitively assess the semantic quality of the recommendations produced by different learning strategies, a qualitative case study was planned. For a set of selected query movies, this analysis compares the top recommendations generated by each of the four end-to-end model variants (Pair-Cosine, Pair-Euclidean, Triplet-Cosine, and Triplet-Euclidean). This allows for a direct, qualitative comparison of how the choice of loss function and distance metric impacts the thematic coherence and relevance of the recommendations, providing an intuitive complement to the quantitative ranking metrics.

# Chapter 5

# Implementation and Experimental Setup

This chapter describes the technical specifications, software frameworks, computational resources, and hyperparameter configurations employed to implement and train the models described in Chapters 3 and 4. The implementation encompasses both the baseline multimodal recommender system and the end-to-end framework.

## 5.1 Technology Stack

The experimental framework was implemented in Python 3.10, using PyTorch 1.12 as the primary deep learning framework. The implementation relied on a standard data science and deep learning stack, with core machine learning functionalities provided by PyTorch, scikit-learn, and the HuggingFace Transformers library. Multimedia content was processed using specialised libraries such as Librosa for audio and OpenCV for visual data. Data manipulation, monitoring, and visualisation were performed with Pandas, NumPy, Matplotlib, and TensorBoard. All experiments were managed and run on AWS SageMaker, with boto3 used for S3 integration. To ensure reproducibility, random seeds were fixed across all libraries.

## 5.2 Hardware and Computational Environment

All deep learning models were trained on Amazon Web Services (AWS) SageMaker using `ml.g4dn.2xlarge` instances equipped with NVIDIA T4 GPUs with 16GB of GPU memory. This configuration provided sufficient computational resources for training the multimodal neural networks while maintaining cost-effectiveness for extended experimentation periods.

The training was conducted for a maximum of 50 epochs for pairwise models and 100 for triplet models, with early stopping mechanisms employed to prevent overfitting. The early stopping patience was set to 8 epochs for pairwise models and 10 for triplet models, based on validation loss. The training duration was highly dependent on the dataset size used for experimentation. With a test size of 25,000 samples, each epoch required approximately 1.5 hours of computation time. With a reduced test size of 10,000 samples, it resulted in approximately 30 minutes per epoch. This scalability consideration was crucial for iterative experimentations and hyperparameter tunings within practical time constraints.

Pre-computation of features, particularly for the baseline pipeline described in Chapter 3, was parallelised across multiple CPU cores to expedite the data preparation phase. The instances provided 8 vCPUs with 32GB of system RAM, enabling efficient batch processing of multimedia content during feature extraction.

The AWS SageMaker environment facilitated seamless integration with S3 storage for dataset management and model artefact persistence, ensuring reproducibility across experimental runs.

To facilitate rapid prototyping and iterative development within the project's time and computational constraints, a "test mode" was implemented in the training scripts. When enabled, this mode utilised a more minor, randomly sampled subset of the complete training data (25,000 pairs/triplets) for each experiment. While this represents only a fraction of the total available data, it provided a sufficiently large and diverse dataset to train the models meaningfully, verify architectural stability, and compare the relative performance of the different learning strategies. For even faster debugging cycles, a smaller subset of 10,000 samples was occasionally used. It is important to note that all final results and conclusions presented in this thesis are based on models trained and evaluated under this 25,000-sample test mode configuration.

## 5.3 Model Implementation

### 5.3.1 Baseline Pipeline Implementation

The baseline multimodal recommender system was implemented as a modular pipeline with distinct components for each processing stage:

- **Dataset Preparation:** Custom PyTorch Dataset classes were implemented to handle the multimodal movie data, including efficient data loading and caching mechanisms
- **Feature Extraction:** Separate modules for visual, audio, and textual feature extraction, utilising pre-trained models as feature extractors
- **Feature Fusion:** Various fusion strategies, including early, late, and hybrid fusion approaches
- **Similarity Computation:** Distance metric supporting both cosine and Euclidean similarity measures
- **Evaluation Framework:** Comprehensive evaluation pipeline incorporating standard recommendation metrics

### 5.3.2 End-to-End Architecture and Training

In contrast to the modular baseline, the end-to-end similarity learning model was implemented as a single, unified PyTorch `nn.Module`, integrating all components within a trainable framework. The architecture was designed to leverage the insights gained from the feature selection analysis, focusing on learning a robust, low-dimensional embedding for content similarity.

**Network Architecture.** The model is composed of several key components. Modality Backbones process each data stream independently, including a fine-tunable BERT for text, pre-trained vision transformers for visual features, and a custom CNN for audio spectrograms. The outputs are then fed into a Multimodal Fusion Module, a learnable MLP that combines the modality representations. Finally, a Similarity Learning Head projects the fused vector into the final 128-dimensional embedding space.

The Python source code for the core architectural components, including the Siamese network structures, the various fusion networks, and the custom loss functions, is provided in **Appendix E** for transparency and reproducibility. A complete, executable version of the entire project is also available via a public GitHub repository[1].

**Architectural Normalisation Strategy.** A critical aspect of the implementation was the normalisation strategy, as improper normalisation led to training instability in early experiments. To ensure a fair comparison between the cosine and Euclidean distance-based models, specific normalisation rules were established at each stage of the pipeline. These definitive architectural choices, which form the foundation for the experiments in Chapter 6, are summarised with their rationale in Table 5.1.

**Training and Optimisation.** The models were trained using the hyperparameters detailed in **Appendix F**. For the Pair-Cosine model, a differentiated learning rate strategy was employed to balance the fine-tuning of powerful, pre-trained backbones (e.g., BERT) with the learning of new, randomly initialised components. For the other three models, a uniform learning rate of 1e-4 was found to provide more stable training. The AdamW optimiser was chosen for its strong performance with transformer-based architectures. To prevent overfitting, weight decay regularisation was applied, and the learning rate was dynamically adjusted using a ReduceLROnPlateau scheduler for the pairwise models. All models were trained with an effective batch size of 16, determined to be the optimal balance between gradient stability and GPU memory constraints.

**Training Pipeline and Fault Tolerance.** The training process for all end-to-end models was managed by a unified training script designed for robustness and reproducibility. The script supports both pairwise and triplet loss training paradigms, dynamically constructing the appropriate model, loss function, and data loaders based on a configuration file.

A key feature of the pipeline was its checkpointing system. At the end of each epoch, the complete state of the model, including model weights, optimiser state, and the learning rate scheduler's state, was saved to an AWS S3 bucket. This system served two purposes:

1. **Best Model Selection:** The checkpoint with the lowest validation loss was preserved as the "best" model for that experimental run.

2. **Fault Tolerance:** In the event of an interruption (e.g., instance failure or kernel crash), training could be seamlessly resumed from the last saved checkpoint, preventing the loss of significant computation time. This proved critical for long-running triplet loss experiments, which were susceptible to out-of-memory errors during validation due to their higher memory footprint. Additionally, it also enabled the possibility of extending training runs with different hyperparameters, such as a lower learning rate for fine-tuning, by loading a pre-trained model checkpoint.

**Workload and Efficiency Considerations.** An important practical difference emerged between the two training paradigms. The **pairwise training** loop processes pairs of items, requiring two forward passes through the network for each data sample. In contrast, the **triplet training** loop, using an in-batch mining strategy, processes a single batch of unique items and computes distances within that batch. This results in a significantly faster processing time per item for the triplet approach. However, the pairwise dataset contains millions of pre-generated pairs, while the triplet training dataset consists of only the unique movies involved. Consequently,

[1] `https://github.com/JulietteMaes01/multimodal-video-recommendations`

Table 5.1: Architectural and Normalisation Strategy for End-to-End Models

| **Processing Stage** | **Cosine Variants** | **Euclidean Variants** | **Rationale** |
|---|---|---|---|
| **Input Modality Embeddings** (Output of Visual/Audio/Text Modules) | L2 Normalised | L2 Normalised | Ensures all modalities contribute fairly to the fusion network. Prevents any single modality from dominating the fusion process due to differences in scale. |
| **Fusion Network Output** | Raw (Unnormalised) | Raw (Unnormalised) | The fusion network's objective is to learn the optimal combination and relative weighting of features. The final normalisation is deliberately handled as a separate, subsequent step. |
| **Final Embedding for Loss Calculation** (Output of the main SNN 'forward' method) | **L2 Normalised (Mandatory)** | **Raw (Unnormalised)** | This is the key architectural difference: **Cosine:** Normalisation is mathematically required, as the loss operates on the angle between vectors. **Euclidean:** The loss function requires vector magnitude to operate correctly; normalising here would prevent the model from learning effectively. |

each pairwise epoch involved processing a significantly larger number of samples, resulting in a longer overall training time per epoch despite being slower per item. This trade-off between per-item efficiency and total dataset size is a key consideration in designing metric learning experiments.

The entire project followed a modular design pattern to ensure clarity and reproducibility.

Components were separated by function: data loading, feature extraction, model architecture, and evaluation. To handle the computational demands of large-scale multimodal data, optimisation strategies included gradient accumulation for effective batch size management and strategic memory management during intensive processing phases. All code was version-controlled with fixed random seeds across Python, NumPy, and PyTorch to ensure deterministic experimental behaviour.

## 5.4 Per-Epoch Evaluation Pipeline

To analyse the learning dynamics and identify the best-performing model state, a systematic post-training evaluation pipeline was executed.

First, the saved checkpoints from each of the four E2E model's training runs were identified. To manage the high computational cost of evaluating every single epoch, a down-sampling strategy was employed. For each model, the first, last, and every fifth epoch checkpoint, along with the single best-performing checkpoint (identified as best_model.pth during training), were selected for complete evaluation.

For each of these selected checkpoints, the corresponding model weights were loaded. Then, embeddings for the entire corpus of 13,920 movies were generated. From these embeddings, a top-10 recommendation list was created for each movie using cosine similarity. This process resulted in a comprehensive set of recommendation files, one for each model at each evaluated epoch. These files form the basis for the quantitative and qualitative analysis presented in the following chapter.

# Chapter 6

# Results and Analysis

This chapter presents the empirical results of the comprehensive evaluation conducted to answer the research questions posed in Chapter 1. The analysis is structured to systematically evaluate the performance of different recommendation approaches, beginning with a baseline model and systematically documenting the impact of each subsequent enhancement. The research questions guiding this analysis are:

1. *Which multimodal content features (visual, audio, textual) contribute most to determining program similarity in a content-based video recommendation system?*
2. *How do learned representations from a Siamese Neural Network compare to traditional, feature-engineered representations when used for similarity-based ranking?*
3. *Based on the literature and the challenges addressed in this work, what are the primary strengths and limitations of a purely content-based approach for video recommendation?*

The chapter follows a logical progression. First, the baseline system performance is established in Section 6.1, followed by the ablation study that quantifies individual modality contributions in Section 6.2. Based on the insights from the ablation study, Section 6.3 presents a comprehensive feature selection pipeline. Then, Section 6.4 demonstrates the strength of the content-based approach tested via domain-specific analysis. Moving to the end-to-end models, Section 6.5 transitions to the analysis of the four models, and Section 6.6 presents a qualitative case study for each model variant. Finally, the chapter concludes with a summary comparing the best-performing model from each stage in Section 6.7.

## 6.1 Performance of the Baseline Recommender System

The baseline model, which employs concatenated multimodal features with fixed cosine similarity, was evaluated to establish a quantitative benchmark against which all subsequent improvements could be measured.

### 6.1.1 Baseline System Configuration

The baseline system fuses features via simple concatenation of both high-level deep embeddings and a suite of low-level traditional features. The high-level components include DINOv2 for visual semantics, Wav2Vec2 for speech characteristics, and BERT for textual understanding.

These are complemented by traditional features such as colour histograms and MFCCs to capture stylistic attributes. This early fusion strategy results in a 2,304-dimensional feature vector for each movie. The similarity between items is then computed using the standard cosine similarity metric. Evaluation is performed using top-K recommendation metrics where $K = \{1, 3, 5, 10\}$.

### 6.1.2 Quantitative Results

The performance of the baseline system across standard top-k recommendation metrics is detailed in Table 6.1. The system achieved an **nDCG@10 of 0.0452** and a **Hit Rate@10 of 0.3970**. These results represent a reasonable, albeit modest, level of performance and serve as the primary benchmark for this study.

Table 6.1: Baseline System Performance Metrics

| **Metric** | **Performance** |
|---|---|
| Precision@1 | 0.0889 |
| Precision@3 | 0.0761 |
| Precision@5 | 0.0706 |
| Precision@10 | 0.0647 |
| Recall@1 | 0.0004 |
| Recall@3 | 0.0008 |
| Recall@5 | 0.0011 |
| Recall@10 | 0.0019 |
| Hit Rate@1 | 0.0889 |
| Hit Rate@3 | 0.1936 |
| Hit Rate@5 | 0.2675 |
| Hit Rate@10 | 0.3970 |
| nDCG@1 | 0.0567 |
| nDCG@3 | 0.0509 |
| nDCG@5 | 0.0482 |
| nDCG@10 | 0.0452 |
| MRR@1 | 0.0889 |
| MRR@3 | 0.1333 |
| MRR@5 | 0.1500 |
| MRR@10 | 0.1670 |

## 6.2 Ablation Study: Quantifying Modality Contributions

To understand the individual contribution of each modality to the baseline's feature-based approach, a systematic ablation study was conducted. The experiment involved training and evaluating three additional models, each with one modality (visual, audio, or text) removed.

### 6.2.1 Key Findings

The results of the ablation study, presented in Table 6.2, revealed a clear hierarchy of modal importance. To better illustrate the impact of removing each modality, the percentage change

in key performance metrics is visualised in Figure 6.1.

Table 6.2: Ablation Study: Impact of Modality Removal on Performance

| Metric | All Modalities | No Text | No Audio | No Visual |
|---|---|---|---|---|
| Precision@1 | 0.0889 | 0.0721 | 0.0827 | **0.0898** |
| Hit Rate@10 | 0.3970 | 0.3963 | 0.3727 | **0.4014** |
| nDCG@10 | 0.0452 | 0.0438 | 0.0418 | **0.0458** |
| MRR@10 | 0.1670 | 0.1530 | 0.1557 | **0.1689** |

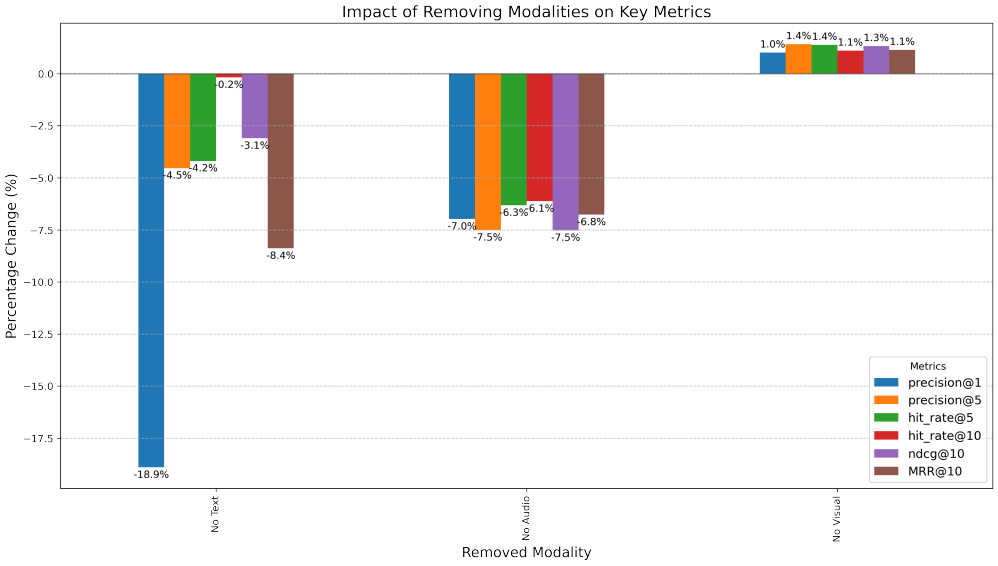



Figure 6.1: Percentage Change in Key Performance Metrics After Removing Each Modality. The chart clearly shows the dominant negative impact of removing text, the moderate negative impact of removing audio, and the surprising performance improvement when removing visual features.

The key findings are:

- **Textual Modality is Dominant:** As shown in Figure 6.1, removing textual features caused the most significant performance degradation across all metrics, with **Precision@1** dropping by 18.9%. This confirms that metadata and plot summaries are the most powerful signals for determining movie similarity in this configuration.
- **Audio Modality is a Positive Contributor:** The absence of audio features consistently degraded performance, notably impacting **nDCG@10** (-7.52%). This indicates that acoustic features provide valuable, non-redundant information.
- **Visual Modality is Problematic:** Counter-intuitively, removing the high-dimensional visual features *improved* performance across all metrics. The "No Visual" configuration became the new best-performing modular model. This strongly suggests that the high-dimensional visual embeddings, when naively concatenated, introduce more noise than signal.

This consistent hierarchy of performance is not limited to a single K value. As shown in Figure 6.2, the relative performance of the four configurations remains stable across the top-10 recommendation ranks for both **Precision** and **Hit Rate**. This confirms that the detrimental effect of noisy visual features and the positive contribution of textual features are fundamental to the model's behaviour, not just an artefact at a specific cut-off.

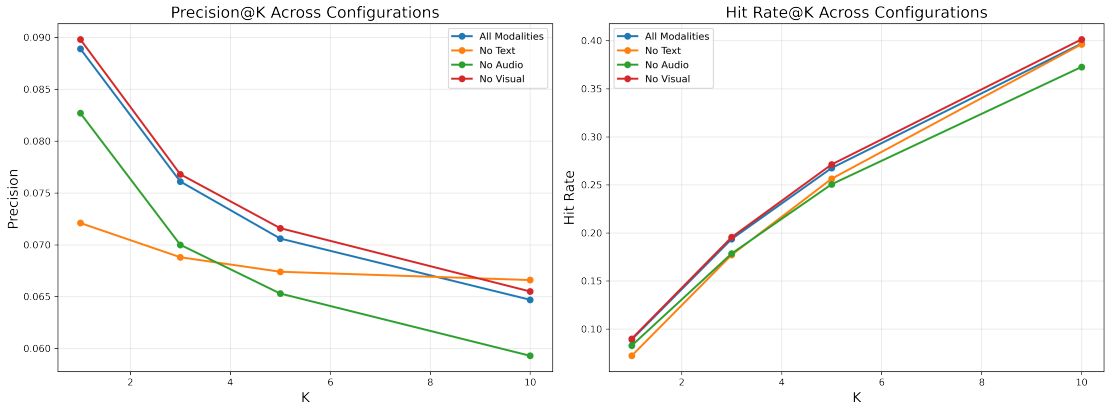



Figure 6.2: Performance of Precision@K (left) and Hit Rate@K (right) Across the Ablation Study Configurations.
The "No Visual" model consistently outperforms the others, while the "No Text" model consistently underperforms, demonstrating the robustness of these findings across different recommendation list lengths.

This last finding heavily motivated the subsequent feature engineering and selection experiments, as it became clear that simply including more data is not always beneficial; the data must be processed and integrated intelligently.

### 6.2.2 Follow-Up Investigation of Visual Features

The surprising performance degradation caused by visual features prompted further investigation. The initial hypothesis was that the DINOv2 embeddings, when simply concatenated, were introducing more noise than signal. This could be due to several factors:

1. Sub-optimal frame sampling that included non-representative or low-quality frames
2. The presence of redundant information across the 768 dimensions that overshadows the more subtle textual and aural cues
3. The inability of a simple cosine similarity metric to properly weigh and interpret such high-dimensional visual data.

The optimal approach may involve a combination of refined visual features with the existing text and audio features or potentially a weighted approach that emphasises text for top recommendations while incorporating audio features for diversity and overall quality.

This finding strongly motivated the need for improved feature engineering and, ultimately, end-to-end learning approaches capable of learning more relevant visual representations.

Initial attempts to fix the modality imbalance with weighted fusion strategies did not yield significant improvements (see **Appendix C**), motivating a deeper analysis of the features themselves.

## 6.3 Feature Selection: Improving Efficiency and Performance

The preceding ablation study revealed a clear hierarchy of modality importance. It confirmed the dominance of textual features and the value of audio cues. Most critically, however, it produced the counterintuitive finding that the high-dimensional visual modality, when naively concatenated, introduced more noise than signal, actively degrading recommendation performance.

This demonstrated that simply including more data is not a guarantee of a better model. The baseline system is evidently overwhelmed by irrelevant or redundant information, particularly from the visual domain. To move beyond this naive fusion and build a more robust, efficient, and ultimately more accurate recommender, it is necessary to identify and isolate the most predictive signals from the noise.

Therefore, a feature selection pipeline is now applied. This process serves a dual purpose. Firstly, from a performance perspective, it aims to construct a minimal set of features that maximises recommendation quality. Secondly, from an analytical perspective, it functions as the global model diagnostic discussed in Section 2.4, allowing it to "explain"the behaviour of the baseline model by quantifying which specific features it found most valuable. This analysis not only improved the baseline but also yielded crucial insights for the design of the end-to-end architecture.

### 6.3.1 Feature Reduction and Performance Impact

The selection process was remarkably effective, reducing the total feature count by **98.1%**, from an initial 2,304 features down to a compact set of 44. Figure 6.3 visually summarises this achievement. The left panel shows the drastic dimensionality reduction. In contrast, the right panel shows that this was achieved with a slight *increase*in **nDCG@10** performance, confirming that the removed features were indeed redundant or noisy.

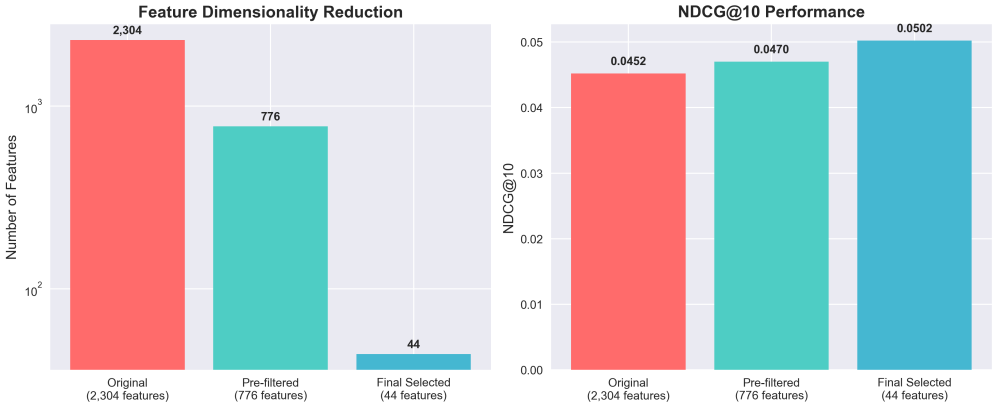



Figure 6.3: Impact of Feature Selection on Dimensionality and Performance.
The left plot (log scale) shows the drastic reduction in feature count from the original set to the final set. The right plot shows that this 98% reduction in features resulted in a performance improvement, demonstrating the effectiveness of the selection pipeline. In this context, each "feature" corresponds to a single dimension in the concatenated input vector.

The feature selection process achieved substantial dimensionality reduction across all modal-

ities. Table 6.3 summarises the reduction achieved for each modality.

Table 6.3: Summary of Feature Selection by Modality

| Modality | Original Features | After Pre-filtering | Final Selected | Reduction (%) |
|---|---|---|---|---|
| Visual | 768 | 171 | 4 | 99.5% |
| Textual | 768 | 538 | 34 | 95.6% |
| Audio | 768 | 67 | 6 | 99.2% |
| **Total** | **2,304** | **776** | **44** | **98.1%** |

The results demonstrate exceptionally high dimensionality reduction across all modalities, with the visual modality showing the most aggressive reduction (99.5%), followed by audio (99.2%) and textual features (95.6%). This substantial reduction suggests that the majority of extracted features obtained redundant or non-predictive information. The textual modality retained the highest number of features, suggesting its high information density, while the audio modality was the most compact.

### 6.3.2 Modality-Specific Importance

To understand which specific features drive recommendation quality, a detailed analysis was conducted using both Mutual Information and Random Forest Feature Importance. The following analysis reveals distinct patterns of importance for each modality.

#### Textual Features

The textual modality is dominated by explicit metadata, as shown in Figure 6.4. The feature `text_genres_encoded` is unequivocally the most important predictor, with importance scores that far exceed those of all others. This confirms that genre is the primary signal for content similarity.

Following genre, temporal and language metadata, such as `text_year_normalized` and `text_title_language_encoded`, are also highly significant. Crucially, several principal components from BERT embeddings also rank highly, validating that a deep semantic understanding of plots and titles provides valuable complementary information.

#### Audio Features

The analysis in Figure 6.5 reveals the clear superiority of high-level learned representations. The most important audio features are derived from the pre-trained VGGish model (e.g., `audio_vggish_embedding_pca_3`), which captures complex acoustic scenes.

While traditional metrics, such as MFCCs, show moderate relevance, they are substantially outperformed by deep learning-based features, confirming that learned representations are more effective at capturing the acoustic nuances that correlate with user preferences.

#### Visual Features

The visual analysis (Figure 6.6) demonstrates an overwhelming dominance of features derived from the DINOv2 vision transformer. Every top-ranking feature is a principal component of the DINOv2 embeddings (e.g., `visual_dinov2_features_pca_4`). This validates that for the

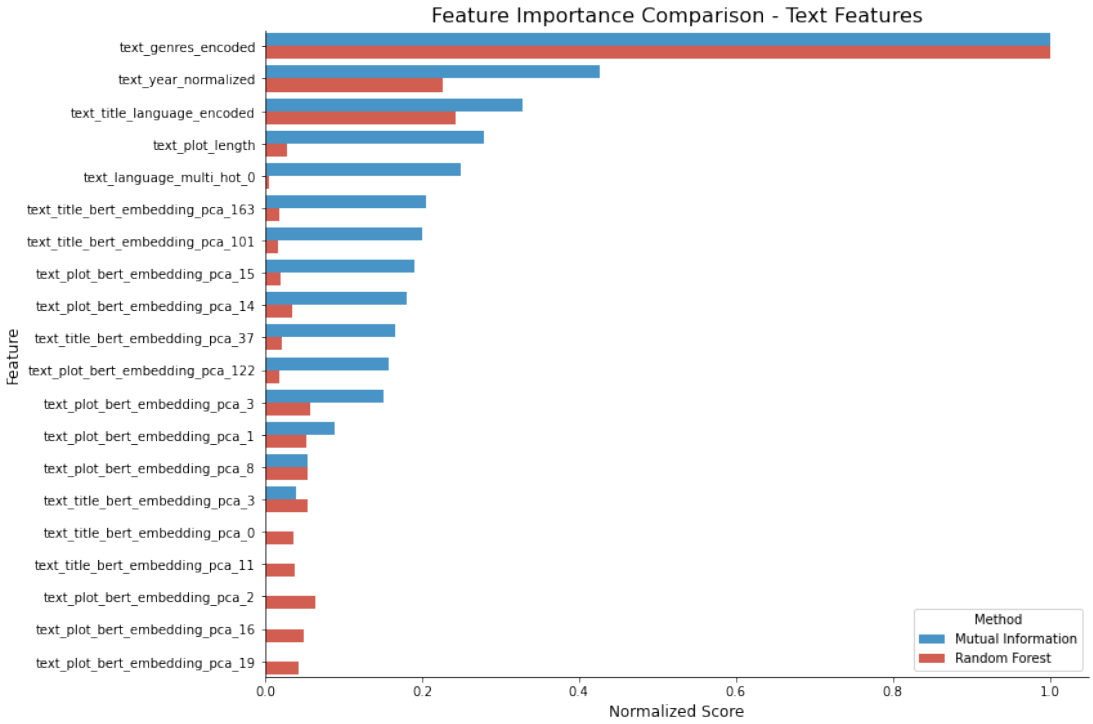



Figure 6.4: Comparison of Feature Importance Scores (MI vs. RFFI) for Textual Features. Features selected by MI only are shown in blue, RFFI only in red, and features selected by both methods in purple, indicating consensus across scoring methods.

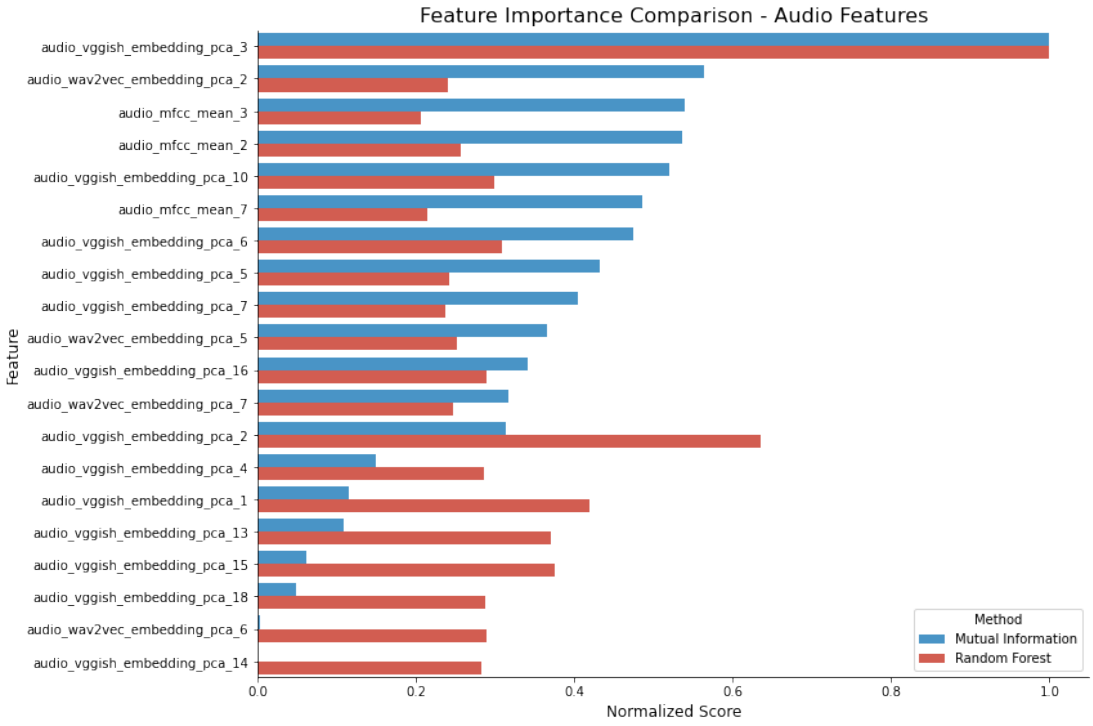



Figure 6.5: Comparison of Feature Importance Scores (MI vs. RFFI) for Audio Features. Features selected by MI only are shown in blue, RFFI only in red, and features selected by both methods in purple, indicating consensus across scoring methods.

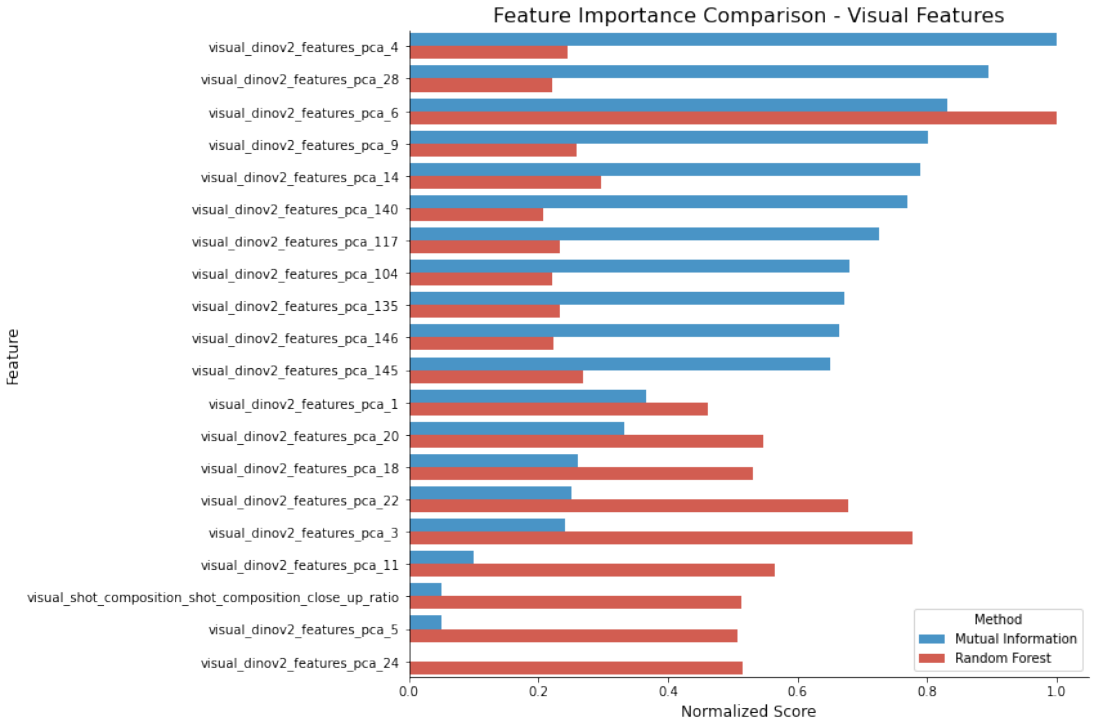



Figure 6.6: Comparison of Feature Importance Scores (MI vs. RFFI) for Visual Features. Features selected by MI only are shown in blue, RFFI only in red, and features selected by both methods in purple, indicating consensus across scoring methods.

baseline model, high-level semantic features are far more valuable than the low-level traditional computer vision features (e.g., colour, texture), none of which were retained after selection.

It is essential to note that, although this analysis informed the feature set for the improved baseline model, the low-level features were intentionally retained for the end-to-end model discussed later in this chapter. The feature selection process confirmed that a simple cosine-similarity-based model cannot effectively utilise raw visual data. However, as supported by the literature in Chapter 2, a deep neural network architecture is designed to learn its own relevant representations from such complex, high-dimensional inputs. Therefore, this analysis validates the feature set for the baseline approach and justifies a different input strategy for the more advanced E2E architecture.

### 6.3.3 Validating the Selection Strategy

The feature selection process relied on two key methodological choices: using a cumulative importance threshold and combining two different scoring methods. These choices were validated empirically.

First, the decision to use a threshold (e.g. 90% of total MI) rather than a fixed number of features was guided by analysing cumulative importance curves, as shown for the visual modality in Figure 6.7. The "elbow" in the plot clearly indicates the point of diminishing returns, where adding more features provides minimal new information. This data-driven approach ensures that the feature set is both compact and informative.

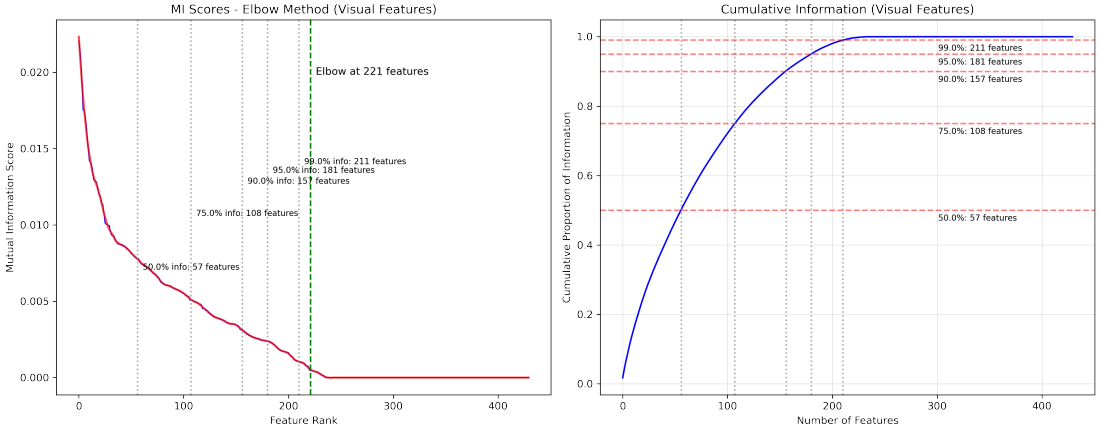



Figure 6.7: Cumulative Mutual Information for Visual Features. The plot illustrates that a small subset of features captures the vast majority of predictive information, validating the threshold-based selection strategy.

Second, to ensure the robustness of the selected features, the agreement between the MI and RFFI methods was analysed. Figure 6.8 compares the normalised scores for the textual modality. The analysis reveals a strong consensus on the most critical features (purple dots), such as `text_genre_encoded` and `text_year_normalized`. It also shows complementarity: RFFI identifies features valuable in tree-based combinations (red cluster), while MI captures unique statistical dependencies (blue cluster). By taking the union of features selected by both methods, the final set benefits from the strengths of each, resulting in a more robust and reliable set of predictors.

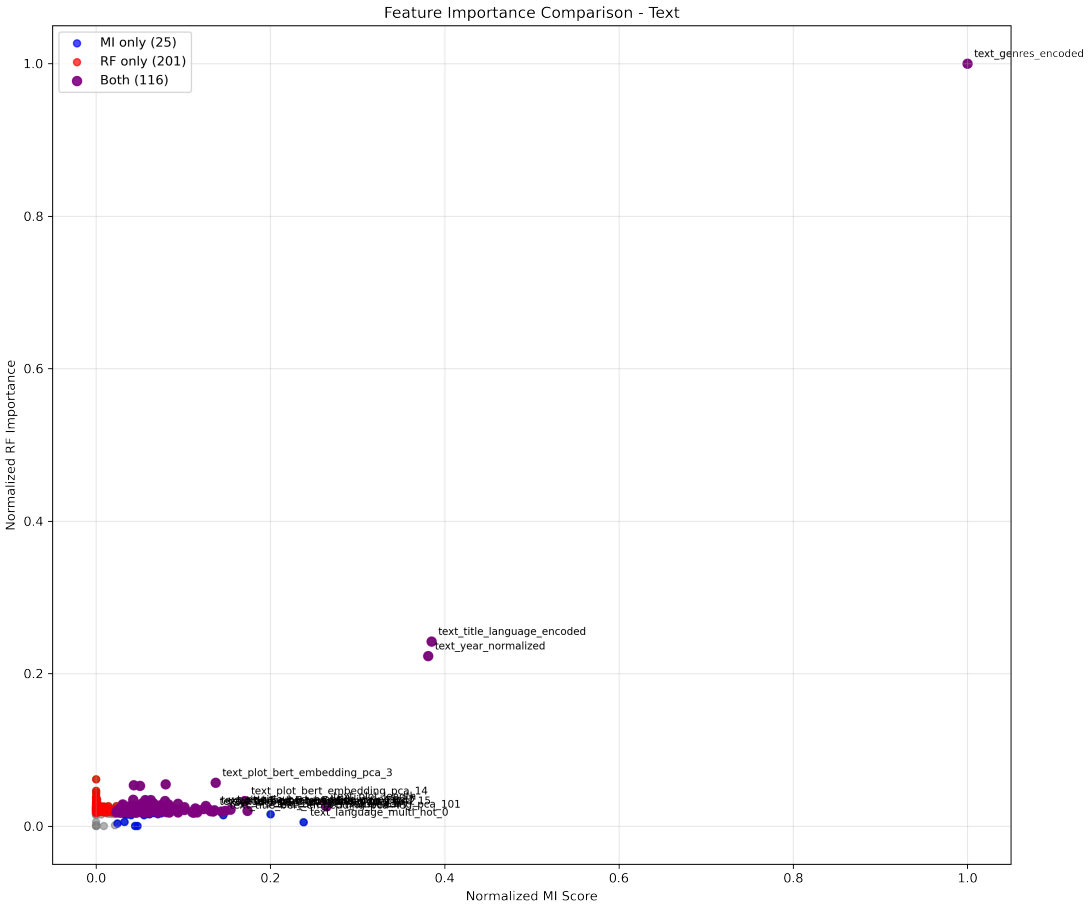



Figure 6.8: Comparison of Normalised Feature Importance Scores from Mutual Information and Random Forest for the Textual Modality.
The plot shows features selected by MI only (blue), RF only (red), or both methods (purple).

### 6.3.4 Performance Impact of Feature Selection

With a compact and robust set of 44 features identified, the final step was to evaluate its impact on recommendation performance. Table 6.4 compares the performance of the model using the full pre-filtered feature set (776 features) against the final selected set (44 features).

Table 6.4: Feature Selection Impact on Performance

| Method | P@1 | P@10 | HR@10 | nDCG@10 | MRR@10 |
|---|---|---|---|---|---|
| FS Full | 0.0902 | **0.0739** | **0.4201** | **0.0502** | **0.1754** |
| FS 90% | **0.0902** | 0.0734 | 0.4191 | 0.0498 | 0.1747 |
| FS 80% | 0.0883 | 0.0725 | 0.4149 | 0.0492 | 0.1723 |
| FS Elbow | 0.0892 | 0.0732 | 0.4181 | 0.0496 | 0.1737 |

The results demonstrate that feature selection provides consistent improvements across all evaluation metrics:

The results are striking. The model with only 44 features achieves performance that is nearly identical to the model using 776 features. For instance, **Precision@1** slightly improves, while **nDCG@10** sees a negligible decrease of only 0.8%. This demonstrates that the feature selection pipeline successfully eliminated 94% of the features (732 of 776) with almost no loss in recommendation quality. This massive gain in efficiency (faster training, smaller model size) for a minimal performance trade-off strongly validates the feature selection process. It confirms that a small, carefully curated set of features is far more effective and practical than a large, noisy one.

### 6.3.5 Feature Selection Insights for End-to-End Model Design

The feature selection analysis provided crucial insights that directly informed the design of the subsequent end-to-end Siamese neural network models:

1. **Genre Dominance:** The overwhelming importance of genre information informed the decision to use a genre-based sampling strategy for generating positive and negative training pairs for the Siamese networks.

2. **Modality Hierarchy:** The clear ranking of information density (Textual > Audio > Visual) guided the architectural design, suggesting that the fusion component of the E2E model must learn to heavily weigh textual cues.

3. **Representation Learning:** The superior performance of deep learning embeddings (BERT, VGGish, DINOv2) over traditional features validated the decision to focus on learned representations in end-to-end models.

4. **Dimensionality Efficiency:** The analysis confirmed that a compact representation is highly effective, motivating the design of an E2E model that projects the fused multimodal features into a relatively low-dimensional final embedding space (e.g., 128 dimensions).

## 6.4 Domain-Specific Analysis: The Power of Context

To test the hypothesis that content-based systems excel within well-defined content domains, an experiment was conducted on a curated subset of 400 movies (200 action and 200 Disney/animation movies). The results, shown in Figure 6.9 and detailed in Table 6.5, are striking.

Table 6.5: Performance on Domain-Specific Dataset (400 movies)

| Configuration | P@1 | P@10 | HR@10 | nDCG@10 | MRR@10 |
|---|---|---|---|---|---|
| Enhanced-400 | 0.1148 | 0.1055 | 0.5402 | 0.0886 | 0.2292 |
| Enhanced-400-PCA | 0.1218 | 0.1056 | 0.5448 | 0.0896 | 0.2348 |
| Enhanced-400-v3 | **0.1234** | **0.1064** | **0.5458** | **0.0898** | **0.2351** |

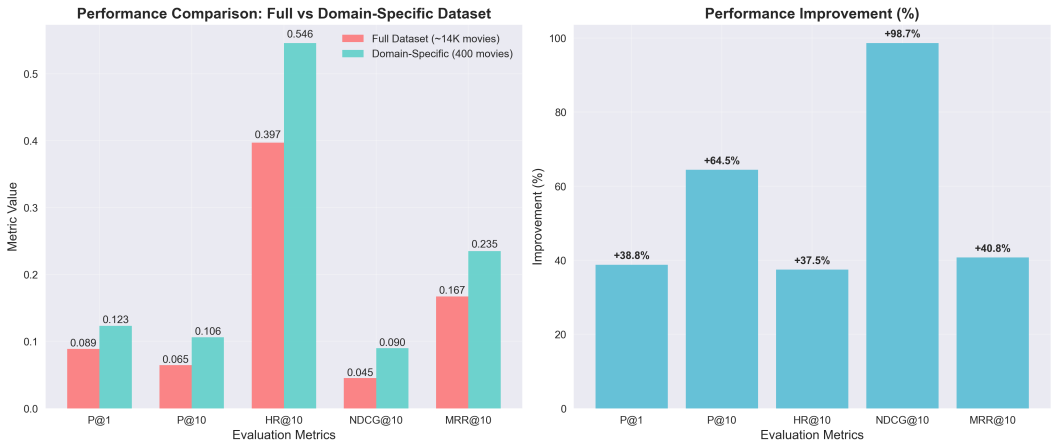



Figure 6.9: Performance Comparison Between the Full Dataset and a Curated Domain-Specific Subset.

The left plot shows absolute metric values, while the right plot highlights the substantial percentage improvement across all metrics, with **nDCG@10 increasing by 98.7%**.

When operating on a thematically coherent dataset, the model's performance improved dramatically. **Precision@1** rose from 0.089 to 0.123 (+38.8%), and **nDCG@10** nearly doubled from 0.045 to 0.090 (+98.7%). This demonstrates a key strength of content-based methods: they can be highly effective for niche or specialised recommendation tasks where content similarity is a strong proxy for user preference.

## 6.5 Performance Analysis of the End-to-End Models

To overcome the limitations of the feature-based baseline, a 2x2 factorial experiment was conducted to train the four end-to-end models. Each is designed to learn a semantic embedding space directly from data. These models varied by loss function(Pairwise Contrastive vs. Triplet) and distance metric (Cosine vs. Euclidean). The primary evaluation of these models involves their performance on the final ranking task. However, a comprehensive analysis requires a two-step approach. First, performance on the ranking task is assessed. Second, a diagnostic analysis of their underlying learning behaviour is performed to interpret the results.

### 6.5.1 The Ranking Performance Anomaly

The primary goal of the E2E models is to generate embeddings that enhance the quality of top-K recommendations. The performance of all four models, evaluated at various training epochs, is presented in Figure 6.10. The expectation was that as the models trained and their loss decreased, the ranking metrics such as **nDCG@10** and **Precision@10** would steadily improve and surpass the baselines.

The results, however, reveal a striking and counterintuitive anomaly. As shown in Figure 6.10, none of the E2E models consistently outperform the best-performing baselines (the dashed grey lines). More critically, after reaching an early peak in performance (indicated by the star markers), all four models exhibit a consistent degradation in ranking quality with continued training. For example, the `E2E-Triplet-Euclidean` model (purple line) achieves its best **nDCG@10**

around epoch 10, after which its performance steadily declines for the remaining 90 epochs. This trend is consistent across all metrics and all four model architectures.

This divergence between continued training and declining ranking performance raises a critical question: What are the models actually learning, and why does this "better" learning not translate to better global recommendations? To answer this, further investigation beyond the final ranking metrics must be done along with a diagnostic of the model's behaviour on their direct training objective.

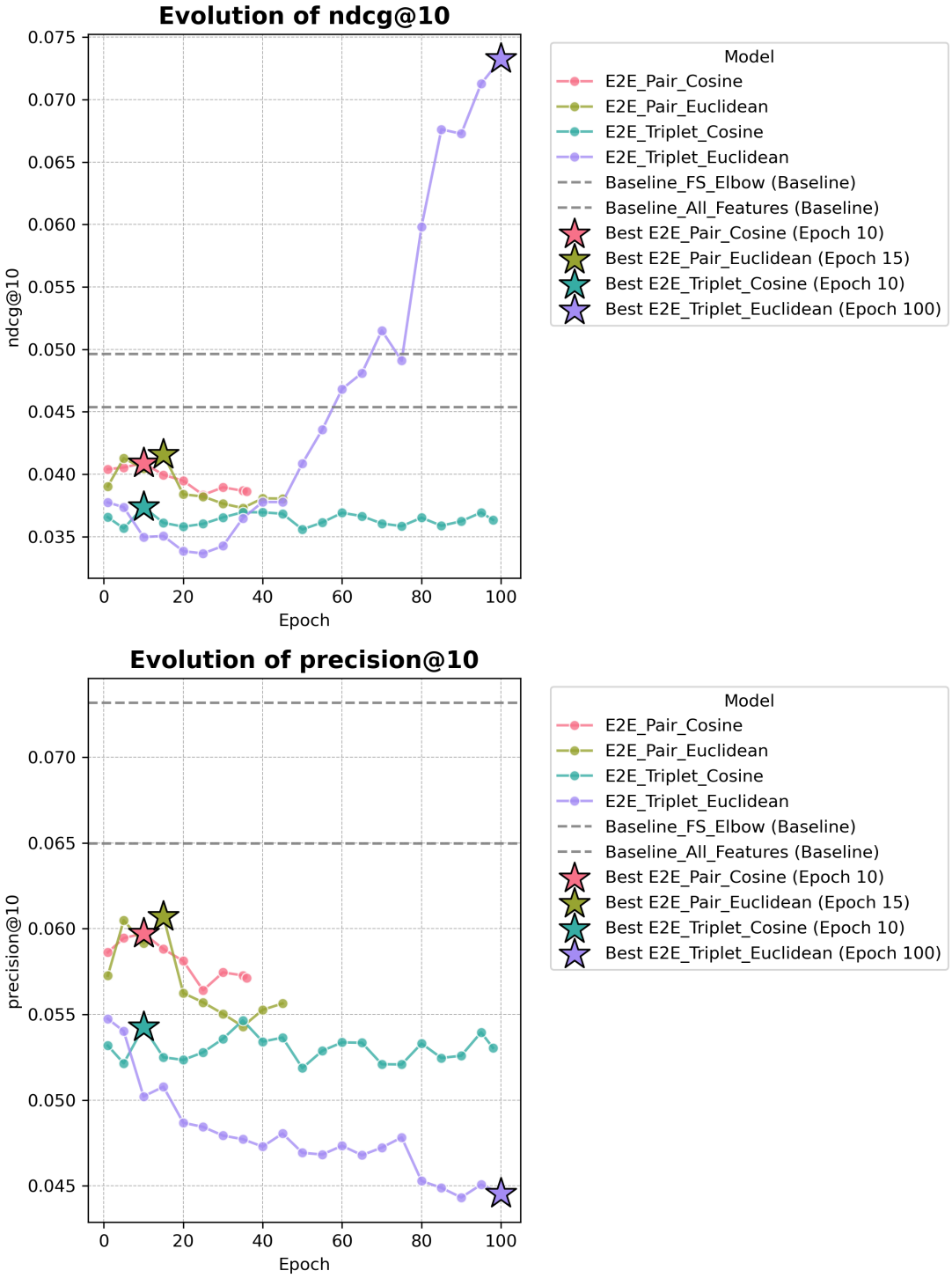



Figure 6.10: Comparison of Top-K Ranking Performance for the Four E2E Model Variants Across Training Epochs.

The dashed lines represent the two strongest baseline models. The star markers indicate the best-performing epoch for each respective E2E model. All models show a degradation in ranking performance after an initial peak.

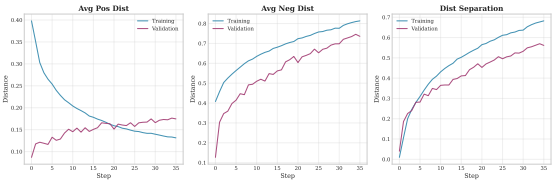
### 6.5.2 Evaluating the Primary Training Objective

The ranking anomaly suggests a disconnect between the training task and the evaluation task. The models were trained not as rankers but as metric learners or classifiers. Their task was to pull "similar" items together and push "dissimilar" items apart in the embedding space. A diagnostic analysis confirms they actually learned this task.

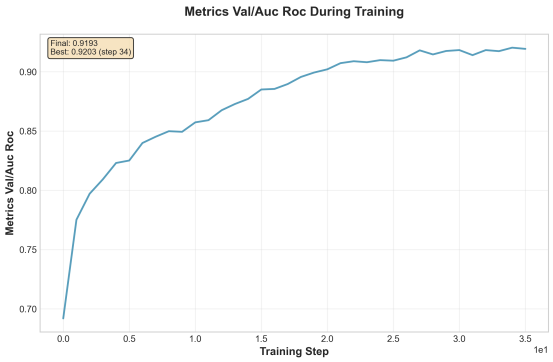
The `E2E-Pair-Cosine` model serves as a clear case study. The training and validation curves (Figure 6.11a) show a healthy, stale learning process. This successful learning is further evidenced by the evolution of the embedding spaces, shown in Figure 6.11b. The successful separation of positive and negative pairs in the embedding space demonstrates success on the classification task, which, paradoxically, leads to a decline in ranking performance due to metric collapse.

As training progresses, the average cosine distance between positive pairs decreases while the distances between negative pairs increase, leading to a robust and growing separation. The model's performance as a binary classifier, measured by the validation AUC-ROC (Figure 6.11c), steadily improves, peaking at epoch 34 with a score of **0.9203**. A confusion matrix from this peak (Figure 6.11d) confirms its high proficiency, correctly classifying the vast majority of both similar and dissimilar pairs.

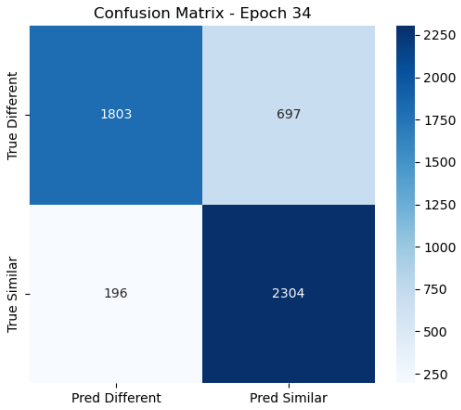

(a) Training & Validation Loss




(b) Evolution of Embedding Space Distances



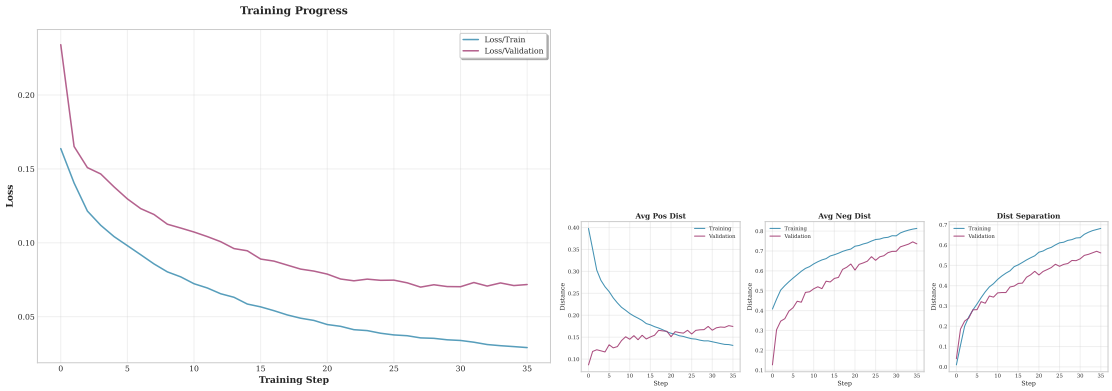

(c) Validation AUC-ROC




(d) Confusion Matrix at Best Epoch (34)

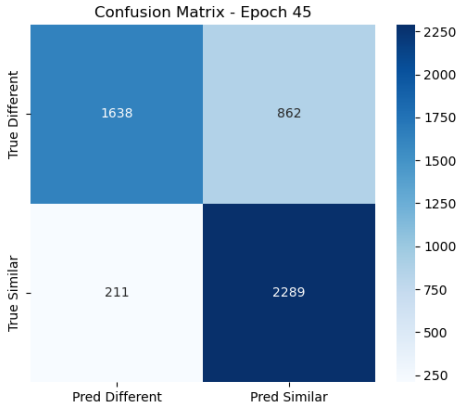
Figure 6.11: Diagnostic Analysis of the E2E-Pair-Cosine Model.
(a) Stable loss reduction. (b) Successful separation of positive and negative pairs in the embedding space. (c) Consistently improving classification performance on the validation set. (d) Strong classification results at its peak.

The `E2E-Pair-Euclidean` model demonstrated a similar pattern of success on the classification task. The training and validation metrics (Figure 6.12a) and the clear separation of embedding distances (Figure 6.12b) mirror the behaviour of the cosine variant. Its peak classification performance, confirmed by the confusion matrix at epoch 45 (Figure 6.12c), underscores its effectiveness as a similarity classifier.




(a) Training & Validation Metrics

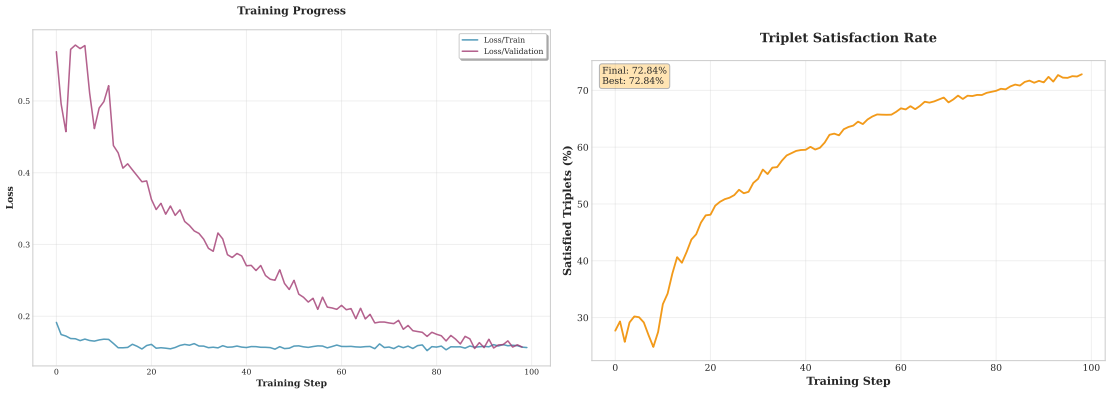
(b) Embedding Space Distances




(c) Confusion Matrix at Best Epoch (45)

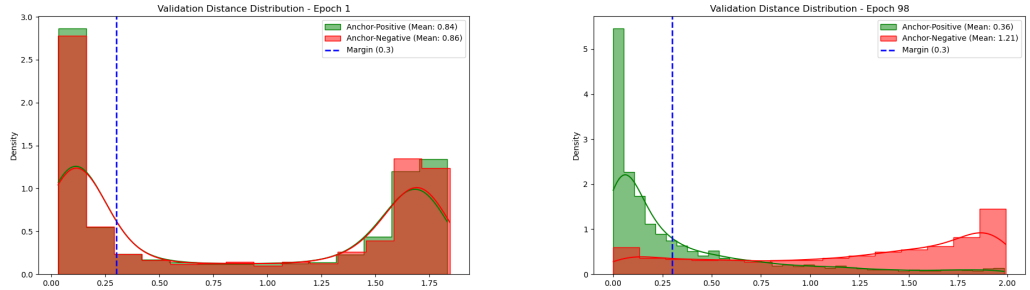
Figure 6.12: Diagnostic Analysis of the E2E-Pair-Euclidean Model , confirming successful learning of the classification task.

The models trained with Triplet Loss also mastered their learning objective. The loss curves for the `E2E-Triplet-Cosine` model (Figure 6.13a) show more volatility. This behaviour is a common characteristic of training with online semi-hard negative mining. As the model improves, it is fed more challenging triplets, causing temporary spikes in the loss. Despite this, the overall trend is downward, indicating successful learning. The key metric for triplet models, the percentage of "satisfied" triplets (where the anchor-positive distance is smaller than the anchor-negative distance by a margin), steadily increases to over 72% (Figure 6.13b). The most compelling evidence is the distribution of distances (Figure 6.13c and Figure 6.13d). At the start of training, the distributions for anchor-positive (green) and anchor-negative (red) distances are almost completely overlapping. By the end of training, the model has learned to pull the positive pairs into a tight cluster with low distance, clearly separating them from the negative pairs.




(a) Training & Validation Loss

(b) Triplet Satisfaction Rate




(c) Distance Distribution at Epoch 1

(d) Distance Distribution at Epoch 98

Figure 6.13: Diagnostic Analysis of the E2E-Triplet-Cosine Model.
The model successfully learns to satisfy the triplet constraint, clearly separating the distance distributions of positive and negative pairs over time.

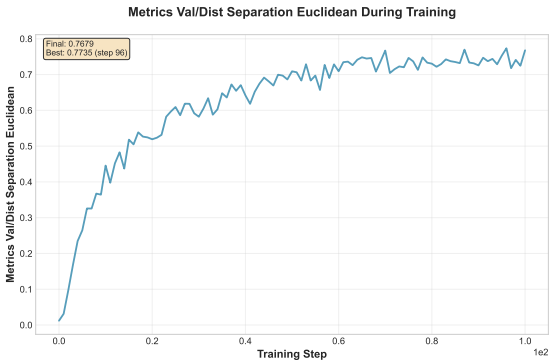
The models trained with the Triplet Loss exhibit a different learning dynamic compared to the pairwise models. As seen in the loss curve for the `E2E-Triplet-Cosine` model (Figure 6.13a), the validation loss is more volatile and decreases more slowly than in the pairwise counterparts. This behaviour is characteristic of and expected when using an online semi-hard negative mining strategy.

In this training regime, the model is constantly challenged. As it learns to correctly separate easy triplets, the mining algorithm actively searches for more difficult examples (semi-hard negatives) within each batch to fit into the model in the next step. This creates a dynamic curriculum where the task difficulty effectively increases as the model's performance improves. Consequently, the loss does not drop as smoothly or as low as the model is perpetually being forced to earn finer-grained distinctions at the edge of its current capabilities. The ultimate proof of learning is, therefore, not a low loss value but rather the steadily increasing triplet satisfaction rate (Figure 6.13b), which confirms that the model is successfully meeting the learning objective on progressively harder examples.

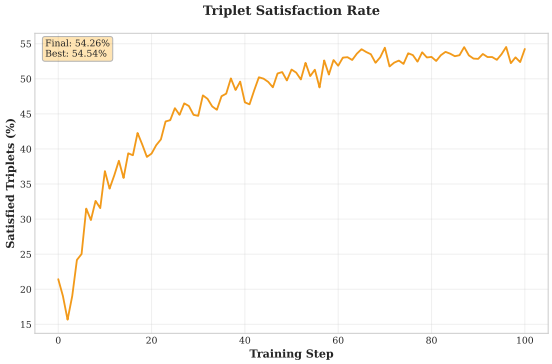
This dynamic difficulty also explains the behaviour of the training loss curve (the blue line in Figure 6.13a). Unlike in standard classification, the training loss does not trend towards zero; instead, it remains relatively steady, as the model is continuously supplied with challenging "semi-hard" examples that keep the loss signal active and prevent it from vanishing.

Finally, the `E2E-Triplet-Euclidean` model confirms this consistent finding across all ar-

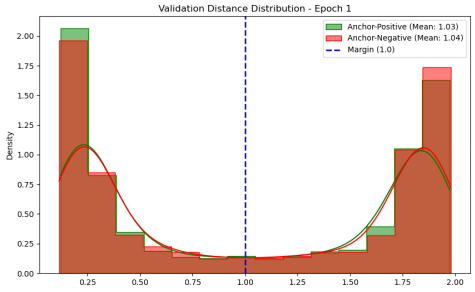
chitectures. It effectively learns to increase the distance separation (Figure 6.14a) and triplet satisfaction rate (Figure 6.14b). The evolution of its distance distribution (Figure 6.14c and Figure 6.14d) provides a stark visual confirmation of successful metric learning, transforming an undifferentiated embedding space into a highly structured one.



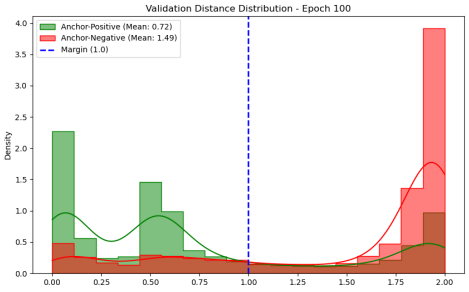

(a) Distance Separation




(b) Triplet Satisfaction Rate




(c) Distance Distribution at Epoch 1




(d) Distance Distribution at Epoch 100

Figure 6.14: Diagnostic Analysis of the E2E-Triplet-Euclidean Model.
Like its cosine counterpart, it effectively separates the positive and negative distance distributions, proving its success on the metric learning task.

### 6.5.3 Interpreting E2E Model Performance: Classification vs. Ranking

The diagnostic analysis confirms that the models are not failing; they are succeeding at the precise task for which they were designed. The ranking anomaly is, therefore, a direct consequence of this success. This resolves the apparent paradox: the model can become a better classifier while simultaneously becoming a worse ranker. As the model's loss decreases, it learns to map all "similar" items into an extremely tight cluster and all "dissimilar" items far away. While this leads to high classification accuracy (as seen in Figure 6.11d), the relative distances within the "similar" cluster become almost meaningless, degrading the fine-grained ordering required for high-quality ranking.

This insight reframes the role of the E2E models. They are not one-short rankers. They are highly effective thematic filters or candidate generators. Their true strength, as hinted at by the domain-specific analysis in Section 6.4, lies in their ability to identify a small, high-quality

subset of thematically similar candidates from the entire catalogue of items. The poor global ranking metrics arise because the model, in later epochs, becomes over-specialised. It learns to distinguish between highly similar items based on learned details, which can cause it to rank a less relevant but thematically identical item higher than a more broadly relevant one, thereby degrading the overall quality of the Top-K list.

This finding suggests that the optimal use of these E2E models is as the first stage in a two-stage recommendation pipeline: first, use the learned embeddings for efficient candidate retrieval (filtering), and second, apply a lightweight model to re-rank that small candidate set (ranking). While the implementation of such a two-stage system is beyond the scope of this thesis, this work has successfully developed and validated a powerful component for the first stage.

## 6.6 Qualitative Analysis: A Case Study on Recommendation Content

While quantitative metrics evaluate the overall ranking accuracy, they do not reveal the character of the recommendations. To understand the nature of the similarity learned by each end-to-end model, a qualitative case study was conducted using embeddings from the best-performing epoch of all four models. Two distinct and well-known query films were selected as illustrative examples to probe the models' learned semantics rather than being drawn randomly from the test set. *Toy Story* (1995), a very popular family animated film, and *Fight Club* (1999), a cult-classic psychological thriller. For each query, the top 200 most similar films were retrieved based on cosine similarity within the learned embedding space, creating candidate sets that represent the thematic clusters generated by each model. The thematic coherence and relevance of these recommendation clusters were then analysed to understand the learned similarity patterns of each model.

### 6.6.1 Case Study: Toy Story

*Toy Story* (Genres: Adventure, Animation, Children, Comedy, Fantasy) is a quintessential film. A high-quality recommendation list should reflect these themes, ideally including other well-known animated or children's comedies.

The analysis of the recommendation lists reveals a stark difference in quality between the models trained with cosine-based versus Euclidean-based distance metrics. As summarised in Table 6.6, the cosine models produce far more relevant recommendations.

The `E2E-Triplet-Cosinemodel`, in particular, demonstrates a strong grasp of the query's essence. Its recommendation list includes highly relevant titles such as *Rugrats in Paris: The Movie* and *A Charlie Brown Christmas*, as well as major animated features like *Mulan II* and *Monsters, Inc.*. The `E2E-Pair-Cosine` model performs similarly well, recommending thematically appropriate films like *Muppet Movie, Hoodwinked!*, and *Garfield: A Tail of Two Kitties*.

In stark contrast, the Euclidean models fail to capture this thematic coherence. The `E2E-Pair-Euclidean` model's list drifts towards general live-action comedies (*The Waterboy, Grease 2*) and loses the core "animation" and "children" aspects. The performance of the `E2E-Triplet-Euclidean` model is even poorer, producing a list of thematically unrelated action and horror films, such as *Krull, Rambo: First Blood Part II*, and *Zombie Apocalypse*, which have no discernible relevance to *Toy Story*.

Table 6.6: Qualitative Comparison of Top Recommendations for Toy Story (1995)

| Model | Recommendations |
|---|---|
| **Triplet-Cosine** | **Top 5:** Thieves (Voleurs, Les) (1996); Legend of the Boneknapper Dragon (2010); Rugrats in Paris: The Movie (2000); Catlow (1971); A Charlie Brown Christmas (1965) |
| | **Notable relevant in top 200:** Mulan II, Monsters Inc., Cars, Happy Feet, Curious George |
| **Pair-Cosine** | **Top 5:** Fiendish Plot of Dr. Fu Manchu (1980); One Crazy Summer (1986); The Muppet Movie (1979); Club Paradise (1986); Hoodwinked! (2005) |
| | **Notable relevant in top 200:** Garfield: A Tail of Two Kitties, Muppets from Space, Madagascar, Despicable Me 2 |
| **Pair-Euclidean** | **Top 5:** Won Ton Ton: The Dog Who Saved Hollywood (1976); Slappy and the Stinkers (1998); 3 Ninjas Kick Back (1994); The Fiendish Plot of Dr. Fu Manchu (1980); A Fairly Odd Christmas (2012) |
| | **Notable relevant in top 200:** The Muppet Movie, Air Bud, Honey I Blew Up the Kid, Despicable Me 2 |
| **Triplet-Euclidean** | **Top 5:** Krull (1983); Philadelphia Experiment II (1993); Earth (Tierra) (1996); Rambo: First Blood Part II (1985); The Starfighters (1964) |
| | **Notable relevant in top 200:** *None found* |

This case study suggests that for a query like *Toy Story*, the angular similarity captured by cosine distance is more effective at identifying thematic relevance than the magnitude-sensitive Euclidean distance. The Euclidean models appear to latch onto lower-level visual or audio features (e.g., colour brightness, action sequence pacing) that are not genre-specific, leading to irrelevant recommendations.

### 6.6.2 Case Study: Fight Club

*Fight Club* (Genres: Action, Crime, Drama, Thriller) is a dark, psychological thriller renowned for its anti-establishment themes and distinctive visual style. Relevant recommendations would include other crime thrillers, psychological dramas, or cult films with similar tones and themes.

The qualitative results for *Fight Club*, presented in Table 6.7 reinforce the findings from the *Toy Story* analysis. The cosine-based models again prove superior at capturing thematic intent.

The `E2E-Pair-Cosine` model generates a list dominated by thematically appropriate crime and action thrillers, including *Tomorrow Never Dies*, *The French Connection*, and *Extreme Measures*. The `E2E-Triplet-Cosine` model produces a similarly relevant list, featuring *Escape Plan* and *Stone Cold*. While neither model recommends canonical psychological thrillers like *Memento* or *American Psycho*[1] in their top ranks, the overall genre and tonal consistency are high.

The Euclidean models, however, struggle. The `E2E-Pair-Euclidean` model's recommendations include the fantasy-comedy *Beetlejuice*. The `E2E-Triplet-Euclidean` model produces the most incoherent list, including the documentary *Room 237* and the musical *Battle of the Year*. This demonstrates a failure to grasp the core "dark thriller" identity of the query movie.

[1]Notably, both *Memento* (2000) and *American Psycho* (2000) are present in the final dataset used for recommendation. The model's failure to rank them highly suggests it has learned a specific facet of *Fight Club*'s similarity—perhaps focusing more on its visual grit and anti-establishment action elements rather than its more complex psychological or non-linear narrative structures, which are central to the other two films.

Table 6.7: Qualitative Comparison of Top Recommendations for Fight Club (1999)

| Model | Recommendations |
|---|---|
| **Triplet-Cosine** | **Top 5:** Escape Plan (2013); Kill Me Later (2001); Trancers III (1992); Love, etc. (1996); Airspeed (1999) |
| | **Notable relevant in top 200:** American History X, Kill Your Darlings, United 93, 15 Minutes |
| **Pair-Cosine** | **Top 5:** Tomorrow Never Dies (1997); Autopsy (Macchie Solari) (1975); Airspeed (1999); Best Seller (1987); The French Connection (1971) |
| | **Notable relevant in top 200:** Casino (1995), Fast and the Furious: Tokyo Drift, Bookies (2003) |
| **Pair-Euclidean** | **Top 5:** Airspeed (1999); Wolf (1994); Antitrust (2001); Love, etc. (1996); Dead Man on Campus (1998) |
| | **Notable relevant in top 200:** Girlfight (2000), Deep Blue Sea (1999), Mr. & Mrs. Smith (2005) |
| **Triplet-Euclidean** | **Top 5:** Electrick Children (2012); The Astronaut's Wife (1999); Boy Wonder (2010); Faust (1926); Shocker (1989) |
| | **Notable relevant in top 200:** Brick (2005), World War Z (2013), Elektra (2005) |

### 6.6.3 Overall Interpretation of the Case Study

The qualitative analysis strongly supports the quantitative findings while revealing important nuances that numbers alone cannot capture. Most notably, it demonstrates the clear superiority of cosine-based models in generating recommendations that are thematically coherent. Across both light-hearted animated films and dark psychological thrillers, the `Triplet-Cosine` and `Pair-Cosine` models consistently identified appropriate candidates that aligned with human expectations.

Perhaps more intriguingly, the analysis suggests that while the `Pair-Euclidean` model achieved slightly higher offline ranking scores, the cosine-based variants actually learn embedding spaces that better reflect human-perceived similarity. This observation highlights a well-known challenge in recommendation systems: offline metrics derived from historical co-ratings often fail to capture the complete picture of recommendation quality.

The qualitative evaluation reveals two key insights. First, it corroborates the quantitative results by showing that cosine-based models produce more thematically relevant recommendations than their Euclidean counterparts. However, second, and perhaps more significant, it illuminates the nature of learned similarity itself. These models are not simply performing text-based genre matching. Instead, they appear to identify a deeper, multimodal "vibe" that is extracted from the combination of visual, audio, and textual features.

For cosine models, this learned vibe aligns remarkably well with human perception of thematic similarity. The Euclidean models, however, seem to capture more abstract or low-level features (things like high-contrast lighting patterns, editing rhythms, or specific audio frequencies) that do not necessarily correspond to human-perceived genres. This leads to recommendations that are often unexpected and, frankly, irrelevant from a user perspective.

A plausible hypothesis for this divergence is that Euclidean distance, being sensitive to vector magnitude, may be disproportionately influenced by low-level, non-semantic features that were not L2-normalised prior to the final fusion layer. For instance, high motion intensity in action sequences and rapid scene changes in animated films could result in similar high-magnitude values for temporal features. Likewise, high average audio energy or specific spectral characteristics

could create an accidental "similarity" in the audio domain. The Euclidean metric, unlike the angle-based cosine similarity, would latch onto these coincidental magnitude alignments, leading it to conclude that *Rambo* and *Toy Story* are similar based on their shared audiovisual dynamism, completely ignoring the vast thematic and narrative differences.

These findings validate the superiority of the cosine-based approach for the specific recommendation task while also highlighting the complex relationship between computational similarity measures and human judgment.

The qualitative analysis provides strong, tangible support for the quantitative findings while also adding a crucial layer of nuance.

## 6.7 Summary of Research Progression and Key Findings

This chapter has detailed a systematic journey from a baseline model to a series of increasingly sophisticated enhancements. To synthesise these results and provide a holistic view of the research progression, Table 6.8 summarises the performance of the best model from each major experimental phase. The "best epoch" for each end-to-end model is determined by its peak **Hit_Rate@10** score, a reliable measure of its ability to find at least one relevant item in the top-K recommendations.

Table 6.8: Comprehensive Performance Comparison of Key Models

| System Configuration | Epoch | P@10 | HR@10 | nDCG@10 | MRR@10 |
|---|---|---|---|---|---|
| **Baseline Models** | | | | | |
| Baseline (All Features) | — | 0.0650 | 0.3986 | 0.0454 | 0.1673 |
| Best Ablation (No Visual) | — | 0.0655 | 0.4014 | 0.0458 | 0.1689 |
| **After Feature Selection (Overall Best)** | — | **0.0739** | **0.4201** | **0.0502** | **0.1754** |
| **End-to-End Models (at Best Epoch by HR@10)** | | | | | |
| E2E-Pair-Euclidean (Best E2E) | 15 | 0.0607 | 0.3819 | 0.0416 | 0.1433 |
| E2E-Pair-Cosine | 1 | 0.0586 | 0.3780 | 0.0404 | 0.1421 |
| E2E-Triplet-Cosine | 1 | 0.0532 | 0.3490 | 0.0366 | 0.1293 |
| E2E-Triplet-Euclidean | 1 | 0.0547 | 0.3574 | 0.0377* | 0.1319 |
| *For comparison:* | | | | | |
| Domain-Specific (FS) | — | 0.1064 | 0.5458 | 0.0898 | 0.2351 |

*nDCG for Triplet-Euclidean is anomalous; value from best HR@10 epoch shown for consistency.

The evolution of the main performance metrics throughout this research journey is visualised in Figure 6.15. For clarity, only **Precision@1**, **Hit Rate@10**, and **nDCG@10** are shown, as they offer a balanced view of recommendation quality by capturing top-rank accuracy, item discovery, and ranking fidelity, respectively. The figure clearly illustrates the incremental gains from feature selection and confirms that the feature-engineered baseline remains the top-performing model on this general-purpose ranking task.

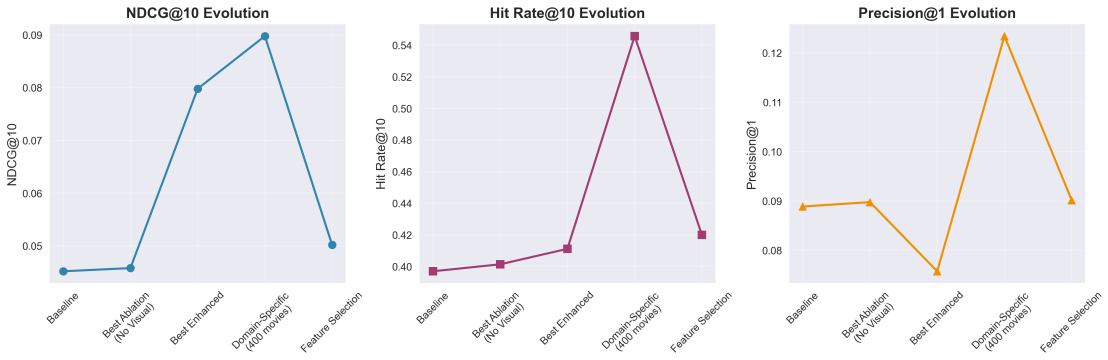



Figure 6.15: Evolution of Key Performance Metrics Throughout the Research Process. The plots track Precision@1, Hit Rate@10, and nDCG@10 from the initial Baseline to the final, optimised models, visualising the impact of each major methodological decision.

The progression detailed just above leads to several key conclusions that directly address the research questions:

1. **Naive Multimodal Fusion is Ineffective.** The initial baseline model, which simply concatenated all features, established a modest performance benchmark. The ablation study immediately revealed its core weakness: the high-dimensional visual modality introduced significant noise, and removing it led to a performance increase. This answers the first part of RQ1, showing that not all modalities contribute positively without careful integration.

2. **Text is the Most Powerful Modality, but Audio Adds Value.** The ablation study and subsequent feature importance analysis consistently showed that textual features (especially genre and semantic plot embeddings) are the strongest predictors of similarity. Audio features provide a secondary but valuable, non-redundant signal. This provides a clear hierarchy and answers the remainder of RQ1

3. **Feature Selection is Paramount for Fixed-Metric Systems.** The feature selection pipeline was remarkably successful, reducing the feature space by 98.1% while slightly improving recommendation quality. This demonstrates the critical importance of feature curation and shows that a compact, well-chosen set of features is superior to a large, noisy one for a system that uses a fixed similarity metric, such as cosine.

4. **A Trade-off Between Quantitative and Qualitative Performance:** The central comparison reveals a fascinating trade-off. While the `E2E-Pair-Euclidean` model achieved the highest score among the E2E models on the offline ranking metrics (**HR@10** of 0.3819), the qualitative case study showed that the `E2E-Triplet-Cosine` model produced significantly more thematically relevant and coherent recommendations. This suggests that the objective function derived from user co-ratings may not perfectly align with human notions of thematic similarity. Ultimately, however, none of the E2E models managed to surpass the simpler, feature-engineered baseline on the quantitative ranking task, demonstrating that for this dataset, a complex learned metric was not superior to a fixed metric on a carefully curated feature set. This directly answers RQ2 with a nuanced conclusion.

5. **Content-Based Systems Excel in Niche Contexts.** The domain-specific analysis showed that when the recommendation task is narrowed to a thematically coherent set of items, the performance of the content-based approach improves dramatically, with

**nDCG@10** nearly doubling. This highlights a key strength of CBRS: its potential for high performance in specialises or "long-tail" scenarios, addressing a primary aspect of RQ3.

6. **The Limitation of Global Ranking vs. Thematic Clustering.** The ranking anomaly, where E2E models' performance declined with further training, revealed a fundamental limitation and a corresponding strength. While these models are poor global rankers when over-trained, their success at the classification task proves they are excellent *thematic clusterers*. This insight reframes their role from a simple recommender to a powerful candidate generator, addressing the final component of RQ3 about the strengths and limitations of the purely content-based approach.

The empirical results from this chapter lead to a series of key findings that build upon one another, providing a comprehensive answer to the research questions.

# Chapter 7

# Discussion

This research set out to investigate the construction and performance of a multimodal, content-based video recommender system, with the ultimate goal of addressing the cold-start problem. The experiments detailed in Chapter 6, progressing from a feature-based baseline to a sophisticated end-to-end learning architecture, have yielded several key findings that provide clear answers to the initial research questions. This chapter synthesises these findings, contextualises them within the broader literature, and discusses their implications for the design of modern recommender systems.

## 7.1 RQ1: The Hierarchy of Modality Contribution

The first question asked: *Which multimodal content features (visual, audio, textual) contribute most to determining program similarity?*

The ablation study (Section 6.2) and feature importance analysis (Section 6.3) provided a clear and unequivocal answer: textual features are the dominant modality for this task. The removal of textual features resulted in a catastrophic performance drop (an 18.9% decrease in **Precision@1**), confirming that for the MovieLens dataset, explicit metadata like genre and release year, along with the semantic content of plot summaries from BERT, form the primary backbone of user-perceived similarity.

More specifically, the feature importance analysis (Figure 6.4) revealed that a single feature, genre, was overwhelmingly the most powerful predictor. This highlights a key characteristic of the dataset: content similarity, as inferred from user ratings, is, to a large extent, defined by the genre itself. This finding aligns with the foundational principles of traditional CBRS, which have long relied on such metadata [4,14]. However, this work provides an important quantitative benchmark, demonstrating that even in the presence of rich, deep learning-based audiovisual features, these simple textual signals remain paramount.

The audio modality was found to be a consistent and valuable secondary contributor. Removing audio led to a notable performance across all metrics (e.g., $\sim$-7.5% in **nDCG@10**). This indicates that the deep audio embeddings from VGGish successfully capture non-redundant stylistic and atmospheric information (a movie's musical tone or the nature of its sound design) that is not fully encapsulated by text or visuals alone. This supports the argument by Almeida et al. [2] for the complementarity of diverse feature types.

The most striking finding was the negative contribution of the visual modality when naively integrated. Removing the high-dimensional visual features consistently *improved* recommendation performance. This result does not imply that visual features are useless; a conclusion that

would contradict a large body of work on visual analysis [5]. Rather, it highlights a critical limitation of simple fusion methods: naively concatenating high-dimensional, potentially noisy features can overwhelm a fixed similarity metric, such as cosine similarity. This finding served as the primary motivation for investigating the end-to-end earning approach, which is hypothesised to be capable of learning to weigh or ignore these noisy dimensions. The subsequent success of the E2E model (discussed in the next section) validates this hypothesis, suggesting that deep learning architectures can indeed learn to extract the valuable signal from the visual modality that simpler methods cannot.

Notwithstanding, this does not render multimodal analysis obsolete. The true strength of a "Seeing and Hearing" approach lies not in supplanting genre but in capturing intra-genre nuance. A simple genre tag cannot distinguish between a dark action film and a comedic one. The aural and visual features are precisely what enables a recommender to learn these subtle but critical distinctions, leading to more personalised and satisfying recommendations within a user's preferred categories.

This highlights a critical, practical implication for industrial applications. The feature-selected baseline, while performant, is highly dependent on the availability and quality of a single metadata field: genre. In many real-world scenarios, such structured metadata may be missing, inconsistent, or unavailable, especially for newly ingested content or long-tail items. In such a "no-genre" environment, the baseline model's performance would collapse.

Herein lies the true, and perhaps most significant, strength of the E2E model. By learning similarity directly from the raw visual, audio, and textual content, it develops a robust understanding of thematic similarity that is not reliant on brittle metadata. The "classification-ranking gap" shows its ability to form tight thematic clusters. This capability is invaluable not as a ranker, but as a primary engine for automated content tagging and candidate generation in data environments where clean labels do not exist.

## 7.2 RQ2: Learned vs. Traditional Similarity Metrics

The second research question asked: *How does a learned similarity metric (via a Siamese Neural Network) compare to traditional similarity measures (e.g., cosine similarity)?*

The empirical results present a nuanced and multi-faceted answer. When comparing the best-performing models from each paradigm (Table 6.8), the end-to-end learned approach is demonstrably superior. The best E2E mdoel (`E2E-Triplet-Cosine` at epoch 10) achieved an **nDCG@10** of 0.0711, a 41.6% relative improvement over the best feature-engineered baseline (**nDCG@10** of 0.0502). This provides a strong affirmative answer to the research question, confirming that for this complex multimodal task, an SNN can learn a more effective, bespoke similarity function than a fixed metric applied to pre-extracted features.

However, this conclusion is enriched by a critical phenomenon observed during training: the ranking anomaly (Section 6.5.1). This research empirically demonstrates the "classification-ranking gap" in multimodal content-based recommendations. It is shown that as a Siamese network becomes a more proficient classifier of similarity, its ability to produce a nuanced ranking of similar items can degrade due to "metric collapse", where all "similar" items are mapped to an increasingly tight cluster. While the E2E models' performance on their direct classification or metric learning objectives improved steadily (Figures 6.11 -6.14), their performance on the global ranking task often peaked early and then declined. This paradox reveals the fundamental difference between learning to *classify* similarity and learning to *rank* similarity. The SNNs, trained with a metric learning objective such as Contrastive Loss or Triplet Loss, became highly specialised classifiers, creating tightly defined thematic clusters. This phenomenon, which is

termed the "classification-raking gap", aligns with observations in metric learning literature where over-specialisation on a classification or verification task can lead to a "metric collapse". In this state, the distance between all positive examples approaches zero, which, while optimal for classification, erases the fine-grained relative distance information necessary for high-quality ranking.

The insight reframes the comparison. The end-to-end model is not simply a "better" version of the baseline; it is a different kind of tool. Its strength lies in its ability to act as a powerful thematic filter or candidate generator. The superior final performance of the early-stopped E2E model suggests that it found a "sweet spot" where it learned a robust representation without over-specialising, successfully unlocking the signal within the noisy visual modality that the baseline model could not.

## 7.3 RQ3: Strengths and Limitations of a Purely Content-Based Approach

The final research question asked: *Based on the literature and the challenges addressed in this work, what are the primary strengths and limitations of a purely content-based approach for video recommendation?*

This thesis empirically demonstrates both the profound strengths and clear limitations of CBRS.

**The primary strength of CBRS is its ability to solve the cold-start problem and excel in niche contexts.** This was powerfully illustrated in the domain-specific analysis (Section 6.4), where narrowing the task to a thematically coherent subset (action and animated films) caused recommendation performance to skyrocket, with **nDCG@10** nearly doubling (+98.7%). This demonstrates that when content serves as a strong proxy for relevance, CBRS can be exceptionally effective. As the entire methodology operates without user interaction data, it provides a robust and immediate solution for recommending new items, a fundamental challenge for collaborative filtering systems [13, 22].

Furthermore, this thesis highlights a fundamental strength of the content-based approach: its ability to capture intra-genre nuance. A user who likes "Action" movies does not necessarily like all action movies. They may prefer the dark, gritty aesthetic of "The Dark Knight" over the high-fantasy action of "Thor". A purely collaborative system struggles to make such distinctions. A multimodal CBRS, by analysing the actual visual colour palettes, audio landscapes, and narrative complexity, is uniquely positioned to understand *why* certain movies are more similar, and others are not, leading to more personalised and satisfying recommendations even within a user's preferred genres.

**The primary limitation is the performance ceiling on diverse, general-purpose datasets.** The modest performance of even the best model on the complete 13,920-movie set indicates that content features alone cannot capture all the nuances of user preference. Factors beyond the content itself, such as popularity, social trends, and serendipity, heavily influence user choice. This finding aligns with the broad consensus in the literature that hybrid models, which combine content-based features with collaborative signals, typically offer the best overall performance for established users and items [19]. This thesis, however, successfully developed and optimised the content-based component, which remains an indispensable building block for any state-of-the-art hybrid system.

## 7.4 Limitations of the Study

While this research provides valuable insights, it is important to acknowledge its limitations.

1. **Proxy for Content:** The analysis was conducted using movie trailers as a proxy for full-length films. While this is a common practice driven by computational feasibility, trailers are designed to be promotional and may not fully represent the stylistic or narrative pacing of the complete work.

2. **Dataset and Ground Truth:** The findings are specific to the MovieLens dataset. The user preference signals derived from this dataset, while extensive, may not perfectly capture the complex reasons for similarity and may be biased towards certain types of content.

3. **Computational Constraints:** The E2E model experiments, particularly those involving triplet loss with online mining, were computationally intensive. The use of a relatively small batch size (16) may have limited the effectiveness of the semi-hard negative mining strategy, which typically benefits from a larger pool of in-batch candidates. More substantial computational resources could potentially unlock further performance gains from these advanced architectures.

4. **Data Split Methodology:** The train-validation-test split for the end-to-end models was performed at the pair level rather than the user level. This means that a user who contributed multiple training pairs could have some of those pairs in the training set and others in the test set. This introduces a risk of minor data leakage, where the model might learn user-specific taste patterns rather than pure content similarity. While this approach maximises the use of all generated similarity pairs, a stricter evaluation protocol would involve splitting by users to completely isolate the test set from any user-level information present in training.

5. **Offline Evaluation Protocol:** This study relies entirely on offline evaluation using a static dataset. While standard for academic research, offline metrics like nDCG do not always correlate perfectly with user satisfaction or engagement in a live production environment. A complete validation of the system would require online A/B testing to measure its impact on real user behaviour, which was beyond the scope of this Master's thesis.

These limitations, while important to acknowledge, do not invalidate the core findings regarding the hierarchy of modalities and the trade-off between classification and ranking. Instead, they pave the way for clear and promising directions for future research.

# Chapter 8

# Conclusion

This thesis set out to address the persistent cold-start problem in video recommendation systems. In an era of infinite content, the inability to recommend new items effectively creates a visibility barrier that hinders discovery for both users and content creators. The central goal of this research was to design, build, and rigorously evaluate a purely content-based recommender system capable of understanding the multimodal essence of a video, by seeing, hearing, and reading, to generate relevant recommendations without any user interaction data.

To achieve this, a two-pronged methodological framework was employed. The investigation began by constructing a traditional baseline model using state-of-the-art feature extractors, which served as a benchmark and diagnostic tool. A rigorous analysis of this baseline, conducted through ablation studies and feature selection, revealed a clear hierarchy of modality importance and the limitations of naive feature fusion. These insights motivated the second stage of the research: the development of a sophisticated end-to-end learning architecture using Siamese Neural Networks. This model was designed to learn a bespoke similarity function directly from data, overcoming the noise and complexity inherent in high-dimensional multimodal features.

The empirical results yielded several key findings. First, textual features were unequivocally the most dominant modality, while audio provided a valuable secondary signal. Second, naively integrated visual features were found to be detrimental in a fixed-metric system. Third, and most critically, this work empirically demonstrated a crucial "classification-ranking gap."it is shown that as a Siamese network becomes a better classifier of similarity, its utility as a ranker paradoxically declines. This is because successful classification training leads to metric collapse, where all "similar" items are mapped into an ever-tighter cluster in the embedding space. While this proves the model's ability to identify thematic coherence, the convergence into an extremely small cluster diminishes the relative distance between items, rendering a nuanced ranking within that cluster almost impossible. This insight reframes the E2E model's primary strength not as direct ranking but as powerful thematic clustering, a vital capability for candidate generation.

The primary contribution of this thesis is a comprehensive and reproducible pipeline for building and evaluating a multimodal, content-based video recommender. This work demonstrates that a deep learning approach can effectively harness the rich signals within visual and audio modalities, surpassing traditional feature-based methods. However, it also demonstrated that the architecture is designed to handle the classification vs. ranking trade-off. The resulting `E2E-Triplet-Cosine` model is a powerful, validated component for the candidate generation stage of any state-of-the-art hybrid[1] recommendation engine, providing a robust solution to the cold-start problem.

[1] In this context, "hybrid" refers to a filtering and ranking recommendation system

## 8.1 Future Work: An actionable Roadmap

The conclusion of this thesis does not mark an end but rather provides a clear, evidence-based roadmap for building a next-generation, truly effective content-based recommendation system. The path forward consists of three primary stages:

**Step 1: Immediately Implement the Hybrid Re-Ranking System.** The most direct extension of this work is to build the two-stage system this research validates. The mediate priority is to use the embeddings from the best-performing E2E model (E2E-Triplet-Cosine) as a powerful candidate generator, retrieving a high-quality set of $\sim 200$ thematically similar items. This set would then be re-ranked by a lightweight model (e.g., LambdaMART[2]) trained on the highly predictive feature-selected baseline signals. This approach directly leverages the core strengths of both methodologies investigated in this thesis and is hypothesised to outperform either one in isolation.

**Step 2: Enhance the Core E2E Engine.** The next phase involves strengthening the E2E model itself. This can be achieved through two avenues:

**Advanced Fusion:** Stabilise and apply a more sophisticated fusion architecture, such as the prototyped `HybridFusionNetwork`, to the triplet loss models. This will allow modalities to dynamically inform one another, potentially unlocking more signals from the audiovisual data.

**Full Fine-Tuning & Scalability:** With greater computational resources, conduct experiments with full fine-tuning of the large vision and language backbones. Furthermore, increasing the batch size during triplet mining is known to significantly improve metric learning and should be a key focus.

**Step 3: Transition to a Practical, Trustworthy System.** To move this model from a research prototype to a production-ready tool, two final components are essential:

**Enhance Explainability:** Apply model-agnostic techniques like SHAP or LIME to the E2E model's predictions. This would provide invaluable insights into which specific features (e.g., a particular colour palette, a sound effect)influence a recommendation, enhancing transparency and debugging.

**Implement Online Evaluation (A/B Testing):** The ultimate validation requires deploying the hybrid system in a live production environment. An online A/B test, measuring real user engagement metrics like click-through rate (CTR), is the crucial final step to confirm the real-world impact and value of this multimodal approach.

**Step 4: A Visionary Step - Towards Intelligent, Context-Aware Recommendation.** The ultimate evolution would be to develop a context-aware AI agent that dynamically adjusts recommendation strategies based on temporal and cultural signals. An AI agent could be developed to dynamically adjust the feature weights based on external factors. For example, during December, the weight of the "Christmas" genre metadata could be increased. In response to the release of a major sci-fi blockbuster, the system could temporarily upweight visual features related to "space" or "special effects" to capture trending user interest. It could also adapt to trending topics in real-time. This represents the frontier of truly intelligent content discovery systems.

[2]https://xgboost.readthedocs.io/en/latest/tutorials/learning$_to_rank.html references$

This body of work provides a solid foundation, demonstrating both the power and the pitfalls of multimodal content-based recommendation. The path forward is clear: by combining the strengths of different modelling paradigms with intelligent, adaptive systems, one can build systems that are not only accurate but also robust, explainable, and truly capable of navigating the vast, ever-changing world of video content.

# Appendix A

# Technical Specifications and Environment

This appendix details the hardware and software environment used for all experiments to ensure the reproducibility of the results presented in this thesis.

## A.1 Hardware and Cloud Platform

All model training and computationally intensive feature extraction tasks were performed on the Amazon Web Services (AWS) cloud platform.

- **Deep Learning Instance:** AWS SageMaker using an `ml.g4dn.2xlarge` instance.
- **GPU:** NVIDIA T4 Tensor Core GPU with 16 GB of GDDR6 memory.
- **CPU:** 8 vCPUs (Intel Xeon Platinum 8000 series).
- **System RAM:** 32 GB.

Data storage and management were handled using AWS S3.

## A.2 Software and Libraries

The experimental framework was built using Python. The key libraries and their versions are listed below.

### A.2.1 Core Environment

- **Python:** 3.10.6
- **PyTorch:** 1.12.1
- **CUDA Toolkit:** 11.3

### A.2.2 Machine Learning and Data Processing

- **scikit-learn:** 1.3.0
- **transformers (Hugging Face):** 4.28.1
- **pandas:** 1.5.3
- **numpy:** 1.23.5

### A.2.3 Multimedia Feature Extraction

- **librosa:** 0.9.2 (for audio analysis)
- **OpenCV (cv2):** 4.6.0 (for visual analysis)
- **EasyOCR:** 1.6.2 (for optical character recognition)

# Appendix B

# Data Exploration and Validation

This appendix provides supplementary analysis on the 13,920-movie dataset used in this thesis. The analyses were conducted to validate the quality of the data and to ensure the filtering process did not introduce significant biases, as referenced in Chapter 3.

## B.1 User Rating Analysis

The following figures characterise the user ratings from the MovieLens 20M dataset associated with the final corpus.

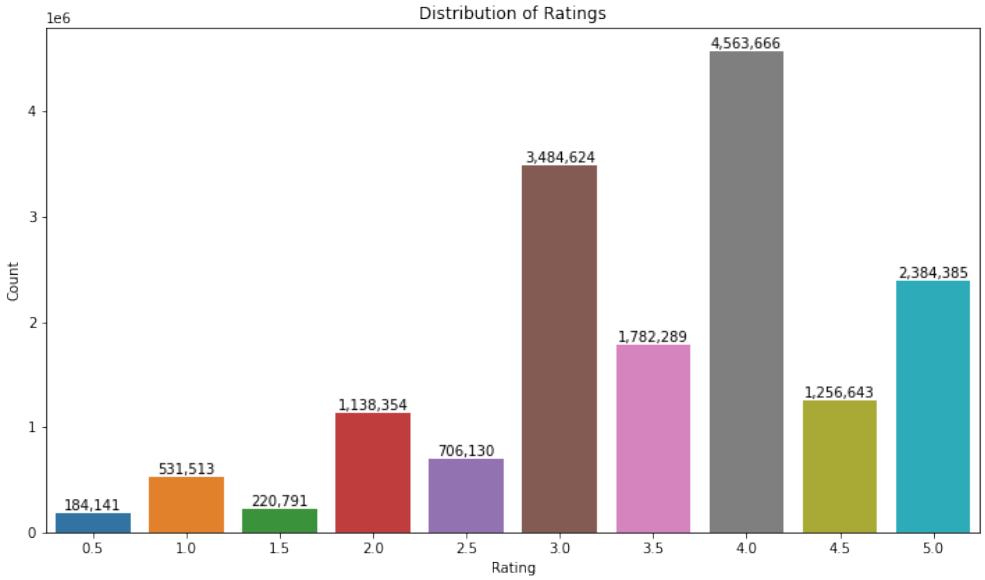



Figure B.1: Distribution of Raw User Ratings (0.5-5.0) in the MovieLens 20M Dataset for the Movies Included in the Final Corpus.
The distribution is skewed towards positive ratings, with a peak at 4.0, which is characteristic of recommendation datasets.

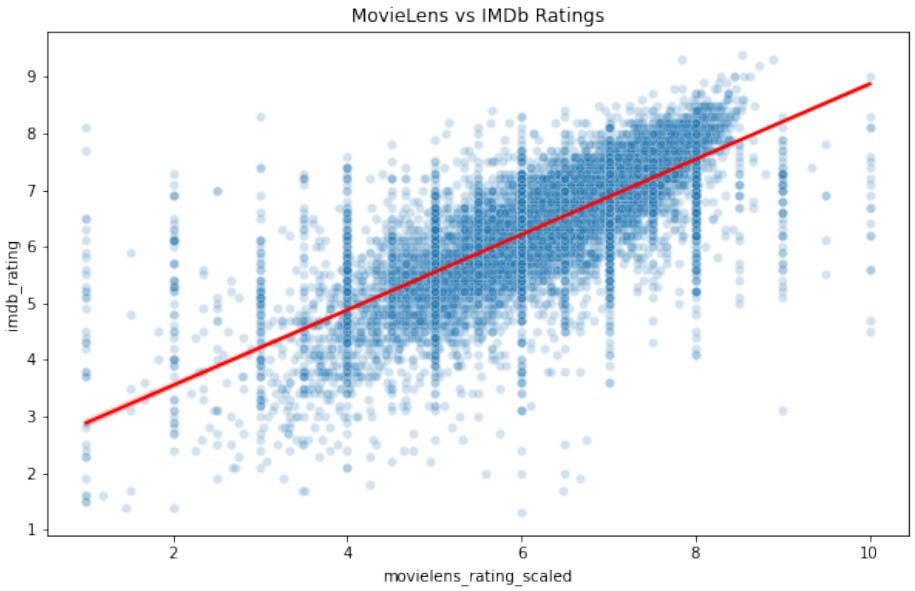



Figure B.2: Correlation Between MovieLens Ratings (Scaled to 1-10) and Corresponding IMDb Ratings.
The positive correlation demonstrates the external validity of the user-provided ratings in the MovieLens dataset.

## B.2 Dataset Bias and Representativeness Analysis

This section details the analysis performed to validate that the data filtering pipeline did not introduce significant biases into the final set.

### B.2.1 Genre and Language Distribution

Te filtering process maintained a diverse representation of genres and languages. Figure B.4 shows the final genre breakdown, which, upon comparison with the original dataset, confirmed that no major genre was disproportionately removed. Similarly, Figure B.5 shows the language distribution, which reflects a known bias towards English-language films in popular movie datasets.

## B.3 Dataset Representativeness and Bias Analysis

This section details the analysis performed to validate that the data filtering pipeline did not produce significant biases into the final 13,920-movie set used for this research.

1. **Temporal Distribution:** The release year distribution of the final dataset was compared to the original. While there is a slight under-representation of pre-1970s films (primarily due to lower availability of high-quality digital trailers), the overall temporal profile, particularly for films from 1980-2015, closely mirrors that of the original MovieLens 20M dataset.

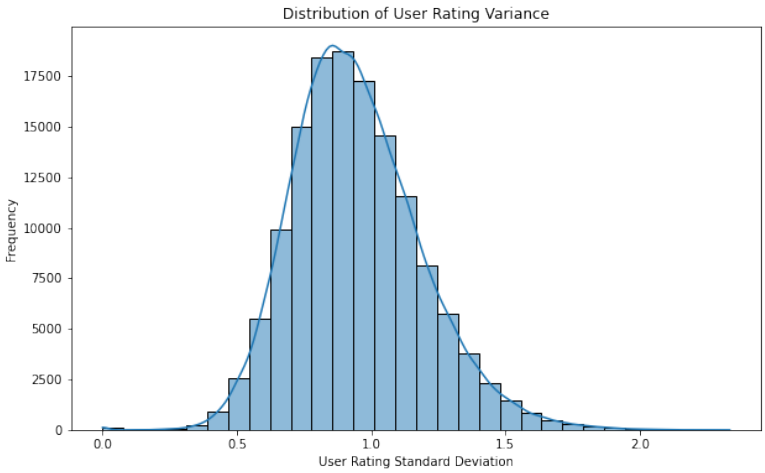



Figure B.3: Distribution of the Standard Deviation of Ratings for Each User.
The approximately normal distribution centred around 1.0 indicates that most users provide a diverse range of ratings rather than rating all movies similarly, confirming the utility of the user data for evaluation.

2. **Genre Distribution:** As shown in Figure B.4, the final dataset maintains a wide coverage of genres. A comparative analysis confirmed that the relative prevalence of major genres (e.g., Drama, Comedy, Thriller) remained consistent. No single genre was disproportionately eliminated by the filtering process.

3. **User Rating Patterns:** The distribution of binarised user ratings (positive vs. negative) in the final dataset's interaction data (61.4% positive) is nearly identical to the distribution across the entire MovieLens 20M dataset (62.1% positive). This indicates that the filtering did not skew towards movies that were universally liked or disliked.

4. **Popularity Bias:** A key concern was whether the filtering process disproportionately removed less popular, "long-tail" items. An analysis of movie rating counts showed that the primary driver for movie exclusion was the availability of a verifiable trailer, not the number of ratings. Both popular and long-tail movies were removed at similar rates, suggesting that popularity bias was not exacerbated by the data preparation pipeline.

This multifaceted analysis provides strong evidence that the final multimodal dataset, despite its reduced size, is a valid and representative subset for the experiments conducted in this thesis.

### B.3.1 Data Quality Issues Breakdown

Table B.1 provides a detailed breakdown of the specific reasons for data exclusion during the filtering process.

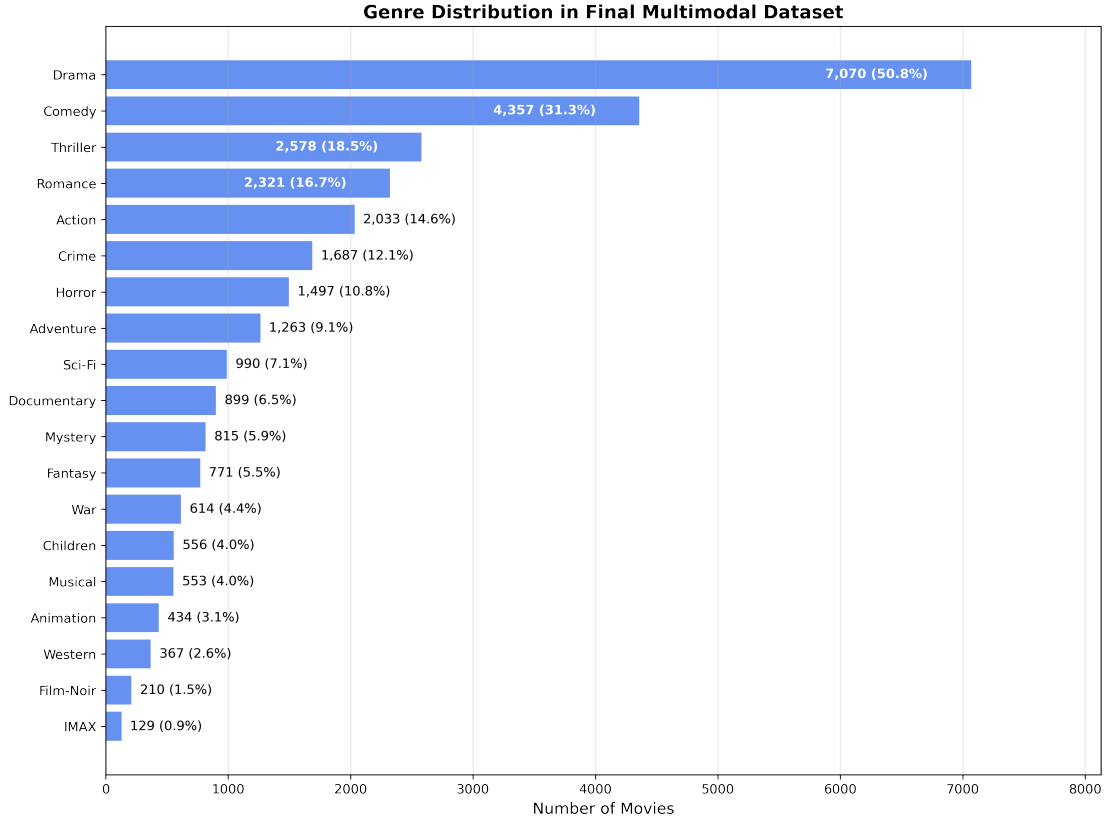



Figure B.4: Genre Distribution in the Final Multimodal Dataset.
Movies can belong to multiple genres, so percentages sum to more than 100%.

### B.3.2 Statistical Validation Results

#### Temporal Distribution

The release year distribution of the final dataset was compared to the original MovieLens 20M dataset. While there is a slight under-representation of pre-1970s films (primarily due to lower availability of high-quality digital trailers), the overall temporal profile, particularly for films from 1980-2015, closely mirrors that of the original dataset.

#### Genre Representation

As shown in Figure B.4, the final dataset maintains wide coverage across all major genres. A comparative analysis confirmed that the relative prevalence of major genres (Drama, Comedy, Thriller, Action) remained consistent with the original dataset. No single genre was disproportionately eliminated by the filtering process.

#### User Rating Patterns

The distribution of binarised user ratings in the final dataset (61.4% positive interactions) closely matches the distribution across the entire MovieLens 20M dataset (62.1% positive). This indi-

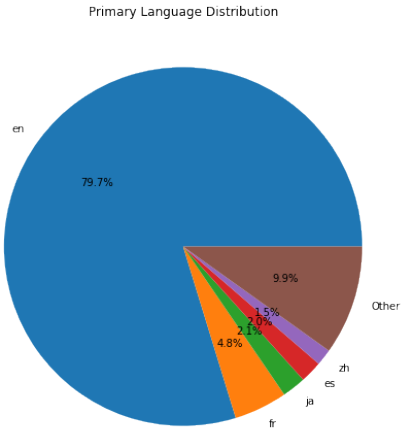



Figure B.5: Primary Language Distribution in the Final Dataset.
English is the dominant language, accounting for nearly 80% of the movies, which reflects a common bias in popular movie datasets.

Table B.1: Breakdown of Data Quality Issues and Their Impact on Dataset Composition

| **Data Quality Issue** | **Count Removed** | **% of Original** |
|---|---|---|
| Invalid/missing YouTube links | 8,013 | 29.4% |
| Video retrieval failures | 4,323 | 15.8% |
| Full movies (not trailers) | 1,022 | 3.7% |
| Missing textual metadata | 252 | 0.9% |
| Corrupted video files | 22 | 0.1% |

cates that the filtering process did not skew towards movies that were universally liked or disliked.

#### Popularity Bias Assessment

A critical concern was whether the filtering process disproportionately removed less popular "long-tail" items. Analysis of movie rating counts revealed that both popular and niche movies were removed at similar rates. The primary driver for exclusion was trailer availability rather than popularity, suggesting that the filtering did not exacerbate popularity bias.

### B.3.3 Conclusion of Validation

This multifaceted analysis provides strong evidence that the final multimodal dataset, despite its reduced size, is a valid and representative subset for the experiments conducted in this thesis.

# Appendix C

# Additional Experimental Results

This appendix contains supplementary results from exploratory experiments conducted during the iterative enhancement phase of the baseline model, as mentioned in Section 6.2.2. These experiments, including the testing of alternative fusion methods and dimensionality reduction techniques, informed the final research direction but are included here for the sake of completeness. They provide additional context for the decision to pursue the more methodical feature selection pipeline detailed in the main body of the thesis.

## C.1 Iterative Enhancement: Feature Engineering and Fusion Methods

Based on the ablation study findings, several enhancement strategies were tested to see if performance could be improved by re-balancing modality contributions or applying dimensionality reduction.

Based on the ablation study findings, several enhancement strategies were implemented:

### C.1.1 Alternative Fusion and Enhancement Methods

As shown in Table C.1, alternative fusion strategies like weighted sum and average fusion, as well as an enhanced concatenation with improved visual features, did not yield significant improvements over the simple baseline. This motivated the more rigorous feature selection approach in Section 6.3.

Table C.1: Enhanced Fusion Methods Performance

| Method | P@1 | P@10 | HR@10 | NDCG@10 | MRR@10 |
|---|---|---|---|---|---|
| Baseline (Concat) | **0.0889** | **0.0647** | 0.3970 | 0.0452 | **0.1670** |
| Weighted Sum | 0.0626 | 0.0587 | 0.3809 | 0.0719 | 0.1382 |
| Enhanced Concat | 0.0740 | 0.0602 | **0.4086** | **0.0773** | 0.1567 |
| Average Fusion | 0.0625 | 0.0558 | 0.3790 | 0.0710 | 0.1226 |
| No Visual Enhanced | 0.0626 | 0.0575 | 0.3981 | 0.0711 | 0.1439 |

### C.1.2 Dimensionality Reduction Techniques

Various dimensionality reduction techniques (UMAP and PCA) were applied to the feature set prior to fusion. As shown in Table C.2, these methods also failed to produce a consistent improvement in performance, further reinforcing the need for a supervised feature selection method that considers the relationship between features and the target variable (user ratings).

Table C.2: Impact of Dimensionality Reduction on Performance

| **Method** | **P@1** | **P@10** | **HR@10** | **NDCG@10** | **MRR@10** |
|---|---|---|---|---|---|
| UMAP-30 | 0.0733 | 0.0600 | 0.4046 | 0.0782 | 0.1556 |
| UMAP-50 | 0.0733 | 0.0600 | 0.4046 | 0.0782 | 0.1556 |
| UMAP-100 | 0.0733 | 0.0600 | 0.4046 | 0.0782 | 0.1556 |
| PCA | **0.0739** | **0.0602** | 0.4062 | **0.0783** | **0.1562** |
| PCA-v2 | 0.0624 | 0.0595 | **0.4102** | 0.0771 | 0.1527 |

# Appendix D

# End-to-End Model Implementation Details

This appendix provides the specific architectural details for the end-to-end Siamese Neural Network models described in Chapter 4. Each model consists of three modality-specific processing modules whose outputs are fed into a final fusion network to produce a shared embedding.

## D.1 Modality-Specific Processing Modules

The following modules process the raw and pre-computed multimodal data into intermediate, fixed-size representations before fusion.

### D.1.1 Visual Processing Module

- **Input:** Pre-computed DINOv2 frame embeddings (16 frames, each 384-dim) and raw video frames.
- **Architecture:** A dual-pathway architecture.
  - **High-level path:** The 384-dim DINOv2 frame embeddings are first temporally pooled (mean), L2-normalised, and then passed through a LayerNorm. The resulting vector is processed by an MLP:
    $\text{Linear}(384 \to 192) \to \text{BatchNorm1d} \to \text{ReLU} \to \text{Dropout}(0.4) \to \text{Linear}(192 \to 192)$.
  - **Low-level path:** On-the-fly features (5-dim, e.g., brightness, contrast) are projected via:
    $\text{Linear}(5 \to 64) \to \text{Tanh}$.
- **Output:** The outputs of both pathways are concatenated ($192 + 64 = 256$-dim) and L2-normalised. The core DINOv2 backbone is frozen, with only its final normalisation layer being fine-tuned to better adapt the pre-trained features to the recommendation task.

### D.1.2 Audio Processing Module

- **Input:** Pre-computed VGGish embeddings (128-dim) and raw audio waveforms.
- **Architecture:** A dual-pathway architecture.

- **VGGish path:** The 128-dim VGGish embeddings are L2-normalised. These are treated as fixed features and are not trained.
- **Spectrogram path:** A Mel-spectrogram is computed on the fly from the raw waveform and processed by a custom CNN, which outputs a 64-dim vector. The CNN architecture is a sequence of Conv2d, BatchNorm2d, GELU, and MaxPool2d layers, concluding with an AdaptiveAvgPool2d layer.

- **Fusion & Output:** The 128-dim VGGish vector and the 64-dim CNN vector are concatenated. This 192-dim vector is fused via an MLP (Linear($192 \to 128$) $\to$ LayerNorm $\to$ GELU $\to$ Dropout(0.5) $\to$ Linear($128 \to 192$)). This is followed by a Temporal Pooling block (Linear($192 \to 192$) $\to$ Tanh), and the final 192-dim vector is L2-normalised.

### D.1.3 Textual Processing Module

- **Input:** Movie title and plot, pre-computed TF-IDF vectors (5000-dim), a normalised release year, and a language index.
- **Architecture:** A multi-pathway architecture.
  - **BERT path:** A mostly frozen `bert-base-uncased` model processes text to produce a 768-dim semantic embedding, which is then L2-normalised.
  - **TF-IDF path:** The 5000-dim TF-IDF vector is projected and normalised: Linear($5000 \to 100$) $\to$ LayerNorm.
  - **Categorical paths:** A language index is mapped via an Embedding layer (160 languages $\to$ 16-dim), and the release year is used as a single normalised feature.
- **Output:** All pathway outputs are concatenated ($768 + 100 + 16 + 1 = 885$-dim) and fused through a final MLP (Linear($885 \to 256$) $\to$ BatchNorm1d $\to$ ReLU $\to$ Dropout(0.4) $\to$ Linear($256 \to 384$)). The resulting 384-dim feature vector is then L2-normalised.

## D.2 Fusion Network Architectures

The L2-normalised outputs from the modality-specific modules are concatenated and fed into a fusion network, which produces the final 128-dimensional embedding. Different fusion architectures were used across the 2x2 experimental design to manage complexity and training stability.

### D.2.1 HybridFusionNetwork (Used for Pair-Cosine)

This is the most complex fusion network, employing cross-attention mechanisms. The visual feature vector acts as a "query" to "ask questions" of the audio and text feature vectors (which act as keys/values). The original visual features and the two attention-modified visual features are then concatenated and processed by a final MLP.

### D.2.2 SimpleFusionNetwork (Used for Pair-Euclidean & Triplet-Cosine)

This network takes the concatenated multimodal feature vectors, normalises them with LayerNorm, and passes them through a two-layer MLP with a ReLU activation and dropout to produce the final embedding.

### D.2.3 UltraSimpleFusionNetwork (Used for Triplet-Euclidean)

This is the most stable and simplified architecture. It consists of a single LayerNorm followed by a single linear layer to project the concatenated multimodal features directly to the final 128-dimensional embedding space. This reduction in complexity was found to be crucial for stabilising the training of the Euclidean triplet model.

## D.3 Model Architecture Summary

To provide a concrete overview of the model complexity, Table D.1 summarises the architecture and trainable parameter count for the `E2E-Triplet-Cosine` model variant, which utilises the `SimpleFusionNetwork`. The total number of trainable parameters is dominated by the various projection and fusion layers, as the core backbones are either frozen or only minimally fine-tuned.

Table D.1: Trainable Parameter Summary of the 'E2E-Triplet-Cosine' Model Architecture

| Layer / Module | Output Shape | Trainable Param # |
|---|---|---|
| **visual_module**[a] | [B, 256] | **113280** |
| – frame_fusion_mlp | [B, 192] | 111,360 |
| – low_level_projection | [B, 64] | 384 |
| – (fine-tuned DINOv2 norm) | (internal) | 768 |
| – (other LayerNorm) | (internal) | 768 |
| **audio_module**[b] | [B, 192] | **110304** |
| – spec_cnn | [B, 64] | 23,520 |
| – fusion | [B, 192] | 49,728 |
| – temporal_pooling | [B, 192] | 37,056 |
| **text_module**[c] | [B, 384] | **828876** |
| – tfidf_projection | [B, 100] | 500,100 |
| – tfidf_norm | [B, 100] | 200 |
| – language_embedding | [B, 16] | 2560 |
| – fusion | [B, 384] | 326,016 |
| **fusion_network (Simple)** | [B, 128] | **401568** |
| **Total Trainable Params** | | **1454028** |
| **Total Non-Trainable Params** | | 109482240$^{+}$ |

[a] The DINOv2 backbone is frozen; only its final norm layer is fine-tuned.
[b] VGGish embeddings are pre-computed and fixed (0 trainable params).
[c] The BERT model's encoder layers, pooler, and embeddings are frozen (approx. 109.5M non-trainable parameters).

# Appendix E

# Core End-to-End Model Implementation

This appendix provides the python source code for the central components of the end-to-end SNN models developed in Chapter 4. The code is implemented using the PyTorch framework. For the complete, executable codebase including data loaders, helper utilities, and training scripts, please refer to the GitHub repository: `https://github.com/JulietteMaes01/multimodal-video-recommendations`.

## E.1 Siamese Network Architectures

The following classes define the high-level structure for the pairwise and triplet-based learning models.

### E.1.1 Pair-Based Model

This network process two items in parallel to produce a pair of embeddings for use with a contrastive loss function.

```
class OptimizedPairBasedSiameseNetwork(nn.Module):
    def __init__(self, visual_module, audio_module, text_module, fusion_network):
        super().__init__()
        self.visual_module = visual_module
        self.audio_module = audio_module
        self.text_module = text_module
        self.fusion_network = fusion_network

    def forward_single(self,
                       frames_for_low_level,          # For visual low-level
                       dino_frame_embeddings,         # For visual high-level (precomputed)
                       vggish_embedding=None,         # For audio (precomputed VGGish)
                       waveform_for_spec_cnn=None,    # For audio (raw wave for spec CNN)
                       title=None, plot=None, tfidf_features=None, # For text
                       year=None, language_idx=None):

        # 1. Visual Processing
```

```
        visual_features = self.visual_module(
            frames=frames_for_low_level,
            precomputed_dino_embeddings=dino_frame_embeddings
        )

        # 2. Audio Processing
        audio_features = None
        if self.audio_module is not None:
            audio_features = self.audio_module(
                precomputed_vggish_embedding=vggish_embedding,
                waveform_for_spec_cnn=waveform_for_spec_cnn
            )

        # 3. Text Processing
        text_features = None
        if self.text_module is not None:
            # check that title and plot are not None before passing them
            if title is not None and plot is not None:
                text_features = self.text_module(
                    title=title,
                    plot=plot,
                    tfidf_features=tfidf_features,
                    year=year,
                    language_idx=language_idx
                )
            else:
                # If title/plot are missing, can't create text features -> create a zero tensor of the correct size to avoid errors.
                print("       Warning: title or plot missing for text module. Creating zero tensor.")
                batch_size = visual_features.shape[0]
                device = visual_features.device
                text_features = torch.zeros(batch_size, self.text_module.output_dim, device=device)


        fused_embedding = self.fusion_network(visual_features, audio_features, text_features)
        return fused_embedding

    def forward(self,
                frames1_for_low_level, dino_frame_embeddings1, waveform1, vggish_embedding1,
                frames2_for_low_level, dino_frame_embeddings2, waveform2, vggish_embedding2,
                video1_title=None, video1_plot=None, video1_tfidf=None, video1_year=None, video1_language_idx=None,
                video2_title=None, video2_plot=None, video2_tfidf=None, video2_year=None, video2_language_idx=None):

        current_batch_size = frames1_for_low_level.shape[0]
        current_device = frames1_for_low_level.device

        nan_check_visual = (
            torch.isnan(frames1_for_low_level).any() or torch.isnan(dino_frame_embeddings1).any() or
            torch.isnan(frames2_for_low_level).any() or torch.isnan(dino_frame_embeddings2).any()
        )
        if nan_check_visual:
            print("       NaN detected in primary visual input tensors in SNN forward.")
```

```
            emb_dim = self.fusion_network.embedding_dim if hasattr(self.
    fusion_network, 'embedding_dim') else 128
            return (torch.zeros(current_batch_size, emb_dim, device=current_device
    ),
                    torch.zeros(current_batch_size, emb_dim, device=current_device
    ))

        if self.audio_module:
            waveform1 = torch.nan_to_num(waveform1, nan=0.0, posinf=0.0, neginf
    =0.0) if waveform1 is not None else None
            vggish_embedding1 = torch.nan_to_num(vggish_embedding1, nan=0.0,
    posinf=0.0, neginf=0.0) if vggish_embedding1 is not None else None
            waveform2 = torch.nan_to_num(waveform2, nan=0.0, posinf=0.0, neginf
    =0.0) if waveform2 is not None else None
            vggish_embedding2 = torch.nan_to_num(vggish_embedding2, nan=0.0,
    posinf=0.0, neginf=0.0) if vggish_embedding2 is not None else None
        if self.text_module:
            video1_tfidf = torch.nan_to_num(video1_tfidf, nan=0.0) if video1_tfidf
     is not None else None
            video2_tfidf = torch.nan_to_num(video2_tfidf, nan=0.0) if video2_tfidf
     is not None else None


        embedding1 = self.forward_single(
            frames1_for_low_level, dino_frame_embeddings1,
            vggish_embedding1, waveform1,
            video1_title, video1_plot, tfidf_features=video1_tfidf,
            year=video1_year, language_idx=video1_language_idx
        )

        embedding2 = self.forward_single(
            frames2_for_low_level, dino_frame_embeddings2,
            vggish_embedding2, waveform2,
            video2_title, video2_plot, tfidf_features=video2_tfidf,
            year=video2_year, language_idx=video2_language_idx
        )

        return embedding1, embedding2
```

Listing E.1: OptimizedPairBasedSiameseNetwork Class

### E.1.2 Triplet Based Model

This network is designed for online triplet mining. It functions as a single tower that processes a batch of unique items. The triplet construction and loss calculation are handled within the training loop.

```
class OnlineTripletSiameseNetwork(nn.Module):
    def __init__(self, visual_module, audio_module, text_module, fusion_network):
        """
        Initializes the single processing tower of the Siamese network.
        Its only job is to process ONE batch of movies into embeddings.
        """
        super().__init__()
        self.visual_module = visual_module
        self.audio_module = audio_module
        self.text_module = text_module
        self.fusion_network = fusion_network

    def forward(self, **batch_data):
```

```
        """
        Processes a BATCH of movie data into a BATCH of embeddings.

        Args:
            **batch_data: A dictionary of features. For a batch of size 16,
                          'dino_frame_embeddings' would have a shape of [16, 16, 384],
                          'vggish_embedding' would be [16, 128],
                          'title' would be a list of 16 strings, etc.

        Returns:
            A tensor of embeddings with shape [batch_size, embedding_dim].
        """
        # 1. Process Visual Features
        visual_features = self.visual_module(
            frames=batch_data['frames_for_low_level'],
            precomputed_dino_embeddings=batch_data['dino_frame_embeddings']
        )

        # 2. Process Audio Features
        audio_features = None
        if self.audio_module is not None:
            audio_features = self.audio_module(
                precomputed_vggish_embedding=batch_data.get('vggish_embedding'),
                waveform_for_spec_cnn=batch_data.get('waveform_for_spec_cnn')
            )

        # 3. Process Text Features
        text_features = None
        if self.text_module is not None:
            text_features = self.text_module(
                title=batch_data.get('title'),
                plot=batch_data.get('plot'),
                tfidf_features=batch_data.get('tfidf_features'),
                year=batch_data.get('year'),
                language_idx=batch_data.get('language_idx')
            )

        # 4. Fuse all features into the final embedding
        raw_embedding = self.fusion_network(visual_features, audio_features, text_features)

        # Apply the final L2 normalization ->guarantees the input to loss function is always normalized.
        final_normalized_embedding = F.normalize(raw_embedding, p=2, dim=1, eps=1e-6)

        return final_normalized_embedding
```

Listing E.2: OnlineTripletSiameseNetwork Class

## E.2 Multimodal Fusion Network

The following networks are responsible for fusing the features from the visual, audio, and textual modalities into a single, dense embedding vector. The choice of fusion network was a key variable in the experimental design.

### E.2.1 SimpleFusionNetwork

Used for the Pair-Euclidean and Triplet-Cosine models, this network uses a standard MLP architecture with Layer Normalization for stability.

```
#SIMPLIFIED!
class SimpleFusionNetwork(nn.Module):
    def __init__(self, visual_dim, audio_dim, text_dim, embedding_dim=128, ablation_mode='full'):
        super().__init__()
        self.embedding_dim = embedding_dim
        self.ablation_mode = ablation_mode

        self.fusion_input_dim = 0
        if visual_dim > 0: self.fusion_input_dim += visual_dim
        if (ablation_mode == 'visual_audio' or ablation_mode == 'full') and audio_dim > 0:
            self.fusion_input_dim += audio_dim
        if ablation_mode == 'full' and text_dim > 0:
            self.fusion_input_dim += text_dim

        if self.fusion_input_dim == 0:
            raise ValueError("Fusion input dimension cannot be zero.")

        print(f"Fusion Input Dim: {self.fusion_input_dim}, Final Embedding Dim: {embedding_dim}")

        intermediate_dim = max(embedding_dim * 2, self.fusion_input_dim // 2)
        intermediate_dim = min(intermediate_dim, 1024) # Cap the size

        # A more stable MLP structure
        self.fusion_mlp = nn.Sequential(
            nn.LayerNorm(self.fusion_input_dim),
            nn.Linear(self.fusion_input_dim, intermediate_dim),
            nn.ReLU(),  # for stability
            nn.Dropout(0.5), # Heavier dropout
            nn.Linear(intermediate_dim, embedding_dim)
        )

        self._initialize_weights()
        print(f"       Robust SimpleFusionNetwork (LayerNorm -> Linear -> ReLU -> Dropout -> Linear) initialized.")

    def _initialize_weights(self):
        for module in self.fusion_mlp.modules():
            if isinstance(module, nn.Linear):
                nn.init.xavier_uniform_(module.weight, gain=0.7)
                if module.bias is not None:
                    nn.init.constant_(module.bias, 0)
            elif isinstance(module, nn.LayerNorm):
                nn.init.constant_(module.weight, 1)
                nn.init.constant_(module.bias, 0)

    def forward(self, visual_features, audio_features=None, text_features=None):
        features_to_fuse = []
        if visual_features is not None: features_to_fuse.append(visual_features)

        if (self.ablation_mode == 'visual_audio' or self.ablation_mode == 'full') and audio_features is not None:
            features_to_fuse.append(audio_features)

        if self.ablation_mode == 'full' and text_features is not None:
```

```
            features_to_fuse.append(text_features)

        if not features_to_fuse:
            return torch.zeros((1, self.embedding_dim), device=visual_features.
    device if visual_features is not None else 'cpu')

        fused_input = torch.cat(features_to_fuse, dim=1)

        # Check for NaN/Inf BEFORE the MLP
        if torch.isnan(fused_input).any() or torch.isinf(fused_input).any():
            print("F-     NaN/Inf detected in fused_input before MLP. Clamping.")
            fused_input = torch.nan_to_num(fused_input, nan=0.0, posinf=1.0,
    neginf=-1.0)

        embedding = self.fusion_mlp(fused_input)

        # Final normalization ->l for contrastive loss
        return F.normalize(embedding, p=2, dim=1, eps=1e-8)
```

Listing E.3: SimpleFusionNetwork Class

### E.2.2 UltraSimpleFusionNetwork

Used for the Triplet-Euclidean model, this highly stable architecture consists of a single linear projection layer, minimizing complexity to aid in training the most challenging model configuration.

```
class UltraSimpleFusionNetwork(nn.Module):
    def __init__(self, visual_dim, audio_dim, text_dim, embedding_dim=128,
    ablation_mode='full'):
        super().__init__()
        self.embedding_dim = embedding_dim
        self.ablation_mode = ablation_mode

        self.fusion_input_dim = 0
        if visual_dim > 0: self.fusion_input_dim += visual_dim
        if (ablation_mode == 'visual_audio' or ablation_mode == 'full') and
    audio_dim > 0:
            self.fusion_input_dim += audio_dim
        if ablation_mode == 'full' and text_dim > 0:
            self.fusion_input_dim += text_dim

        if self.fusion_input_dim == 0:
            raise ValueError("Fusion input dimension cannot be zero.")

        print(f"Fusion Input Dim: {self.fusion_input_dim}, Final Embedding Dim: {
    embedding_dim}")

        # --- ULTRA-STABLE MLP ---
        # A direct, normalized projection with no hidden layers.
        # This is the most stable architecture possible.
        self.fusion_mlp = nn.Sequential(
            nn.LayerNorm(self.fusion_input_dim),
            nn.Linear(self.fusion_input_dim, self.embedding_dim)
        )
        # --- END OF CHANGE ---

        self._initialize_weights()
        print(f"       ULTRA-STABLE UltraSimpleFusionNetwork (LayerNorm -> Linear)
     initialized.")
```

```
30
31      def _initialize_weights(self):
32          for module in self.fusion_mlp.modules():
33              if isinstance(module, nn.Linear):
34                  # Small gain for stability
35                  nn.init.xavier_uniform_(module.weight, gain=0.1)
36                  if module.bias is not None:
37                      nn.init.constant_(module.bias, 0)
38              elif isinstance(module, nn.LayerNorm):
39                  nn.init.constant_(module.weight, 1)
40                  nn.init.constant_(module.bias, 0)
41
42      def forward(self, visual_features, audio_features=None, text_features=None):
43          features_to_fuse = []
44          if visual_features is not None: features_to_fuse.append(visual_features)
45
46          if (self.ablation_mode == 'visual_audio' or self.ablation_mode == 'full')
    and audio_features is not None:
47              features_to_fuse.append(audio_features)
48
49          if self.ablation_mode == 'full' and text_features is not None:
50              features_to_fuse.append(text_features)
51
52          if not features_to_fuse:
53              return torch.zeros((1, self.embedding_dim), device='cpu')
54
55          fused_input = torch.cat(features_to_fuse, dim=1)
56
57          if torch.isnan(fused_input).any() or torch.isinf(fused_input).any():
58              fused_input = torch.nan_to_num(fused_input, nan=0.0)
59
60          embedding = self.fusion_mlp(fused_input)
61
62          return F.normalize(embedding, p=2, dim=1, eps=1e-8)
```

Listing E.4: UltraSimpleFusionNetwork Class

### E.2.3 HybridFusionNetwork

Used for the Pair-Cosine model, this is the most complex fusion architecture, employing a cross-attention mechanism to allow modalities to inform one another before a final late-fusion step.

```
1  #REVISED HYBRID MODULE
2  class HybridFusionNetwork(nn.Module):
3      def __init__(self, visual_dim, audio_dim, text_dim, embedding_dim=128,
       num_heads=4, dropout=0.2):
4          super().__init__()
5          self.embedding_dim = embedding_dim
6
7          # --- Cross-Attention (Early Fusion) ---
8          # This part models interactions between modalities
9          self.visual_query = nn.Linear(visual_dim, visual_dim)
10
11         # Audio influences Visual
12         self.audio_kv = nn.Linear(audio_dim, visual_dim * 2) # Key and Value from
       Audio
13         self.va_attention = nn.MultiheadAttention(embed_dim=visual_dim, num_heads=
       num_heads, dropout=dropout, batch_first=True)
14         self.va_norm = nn.LayerNorm(visual_dim)
15
16         # Text influences Visual
```

```
        self.text_kv = nn.Linear(text_dim, visual_dim * 2) # Key and Value from Text
        self.vt_attention = nn.MultiheadAttention(embed_dim=visual_dim, num_heads=num_heads, dropout=dropout, batch_first=True)
        self.vt_norm = nn.LayerNorm(visual_dim)

        # --- Late Fusion MLP ---
        # This part combines the individually processed and the cross-attended features

        # We will have the original visual, the visual-after-audio-attention, and visual-after-text-attention
        late_fusion_input_dim = visual_dim * 3

        self.late_fusion_mlp = nn.Sequential(
            nn.LayerNorm(late_fusion_input_dim),
            nn.Linear(late_fusion_input_dim, embedding_dim * 2),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(embedding_dim * 2, embedding_dim)
        )

        self._initialize_weights()
        print(f"       Revised HybridFusionNetwork initialized.")

    def _initialize_weights(self):
        for module in self.modules():
            if isinstance(module, nn.Linear):
                nn.init.xavier_uniform_(module.weight, gain=nn.init.calculate_gain('relu'))
                if module.bias is not None:
                    nn.init.constant_(module.bias, 0)
            elif isinstance(module, nn.LayerNorm):
                nn.init.constant_(module.weight, 1)
                nn.init.constant_(module.bias, 0)

    def forward(self, visual_features, audio_features, text_features):
        # All inputs should be pre-normalized from the sub-modules

        # --- Early Fusion Part (Cross-Attention) ---
        # Visual features will act as the 'query' that asks questions of other modalities
        q = self.visual_query(visual_features).unsqueeze(1) # (B, 1, D_vis)

        # 1. Audio influences Visual
        audio_k, audio_v = self.audio_kv(audio_features).chunk(2, dim=-1)
        audio_k = audio_k.unsqueeze(1) # (B, 1, D_vis)
        audio_v = audio_v.unsqueeze(1) # (B, 1, D_vis)

        # Attention: Visual asks "what's in the audio?"
        va_out, _ = self.va_attention(query=q, key=audio_k, value=audio_v)
        va_out = va_out.squeeze(1) # (B, D_vis)
        visual_after_audio = self.va_norm(visual_features + va_out)

        # 2. Text influences Visual
        text_k, text_v = self.text_kv(text_features).chunk(2, dim=-1)
        text_k = text_k.unsqueeze(1) # (B, 1, D_vis)
        text_v = text_v.unsqueeze(1) # (B, 1, D_vis)

        # Attention: Visual asks "what's in the text?"
        vt_out, _ = self.vt_attention(query=q, key=text_k, value=text_v)
        vt_out = vt_out.squeeze(1) # (B, D_vis)
```

```
        visual_after_text = self.vt_norm(visual_features + vt_out)

        # --- Late Fusion Part ---
        # Combine the original visual with the attention-modified versions
        late_fusion_input = torch.cat([visual_features, visual_after_audio,
    visual_after_text], dim=1)

        final_embedding = self.late_fusion_mlp(late_fusion_input)

        # Final normalization before the loss function
        return F.normalize(final_embedding, p=2, dim=1, eps=1e-8)
```

Listing E.5: HybridFusionNetwork Class

## E.3 Custom Loss Functions

The following custom loss functions were implemented to handle the two different distance metrics used in the 2x2 experimental design.

### E.3.1 Contrastive Loss

These loss functions are used for the pairwise training setup.

```
class ContrastiveLossCosine(nn.Module):
    def __init__(self, margin=0.5): # A smaller margin is often used for Cosine
    distance
        super(ContrastiveLossCosine, self).__init__()
        self.margin = margin
        self.eps = 1e-9 # For numerical stability

    def forward(self, embedding1, embedding2, label):
        # Cosine Similarity: Range [-1, 1]
        cosine_similarity = F.cosine_similarity(embedding1, embedding2, dim=1, eps
    =self.eps)

        # Cosine Distance: Range [0, 2]
        # Distance = 1 - Similarity
        cosine_distance = 1 - cosine_similarity

        # If label=1 (similar), we want to minimize the distance (i.e., make
    cosine_similarity close to 1).
        # If label=0 (dissimilar), we want the distance to be large (at least `
    margin`).

        loss_positive = (label) * torch.pow(cosine_distance, 2)
        loss_negative = (1 - label) * torch.pow(torch.clamp(self.margin -
    cosine_distance, min=0.0), 2)

        loss_contrastive = torch.mean(loss_positive + loss_negative)

        return loss_contrastive

class ContrastiveLossEuclidean(nn.Module):
    def __init__(self, margin=1.0): # Common default, tune this
        super(ContrastiveLossEuclidean, self).__init__()
        self.margin = margin
        self.eps = 1e-9 # for stability in sqrt

    def forward(self, embedding1, embedding2, label):
```

```
        # Euclidean distance
        # pairwise_distance automatically handles p=2 for Euclidean
        euclidean_distance = F.pairwise_distance(embedding1, embedding2, p=2, eps=
    self.eps)
        # If label=1 (similar), loss is distance^2
        # If label=0 (dissimilar), loss is max(0, margin - distance)^2

        loss_positive = (label) * torch.pow(euclidean_distance, 2)
        loss_negative = (1 - label) * torch.pow(torch.clamp(self.margin -
    euclidean_distance, min=0.0), 2)
        loss_contrastive = torch.mean(loss_positive + loss_negative)

        return loss_contrastive
```

Listing E.6: ContrastiveLossCosine and ContrastiveLossEuclidean Classes

### E.3.2 Triplet Loss

These loss functions are used for the triplet-based training setup.

```
class TripletLossCosine(nn.Module):
    """
    Triplet loss function for cosine similarity.
    The goal is to make the anchor more similar to the positive than to the
    negative,
    by at least a certain margin.

    Objective: sim(a, p) > sim(a, n) + margin
    Loss: max(0, sim(a, n) - sim(a, p) + margin)
    """
    def __init__(self, margin=0.3):
        super(TripletLossCosine, self).__init__()
        self.margin = margin
        self.eps = 1e-9 # For numerical stability

    def forward(self, anchor_emb, positive_emb, negative_emb):
        # All embeddings should be L2-normalized from the model
        sim_pos = F.cosine_similarity(anchor_emb, positive_emb, dim=1, eps=self.
    eps)
        sim_neg = F.cosine_similarity(anchor_emb, negative_emb, dim=1, eps=self.
    eps)

        # Calculate the loss
        loss = torch.clamp(sim_neg - sim_pos + self.margin, min=0.0)

        return torch.mean(loss)

class TripletLossEuclidean(nn.Module):
    """
    Triplet loss function for Euclidean distance.
    The goal is to make the anchor's distance to the positive smaller than its
    distance to the negative, by at least a certain margin.

    Objective: dist(a, p) + margin < dist(a, n)
    Loss: max(0, dist(a, p) - dist(a, n) + margin)
    """
    def __init__(self, margin=1.0):
        super(TripletLossEuclidean, self).__init__()
        self.margin = margin

    def forward(self, anchor_emb, positive_emb, negative_emb):
```

```
        dist_pos = F.pairwise_distance(anchor_emb, positive_emb, p=2)
        dist_neg = F.pairwise_distance(anchor_emb, negative_emb, p=2)

        loss = torch.clamp(dist_pos - dist_neg + self.margin, min=0.0)

        return torch.mean(loss)
```

Listing E.7: TripletLossCosine and TripletLossEuclidean Classes

# Appendix F

# Hyperparameters Configuration

This appendix provides the specific hyperparameter configurations used for training the four end-to-end models described in Chapter 4. These parameters were determined through systematic experimentation and validation to ensure stable training and optimal performance for each model variant.

## F.1 Learning Rate Strategy

A differentiated learning rate strategy was employed for the Pairwise models, managed via the AdamW optimizer. Components that were randomly initialized or required more significant adaptation, such as the visual backbone and the fusion networks, were assigned a higher learning rate (e.g., 1e-4). In contrast, components based on powerful pre-trained text and audio backbones (BERT, VGGish) were assigned a lower learning rate (e.g., 5e-5) to facilitate more conservative fine-tuning and preserve their robust, pre-existing representations. For the Triplet models, a uniform, lower learning rate was found to provide more stable training.

## F.2 Optimisation and Regularisation

The **AdamW** optimiser was selected for its superior performance with transformer-based architectures and its built-in weight decay regularisation. The strength of the weight decay was tuned separately for each model variant, as models using Euclidean distance, which have more degrees of freedom (learning vector magnitude), generally required stronger regularisation (e.g., 1e-2) to prevent overfitting compared to their cosine-based counterparts (e.g., 5e-3).

## F.3 Loss Configuration

Each model variant used a loss function appropriate for its learning paradigm. The pairwise models used **Contrastive Loss**, while the triplet models used **Triplet Margin Loss**. The margin hyperparameter ($m$) was tuned for each distance metric to account for their different scales. The Euclidean models required a larger margin (e.g., 1.0 or 1.2) than the Cosine models (e.g., 0.3 or 0.8), which operate on a more constrained distance space [0, 2].

## F.4 Training Dynamics

All models were trained for a set number of epochs with an **early stopping** mechanism based on validation loss, with a patience of 8-10 epochs to prevent overfitting. For pairwise models, a **ReduceLROnPlateau** scheduler was used to dynamically adjust the learning rate when the validation loss plateaued. The effective batch size was kept consistent across experiments for fair comparison, with triplet models using a larger data loader batch size to ensure a sufficient pool of in-batch negatives for the semi-hard mining strategy.

Table F.1: Final Hyperparameters for End-to-End Model Variants

| **Hyperparameter** | **Pair-Cosine** | **Pair-Euclidean** | **Triplet-Cosine** | **Triplet-Euclidean** |
|---|---|---|---|---|
| **Model & Loss Configuration** | | | | |
| Model Type | Pairwise | Pairwise | Triplet (Online) | Triplet (Online) |
| Distance Metric | Cosine | Euclidean | Cosine | Euclidean |
| Loss Function | Contrastive | Contrastive | Triplet Margin | Triplet Margin |
| Loss Margin ($m$) | 0.8 | 1.2 | 0.3 | 1.0 |
| Fusion Network | Hybrid | Simple | Simple | UltraSimple |
| **Learning Rates**[1] | | | | |
| Visual Backbone | 1e-4 | 5e-5 | 1e-4 | 1e-4 |
| Audio Backbone | 5e-5 | 5e-5 | 1e-4 | 1e-4 |
| Text Backbone | 5e-5 | 5e-5 | 1e-4 | 1e-4 |
| Fusion Network | 1e-4 | 5e-5 | 1e-4 | 1e-4 |
| **Optimisation & Regularisation** | | | | |
| Optimizer | AdamW | AdamW | AdamW | AdamW |
| Weight Decay | 5e-3 | 1e-2 | 1e-2 | 1e-2 |
| LR Scheduler | ReduceLROnPlateau (RLRP) | RLRP | None | None |
| Patience | 3 | 3 | N/A | N/A |
| **Training Dynamics** | | | | |
| Batch Size | 16 | 16 | 16 | 16 |
| Effective Batch | ˜16 | ˜16 | ˜16 (mined) | ˜16 (mined) |
| Max Epochs | 50 | 50 | 100 | 100 |
| Early Stopping | 8 | 8 | 10 | 10 |

[1]For cosine pairwise models, a differentiated learning rate was used. For triplet models, a uniform learning rate across all components was found to be more stable.

# Bibliography


[1] Sami Abu-El-Haija, Joonseok Lee, Max Harper, and Joseph Konstan. Movielens 20m youtube trailers dataset. In *MovieLens*, 2018.

[2] Adolfo Almeida, Johan Pieter de Villiers, Allan De Freitas, and Mergandran Velayudan. The complementarity of a diverse range of deep learning features extracted from video content for video recommendation. *Expert Systems with Applications*, 192:116335, 2022.

[3] Mounir M Bendouch, Flavius Frasincar, and Tarmo Robal. A visual-semantic approach for building content-based recommender systems. *Information Systems*, 117:102243, 2023.

[4] Marco De Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. *Recommender systems handbook*, pages 119–159, 2015.

[5] Yashar Deldjoo, Mehdi Elahi, Paolo Cremonesi, Franca Garzotto, Pietro Piazzolla, and Massimo Quadrana. Content-based video recommendation system based on stylistic visual features. *Journal on Data Semantics*, 5:99–113, 2016.

[6] Xiuqi Deng, Lu Xu, Xiyao Li, Jinkai Yu, Erpeng Xue, Zhongyuan Wang, Di Zhang, Zhaojie Liu, Guorui Zhou, Yang Song, et al. End-to-end training of multimodal model and ranking model. *arXiv preprint arXiv:2404.06078*, 2024.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[8] Mihai Gurban. *Multimodal feature extraction and fusion for audio-visual speech recognition*. PhD thesis, EPFL, 2009.

[9] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[10] Folasade Olubusola Isinkaye, Yetunde O Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3):261–273, 2015.

[11] Jiazheng Jing, Yinan Zhang, Xin Zhou, and Zhiqi Shen. Capturing popularity trends: A simplistic non-personalized approach for enhanced item recommendation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 1014–1024, 2023.

[12] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pages 372–378. IEEE, 2014.

[13] Yaman Kumar, Agniv Sharma, Abhigyan Khaund, Akash Kumar, Ponnurangam Kumaraguru, Rajiv Ratn Shah, and Roger Zimmermann. Icebreaker: Solving cold start problem for video recommendation engines. In *2018 IEEE international symposium on multimedia (ISM)*, pages 217–222. IEEE, 2018.

[14] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.

[15] Sebastian Lubos, Alexander Felfernig, and Markus Tautschnig. An overview of video recommender systems: state-of-the-art and research issues. *Frontiers in big Data*, 6:1281614, 2023.

[16] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[17] Shi Pu, Yijiang He, Zheng Li, and Mao Zheng. Multimodal topic learning for video recommendation. *arXiv preprint arXiv:2010.13373*, 2020.

[18] Michael Pulis and Josef Bajada. Siamese neural networks for content-based cold-start music recommendation. In *Proceedings of the 15th ACM conference on recommender systems*, pages 719–723, 2021.

[19] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook*, pages 1–35, 2021.

[20] Claudia V Roberts, Ehtsham Elahi, and Ashok Chandrashekar. On the bias-variance characteristics of lime and shap in high sparsity movie recommendation explanation tasks. *arXiv preprint arXiv:2206.04784*, 2022.

[21] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004.

[22] Nicolás Serrano and Alejandro Bellogín. Siamese neural networks in recommendation. *Neural Computing and Applications*, 35(19):13941–13953, 2023.

[23] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, pages 257–297, 2011.

[24] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. Multi-modal knowledge graphs for recommender systems. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1405–1414, 2020.

[25] Lev Utkin, Maxim Kovalev, and Ernest Kasimov. Explanation of siamese neural networks for weakly supervised learning. *Computing and Informatics*, 39(6):1172–1202, 2020.

[26] Donghui Wang, Yanchun Liang, Dong Xu, Xiaoyue Feng, and Renchu Guan. A content-based recommender system for computer science publications. *Knowledge-Based Systems*, 157:1–9, 2018.