PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research



UNIVERSITY OF ECHAHID HAMMA LAKHDAR – EL OUED
FACULTY OF EXACT SCIENCES
Computer Science Department



Thesis
submitted in partial fulfillment of the requirements for the degree of

ACADEMIC MASTER

Field: Mathematics and Computer Science
Option: Computer Science
Specialty: Artificial Intelligence and Data Science

Submitted by:

Touka Habita
Malak Elhossen Harrache

A Multimodal Deep Learning Model for Classifying Social Media Content According to Islamic Sharia Principles

Defended on: June 2026

Examination Committee:

M.	**Ahmed Ghenabzia**	MCB	Supervisor
M.	**Mouadh Bali**	MAA	Co-Supervisor
M.	**Brahim Lajdel**	Professor	Chair
M.	**Adel Berhoum**	MCB	Examiner

Academic year: 2025–2026

Abstract

Facebook Reels are short multimodal videos whose meaning is formed through visual and audio content. From an Islamic perspective, some reels may contain elements that conflict with Sharia-based values or negatively affect the recipient religiously orbehaviorally. This motivates the need for automatic reel classification using culturally aware criteria.

This thesis designs and develops a multimodal deep learning system for classifying Facebook Reels from an Islamic Sharia-based perspective. The dataset was collected from Facebook using automated link extraction and browsing-based collection strategies. The collected reels were then manually annotated according to three categories: Good, Acceptable, and Bad. These labels were used during model development, while the final system output was organized into compliant and non-compliant content.

The proposed approach uses pretrained audio and visual encoders to extract separate representations from each reel. Audio embeddings and visual embeddings are then organized, aligned with their labels, and used to train classification models. Multimodal fusion is applied to combine the two modalities, and knowledge distillation is adopted to transfer knowledge from larger teacher models to lighter student models for more efficient deployment.

This work contributes to culturally grounded artificial intelligence by applying Sharia-based annotation criteria to multimodal short-video classification and by building a framework that supports the automatic detection of non-compliant reel content.

Keywords: Multimodal Deep Learning, Islamic Sharia Principles, Social Media Content Classification, Knowledge Distillation.

Résumé

Les Facebook Reels sont de courtes vidéos multimodales dont le sens se construit à travers le contenu visuel et audio. Dans une perspective islamique, certains reels peuvent contenir des éléments contraires aux valeurs fondées sur la Charia ou affecter négativement le récepteur sur les plans religieux ou comportemental. Cela met en évidence la nécessité d'une classification automatique des reels selon des critères sensibles au contexte culturel.

Cette thèse conçoit et développe un système d'apprentissage profond multimodal pour classer les Facebook Reels selon une perspective fondée sur la Charia islamique. Le jeu de données a été collecté à partir de Facebook à l'aide de stratégies d'extraction automatique de liens et de collecte basée sur la navigation. Les reels collectés ont ensuite été annotés manuellement selon trois catégories : Good, Acceptable et Bad. Ces étiquettes ont été utilisées pendant le développement du modèle, tandis que la sortie finale du système a été organisée en contenu conforme et non conforme.

L'approche proposée utilise des encodeurs audio et visuels pré-entraînés afin d'extraire des représentations distinctes de chaque reel. Les embeddings audio et visuels sont ensuite organisés, alignés avec leurs étiquettes, puis utilisés pour entraîner des modèles de classification. La fusion multimodale est appliquée afin de combiner les deux modalités, et la distillation des connaissances est adoptée pour transférer les connaissances de grands modèles enseignants vers des modèles étudiants plus légers, en vue d'un déploiement plus efficace.

Ce travail contribue à l'intelligence artificielle culturellement ancrée en appliquant des critères d'annotation fondés sur la Charia à la classification multimodale de courtes vidéos, et en construisant un cadre qui soutient la détection automatique des contenus de reels non conformes.

Mots-clés : Apprentissage profond multimodal, principes de la charia islamique, classification du contenu des médias sociaux, distillation des connaissances.

الملخص

تعد Facebook Reels مقاطع فيديو قصيرة متعددة الوسائط، يتشكل معناها من خلال المحتوى المرئي والسمعي. ومن منظور إسلامي، قد تحتوي بعض المقاطع على عناصر تتعارض مع القيم المستندة إلى الشريعة، أو تؤثر سلبا في المتلقي من الناحية الدينية أو السلوكية. وهذا يبرز الحاجة إلى التصنيف الآلي لمقاطع Reels باستخدام معايير واعية بالسياق الثقافي.

تصمم هذه المذكرة وتطور نظام تعلم عميق متعدد الوسائط لتصنيف مقاطع Facebook Reels من منظور قائم على الشريعة الإسلامية. تم جمع مجموعة البيانات من Facebook باستخدام استراتيجيات الاستخراج الآلي للروابط والجمع القائم على التصفح. بعد ذلك، تم وسم المقاطع المجمعة يدويا وفق ثلاث فئات: جيد، مقبول، وسيئ. استخدمت هذه الوسوم أثناء تطوير النموذج، بينما نُظم خرج النظام النهائي إلى محتوى متوافق ومحتوى غير متوافق.

تعتمد المقاربة المقترحة على مرمزات سمعية وبصرية مسبقة التدريب لاستخراج تمثيلات منفصلة من كل مقطع. ثم يتم تنظيم التمثيلات السمعية والبصرية ومواءمتها مع وسومها، واستعمالها لتدريب نماذج التصنيف. كما يتم تطبيق الدمج متعدد الوسائط للجمع بين النمطين، واعتماد تقطير المعرفة لنقل المعرفة من نماذج معلم أكبر إلى نماذج طالب أخف، بهدف تحقيق نشر أكثر كفاءة.

يسهم هذا العمل في الذكاء الاصطناعي المرتكز على السياق الثقافي من خلال تطبيق معايير وسم قائمة على الشريعة في تصنيف مقاطع الفيديو القصيرة متعددة الوسائط، وبناء إطار يدعم الكشف الآلي عن محتوى Reels غير المتوافق.

الكلمات المفتاحية: التعلم العميق متعدد الوسائط، مبادئ الشريعة الإسلامية، تصنيف محتوى وسائل التواصل الاجتماعي، تقطير المعرفة.

Contents

Abstract ii

Résumé iii

Arabic abstract iv

List of Figures x

List of Tables xii

List of Abbreviations xiv

General Introduction **1**

1 Background **3**

1.1 Introduction 3

1.2 Social media platforms 3

1.2.1 Definition and main components 3

1.2.2 The dual impact of social media on individuals and society 4

1.2.3 Facebook reels in the Algerian context 4

1.3 The rise of short-form video content 4

1.3.1 Definition and components of short-form video 4

1.3.2 The scientific sensitivity of short videos 5

1.3.3 Social and behavioral influence of reels 6

1.4 Digital video as structured data 6

1.4.1 Technical properties of digital video 7

1.4.2 Computational understanding of short videos 7

1.5 Reels from the perspective of islamic shariah 7

1.5.1 Shariah-related violations and transgressions in reels 8

1.5.2 The impact of reels on the muslim recipient 8

1.5.3 The need to evaluate and guide reels content 8

1.6 Automatic classification and the role of human annotation 9
1.7 Islamic sharia-based content evaluation 9
1.7.1 Content categories in previous islamic media guidelines 10
1.7.2 Boundary cases and content ambiguity 10
1.8 Conclusion 11

2 Theoretical foundations of multimodal learning for short-video classification 12

2.1 Introduction 12
2.2 Multimodal classification 12
2.2.1 Classification types 12
2.2.2 Multimodal learning 13
2.3 Data annotation and ground-truth construction 13
2.3.1 Annotation approaches 14
2.3.2 Label aggregation 14
2.4 Data preprocessing for machine learning 15
2.5 Representation learning (encoders, embeddings, and attention) 15
2.5.1 Encoders 16
2.5.2 Embeddings 16
2.5.3 Attention and transformer encoders 18
2.6 Pretrained models, foundation models, and transfer learning 19
2.6.1 Using pretrained models 19
2.6.2 Audio foundation models 20
2.6.3 Visual and video foundation models 20
2.7 Classification models 21
2.8 Fusion levels 22
2.9 Knowledge distillation and efficient learning 22
2.9.1 Motivation for distillation 23
2.9.2 Types of distillation 23
2.9.3 Common training losses 24
2.10 Related works 24
2.10.1 Zamry et al. — Islamic web content filtering and categorization . . 24
2.10.2 Masadeh et al. — Religious Arabic hatred speech in social networks 25

2.10.3 Das et al. — HateMM: A multimodal dataset for hate video classification . . . 25
2.10.4 Céspedes-Sarrias et al. — MM-HSD: multimodal hate speech detection in videos . . . 25
2.10.5 Zaghouani et al. — MAHED 2025 shared task . . . 25
2.11 Comparative analysis of previous works . . . 25
2.12 Conclusion . . . 27

3 Contribution 1: Dataset Collection and Annotation 28
3.1 Introduction . . . 28
3.2 Chapter roadmap . . . 28
3.3 The need for dataset construction . . . 29
3.4 Collection strategies . . . 29
3.4.1 Initial page-based collection strategy . . . 29
3.4.2 Revised data collection strategy based on account browsing . . . 30
3.5 The labeling website as an operational environment . . . 30
3.6 Storage strategies . . . 31
3.6.1 Initial storage strategy . . . 31
3.6.2 Improved storage strategy . . . 31
3.7 From collected links to usable videos . . . 32
3.8 Human annotation strategy . . . 32
3.9 Annotation scheme and labeling rules . . . 33
3.10 Exploratory data analysis of the final dataset . . . 34
3.11 Conclusion . . . 37

4 Contribution 2: Audio-Visual Representation Learning and Reel Classification 38
4.1 Introduction . . . 38
4.2 Chapter roadmap . . . 38
4.3 Model selection for audio-visual representation . . . 39
4.4 Audio branch: from audio stream to audio embedding . . . 39
4.4.1 Audio stream extraction . . . 40
4.4.2 Audio standardization . . . 40
4.4.3 Kimi-Audio-Whisper encoder . . . 41

4.4.4 Audio embedding generation . . . 41
4.5 Visual branch: from sampled frames to visual embedding . . . 42
4.5.1 Frame selection and sampling . . . 42
4.5.2 SigLIP-Based visual encoder inside SmolVLM-500M . . . 43
4.5.3 Visual feature extraction and embedding generation . . . 43
4.6 Embedding organization and label alignment . . . 44
4.7 Classification dataset preparation . . . 44
4.8 Classification models . . . 45
4.8.1 XGBoost classifier . . . 45
4.8.2 MLP classifier . . . 46
4.9 Evaluation and model selection . . . 47
4.9.1 Overall comparison on fused features . . . 47
4.9.2 Per-class three-class comparison . . . 48
4.9.3 Final model selection . . . 48
4.10 Selected model analysis . . . 49
4.10.1 Fused MLP training behaviour . . . 49
4.10.2 Bad versus Non-Bad output analysis . . . 50
4.11 Conclusion . . . 51

5 Contribution 3: Multimodal knowledge distillation and fusion methodology 52

5.1 Introduction . . . 52
5.2 Chapter roadmap . . . 52
5.3 Motivation for a student-based audio-visual approach . . . 53
5.4 Pipeline overview . . . 54
5.5 Audio knowledge distillation methodology . . . 54
5.5.1 Audio teacher signal . . . 55
5.5.2 Audio student design . . . 55
5.5.3 Audio training signals and exported outputs . . . 56
5.6 Video knowledge distillation methodology . . . 57
5.6.1 Video teacher signal . . . 57
5.6.2 Video student design . . . 57

5.6.3 Video training signals and exported outputs . . . 58
5.7 audio-visual fusion strategy . . . 59
5.7.1 Late fusion rationale and input design . . . 60
5.7.2 Candidate fusion representations . . . 61
5.7.3 Final decision logic . . . 61
5.8 Conclusion . . . 63

6 System implementation **64**
6.1 Introduction . . . 64
6.2 Environment specifications . . . 64
6.3 Tools and libraries . . . 65
6.4 Model evaluation metrics . . . 67
6.5 Labeling platform and data management . . . 68
6.5.1 Collection script implementation . . . 69
6.5.2 Video storage and delivery . . . 70
6.5.3 Backend implementation . . . 70
6.5.4 Frontend and interface screens . . . 71
6.6 Model implementation and runtime cost . . . 73
6.6.1 Teacher-Based pipeline size and memory usage . . . 73
6.6.2 Final KD package size and memory usage . . . 74
6.7 Reels Guard browser extension implementation . . . 75
6.7.1 Extension interface and monitoring . . . 75
6.7.2 Extension workflow and model integration . . . 76
6.7.2.1 Reel detection . . . 76
6.7.2.2 Backend communication . . . 77
6.7.2.3 Model integration . . . 77
6.7.2.4 Report retrieval and visualization . . . 77
6.8 Conclusion . . . 78

General Conclusion **79**

List of Figures

1.1 Comparison between long-form and short-form video. 5

1.2 Digital video as structured multimodal data. 6

1.3 Diagram for the practical motivation behind automatic content classification. 9

2.1 General overview of multimodal learning and its main forms. 13

2.2 General pipeline from data annotation to ground-truth construction. 14

2.3 Examples of embeddings from raw data to numerical representations. . . . 17

2.4 Transformation of raw inputs into fixed-size embeddings for downstream models. 17

2.5 Main components of a Transformer encoder. 18

2.6 Teacher–student knowledge distillation using output and representation-level signals. 24

3.1 Roadmap of the main stages covered in this chapter. 28

3.2 Comparison of reel collection strategies. 30

3.3 Role of the labeling website in data management and annotation support. . 31

3.4 Evolution of the storage strategy from Supabase Storage to a dedicated external server. 32

3.5 Main workflow from the master CSV file to usable videos prepared for annotation. 32

3.6 Human annotation workflow showing how each video is reviewed and labeled within the platform. 33

3.7 Overall pipeline from reel collection to the construction of the final labeled dataset. 34

3.8 Distribution of the final labels in the dataset. 35

3.9 Daily number of first-labeled videos during dataset construction. 36

3.10 Daily percentage of the three quality labels during dataset construction. . . 36

4.1 Overview of the complete pipeline from labeled reels to the final classification model . . . 39
4.2 Audio format standardization before audio embedding extraction. 40
4.3 Audio representation generation using the Kimi/Whisper encoder, mean pooling, and L2 normalization. 42
4.4 Sampling one frame per second from each reel and preparing the selected frames for the visual encoder. 43
4.5 Generation of the final visual embedding from sampled RGB frames 44
4.6 Construction of three feature sets from the extracted embeddings 45
4.7 Confusion matrix of the selected fused MLP classifier on the test set. . . . 49
4.8 Training and validation curves of the selected fused MLP classifier used to monitor convergence and possible overfitting. 50

5.1 Roadmap of the student-based distillation and fusion methodology. 52
5.2 Overall knowledge distillation and fusion pipeline. 54
5.3 Audio distillation flow between the Kimi teacher and the Whisper-base student. 54
5.4 Video distillation flow between the frozen teacher and the student branch. 57
5.5 Fusion flow from audio-video student outputs to the final binary decision. . 60
5.6 Final decision logic of the selected fusion head. 62

6.1 Overall workflow of the web-based labeling and dataset construction system. 69
6.2 Supabase backend database schema 71
6.3 Desktop view of the web-based labeling interface. 72
6.4 Mobile views of the labeling interface. 72
6.5 Comparison between the teacher-based pipeline and the final KD package. 73
6.6 Overall workflow of the Reels Guard browser extension. 75
6.7 Reels Guard extension monitoring interfaces. 76
6.8 Daily report interface generated by the Reels Guard extension. 78

List of Tables

1.1 Main technical properties of digital video 7

2.1 Examples of common audio models and their general structure 20
2.2 Examples of common visual and video models and their general structure. 20
2.3 Common multimodal fusion strategies. 22
2.4 Comparative analysis of selected previous works. 26

3.1 Basic dataset statistics. 35

4.1 Main configuration of the XGBoost classifier. 46
4.2 Main training configuration of the MLP classifier. 46
4.3 Overall performance comparison between XGBoost and MLP using fused audio-visual embeddings. 48
4.4 Per-class comparison between XGBoost and MLP using fused audio-visual embeddings. 48
4.5 Training behaviour of the selected fused MLP classifier. 49
4.6 Binary evaluation of the fused MLP outputs without training a separate binary classifier. 51

5.1 Mixed teacher-student comparison focused on distillation motivation. . . . 53
5.2 General architecture of the Whisper-base audio student backbone. 55
5.3 Audio student training setup and selected checkpoint. 56
5.4 Exported audio outputs used for fusion. 56
5.5 General architecture of the MobileCLIP/FastViT video student backbone. 58
5.6 Video student training setup and selected checkpoint. 59
5.7 Exported video outputs used for fusion. 59
5.8 Candidate fusion representations. 61
5.9 Selected fusion MLP training configuration. 61

5.10 Best three-class fusion heads on student outputs. 62
5.11 Binary adaptation of the selected fusion model. 63

6.1 Environment specifications used in the implementation. 65
6.2 Main tools and libraries used in the project. 66
6.3 Model evaluation metrics used in this work. 68
6.4 Final teacher-based pipeline size and runtime memory usage. 74
6.5 Final KD package size and loading memory cost. 74

List of Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
CSS	Cascading Style Sheets
FFmpeg	Fast Forward MPEG
F1-score	Harmonic mean of precision and recall
GPU	Graphics Processing Unit
HD	High Definition
HTML	HyperText Markup Language
JS	JavaScript
KD	Knowledge Distillation
ML	Machine Learning
MLP	Multilayer Perceptron
NVMe	Non-Volatile Memory Express
RAM	Random Access Memory
RGB	Red, Green, and Blue
SD	Standard Definition
SFTP	Secure File Transfer Protocol
SVM	Support Vector Machine
SSD	Solid-State Drive
TL	Transfer Learning
URL	Uniform Resource Locator
ViT	Vision Transformer
VLM	Vision-Language Model
VPS	Virtual Private Server
VRAM	Video Random Access Memory

General Introduction

Background and Context

Short-form videos have become a dominant content format on social media platforms, carrying not only entertainment value but also cultural messages, behavioral patterns, and value systems. Among these platforms, Facebook Reels—brief, mobile-optimized, vertically-oriented videos (typically under 60 seconds)—have achieved particular prominence, especially in the Algerian context where Facebook remains the most widely used social media platform [1, 2].

The multimodal nature of Reels means their meaning is not formed by visual content alone, but through the complex interaction of visuals, speech, music, text overlays, timing, gestures, and interactive elements [3, 4, 5]. Understanding a reel requires computational analysis of multiple information channels working in concert.

From an Islamic Sharia-oriented perspective, short-form video content raises particular concerns [6]. Reels may contain elements that can be interpreted as violating Islamic principles or implicitly undermine Islamic values. Research on social media and mental health research suggests associations between extensive short-form video use and declining cognitive functioning, increased anxiety and stress, and reduced psychological well-being.

Research Gap and Motivation

Several critical gaps exist in the research landscape:

- **Absence of Task-Specific Datasets:**No publicly available datasets exist specifically for classifying Arabic Facebook Reels according to Islamic Sharia-based criteria.
- **Limited Arabic and Islamic Content Analysis:**Extracting meaningful representations from short videos requires handling temporal compression, variable quality, multilingual audio, and the challenge of capturing meaning that emerges from multimodal interaction.
- **Practical Deployment Barriers:**Large pretrained models capable of generating rich representations are computationally expensive, creating tension between model quality and practical deployability.
- **Integration Gap:**No existing work combines dataset construction, multimodal representation learning,classification, efficiency optimization, and practical system deployment within a coherent framework.

Problem Statement

This thesis addresses three interconnected problems:

- **Data Construction and Annotation:**How can an Arabic Facebook Reels dataset be systematically collected and reliably annotated according to Islamic Sharia-based criteria?
- **Multimodal Representation and Classification:**How can audio and visual modalities be effectively extracted, represented, and fused for short-video classification?
- **Practical System Efficiency:**How can a classification system be made practical for real-world deployment when large pretrained models are computationally expensive?

Hypothesis

- **Hypothesis 1:** Carefully designed collection and annotation strategies can produce high-quality labeled datasets of Arabic Facebook Reels suitable for training culturally and religiously-based classifiers.
- **Hypothesis 2:** Combining audio and visual representations significantly improves short-video classification performance compared with single-modality approaches, as reels communicate meaning through the interaction of multiple information channels.
- **Hypothesis 3:** A lightweight student-based system can preserve the multimodal classification capability of larger teacher models while reducing computational cost substantially (>90%), enabling practical real-world deployment.

Thesis Contributions and Organization

Chapter 1 — Background explains social media platforms, the rise of short-form video, the scientific sensitivity of reels (algorithmic amplification, multimodal persuasion, fragmented attention), and the Islamic Sharia-based evaluative framework that motivates the three-class taxonomy — Good, Acceptable, and Bad. It also justifies the choice of Facebook Reels in the Algerian context.

Chapter 2 — Theoretical foundations of multimodal learning for short-video classification builds the conceptual framework needed for all subsequent chapters. It covers supervised classification, multimodal learning, data annotation, preprocessing, representation learning, pretrained encoders, classifiers, fusion strategies, and knowledge distillation. It closes with a comparative review of related work and identifies the research gap.

Chapter 3— Contribution 1: Dataset Collection and Annotation presents the original dataset built for this thesis. It describes the two collection strategies (page-based, then browsing-based), the labeling platform, the three-annotator rule, the annotation scheme, and the construction of the final labeled CSV dataset.

Chapter4— Contribution 2: Audio-Visual Representation Learning and Reel Classification presents the first classification system. It details the audio branch (Kimi-Audio encoder → embedding), the visual branch (SmolVLM-500M / SigLIP encoder → embedding), the fusion strategy, and the comparison of XGBoost and MLP classifiers on fused features. The fused MLP is selected as the best model, and its outputs are also analyzed under a binary Good+Acceptable vs. Bad reading.

Chapter5— Contribution 3:Multimodal knowledge distillation and fusion methodology addresses the efficiency problem of the teacher-based system. It introduces the student-based pipeline: the Kimi-Audio teacher is distilled into a Whisper-base student for audio, and the SmolVLM/SigLIP teacher is distilled into a MobileCLIP/FastViT student for video. Both students are trained with hybrid signals (teacher embeddings, teacher probabilities, and true labels) and their outputs are combined via a late-fusion head to produce the final three-class prediction.

Chapter 6— System Implementation describes the full practical deployment. It details the execution environment, tools, backend, storage, and labeling platform. It reports the runtime cost of the teacher pipeline against the final KD package (on disk and RAM), and presents the **Reels Guard browser extension** published on Microsoft Edge Add-ons.

Chapter 1

Background

1.1 Introduction

Social media platforms have become major environments for the circulation of digital content, especially short-form video. Because reels combine rapid exposure, high engagement, and multimodal communication, they have become an important subject of analytical study rather than a simple entertainment format.

For this reason, this chapter presents the general background needed to understand short-form video content. It begins with social media platforms and the rise of reels, then discusses their scientific sensitivity, the structure of digital video, the main challenges of automatic understanding, and the role of content evaluation and human annotation.

1.2 Social media platforms

1.2.1 Definition and main components

Social media platforms are online communication services through which users create online communities and share information, ideas, personal messages, and different forms of content, including videos [7]. They have become part of the daily life of many Internet users because they support interaction, access to information, news following, and engagement with online goods and services [7].

In this study, the main content components of social media platforms are summarized as follows:

Posts: regular published content that may include text, images, links, or videos.

Stories: short visual content designed for quick viewing and temporary sharing.

Reels: short-form videos created for fast consumption, engagement, and easy sharing.

Official platform documentation presents posts, stories, and reels as major content formats that can be published across Facebook and Instagram through Meta Business Suite [8].

1.2.2 The dual impact of social media on individuals and society

Social media has become a powerful force shaping individual lives and society at large, producing both significant benefits and serious harms.

- **Positive effects:** Platforms facilitate easy access to information, seamless learning, and global collaboration. Studies show social media enhances accessibility to knowledge and fosters collaborative learning, with platforms like WhatsApp and YouTube improving learner participation by approximately 40% [9].
- **Negative effects:** Social media also drives the spread of superficial and misleading content. Research links social media use to declining psychological resilience, rising polarization, and the erosion of shared civic values [10]. Exposure to vulgar or negative content diminishes moral sensitivity and encourages superficial behavior [11]. Moreover, social comparison on these platforms is associated with increased depression, anxiety, and reduced well-being [12, 13, 14].

1.2.3 Facebook reels in the Algerian context

Facebook occupies a central position in the Algerian social-media landscape and remains the most widely used tracked platform in the country. Its broad reach and continued dominance make it a particularly important environment for understanding the circulation and influence of reels in the Algerian context [2].

1.3 The rise of short-form video content

1.3.1 Definition and components of short-form video

Short-form video is a digital content format characterized by brief, mobile-optimized clips, typically lasting from a few seconds to a few minutes, designed for fast-paced and easily digestible information [15].

The main components of this format include:

- **Temporal brevity**: Extremely short duration, usually under 60 seconds for platforms like TikTok and Reels, focusing on quick consumption [15].
- **Vertical orientation**: Full-screen vertical aspect ratio tailored for smartphone viewing [16, 17].

- **Multimodal communication**: Integration of visuals, audio, text overlays, and interactive elements (e.g., filters, stickers) to enhance engagement [5].
- **Immersive experience**: Seamless, swipe-based navigation that encourages continuous viewing and content sharing [18].

Short-form videos are therefore best understood not as a singular type but as a complex, cross-platform artifact whose meaning is generated through the interplay of these technical and social components [19], as summarized in Figure 1.1.

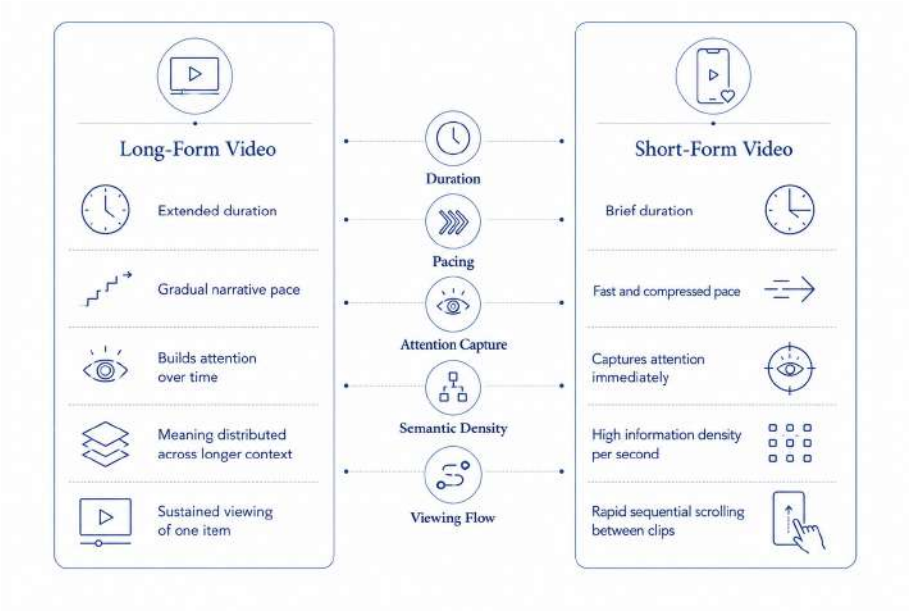


Figure 1.1: Comparison between long-form and short-form video.

1.3.2 The scientific sensitivity of short videos

Short-form videos are scientifically sensitive because their effect results from platform design, rapid exposure, and multimodal persuasion rather than from duration alone. Their importance can be summarized through the following factors:

- **Rapid exposure:** users can watch many clips in a short time, which reduces reflection time and increases dependence on immediate cues [20].
- **Algorithmic amplification:** recommendation systems can increase the visibility of selected clips beyond ordinary chronological exposure [21, 22].
- **Fragmented attention:** frequent switching between clips may reduce sustained attention and viewing satisfaction [23].
- **Multimodal persuasion:** meaning is often produced by the joint effect of visuals, speech, music, text, and timing [3, 4].

These factors explain why short videos should be treated as a serious analytical object rather than as a simple entertainment format.

1.3.3 Social and behavioral influence of reels

Reels can influence users beyond simple information delivery. Because they are consumed in recommendation-driven sequences, their effect is often cumulative rather than isolated.

Their influence can be viewed through three main levels:

- **Cognitive level:** reels compete for attention through speed, novelty, and salience.
- **Affective level:** tone, music, and framing can trigger emotional reactions.
- **Behavioral level:** repeated exposure may shape preferences, imitation, and future engagement with similar content.
- **Cognitive and mental health level:**increased use of short-form videos (e.g., Reels) is associated with poorer cognitive functioning (notably attention and inhibitory control), as well as higher levels of stress and anxiety. These effects appear consistently across different age groups and platforms[24].

1.4 Digital video as structured data

A digital video is more than a collection of pictures. It is a structured spatio-temporal signal in which frames are ordered over time and are often accompanied by an audio track. In many practical cases, additional information is embedded in the clip through captions, subtitles, stickers, or overlaid text. For this reason, video understanding involves the interpretation of multiple interacting information channels rather than the inspection of isolated images [25, 26, 4], as illustrated in Figure 1.2.

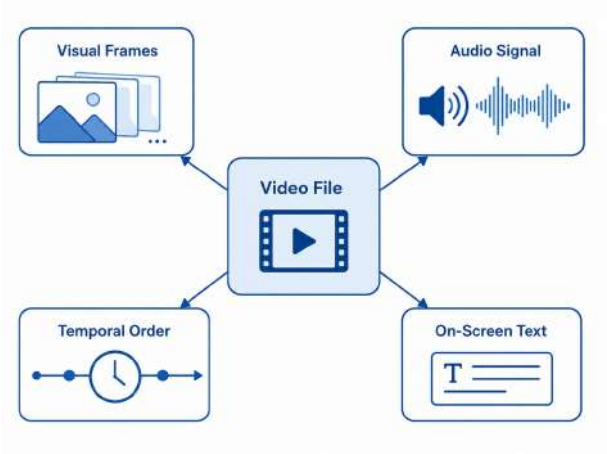


Figure 1.2: Digital video as structured multimodal data.

1.4.1 Technical properties of digital video

From the perspective of computational analysis, several technical properties influence how a video is represented and processed, as listed and explained in Table 1.1:

Property	**Meaning**
Frame rate	Number of frames displayed per second.
Resolution	Pixel dimensions of each video frame.
Duration	Total length of the video clip.
Compression	Encoding process that reduces file size but may affect quality.
Audio track	Speech, music, silence, or other sound events.
Orientation	Video layout, such as vertical, square, or horizontal format.
Text overlays	Captions, subtitles, stickers, or written elements inside the frame.

Table 1.1: Main technical properties of digital video

1.4.2 Computational understanding of short videos

From a computational perspective, short videos present a unique data structure that differs from traditional long-form video. Their analysis requires handling high semantic density within very short temporal windows [27, 28].

Key computational characteristics include:

- **High temporal density**: Meaning is compressed into few seconds, demanding fine-grained feature extraction.
- **No long-range dependency**: Unlike movies or lectures, short videos lack extended narrative structure, making recurrent models less effective.
- **Compression artifacts**: Heavy compression for mobile streaming can degrade edge details, affecting optical flow and object detection.
- **Variable frame rate and resolution**: User-generated content often has non-standard parameters, complicating batch processing.

These properties distinguish short videos from traditional video datasets and motivate specialized pre-processing, sampling strategies, and lightweight architectures suitable for real-time analysis.

1.5 Reels from the perspective of islamic shariah

Reels content is among the most prominent contemporary digital phenomena that require Shariah-based evaluation and an examination of its effects on the Muslim recipient.

1.5.1 Shariah-related violations and transgressions in reels

Reels include a range of Shariah violations. Some of them contain exposure of the *awrah*, or what falls under its ruling, deliberate arousal of desires, or the presentation of scenes that contradict Islamic moral guidelines. Other violations appear in the promotion of deviant behaviours, or in presenting them in an entertaining form that lessens their Shariah-based reprehensibility. In addition, forms of mockery of religious values or the denigration of moral symbols have become widespread. All of this conflicts with the objectives of Shariah in preserving religion and intellect, and in safeguarding the moral integrity of society[6][1]

1.5.2 The impact of reels on the muslim recipient

The uncontrolled use of reels produces negative effects on the Muslim recipient from several aspects[6][1]:

- From the perspective of faith, continuous exposure to fast and successive content weakens the presence of the heart and distracts attention from remembrance of Allah and spiritual meanings. Over time, this establishes a state of constant preoccupation that competes with acts of worship and reduces humility and attentiveness in worship.
- From the moral perspective, the continuous flow of content contributes to normalizing certain behaviours that contradict Shariah values, such as mockery, backbiting, or the display of exaggerated lifestyles.
- Repeated exposure to content that is not Shariah-compliant gradually affects the recipient's standards of what is *Halal* and *Haram*, making reprehensible acts appear familiar.

1.5.3 The need to evaluate and guide reels content

This requires activating platform monitoring tools to block clips that violate Islamic guidelines, while developing artificial intelligence mechanisms that take into account the cultural and religious specificities of society. It also requires strengthening the role of educational and media institutions in producing purposeful alternatives concerned with beneficial values and permissible content. Content creators who offer beneficial material should also be encouraged to invest in these platforms to spread useful knowledge and sound ideas, thereby transforming this tool from a source of harm into a source of benefit, and from a means of wasting time into a means of making it fruitful[6][1].

[1]The original Arabic consultation notes were translated into English by HABITA Anwaar for inclusion in this thesis.

1.6 Automatic classification and the role of human annotation

The volume and speed of short-video circulation make fully manual analysis difficult in real platform environments. Automatic classification is therefore needed as a scalable support layer. Its practical value appears in:

- supporting large-scale monitoring and organization,
- prioritizing videos that require closer human review,
- enabling structured filtering and downstream analysis.

This logic is already reflected in platform practice. Meta and YouTube explain that automated systems are used to help identify, organize, or prioritize content, while human review remains necessary for contextual decisions [29, 30, 31, 32], as illustrated in Figure 1.3.

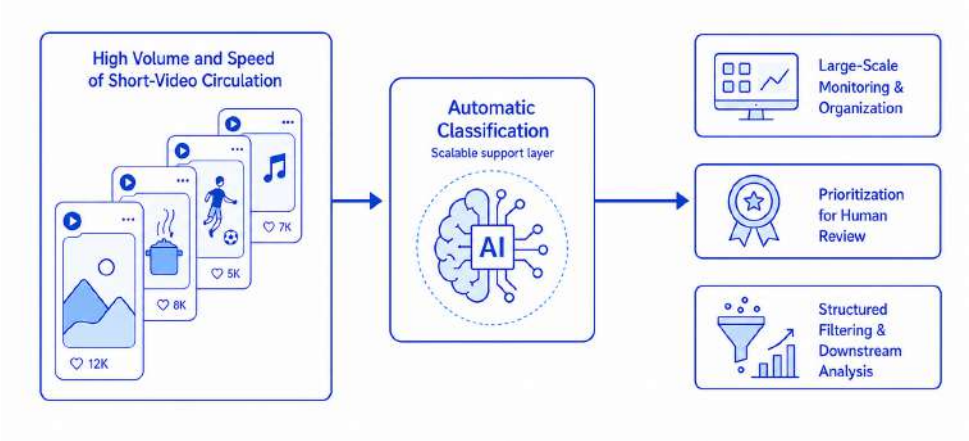


Figure 1.3: Diagram for the practical motivation behind automatic content classification.

In supervised classification, human annotation provides the ground-truth labels that the model learns from. This is especially important in Islamic-based video analysis, where labels cannot be derived reliably from metadata alone because the categories require interpretation.

Annotation research also shows that inter-coder agreement is important for conceptual clarity, and that repeated labeling can improve effective label quality [33, 34, 35]. For this reason, annotation is treated in this thesis as a central methodological component, not as a minor preprocessing step.

1.7 Islamic sharia-based content evaluation

Islamic Sharia-based evaluation considers content benefit, permissibility, and conflict with Islamic principles.

1.7.1 Content categories in previous islamic media guidelines

Previous Islamic media guidelines and studies show that social media content can be evaluated according to its benefit, permissibility, and possible conflict with Islamic principles. These sources provide a close conceptual basis for organizing content into graded categories.

The main categories can be summarized as follows:

- **Beneficial and positive content:** content that spreads truthful, beneficial, and positive information, supports Islamic values, strengthens religious understanding, and contributes to good conduct and social harmony [36, 37].
- **Permissible content:** content that remains within lawful and acceptable boundaries, as long as it does not violate Shariah principles or promote prohibited elements [37].
- **Prohibited or harmful content:** content that damages *aqidah*, *syariah*, or *akhlak*, or includes elements such as deviant beliefs, mockery of Islam, superstition, slander, false information, immoral behavior, or lifestyles contrary to Islamic norms [36, 37].

Based on this literature, the present thesis later operationalizes these orientations into three practical annotation classes: *Good*, *Acceptable*, and *Bad*.

1.7.2 Boundary cases and content ambiguity

Previous Islamic media guidelines show that not all digital content can be evaluated only through an explicit violation or a clear benefit. Some digital content may appear permissible at first glance, but its full evaluation can change when implicit cues, brief elements, or contextual meanings are considered [36, 37].

The main sources of ambiguity include:

- **Implicit religious concern:** the problematic meaning may appear through suggestion, behavior, symbolism, or music rather than through a direct statement.
- **Cross-modal contradiction:** the visual content may seem acceptable, while the audio, text overlay, or context changes the overall meaning.
- **Short-lived cues:** an important word, gesture, symbol, or scene may appear briefly and affect the final judgment.

1.8 Conclusion

This chapter provided the general background for the study of Arabic short-video classification. It first described social media as a recommendation-driven content ecosystem and then explained the emergence of short-form video as a high-density, mobile-native content format. It also clarified why reels are scientifically sensitive: they combine rapid exposure, repetition, algorithmic amplification, fragmented attention, and multimodal persuasion .

The chapter also justified the choice of Facebook Reels in the Algerian context and showed that digital video should be treated as structured multimodal data composed of visual, audio, textual, and temporal components.

Chapter 2

Theoretical foundations of multimodal learning for short-video classification

2.1 Introduction

The previous chapter introduced the background of this thesis by explaining short-form videos, their multimodal nature, and the need for automatic classification supported by human annotation. Based on that foundation, this chapter presents the theoretical concepts needed to understand the methods and technical choices used in the following chapters.

It covers supervised classification, data preparation, numerical representations, pretrained models, classifiers, multimodal fusion, knowledge distillation, and evaluation. It also reviews related works.

2.2 Multimodal classification

Classification is a supervised learning task in which a model assigns an input sample to one class from a predefined set of classes. The model learns from labeled examples, where each sample is associated with a known target label, and uses this learning to classify unseen samples [38].

2.2.1 Classification types

The main classification types can be summarized as follows:

- **Binary classification:** a classification setting in which each sample belongs to one of two possible classes.

- **Multiclass classification:** a classification setting in which each sample belongs to one class among more than two possible classes [39].

2.2.2 Multimodal learning

Multimodal learning refers to machine learning approaches that process and relate information from more than one modality. A modality is a form of information or data source, such as text, image, audio, or video [3].

The main forms of multimodal learning include:

- **Multimodal representation:** learning shared or coordinated representations across modalities. For example, CLIP learns image–text representations from paired images and captions [40].
- **Multimodal alignment:** finding correspondences between elements from different modalities. For example, visual grounding aligns text with the corresponding image regions or objects [41].
- **Multimodal fusion:** combining modalities into a unified model or decision process. For example, camera and LiDAR features can be fused for 3D object detection in autonomous driving [42].

These forms and their relation to the main input modalities are shown in Figure 2.1.

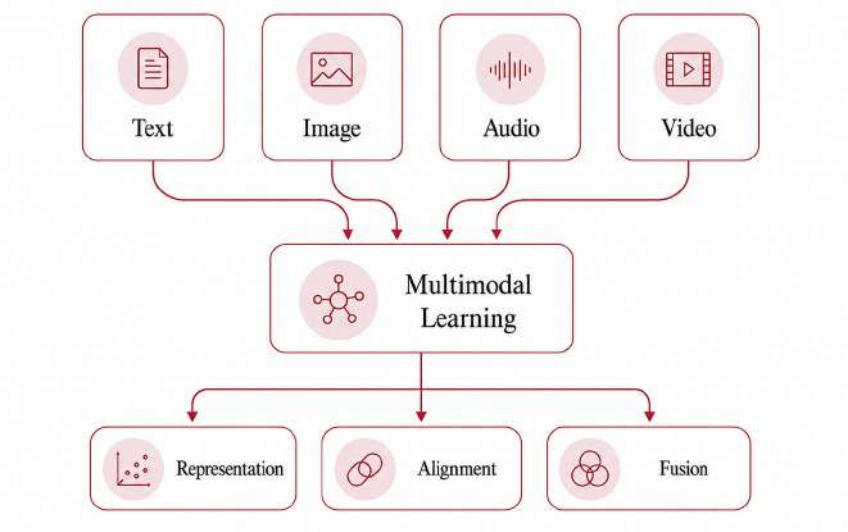


Figure 2.1: General overview of multimodal learning and its main forms.

2.3 Data annotation and ground-truth construction

The main concepts related to annotation are:

- **Data annotation:** the process of assigning meaningful labels to raw data, such as images, text, audio, or video, so that it can be used by machine learning models [43].
- **Ground truth:** the verified reference label used as the correct target for training, validation, or evaluation [44].

2.3.1 Annotation approaches

Annotation can be performed in several ways depending on the task and the required reliability [45, 46, 47]:

- **Manual annotation:** labels are assigned by human annotators according to predefined labeling instructions.
- **Automatic labeling:** labels are generated using rules, metadata, or existing machine learning models.
- **Semi-automatic annotation:** automatic labels are suggested first, then reviewed or corrected by humans when needed.
- **Expert annotation:** labels are provided by annotators with specific domain knowledge when the task requires specialized judgment.

2.3.2 Label aggregation

When several annotations are available for the same sample, an aggregation method is needed to produce the final label [48]. Common strategies include:

- **Majority voting:** the final label is the one selected by most annotators [34].
- **Weighted voting:** more importance is given to some annotators or annotation sources according to reliability or expertise [48].
- **Expert adjudication:** an expert reviews conflicting labels and decides the final class [49].

  This general progression from annotation to final ground-truth construction is illustrated in Figure 2.2.

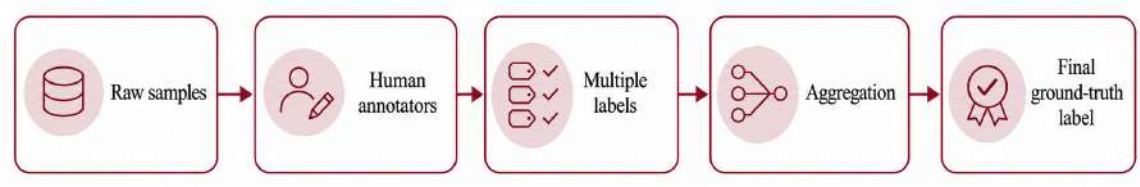


Figure 2.2: General pipeline from data annotation to ground-truth construction.

2.4 Data preprocessing for machine learning

Data preprocessing is the stage that prepares raw data before it is used by machine learning models. Its goal is to make the data cleaner, more consistent, and suitable for later use in the learning pipeline [50]. The main preprocessing operations include:

- **Cleaning:** removing invalid, corrupted, duplicated, or unusable samples that may affect the learning process [50, 51].
- **Handling missing values:** managing incomplete entries by removing, filling, or otherwise treating unavailable values [52].
- **Normalization and standardization:** transforming values into a consistent numerical scale, such as scaling features or centering them around the mean [53].
- **Label encoding:** converting class names or categorical labels into numerical values that can be used by machine learning algorithms [54].

After preprocessing, the dataset must be organized to keep each sample correctly linked to its related information:

- **Metadata:** descriptive information about each sample, such as identifier, file path, label, or processing status [55].
- **Manifest files:** structured files that list data samples and link them with their paths, labels, or processing outputs [56].
- **Keeping alignment:** maintaining the correct correspondence between raw inputs, extracted features or embeddings, and final labels.

Good dataset organization prevents mismatches between samples and labels. This is especially important when preprocessing produces separate outputs.

2.5 Representation learning (encoders, embeddings, and attention)

Representation learning is the process of transforming raw data into useful numerical representations that capture meaningful patterns in the input [57]. Instead of using raw pixels, waveform samples, or text tokens directly, machine learning models often rely on learned representations that make the data easier to process.

In this context, **raw data** refers to the original form of data, such as pixels, audio samples, frames, or text tokens. A good representation reduces the complexity of raw data and provides downstream models with more informative features for learning tasks.

2.5.1 Encoders

An encoder is a model component that transforms an input into a hidden representation or embedding. In neural models, this is commonly described as mapping an input into an internal code or feature representation [26].

The main encoder types are:

- **Audio encoder:** processes audio inputs, such as waveforms or log-Mel features, and produces audio representations across time [58].
- **Visual encoder:** processes images or frames and produces visual representations, often from patches or visual tokens [59].
- **Text encoder:** processes textual inputs, such as words or tokens, and transforms them into text representations or embeddings [26].
- **Multimodal encoder:** represents or aligns information from more than one modality, such as image and text or audio and visual data [3].

2.5.2 Embeddings

An embedding is a numerical vector representation that maps data into a vector space where it can be processed by machine learning models [60]. Embeddings can be produced from different types of data:

- **Audio embedding:** a vector representation extracted from audio features or audio encoder outputs.
- **Visual embedding:** a vector representation extracted from images, frames, patches, or visual encoder outputs.
- **Text embedding:** a vector representation of text tokens or sequences, commonly used as input to neural language models.
- **Fused embedding:** a combined representation formed by merging features from more than one modality.

The transformation from raw inputs into numerical embeddings is illustrated in Figure 2.3.

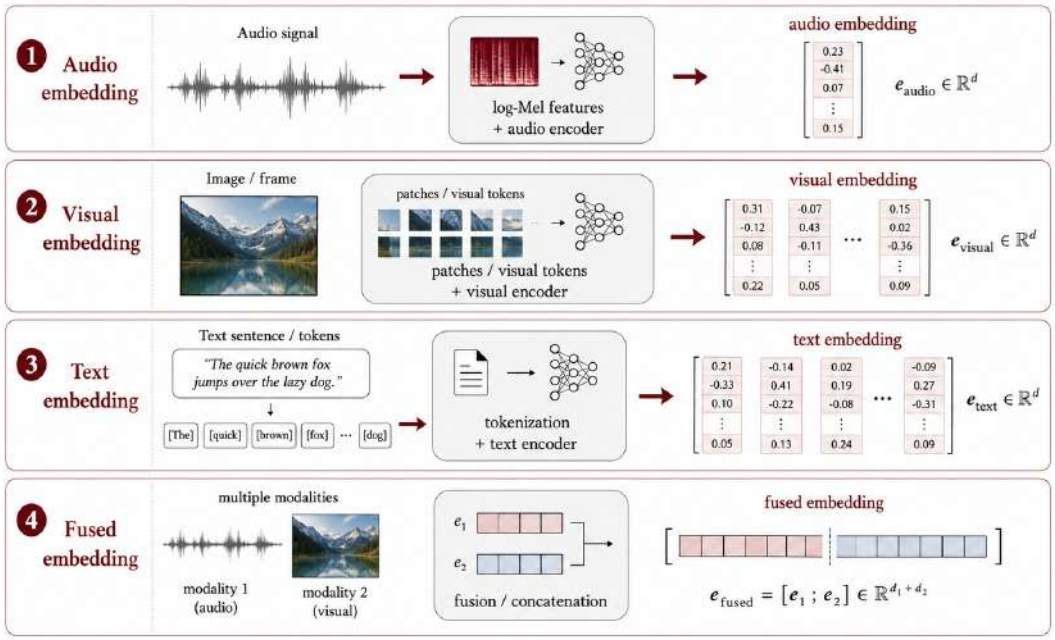


Figure 2.3: Examples of embeddings from raw data to numerical representations.

In practice, embeddings are not always produced as a single vector directly. Many neural models first produce a sequence of feature vectors, such as temporal features, frame-level features, patch-level features, or hidden states.

Before these features can be used by downstream machine learning models, they are usually transformed into one fixed-size vector through the process illustrated in Figure 2.4, as follows:

- **Sequence features:** the model first produces multiple feature vectors, such as temporal features, frame-level features, or hidden states.
- **Pooling:** the sequence is summarized into a smaller representation, for example by averaging feature vectors using mean pooling [61].
- **Normalization:** the resulting vector may be rescaled, such as with L2 normalization, to make vector magnitudes more consistent [62].
- **Fixed-size vector:** the final result is a stable numerical representation with the same dimension for all samples, making it suitable for classifiers that expect numerical feature vectors [63].

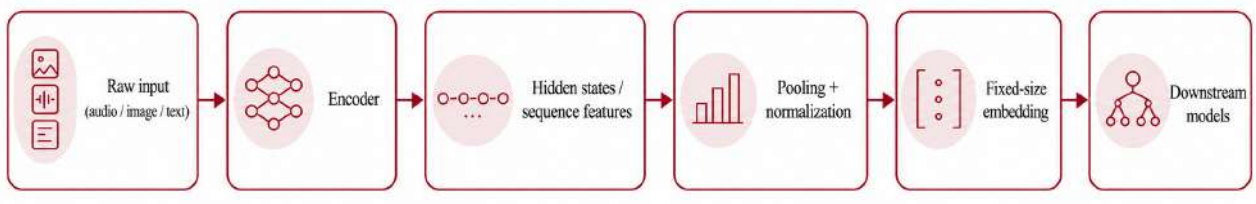


Figure 2.4: Transformation of raw inputs into fixed-size embeddings for downstream models.

2.5.3 Attention and transformer encoders

Attention is a mechanism that allows a model to focus on the most relevant parts of the input when building a representation [64].

The main forms of attention discussed in this context are:

- **Self-attention:** a form of attention where elements of the same sequence attend to each other to capture internal relationships.
- **Multi-head attention:** a structure that applies several attention heads in parallel, allowing the model to capture different types of relationships.

A Transformer encoder is a neural architecture that builds input representations mainly through self-attention and feed-forward layers, without relying on recurrent or convolutional sequence processing [64]. Its main components, illustrated in Figure 2.5, are:

- **Input embedding:** converts the input into numerical vectors. In text models, tokens are embedded, in vision models, image patches can be treated as visual tokens [59].
- **Positional encoding:** adds information about the position or order of elements, since attention alone does not encode order.
- **Self-attention layer:** models relationships between elements inside the same input sequence.
- **Feed-forward network:** applies a neural transformation to each position after attention.
- **Layer normalization:** stabilizes the activations of neural layers and helps training [65].

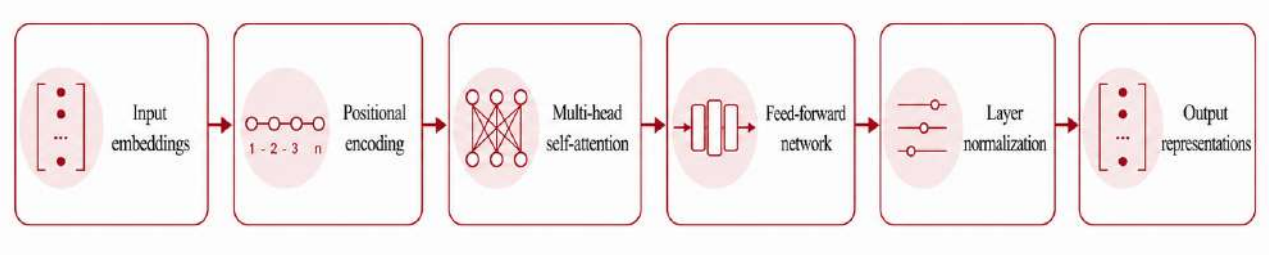


Figure 2.5: Main components of a Transformer encoder.

2.6 Pretrained models, foundation models, and transfer learning

The main concepts are:

- **Pretrained model:** a model whose weights have already been learned from previous training and can be loaded for reuse in another task [66].
- **Foundation model:** a model trained on broad data at scale and designed to be adapted to a wide range of downstream tasks [67].
- **Transfer learning:** the reuse of knowledge learned from one task or dataset to improve learning on another task [68].

2.6.1 Using pretrained models

Pretrained models can be reused in different ways:

- **Feature extraction:** the pretrained model is used to produce numerical features, while a separate classifier is trained on top of them [68] .
- **Freezing:** the pretrained model parameters are kept fixed, so only the new task-specific layers are trained [68] .
- **Fine-tuning:** the pretrained model continues training on a smaller task-specific dataset instead of starting from random weights[69] .
- **Partial fine-tuning:** only selected layers are unfrozen and updated, while the rest of the model remains frozen [68] .

When pretrained models are adapted to new tasks, several components are commonly used:

- **Backbone:** the main feature-extracting part of the model, usually reused from a pretrained architecture[68].
- **Projection head:** a small module that maps representations from one vector space to another, often for alignment or representation learning[70].
- **Classification head:** a task-specific module that maps extracted features to class scores or labels[71].

2.6.2 Audio foundation models

Audio foundation models and audio encoders convert audio inputs into numerical representations, such as hidden states or audio embeddings.

The following examples illustrate common audio models and their general structure, as shown in Table 2.1.

Table 2.1: Examples of common audio models and their general structure

Audio model	**Input**	**Architecture**	**Output**
Whisper [58]	Log-Mel spectrogram features extracted from the audio signal.	An encoder-decoder Transformer structure designed for speech recognition and translation.	Hidden representations from the encoder and decoded text tokens from the decoder.
Kimi-Audio [72]	Audio signals or processed audio features.	A large audio foundation model built to process and understand different forms of audio information.	Audio representations that can support downstream tasks such as classification or distillation-based learning.
Wav2Vec2 [73]	Raw speech waveform.	A feature encoder followed by contextual Transformer layers.	Contextual speech representations that can be adapted to speech-related tasks.

2.6.3 Visual and video foundation models

Visual and video foundation models convert images, frames, or frame sequences into visual representations, such as hidden states or embeddings.

The following examples illustrate common visual and video models and their general structure, as shown in Table 2.2.

Table 2.2: Examples of common visual and video models and their general structure.

Model	**Input**	**Architecture**	**Output**
Vision Transformer / ViT [59]	Image patches extracted from an input image or frame.	A patch embedding layer followed by Transformer encoder layers.	Visual representations or image embeddings.

Continued on next page

Table 2.2 – continued from previous page			
Model	**Input**	**Architecture**	**Output**
SmolVLM-500M [74, 75]	Images and text prompts.	A SigLIP-based image encoder connected to a compact language decoder.	Vision-language representations or generated textual responses.
VideoMAE [76]	A sequence of video frames.	Masked video modeling with a Transformer encoder.	Video-level representations that capture spatial and temporal information.
Vidi [77, 78]	Video content, often combined with textual instructions or queries.	A multimodal video-language architecture for processing visual and temporal information.	Video-aware representations or responses for video understanding tasks.

2.7 Classification models

In many neural classification pipelines, the classifier does not operate directly on raw data. Instead, it receives the learned embedding produced by previous encoders and maps it to class scores through a classification head [71]. These raw class scores are called logits [79]. They can be converted into probabilities using a softmax function, and the predicted class is usually selected as the class with the highest score [80].

Examples of commonly used classification models include:

- **Multilayer Perceptron (MLP):** a neural network classifier composed of fully connected layers that map input features to output classes [63].
  - **Input layer:** receives the embedding vector as numerical input.
  - **Hidden layers:** learn nonlinear relationships from the input features.
  - **Activation function:** introduces nonlinearity, commonly using functions such as ReLU.
  - **Dropout:** reduces overfitting by randomly disabling some activations during training [81].
  - **Output layer:** produces class scores for the target classes.
- **XGBoost:** an optimized gradient boosting method based on tree learners [82].
  - **Gradient boosting:** builds models sequentially, where each new model improves the errors of previous ones.
  - **Decision trees:** split the feature space to support prediction.

  - **Vector features:** embeddings can be used as structured numerical inputs for tree-based classification.

- **Support Vector Machine (SVM):** a supervised learning method used for classification by learning a decision boundary between classes [83].
  - **Feature space:** the numerical space where input vectors are represented.
  - **Decision boundary:** the separating surface learned by the model to distinguish between classes.
  - **Support vectors:** training samples that influence the learned decision function.

2.8 Fusion levels

Fusion strategies are commonly described according to the stage at which information from different modalities is combined, as shown in Table 2.3

Table 2.3: Common multimodal fusion strategies.

Fusion strategy	**Description**
Early fusion / feature-level fusion	Information from different modalities is combined before the final classification stage, usually at the feature or representation level [84].
Concatenation	A simple early fusion method where feature vectors from different modalities are joined into one larger representation [84].
Intermediate fusion	Modalities are combined inside the model, usually through hidden layers or shared representations that learn interactions between modalities [85].
Late fusion / decision-level fusion	Each modality is processed separately and produces its own scores, probabilities, or decisions, which are then combined to produce the final output [84].
Cross-attention fusion	One modality attends to another modality in order to model cross-modal relationships more explicitly [86].

2.9 Knowledge distillation and efficient learning

Knowledge distillation is a training method that transfers knowledge from a large teacher model to a smaller student model [87, 88].

- **Teacher model:** a larger or stronger model that provides guidance signals for training another model.

- **Student model:** a smaller model trained to imitate or benefit from the teacher while keeping lower computational cost.
- **Teacher-student paradigm:** a learning framework where the student is optimized using information produced by the teacher.

2.9.1 Motivation for distillation

Knowledge distillation is useful for several reasons [87, 88]:

- **Heavy teacher models:** large models may provide strong predictions but can be expensive to deploy or run repeatedly.
- **Efficient student models:** smaller models usually require less memory and computation during inference.
- **Knowledge transfer:** the student can learn from richer teacher outputs instead of learning only from hard class labels.
- **Lower inference cost:** distillation can make a model more practical when runtime memory, speed, or deployment resources are limited.

2.9.2 Types of distillation

The main types of distillation include[87, 88, 89]:

- **Response-based distillation:** the student learns from the final outputs of the teacher, such as logits or class probabilities.
- **Logit or probability distillation:** the student is trained to match the soft outputs produced by the teacher model.
- **Feature-based distillation:** the student learns from intermediate representations or hidden features produced by the teacher.
- **Embedding distillation:** the student tries to produce embeddings close to the teacher embeddings in the representation space.
- **Hybrid distillation:** multiple signals are combined, such as teacher embeddings, teacher probabilities, and true labels.

The main teacher–student distillation signals are illustrated in Figure 2.6.

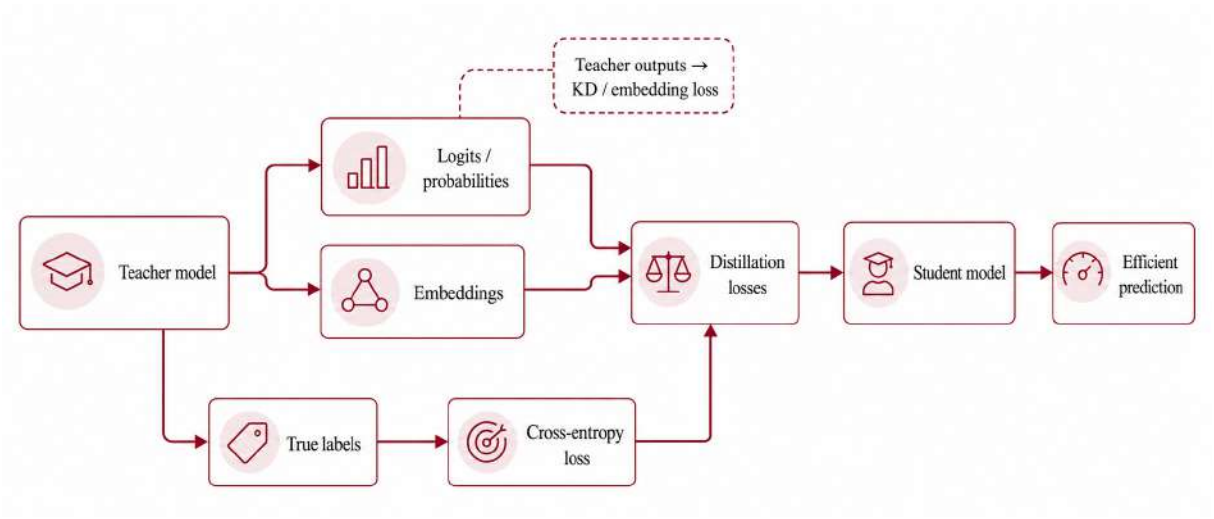


Figure 2.6: Teacher–student knowledge distillation using output and representation-level signals.

2.9.3 Common training losses

Several losses can be used in distillation-based training:

- **Cross-entropy loss:** learns directly from the true class labels in a classification task [79].
- **KD loss:** aligns the student probability distribution with the teacher probability distribution, commonly using softened outputs [87, 90].
- **Embedding matching loss:** measures the distance between teacher and student representations [89, 88].
- **MSE loss:** measures the squared difference between two vectors or tensors [91].

2.10 Related works

This section reviews related studies on religion-oriented classification, religious hate speech detection, harmful video classification, and Arabic multimodal content analysis.

2.10.1 Zamry et al. — Islamic web content filtering and categorization

Zamry et al. proposed an Islamic web content filtering and categorization approach for deviant teaching websites [92]. Their study classified Islamic web pages into three categories: *deviate*, *suspicious*, and *clean*. The method used textual representation techniques such as TF-IDF, Entropy, and Modified Entropy, followed by a Support Vector Machine classifier.

2.10.2 Masadeh et al. — Religious Arabic hatred speech in social networks

Masadeh et al. proposed a machine learning framework for detecting religious Arabic hatred speech in social networks [93]. The study relied on machine learning classification over Arabic textual content to identify harmful religious speech.

2.10.3 Das et al. — HateMM: A multimodal dataset for hate video classification

Das et al. introduced HateMM, a multimodal dataset for hate video classification [94]. The study proposed a dataset for classifying videos into hateful and non-hateful categories using more than one modality.

2.10.4 Céspedes-Sarrias et al. — MM-HSD: multimodal hate speech detection in videos

Céspedes-Sarrias et al. proposed MM-HSD for multimodal hate speech detection in videos [95]. Their work integrates video frames, audio, speech transcripts, and on-screen text, using cross-modal attention to combine information across modalities.

2.10.5 Zaghouani et al. — MAHED 2025 shared task

Zaghouani et al. presented the MAHED 2025 shared task on multimodal detection of hope and hate emotions in Arabic content [96]. The shared task included Arabic hate and hope classification, multi-task learning, and multimodal detection of hateful Arabic memes.

2.11 Comparative analysis of previous works

After reviewing the selected studies, this section compares their main contributions and limitations in relation to the present thesis. The comparison focuses on research focus, data type, modality, methodological contribution, and the remaining gap in Arabic short-video Islamic-based classification.

Table 2.4: Comparative analysis of selected previous works.

Study	**Year**	**Research focus**	**Data / modality**	**Main contribution**	**Relevance and limitation**
Zamry et al.	2014	Islamic web content filtering	Web pages / text	Classified Islamic web pages into *deviate*, *suspicious*, and *clean* using textual features and SVM.	Closest conceptually because it uses a religion-oriented three-class scheme. However, it is limited to textual web pages and does not address Arabic short videos or multimodal data.
Masadeh et al.	2022	Religious Arabic hate speech detection	Social-network text	Proposed a machine learning framework for detecting religious Arabic hate speech.	Strengthens the Arabic religious-content direction, but remains limited to textual data and hate-speech detection rather than broader Islamic-based classification.
Das et al.	2023	Hate video classification	Multimodal videos	Introduced HateMM, a multimodal dataset for hate video classification.	Technically close because it treats video as multimodal content. However, it focuses on binary hate/non-hate detection and is not Arabic or Islamic-based.
Céspedes-Sarrias et al.	2025	Multimodal hate speech detection in videos	Frames, audio, transcripts, and on-screen text	Proposed MM-HSD using cross-modal information for hate speech detection in videos.	Highly relevant technically because it combines multiple video modalities. However, its task is hate speech detection, not *Good*, *Acceptable*, and *Bad* classification.
Zaghouani et al.	2025	Arabic harmful and emotional content analysis	Arabic text and multimodal memes	Introduced MA-HED 2025 for Arabic hope/hate detection and multimodal hateful meme analysis.	Relevant to Arabic multimodal sensitive-content analysis, but focuses mainly on text and memes rather than Arabic short videos with audio.

The comparative analysis highlights how each study relates to the present thesis from a different perspective.

2.12 Conclusion

This chapter established the theoretical basis required to understand the methodological choices of the thesis. It clarified how audio-visual data can be represented, combined, and classified using modern machine learning approaches, and explained the role of annotation, pretrained models, distillation, and evaluation in building reliable classification systems.

By connecting these concepts to previous studies, the chapter also showed that existing work does not fully address Arabic short-video classification within an Islamic-based evaluative framework. This gap motivates the methodological design presented in the next chapter.

Chapter 3

Contribution 1: Dataset Collection and Annotation

3.1 Introduction

This chapter describes how the dataset was built for the practical part of the thesis. The focus is methodological. It explains how reels were collected, why the initial collection plan was changed, how videos were prepared for annotation, how human labels were produced, and how the final CSV dataset was derived. Detailed technical implementation, software structure, interface design, and backend logic are left to the implementation chapter.

The discussion extends the methodological need for human annotation established earlier in the thesis and turns it into a concrete data-building workflow [33, 35].

3.2 Chapter roadmap

This roadmap outlines the chapter structure, from dataset need and collection strategies to annotation, dataset construction, and EDA, as shown in Figure 3.1.



Figure 3.1: Roadmap of the main stages covered in this chapter.

3.3 The need for dataset construction

No suitable public dataset was available for this task, specifically an Arabic Facebook Reels dataset labeled according to the three target classes used in this work.

The need to construct the dataset was driven by three main considerations:

- **Task-specific labels:** the study targets a value-oriented classification problem that is not covered by available public reel datasets.
- **Recent content:** reel content changes rapidly, so recent videos are more suitable than old material.
- **Realistic collection:** the strength of the dataset depends on collecting reels in a way that reflects actual viewing conditions rather than relying on a narrow or artificial source selection.

For these reasons, dataset construction was not a secondary preprocessing step. It was a central methodological requirement of the study.

3.4 Collection strategies

Data collection was a central stage in this work, and several approaches were tested before selecting the strategy that best suited the study objectives.

3.4.1 Initial page-based collection strategy

The first collection strategy used an automated tool to extract Facebook Reel links from pages selected manually in advance. This stage was useful as an exploratory start because it produced an initial set of reel URLs and confirmed that automatic link collection was feasible.

The procedure followed three simple steps:

- select a set of Facebook pages,
- extract their reel links automatically,
- aggregate the collected URLs into a CSV file.

This strategy was ultimately discarded for the final dataset because several methodological limitations became clear during collection:

- it depended on a fixed set of manually selected pages,
- a single page could contribute many reels, which reduced diversity,

- the collected corpus risked over-representing a limited set of content sources,
- the resulting dataset did not reflect actual reel exposure with sufficient realism for the final study.

3.4.2 Revised data collection strategy based on account browsing

A second strategy was then adopted. Instead of collecting reels from fixed pages, reel links were captured during real browsing sessions inside Facebook Reels. A general-purpose script monitored navigation through reels and stored the discovered links in a master CSV file [1, 97].

To improve coverage and reduce personalization bias, two account types were used:

- real accounts belonging to actual users,
- newly created accounts intended to reduce the effect of an existing personal recommendation history.

This combination increased the diversity of the collected reels. It also brought the dataset closer to real platform consumption, as shown in Figure 3.2.

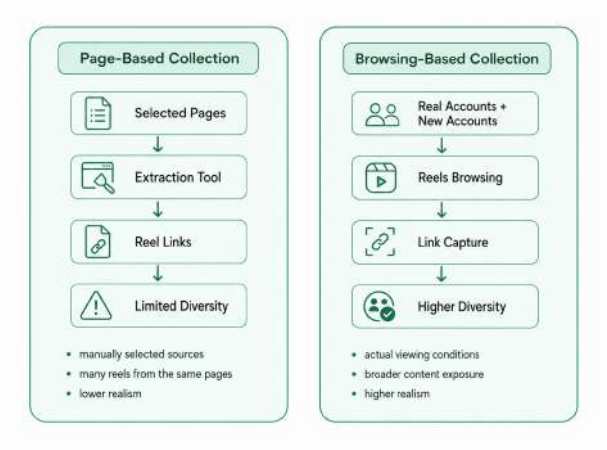


Figure 3.2: Comparison of reel collection strategies.

3.5 The labeling website as an operational environment

Once reel links had been collected, the project required a controlled environment for managing the data and recording human judgments. A dedicated website was therefore used as the operational workspace of the dataset.

The website served two basic functions:

- **data management:** importing link lists, organizing the video flow, and tracking annotation progress,
- **annotation support:** presenting each video to the annotator and recording the selected label and auxiliary flags.

Access to the website was controlled. Annotator accounts were created by the administrator, and the platform did not allow open public registration.

This moved the process away from scattered manual handling. Links, videos, and human judgments became part of one structured workflow, as shown in Figure 3.3.



Figure 3.3: Role of the labeling website in data management and annotation support.

3.6 Storage strategies

A reliable storage strategy was necessary to support the growth of the collected reel dataset. The initial setup was therefore revised when it became insufficient for the required scale.

3.6.1 Initial storage strategy

At the beginning of the project, videos were stored in a limited local storage setup integrated with the platform. This arrangement was suitable for a small pilot dataset because it simplified file handling and early data organization. However, it became insufficient once the corpus started to grow. The available storage space was quickly exhausted, which restricted the continuation of data collection at the scale required for the study.

3.6.2 Improved storage strategy

To overcome this limitation, the storage design was revised. Videos were moved to a dedicated external server with larger capacity. This change made it possible to continue collecting, storing, and serving a much larger number of reels. It also removed the earlier

storage bottleneck and made the dataset pipeline more sustainable for long-term growth, as illustrated in Figure 3.4.

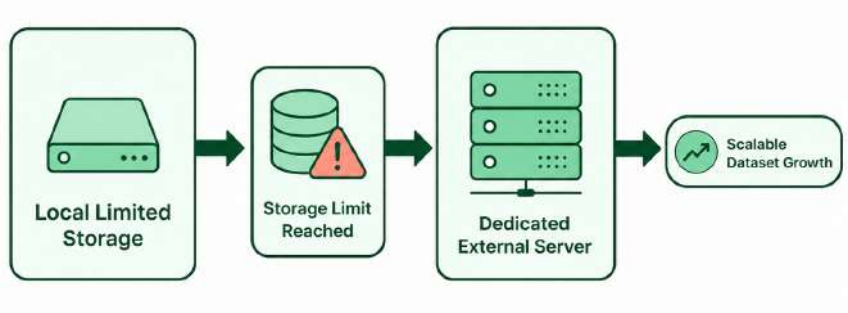


Figure 3.4: Evolution of the storage strategy from Supabase Storage to a dedicated external server.

3.7 From collected links to usable videos

The master CSV of collected links was not the final dataset. It was an intermediate resource.

The general workflow, illustrated in Figure 3.5,was as follows:

- collect reel links into a master CSV,
- import the CSV into the administrative workflow,
- download the corresponding videos,
- assign each video a stable internal reference in the dataset,
- present the video in the labeling interface for annotation.

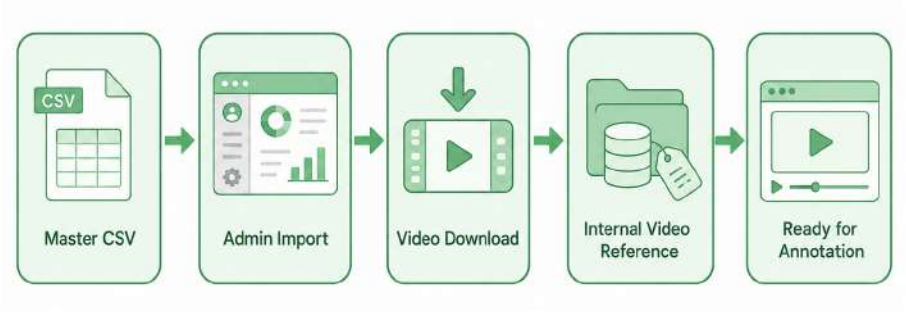


Figure 3.5: Main workflow from the master CSV file to usable videos prepared for annotation.

3.8 Human annotation strategy

Human annotation provided the ground truth for the dataset. The annotators were carefully selected volunteers from a known social circle. They were recruited through

direct contact, informed of the labeling rules and class definitions, and granted access to the platform only after agreeing to follow the same annotation criteria.

A fixed annotation rule was adopted:

- each video was labeled by exactly three annotators.

This rule served several purposes:

- reducing the effect of a single individual judgment,
- handling borderline and ambiguous cases more carefully,
- producing a more reliable final label through aggregation.

Research on inter-annotator agreement and repeated labeling supports this general logic. When judgments are interpretive rather than purely factual, repeated annotation can improve label reliability and make disagreement informative rather than accidental [33, 35], as illustrated in Figure 3.6.

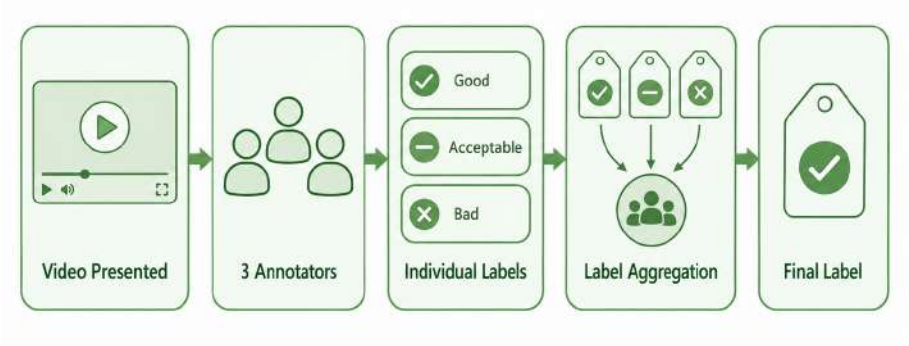


Figure 3.6: Human annotation workflow showing how each video is reviewed and labeled within the platform.

3.9 Annotation scheme and labeling rules

The labeling interface was designed around three primary class buttons. Each video was assigned one main label according to the practical rules described below. These rules were derived from consultation notes prepared by a university doctor specialized in Sharia studies[6][1].

- **Good**: This refers to any content that provides religious or worldly benefit, especially content that promotes Islamic values and morals, such as Islamic knowledge, reminders, good character, as well as beneficial content related to health, education, trade, crafts, work, and useful services . An illustrative example: Good reel example.

[1]The original Arabic consultation notes were translated into English by HABITA Anwaar for inclusion in this thesis.

- **Acceptable**: This refers to any content in which no clear benefit or apparent harm is observed, such as passing or permissible entertainment content that is free from violations or influential messages. An illustrative example: Acceptable reel example.
- **Bad**: This refers to any content that involves religious or moral harm, or weakens values and modesty, even if it is not explicitly prohibited, while avoiding engagement with matters subject to scholarly disagreement. Examples include immodest forms of hijab, uncontrolled gender mixing, deviant ideas, triviality, mockery, and scenes that undermine dignity or reinforce wrong behaviours . An illustrative example: Bad reel example.

In addition to the primary class label, two auxiliary buttons were recorded when relevant:

- **Music:** used when the video contained music.
- **Foreign language:** used when the spoken language was not Arabic.

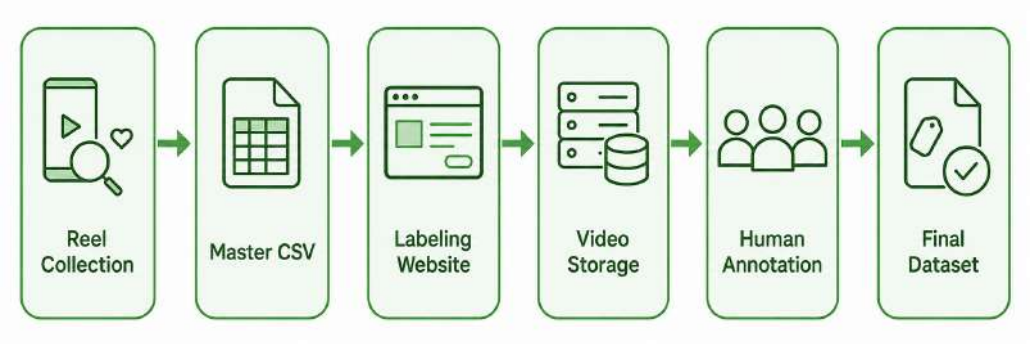


Figure 3.7: Overall pipeline from reel collection to the construction of the final labeled dataset.

3.10 Exploratory data analysis of the final dataset

This section provides a concise descriptive overview of the final dataset. The purpose is not to present an exhaustive statistical analysis, but to highlight the most informative patterns that help describe the size, structure, class composition, and temporal growth of the dataset.

Table 3.1: Basic dataset statistics.

Dataset statistic	**Value**
Total rows	23,672
Total columns	12
Unique classes	3
Unique video paths	23,672
Unique video URLs	23,672

Table 3.1 summarizes the main structural properties of the final dataset. The dataset contains a substantial number of videos, and the equality between the total number of rows and the number of unique video paths and URLs indicates a one-to-one correspondence between records and stored video references. This suggests that the final dataset is structurally clean and consistently organized.

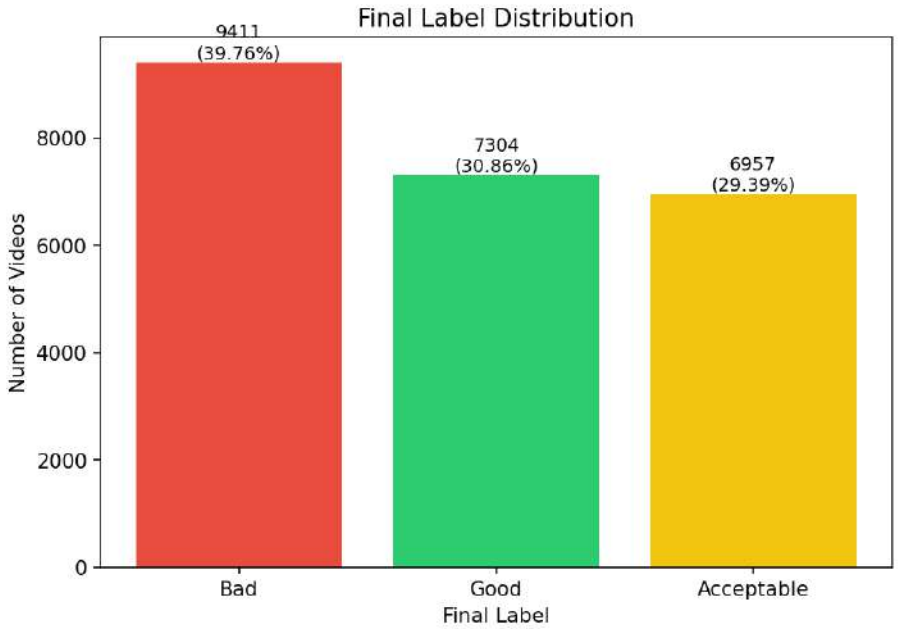


Figure 3.8: Distribution of the final labels in the dataset.

Figure 3.8 shows the class distribution across the three target labels. The *Bad* class is the largest, while *Good* and *Acceptable* appear in relatively close proportions.

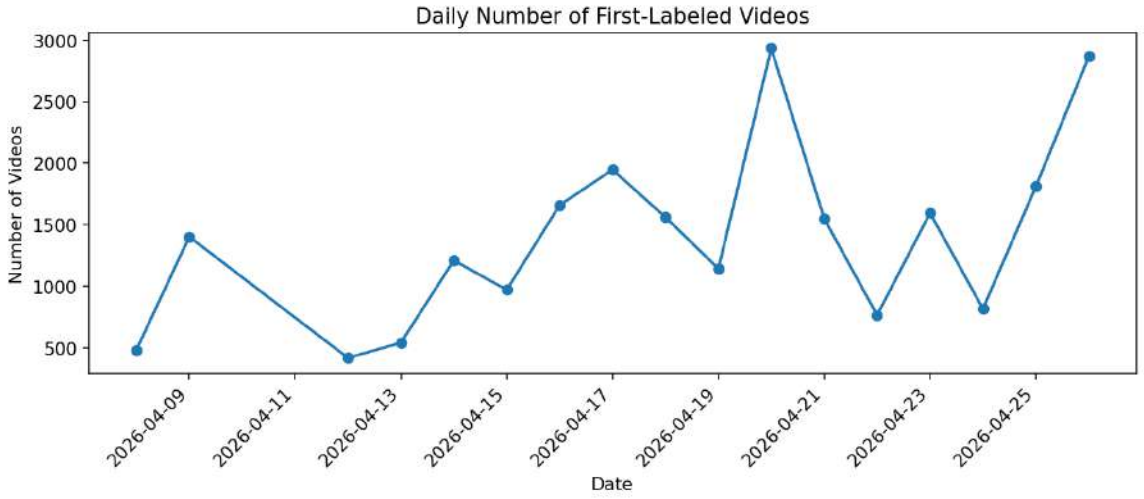


Figure 3.9: Daily number of first-labeled videos during dataset construction.

Figure 3.9 illustrates the temporal evolution of dataset construction through the daily number of first-labeled videos. The curve is not uniform across days, which suggests that data collection and annotation progressed in stages rather than at a constant rate. Such variation is expected in a real annotation workflow involving human reviewers and successive expansion of the corpus.

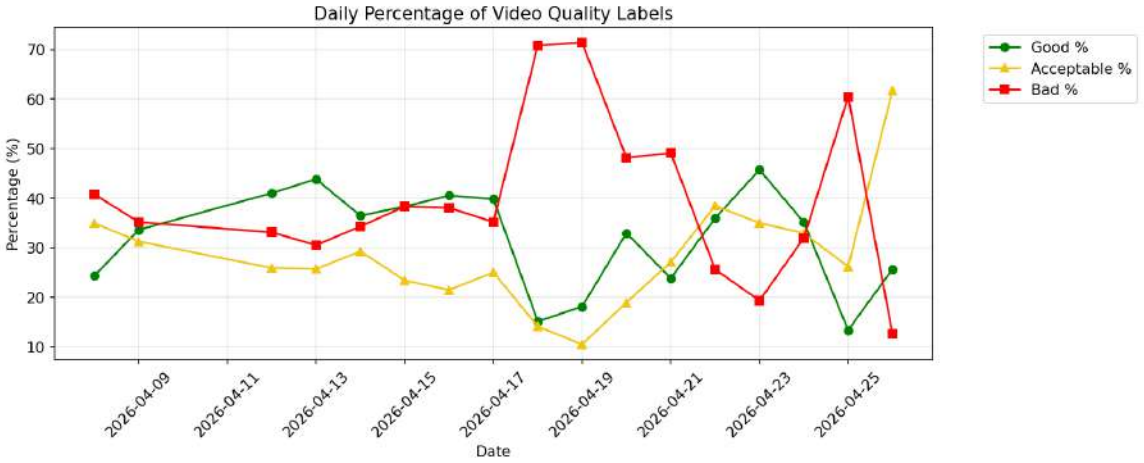


Figure 3.10: Daily percentage of the three quality labels during dataset construction.

Figure 3.10 shows the daily percentage of the three target labels across the annotation period. The proportions vary noticeably from one day to another, which indicates that class composition was not fixed during dataset construction. This variation is consistent with a progressive collection and annotation process.

Overall, these descriptive results show that the final dataset is sufficiently large, structurally consistent, and reasonably distributed across the three target classes. They also confirm that dataset construction was a gradual process shaped by successive collection and annotation stages rather than a single one-time extraction.

3.11 Conclusion

This chapter described the construction of the dataset used in the practical part of the thesis. It began with the collection goals, presented the first page-based strategy, explained why that strategy was not sufficient, and then introduced a browsing-based collection method built around actual reel exposure. It also described the role of the labeling website, the move from a limited storage setup to a dedicated server, the preparation of videos for annotation, the fixed three-annotator rule, and the annotation scheme used for the three classes.

The result of these stages was a structured labeled dataset rather than a loose collection of reel links. The next chapter can therefore focus on implementation details with the data-building process already established.

Chapter 4

Contribution 2: Audio-Visual Representation Learning and Reel Classification

4.1 Introduction

After the reels had been collected and labeled, the next step was to prepare them for machine learning. Raw video files cannot be fed directly into conventional classification models. Each reel contains a sequence of visual frames, an audio signal, and potentially overlaid text or graphics. For this reason, this unstructured information must be transformed into a structured numerical form before any learning can take place.

This chapter explains the transition from labeled reels to classification models. Since a reel does not depend only on its visual content, and part of its meaning may also come from audio, each reel was treated as audio-visual data.

The transition from raw reels to model-ready inputs involved three main steps:

- Extracting an audio representation that captures the sound-related content.
- Extracting a visual representation that captures the visual content.
- Organizing these representations and using them to build the classification model.

In this way, the chapter forms a bridge between the construction of the labeled dataset and the training of classification models. It shows how raw videos were converted into numerical inputs that could be used for learning and evaluation.

4.2 Chapter roadmap

Figure 4.1 outlines the chapter workflow, from video input and audio-visual preparation to embedding extraction, model training, and evaluation.

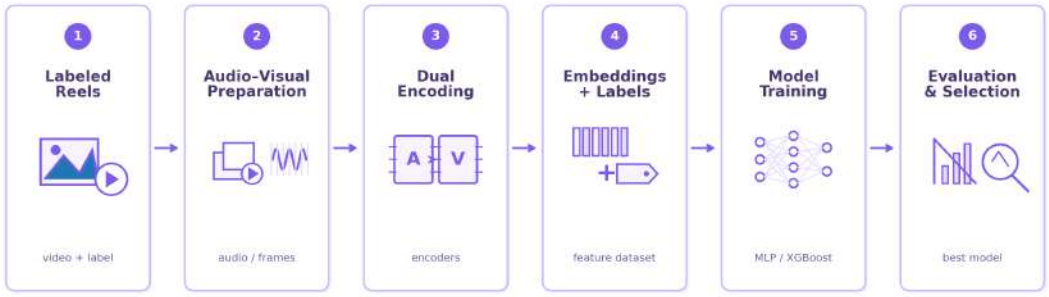


Figure 4.1: Overview of the complete pipeline from labeled reels to the final classification model

4.3 Model selection for audio-visual representation

During the experimental phase, several pretrained models were considered for representing the content of the reels.

The final selection was based on four practical criteria:

- **Modality relevance:** alignment with the two main sources of information in reels: audio and visual content.
- **Reusable representations:** extraction of feature representations that can be stored and reused during classification.
- **Compactness and efficiency:** use of models that provide strong representations while remaining practical under limited GPU and memory resources.
- **Pretrained knowledge:** reliance on large-scale pretrained representations instead of learning features from scratch.

Based on these criteria, two pretrained models were selected:

- Kimi-Audio for audio representation;
- SmolVLM-500M for visual representation.

4.4 Audio branch: from audio stream to audio embedding

Audio may add speech, music, tone, or background sounds to short-video meaning. Therefore, it was extracted, standardized, and converted into a numerical audio embedding before classification.

4.4.1 Audio stream extraction

After each reel was temporarily loaded, the audio stream was separated from the original video. This produced an independent audio input that could be processed without the visual component.

The audio stream extraction was necessary for two main reasons:

- the audio encoder works on audio input, not on the full video file,
- the extracted audio became the direct source of the auditory representation.

4.4.2 Audio standardization

After extracting the audio stream, the audio signal was converted into a unified format before being passed to the encoder.

The adopted audio format was:

- one audio channel in *mono* format,
- a sampling rate of *16 kHz*.

This standardization was necessary for three main reasons:

- The audio could differ in the number of channels, sampling rate, or recording quality,
- Using the audio in its original form could produce inconsistent inputs for the encoder.

When no valid audio was available, a silent audio input was used instead of removing the reel.

This step is summarized in Figure 4.2.

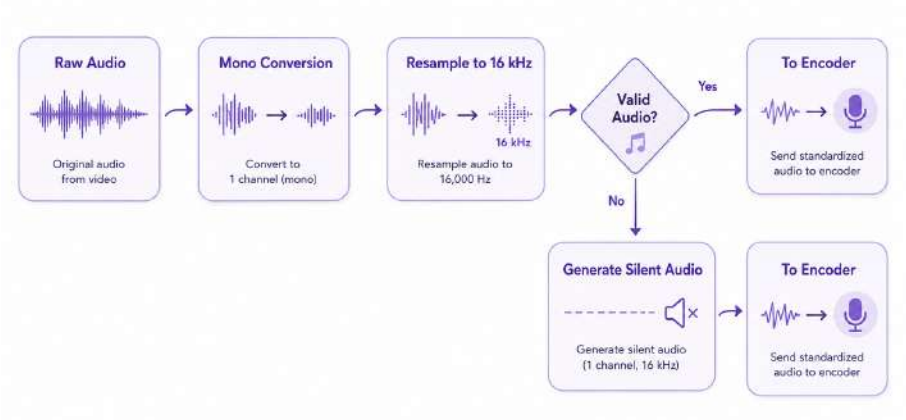


Figure 4.2: Audio format standardization before audio embedding extraction.

4.4.3 Kimi-Audio-Whisper encoder

The audio model used in this work was *Atotti/Kimi-Audio-Whisper-Encoder*. It is based on the *Whisper* architecture [58] and was used to extract temporal audio features from the prepared audio signal.

According to the official configuration, the model is based on *openai/whisper-large-v3*. It uses log-Mel audio features with *128* Mel bins and produces representations with a hidden size of *1280*. The encoder includes *32* layers, *20* attention heads, a feed-forward dimension of *5120*, and a maximum encoder source length of *1500* positions [98].

The encoder was used in the audio branch through the following main components:

- **Input:** standardized audio signal converted into log-Mel features.
- **Convolutional front-end:** the log-Mel features are projected and prepared for the Transformer encoder.
- **Transformer encoder:** the audio sequence is processed through Transformer encoder layers using multi-head self-attention.
- **Output:** temporal audio features representing the sound content of the reel.

4.4.4 Audio embedding generation

After audio preparation, each audio input was passed to the Kimi/Whisper-based audio encoder. For each reel, the encoder produced a sequence of *temporal audio features*. These features represent the audio content across time. However, the classification dataset required one fixed-size representation for each reel, not a sequence of features.

The audio representation generation involved two main operations:

- *Mean pooling* was applied to the temporal audio features to produce one fixed-size vector for each reel:
$$\bar{h} = \frac{1}{T}\sum_{t=1}^{T} h_t$$
- *L2 normalization* was applied to the resulting vector to improve stability and comparability during classification. The final normalized audio embedding was obtained as:
$$e_{\text{audio}} = \frac{\bar{h}}{\|\bar{h}\|_2}$$
where $\bar{h}$ is the pooled audio vector and e_{audio} is the final audio embedding.

As a result, each reel was represented by one *audio embedding* describing its auditory content.

The audio embedding generation process is illustrated in Figure 4.3.

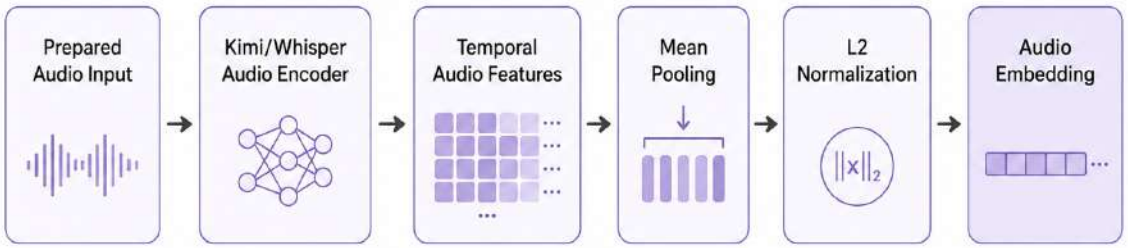


Figure 4.3: Audio representation generation using the Kimi/Whisper encoder, mean pooling, and L2 normalization.

4.5 Visual branch: from sampled frames to visual embedding

Visual content may show scenes, objects, actions, and on-screen elements that support short-video understanding. Therefore, representative frames were sampled and converted into visual embeddings before classification.

4.5.1 Frame selection and sampling

Before extracting visual embeddings, representative frames were selected from each reel instead of processing the full video frame by frame. This reduced computational cost while keeping a consistent visual input for the encoder.

Since the reels had already been prepared with a maximum duration of 20 seconds, the visual sampling strategy was set to one frame per second.

The sampling rule was:

- one frame was sampled from each second of the reel;
- duplicate frame indices were removed when they occurred.

The selected time points were converted into frame indices using the frame rate of each video. The extracted frames were then converted into RGB images .

The frame sampling workflow is shown in Figure 4.4.

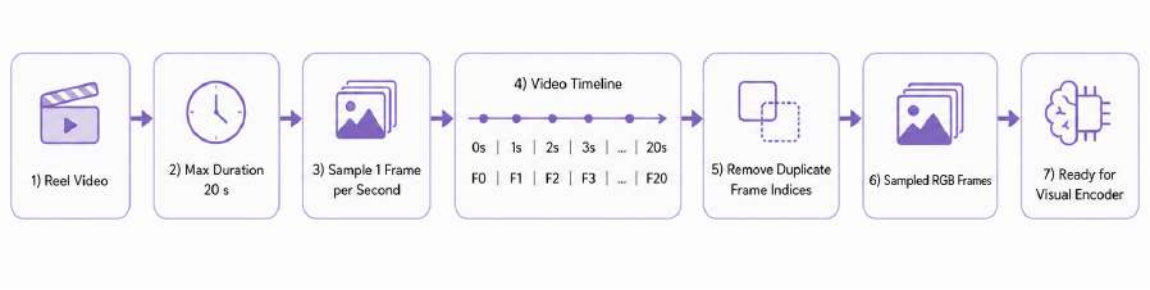


Figure 4.4: Sampling one frame per second from each reel and preparing the selected frames for the visual encoder.

4.5.2 SigLIP-Based visual encoder inside SmolVLM-500M

The visual encoder inside *SmolVLM-500M* is based on the *SigLIP base patch-16/512* architecture. This choice provides a balance between model compactness and the ability to capture meaningful visual information [99, 100].

According to the official configuration, the visual component processes three-channel RGB images with an image size of *512*, a patch size of *16*, a hidden size of *768*, *12* Transformer layers, and *16* attention heads [101].

The encoder was used in the visual branch through the following main components:

- **Input:** RGB images extracted from the selected frames.
- **Patch embedding:** the image is divided into *16* *16* patches and converted into visual tokens.
- **Transformer encoder:** the visual tokens are processed through Transformer layers to extract visual features.
- **Output:** visual features representing the content of the selected frames.

4.5.3 Visual feature extraction and embedding generation

After the selected frames were passed through this encoder, frame-level visual features were obtained. These features were later used to form one visual embedding for each reel.

The final visual embedding was then generated through two main operations:

- the extracted frame-level features were aggregated into a single fixed-size vector representing the visual content of the reel,
- *L2 normalization* was applied to the resulting vector to make the visual representation more stable and comparable across different reels.

The visual embedding generation process is shown in Figure 4.5.

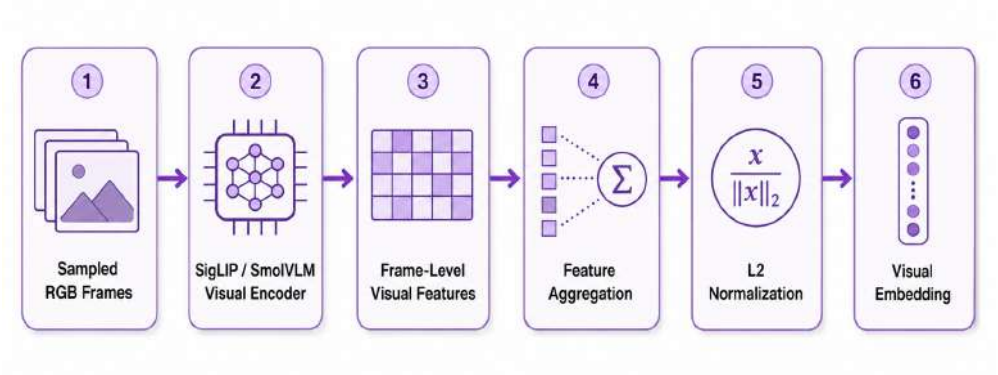


Figure 4.5: Generation of the final visual embedding from sampled RGB frames

4.6 Embedding organization and label alignment

The embeddings were organized and aligned with the dataset labels as follows:

- Audio and visual embeddings were saved separately, allowing each modality to be used independently or combined later into a fused audio-visual representation.
- A manifest file was saved to link each reel with its embeddings, final label, video path, and embedding row positions.
- The final labels were kept with the metadata so that the embeddings could be used as supervised learning inputs.

As a result, the output was an organized embedding-based dataset containing audio embeddings, visual embeddings, metadata, and their corresponding final labels. This dataset was then ready to be used as input for training the classification models.

4.7 Classification dataset preparation

Based on the prepared dataset, three feature sets were constructed from the extracted embeddings. These sets contained the same processed reels, but differed in the type of representation used as input.

The prepared feature sets were:

- **Audio feature set:** used only the audio embeddings and was used to train an audio-based classifier head.
- **Visual feature set:** used only the visual embeddings and was used to train a visual-based classifier head.

- **Fused feature set:** combined the audio and visual embeddings into one representation, which formed the main audio-visual input.

The construction of these feature sets is shown in Figure 4.6.

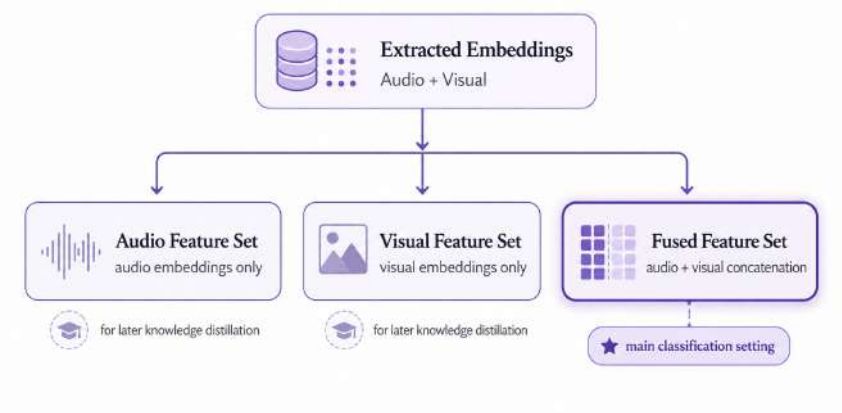


Figure 4.6: Construction of three feature sets from the extracted embeddings

Preparing these three feature sets was important because the project required more than one representation. The fused representation was used as the main classification setting, while the audio-only and visual-only heads were trained for later use in the knowledge distillation stage.

4.8 Classification models

Before training the classifiers, the input representation dimensions were defined according to the extracted embeddings:

- **Audio embedding:** 1280 dimensions.
- **Visual embedding:** 768 dimensions.
- **Fused audio-visual embedding:** 2048 dimensions, obtained by concatenating the audio and visual embeddings.

Both classifiers were trained under the same supervised three-class setting. The target classes were *Good*, *Acceptable*, and *Bad*. Therefore, the following subsections focus only on how each model learned from the extracted embeddings.

4.8.1 XGBoost classifier

The first classification model was an XGBoost classifier. This model is based on boosted decision trees and was used to learn from the extracted embeddings.

In this work, XGBoost was trained on the fused audio-visual representation to be compared with the fused MLP classifier under the same main audio-visual input setting.

The main XGBoost configuration was as follows:

Table 4.1: Main configuration of the XGBoost classifier.

Configuration item	**Value**
Input representation	Fused audio-visual embeddings
Objective	*multi:softprob*, producing probability scores for the three classes
Number of estimators	700 trees
Maximum tree depth	3
Learning rate	0.03
Subsample ratio	0.8
Column sampling by tree	0.7
Regularization	$\alpha = 1.5$ and $\lambda = 8.0$
Evaluation metric during training	*mlogloss*
Early stopping	100 rounds
Class weighting	Balanced class weights

During training, the model was fitted on the training set and monitored using the validation set. The validation performance was used for early stopping, which helped reduce overfitting while keeping the model strong enough to learn from the fused embeddings.

4.8.2 MLP classifier

The second classification model was a multilayer perceptron classifier. Unlike XGBoost, this model used fully connected neural layers to learn from the extracted embeddings.

The main training configuration of the MLP classifier was as follows:

Table 4.2: Main training configuration of the MLP classifier.

Configuration item	**Value**
Maximum number of epochs	30
Activation function	ReLU, defined as $\mathrm{ReLU}(x) = \max(0, x)$
Dropout	0.55

Configuration item	**Value**
Loss function	Cross-Entropy Loss, defined as $$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i),$$ where C is the number of classes, y_i is the true label indicator, and $\hat{y}_i$ is the predicted probability for class i.
Optimizer	AdamW with a learning rate of 310^{-4}
Weight decay	610^{-3}
Batch size	256
Early stopping	Applied with a patience of 6 epochs

Three MLP classifier heads were trained according to the input representation:

- **Audio MLP:** received only the audio embedding as input.
- **Visual MLP:** received only the visual embedding as input.
- **Fused MLP:** received the concatenated audio-visual embedding as input.

For all three settings, the same general MLP structure was used, while the input dimension changed according to the selected feature set. The architecture can be summarized as follows:

$$\text{Input} \rightarrow 256 \rightarrow 128 \rightarrow 3$$

Each hidden layer was followed by a ReLU activation function and dropout. The output layer produced the final class logits.

Before training the MLP models, the input embeddings were standardized using *StandardScaler*. The scaler was fitted on the training set and then applied to the validation and test sets. This provided the neural classifier with features on a consistent scale.

4.9 Evaluation and model selection

The trained classifiers were evaluated to determine which model best handled the fused audio-visual embeddings. This evaluation guided the selection of the final classifier adopted in the system.

4.9.1 Overall comparison on fused features

The two classifiers were first compared using the same train, validation, and test splits. Since the purpose was to compare the models under the same conditions, the same metrics were reported for both models.

Table 4.3: Overall performance comparison between XGBoost and MLP using fused audio-visual embeddings.

Model	Split	Accuracy	Macro F1	Support
XGBoost	Train	84.41%	84.10%	16570
	Validation	76.88%	76.31%	3551
	Test	75.86%	75.46%	3551
MLP	Train	87.62%	87.14%	16570
	Validation	80.65%	79.93%	3551
	Test	79.24%	78.58%	3551

The overall results show that the fused MLP classifier achieved better validation and test performance than XGBoost. This indicated that the neural classifier was more suitable for learning from the extracted fused embeddings.

4.9.2 Per-class three-class comparison

The three-class results were analyzed at the class level to better understand the behavior of each classifier. This made it possible to observe how the models performed across the different classes, beyond the overall performance metrics.

Table 4.4: Per-class comparison between XGBoost and MLP using fused audio-visual embeddings.

Model	Class	Precision	Recall	F1-score	Support
XGBoost	Good	76.12%	77.37%	76.74%	1096
MLP		78.7%	81.93%	80.29%	1096
XGBoost	Acceptable	64.01%	72.29%	67.90%	1043
MLP		68.72%	70.76%	69.72%	1043
XGBoost	Bad	86.74%	77.34%	81.77%	1412
MLP		88.17%	83.43%	85.74%	1412

4.9.3 Final model selection

The final model was selected based on the following observations:

- MLP achieved better performance than XGBoost on the extracted embeddings.
- The decision considered Macro F1-score, result stability, and three-class classification performance.
- Therefore, the fused MLP classifier was adopted as the final model.

4.10 Selected model analysis

The selected fused MLP classifier was further analyzed to understand its learning behavior and practical decision output, and to assess its ability to separate *Bad* reels from the remaining content.

4.10.1 Fused MLP training behaviour

After selecting the fused MLP classifier as the final model, its training behavior was analyzed to verify its stability during learning.

The selected checkpoint was obtained at epoch 17 based on the validation Macro F1-score. The training, validation, and test performance of this checkpoint is summarized in Table 4.5.

Table 4.5: Training behaviour of the selected fused MLP classifier.

Split	**Loss**	**Accuracy**	**Macro F1**	**Support**
Train	32.09%	87.62%	87.14%	16570
Validation	48.96%	80.65%	79.93%	3551
Test	49.92%	79.24%	78.58%	3551

Table 4.5 shows that the fused MLP achieved higher performance on the training set than on the validation and test sets, which is expected in supervised learning. However, the validation and test results remained close to each other, indicating that the model generalized reasonably well.

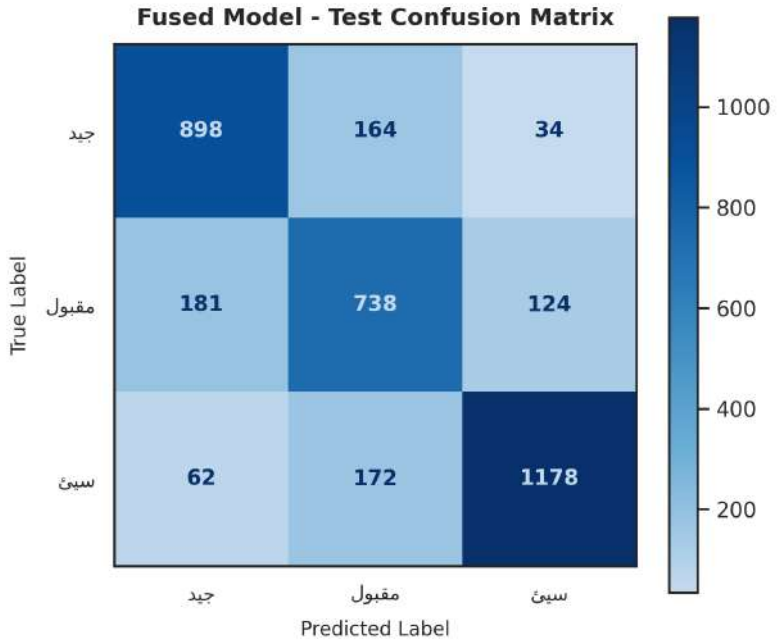


Figure 4.7: Confusion matrix of the selected fused MLP classifier on the test set.

Figure 4.7 shows that most test samples were correctly classified along the diagonal. The strongest result appeared for the *Bad* class, while most errors occurred between the *Good* and *Acceptable* classes, and between the *Acceptable* and *Bad* classes. The learning curves used to monitor convergence and possible overfitting are shown in Figure 4.8.

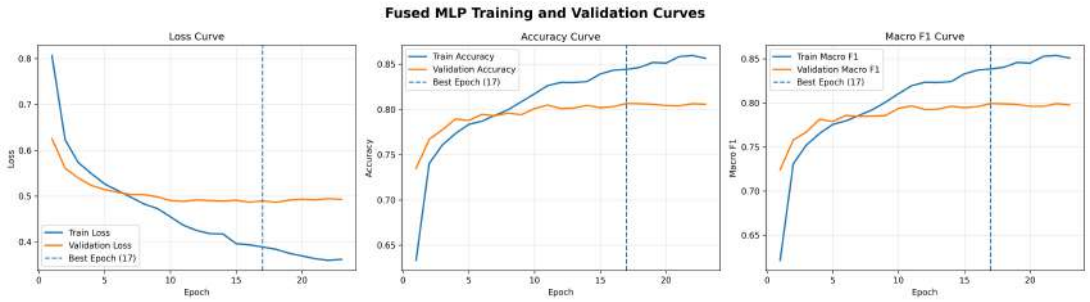


Figure 4.8: Training and validation curves of the selected fused MLP classifier used to monitor convergence and possible overfitting.

Figure 4.8 shows that the training process was stable. The gap between the training and validation curves remained acceptable, which supports the conclusion that the model did not rely only on memorizing the training data.

4.10.2 Bad versus Non-Bad output analysis

After the three-class results of the fused MLP model were analyzed, the model outputs were also examined using a binary reading.

This analysis was based on the following points:

- No separate binary classifier was trained.
- The fused MLP model remained trained on the original three classes: *Good*, *Acceptable*, and *Bad*.
- Only the output probabilities were reorganized during evaluation.
- The goal was to examine how well the model could separate *Bad* reels from the remaining reels.

In this binary interpretation, *Good* and *Acceptable* were grouped as *Non-Bad*, while *Bad* was kept as the target class. The probability of the *Non-Bad* group was computed as:

$$P(\textit{Non-Bad}) = P(\textit{Good}) + P(\textit{Acceptable})$$

The reel was then classified as *Bad* if:

$$P(\textit{Bad}) \geq P(\textit{Non-Bad})$$

Otherwise, it was interpreted as *Non-Bad*.

The obtained results are shown in Table 4.6.

Table 4.6: Binary evaluation of the fused MLP outputs without training a separate binary classifier.

Split	**Accuracy**	**Precision Bad**	**Recall Bad**	**F1 Bad**	**Macro F1**
Train	93.53%	94.43%	88.98%	91.62%	93.18%
Validation	90.03%	89.48%	84.92%	87.14%	89.50%
Test	89.24%	89.49%	82.65%	85.94%	88.61%

The results can be summarized as follows:

- The binary reading produced stable results on the validation and test sets.
- The Macro F1-score shows that the model performed well beyond overall accuracy.
- For the *Bad* class, precision was higher than recall, meaning that *Bad* predictions were generally reliable.
- The binary output analysis confirmed the practical usefulness of the model in separating *Bad* reels from the others.

4.11 Conclusion

This chapter explained how the labeled reels were transformed into numerical representations that could be used for classification. First, pretrained audio and visual models were selected, then each reel was processed through two branches: one for generating audio embeddings and another for generating visual embeddings from sampled frames.

The extracted embeddings were organized and linked with the final labels to build a dataset ready for supervised learning. Three feature settings were prepared: audio-only, visual-only, and fused audio-visual representations. The fused representation was used as the main setting for classification.

XGBoost and MLP classifiers were trained and compared using the three-class label setting: *Good*, *Acceptable*, and *Bad*. The results showed that the fused MLP model was the most suitable classifier for the extracted embeddings.

The selected model was also analyzed through a binary reading of its outputs, where *Good* and *Acceptable* were grouped as *Non-Bad* and compared against *Bad*. This helped evaluate its practical ability to separate unsuitable reels while keeping the original three-class training.

Chapter 5

Contribution 3: Multimodal knowledge distillation and fusion methodology

5.1 Introduction

After completing the first audio-visual reel classification model, the next objective was to reduce its practical cost. The initial model used strong pre-trained audio and visual encoders. These encoders were useful for extracting rich representations, but their size and runtime memory requirements made repeated inference less practical.

This chapter presents the methodology used to build a lighter student-based audio-visual system. The method keeps the useful knowledge of the larger teacher models, while training smaller audio and video student branches. The outputs of these branches are then combined by a fusion head to produce the final prediction.

5.2 Chapter roadmap

Figure 5.1 outlines the chapter structure, from the motivation for student-based distillation to audio and video distillation, audio-visual fusion, and the final practical output.

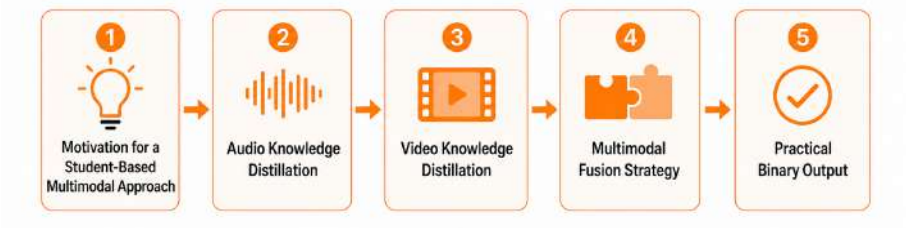


Figure 5.1: Roadmap of the student-based distillation and fusion methodology.

5.3 Motivation for a student-based audio-visual approach

The transfer learning strategy resulted in a strong teacher model, thereby making knowledge distillation feasible; however, the resulting system was heavy in terms of model size and memory usage.

The main motivations for moving to a student-based knowledge distillation strategy were:

- **Limited task-specific data:** the dataset was not large enough to train a strong audio-visual model from zero.
- **Transfer learning dependency:** strong pre-trained encoders were required to obtain useful audio and visual representations.
- **High runtime cost:** the complete teacher-based pipeline required large audio and visual components to be loaded.
- **Need for lighter inference:** smaller student branches make the system more practical for repeated classification.

Therefore, the student-based approach was adopted to reduce the practical burden of the original teacher-based pipeline while preserving its useful modality-specific knowledge. Table 5.1 compares the teacher and student models in terms of scale and practical motivation.

Table 5.1: Mixed teacher-student comparison focused on distillation motivation.

Modality	**Role**	**Model**	**Parameters**	**Complexity & Reason**
Audio	Teacher	Kimi-Audio	∼1.54B	Large audio teacher used to generate soft targets.
	Student	Whisper-base	74M	Smaller audio encoder with lower inference cost.
Video	Teacher	SmolVLM / SmolSigLIP	∼203M	Strong visual teacher used for representation guidance.
	Student	MobileCLIP / FastViT	∼35.7M	Lightweight visual branch suitable for fusion.

This design follows the idea of knowledge distillation, where a smaller model learns from information provided by a larger model [87]. It also matches the audio-visual nature of short-video classification, where audio and visual cues may provide complementary evidence [3].

5.4 Pipeline overview

The proposed pipeline has three main parts. First, the audio student learns from a frozen audio teacher. Second, the video student learns from a frozen visual teacher. Third, the exported outputs of both students are combined by a fusion head.

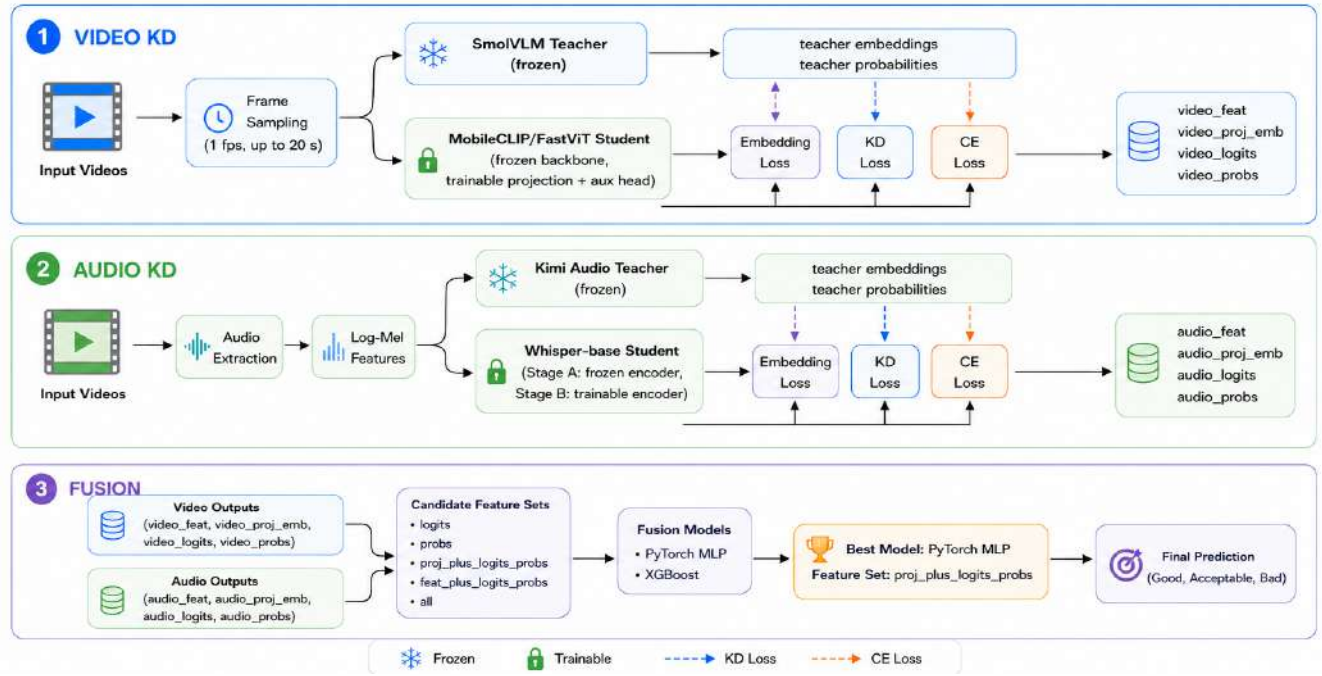


Figure 5.2: Overall knowledge distillation and fusion pipeline.

5.5 Audio knowledge distillation methodology

Figure 5.3 summarizes the audio knowledge distillation pipeline.

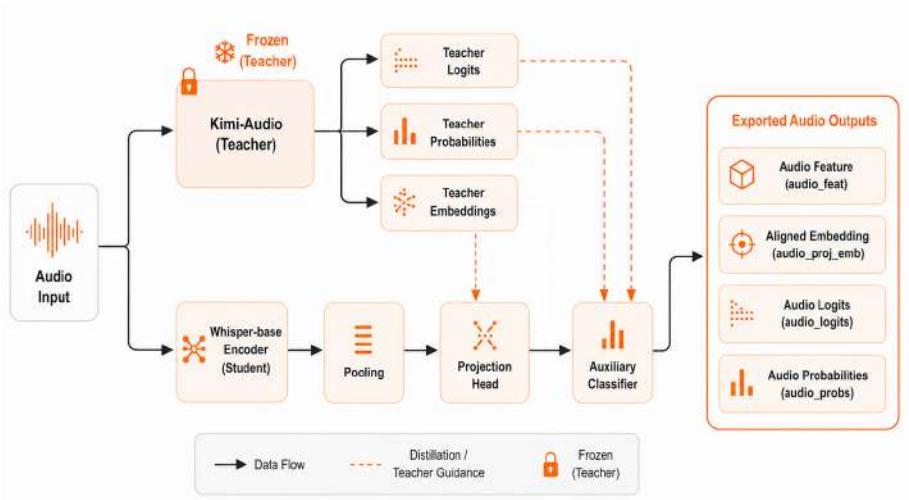


Figure 5.3: Audio distillation flow between the Kimi teacher and the Whisper-base student.

5.5.1 Audio teacher signal

The audio teacher was based on the Kimi audio representation pipeline. Kimi-Audio is a large audio foundation model designed for broad audio understanding tasks [72]. In this methodology, the Kimi-based teacher remained frozen and was used only to provide guidance signals for the audio student.

The audio teacher provided:

- **audio teacher embeddings**, used as representation-level targets;
- **teacher probabilities**, used as soft prediction targets;
- **frozen audio MLP teacher-head outputs**, used as teacher-side classification signals.

5.5.2 Audio student design

The audio student was based on the **Whisper-base encoder**. Whisper is a Transformer-based encoder-decoder model trained for robust speech recognition and speech translation using large-scale weak supervision [58]. In this work, only the encoder side was used as the audio student backbone.

The general architecture of the Whisper-base encoder [102] is summarized below:

Table 5.2: General architecture of the Whisper-base audio student backbone.

Part	**Description**
Input representation	Log-Mel spectrogram with 80 Mel bins.
Convolutional front-end	Projects audio features and reduces temporal length.
Transformer encoder	6 encoder layers, hidden size 512, and 8 attention heads.
Feed-forward blocks	Feed-forward dimension of 2048 inside the encoder layers.
Encoder output	Sequence of hidden states with hidden size 512.

Inside the proposed branch, the Whisper-base encoder was extended with a small task-specific student module composed of:

- **Pooling step:** summarizes the encoder hidden states into a fixed-size audio representation.
- **Projection head:** aligns the student audio representation with the Kimi teacher embedding space.
- **Auxiliary classification head:** produces the student-side prediction for the classification task.

5.5.3 Audio training signals and exported outputs

The audio student was trained using a hybrid distillation objective that combines teacher guidance and direct label supervision. The training used three main signals: teacher embeddings for representation alignment, teacher probabilities for soft prediction guidance, and final labels for direct supervised learning.

These signals were optimized using embedding matching loss, KD loss, and cross-entropy loss. The audio branch was trained in two stages: Stage A adapted the last Whisper encoder layers with the projection and auxiliary heads, while Stage B allowed the full encoder to adapt using smaller learning rates.

Table 5.3: Audio student training setup and selected checkpoint.

Item	**Setting**
Optimizer	AdamW
Training stages	Stage A: `A_last6`; Stage B: `B_all`
Number of epochs	6 epochs in total: 4 epochs for Stage A and 2 epochs for Stage B
Learning rates	Stage A: encoder 510^{-6}, heads 210^{-4}; Stage B: encoder 110^{-6}, heads 710^{-5}
Loss weights	CE: 0.65, KD: 0.25, cosine embedding: 0.07, MSE embedding: 0.03
Batch size	4
Weight decay / gradient clipping	10^{-4} / 1.0
Stage A reference	Epoch 3, `A_last6`: loss 0.5668, valid accuracy 0.6021, valid Macro-F1 0.5993
Selected checkpoint	Epoch 5, `B_all`: loss 0.5463, valid accuracy 0.6043, valid Macro-F1 0.6001
Selection criterion	Best validation Macro-F1 of the audio student output

After training, the audio branch exported the outputs shown in Table 5.4.

Table 5.4: Exported audio outputs used for fusion.

Output	**Meaning**
`audio_feat`	Compact audio representation produced by the audio student branch.
`audio_proj_emb`	Teacher-aligned audio embedding produced by the projection head.
`audio_logits`	Pre-softmax class scores from the audio auxiliary classifier.
`audio_probs`	Audio class probabilities obtained from the student logits.

5.6 Video knowledge distillation methodology

Figure 5.4 summarizes the video knowledge distillation pipeline.

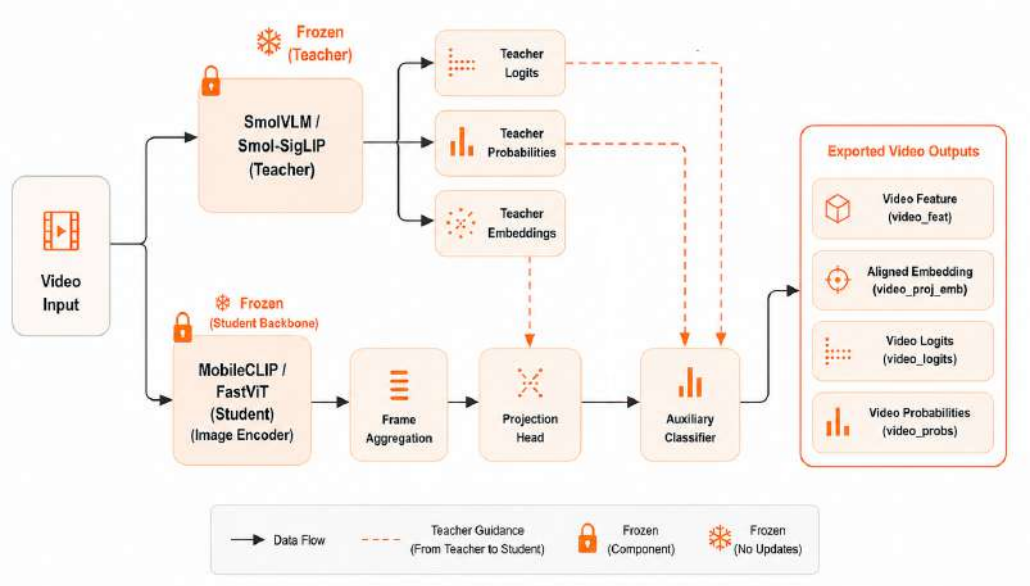


Figure 5.4: Video distillation flow between the frozen teacher and the student branch.

5.6.1 Video teacher signal

The visual teacher was based on the previous SmolVLM / Smol-SigLIP representation pipeline. In this methodology, the visual teacher remained frozen and was used only to produce guidance signals for the video student.

The video teacher provided:

- **visual teacher embeddings**, used as representation-level targets;
- **teacher probabilities**, used as soft class distributions;
- **frozen visual MLP teacher-head outputs**, used as teacher-side classification signals.

5.6.2 Video student design

The video student was based on `fastvit_mci2.apple_mclip2_dfndr2b` from the `timm` model collection. The model card identifies it as a **MobileCLIP v2 image encoder only**, equivalent to the image tower of **MobileCLIP2-S2-OpenCLIP** [103]. The selected image encoder is built on a FastViT-style visual backbone. FastViT is an efficient hybrid vision transformer family using structural reparameterization [104], while MobileCLIP targets efficient image-text representation learning for lower-latency settings [105].

The general architecture of the selected video student backbone is summarized below:

Table 5.5: General architecture of the MobileCLIP/FastViT video student backbone.

Part	**Description**
Input representation	RGB image tensor prepared with the pretrained `timm` transform.
Early visual layers	Extract low-level visual features and reduce spatial resolution.
FastViT stages	Efficient visual representation blocks based on the FastViT family.
Pooling / aggregation	Produces a fixed-size image-level representation.
Image encoder output	Compact MobileCLIP2 visual embedding.

Inside the proposed branch, the MobileCLIP/FastViT image encoder was used as the frame-level visual backbone. The branch was then extended with task-specific components:

- **Frame-level encoding:** extracts a visual representation from each sampled frame.
- **Frame aggregation:** combines frame-level representations into one reel-level visual representation.
- **Projection head:** aligns the student visual representation with the SmolVLM / Smol-SigLIP teacher embedding space.
- **Auxiliary classification head:** produces the student-side visual prediction for the classification task.

During the main video distillation stage, the visual backbone was kept frozen while the projection and classification heads were trained. This allowed the branch to adapt to the classification task without fully retraining the image encoder.

5.6.3 Video training signals and exported outputs

The video student was trained using a hybrid distillation objective. It combined teacher embeddings for visual representation alignment, teacher probabilities for soft prediction guidance, and final labels for direct supervised learning.

These signals were optimized using embedding matching loss, KD loss, and cross-entropy loss. The retained video checkpoint was obtained from Stage A, where the MobileCLIP/FastViT backbone remained frozen and only the projection head and auxiliary classifier were trained.

Table 5.6: Video student training setup and selected checkpoint.

Item	**Setting**
Optimizer	AdamW
Selected stage	`A_frozen`
Trainable parts	Projection head and auxiliary classifier
Frozen part	MobileCLIP/FastViT visual backbone
Input sampling	20 frames at 1 fps, up to 20 seconds
Batch setting	Batch size 2, gradient accumulation 8
Learning rate	Head LR 110^{-4}
Weight decay / gradient clipping	10^{-4} / 1.0
Distillation temperature	2.0
Selected checkpoint	Epoch 1, `A_frozen`: valid accuracy ≈0.71, valid Macro-F1 0.7146

After training, the video branch was exported as a feature extractor. Table 5.7 lists the video outputs used in the fusion stage.

Table 5.7: Exported video outputs used for fusion.

Output	**Meaning**
`video_feat`	Compact visual representation produced by the video student branch.
`video_proj_emb`	Teacher-aligned visual embedding produced by the projection head.
`video_logits`	Pre-softmax class scores from the video auxiliary classifier.
`video_probs`	Video class probabilities obtained from the student logits.

5.7 audio-visual fusion strategy

After training the audio and video students, their outputs were combined through a audio-visual fusion strategy. The fusion stage does not operate on raw audio or raw video. It receives the exported outputs of the two trained student branches.

Figure 5.5 summarizes the audio-visual fusion pipeline.

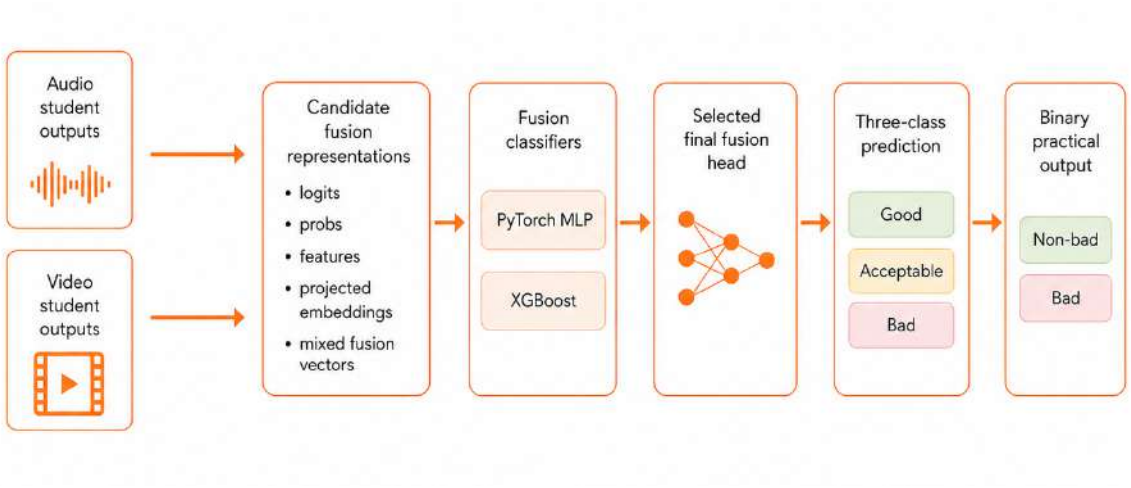


Figure 5.5: Fusion flow from audio-video student outputs to the final binary decision.

5.7.1 Late fusion rationale and input design

Fusion was performed after training the audio and video students separately, rather than jointly during student training.

The late-fusion strategy was preferred for the following reasons:

- **Modality-specific learning:** each student first learns a stable representation for its own modality before being combined with the other branch.
- **Reduced training complexity:** separate training avoids coupling two different encoders, input formats, and optimization behaviors in one process.
- **Better control over distillation:** teacher embeddings, teacher probabilities, and final labels can be applied clearly to each modality before fusion.
- **Flexible fusion design:** after exporting the student outputs, different fusion representations can be tested without retraining the audio and video students.

The fusion stage receives four exported outputs from each branch:

- **Audio outputs:** `audio_feat`, `audio_proj_emb`, `audio_logits`, and `audio_probs`.
- **Video outputs:** `video_feat`, `video_proj_emb`, `video_logits`, and `video_probs`.

These outputs allow the fusion stage to combine both representation-level information and decision-level information from the two student branches.

5.7.2 Candidate fusion representations

Several fusion representations were formed by combining different levels of audio and video information. This allowed the fusion head to use either decision-level signals, representation-level signals, or both.

Table 5.8: Candidate fusion representations.

Candidate representation	**Components**
`logits`	`audio_logits` + `video_logits`
`probs`	`audio_probs` + `video_probs`
`logits_probs`	`audio_logits`, `audio_probs`, `video_logits`, and `video_probs`
`proj_emb`	`audio_proj_emb` + `video_proj_emb`
`feat`	`audio_feat` + `video_feat`
`proj_plus_logits_probs`	Projected embeddings combined with logits and probabilities.
`feat_plus_logits_probs`	Raw student features combined with logits and probabilities.
`all`	All exported audio and video outputs combined.

5.7.3 Final decision logic

The final fusion stage was trained directly on the exported outputs of the audio and video students. Two candidate supervised fusion heads were evaluated: a PyTorch MLP fusion head and an XGBoost fusion head. The selected PyTorch MLP used the `proj_plus_logits_probs` representation. Before classification, the fusion vectors were standardized using a `StandardScaler`.

Table 5.9: Selected fusion MLP training configuration.

Item	**Setting**
Input representation	`proj_plus_logits_probs`
Input dimension	2060
Hidden layers	256 and 128 units
Activation / dropout	ReLU with dropout 0.45
Batch normalization	Not used
Optimizer	AdamW
Learning rate / weight decay	510^{-4} / 0.003
Batch size	256
Maximum epochs / patience	80 epochs / 10 epochs

Item	Setting
Loss function	Weighted Cross-Entropy Loss
Selected epoch	34, with training loss 0.4193

The final fusion head was selected by comparing the best PyTorch MLP and XGBoost results.

Table 5.10: Best three-class fusion heads on student outputs.

Fusion head	Best feature set	Input dim.	Valid Acc.	Valid F1	Test F1
PyTorch MLP	`proj_plus_logits_probs`	2060	79.27%	78.31%	77.73%
XGBoost	`feat_plus_logits_probs`	1804	78.79%	77.78%	77.41%

The PyTorch MLP was selected as the final three-class fusion head because it achieved the strongest validation Macro-F1 and maintained the best test Macro-F1.

The final decision flow of the selected fusion head, illustrated in Figure 5.6, is:

1. extract the exported outputs from the audio student;
2. extract the exported outputs from the video student;
3. build the selected fusion vector `proj_plus_logits_probs`;
4. apply the scaler associated with the selected MLP head;
5. pass the scaled vector through the PyTorch MLP fusion head;
6. produce the final three-class probabilities for Good, Acceptable, and Bad.

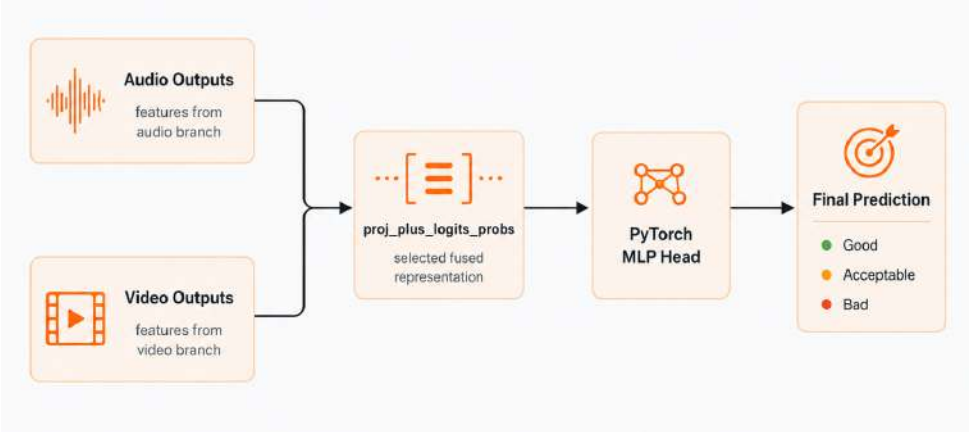


Figure 5.6: Final decision logic of the selected fusion head.

For the practical system, the selected three-class fusion output was converted into a binary output. The conversion follows the predefined mapping:

$$P(\text{Non-bad}) = P(\text{Good}) + P(\text{Acceptable}),$$
$$P(\text{Bad}) = P(\text{Bad}).$$

The selected threshold was 0.5, which was also the best threshold on the validation split.

Table 5.11: Binary adaptation of the selected fusion model.

Split	**Threshold**	**Accuracy**	**Macro-F1**	**Bad-class F1**
Train	0.50	89.55%	88.99%	86.5%
Validation	0.50	89.96%	88.33%	86.61%
Test	0.50	88.96%	88.33%	85.61%

The final student-based system keeps the same audio-visual structure as the teacher-based pipeline, with separate audio and video extractors followed by a fusion head. However, the teacher components were replaced by lighter student branches. This keeps the results close to the original model while reducing model size and runtime memory.

5.8 Conclusion

This chapter presented the methodology used to improve the efficiency of the initial audio-visual classification system through knowledge distillation and audio-visual fusion. The audio branch distilled knowledge from the Kimi-Audio teacher into a Whisper-base student, while the video branch distilled knowledge from the SmolVLM / Smol-SigLIP teacher into a MobileCLIP / FastViT student.

Both student branches were trained using teacher embeddings, teacher probabilities, and final labels. Their exported outputs were then combined using a late-fusion strategy. The selected fusion head produced the final three-class prediction, which was adapted into the binary output used in the practical application.

Overall, the original teacher-based pipeline was transformed into a lighter student-based audio-visual system while preserving the audio-visual structure of the project.

Chapter 6

System implementation

6.1 Introduction

This chapter presents the practical implementation environment of the proposed system for classifying Facebook Reels according to Islamic Sharia-oriented content categories. While the previous chapters described the theoretical background, dataset construction, audio-visual representation, classification methodology, and knowledge distillation approach, this chapter focuses on the technical setting that supported the development of the system.

The chapter outlines the main environment specifications, tools, platforms, and libraries used during implementation. It also explains the role of the key components involved in data handling, model execution, backend management, and browser-based interaction.

The aim of this chapter is to provide a comprehensive and objective overview of how the proposed methodology was translated into a functioning system. It highlights the software ecosystem, execution environment, supporting services, and deployment elements that made the audio-visual classification approach usable in a practical short-video analysis context.

6.2 Environment specifications

The implementation of this project relied on notebook-based environments for embedding extraction, model training, and evaluation. In addition, a VPS server, Supabase backend, and web-based labeling interface were used to support video storage, data management, and annotation.

The main environment specifications are listed below:

Table 6.1: Environment specifications used in the implementation.

Component	Specification
Development environment	Google Colab and Kaggle Notebook environments
Programming language	Python
GPU acceleration	CUDA-enabled GPU when available
Deep learning framework	PyTorch
Embedding extraction device	GPU when available, otherwise CPU
Dataset storage	Kaggle input datasets and external VPS servers
Video serving source	External VPS server
Model and extension backend VPS: Contabo Cloud VPS 10	4 vCPU cores, 8 GB RAM, 75 GB NVMe storage, 200 Mbit/s port
Video storage VPS: Contabo Storage VPS 30	6 vCPU cores, 18 GB RAM, 1 TB SSD storage, 600 Mbit/s port
Backend service	Supabase
Frontend hosting	Cloudflare Pages
Browser extension platform	Microsoft Edge extension environment

6.3 Tools and libraries

The implementation of the proposed system relied on a set of software tools, platforms, and libraries. These tools were selected according to their role in data processing, embedding extraction, knowledge distillation, model training, backend management, frontend deployment, and browser extension integration.

Table 6.2: Main tools and libraries used in the project.

Tool / libraries	Role in the project
Python	Main programming language used for data processing, embedding extraction, model training, evaluation, and deployment scripts.
Google Colab / Kaggle	Notebook-based environments used for experiments, embedding extraction, knowledge distillation, and model training.
KaggleHub	Used to import and access Kaggle datasets inside the notebook environment.
PyTorch	Main deep learning framework used for model loading, inference, tensor handling, neural classifier training, and checkpoint saving.
Hugging Face Transformers	Used to load and use pretrained audio and multimodal models.
Kimi-Audio Whisper Encoder	Audio encoder used for extracting audio embeddings from reel audio streams.
WhisperModel / WhisperFeatureExtractor	Used in the audio knowledge distillation stage to process audio inputs and train the audio student model.
SmolVLM-500M	Visual encoder used for extracting visual embeddings from sampled video frames.
MobileCLIP / FastViT	Visual student backbone used in the video knowledge distillation stage.
timm	Used to load and configure the MobileCLIP/FastViT visual student backbone.
NumPy	Used for numerical operations, feature arrays, logits, probabilities, and embedding matrix handling.
Pandas	Used for CSV loading, metadata processing, label matching, split generation, and report generation.
scikit-learn	Used for dataset splitting, feature standardization, and evaluation metrics such as accuracy, F1-score, and confusion matrices.
XGBoost	Used as a tree-based baseline classifier for fused audio-visual embeddings.
joblib	Used for saving and loading preprocessing objects such as scalers and model packages.

Tool / libraries	Role in the project
FFmpeg	Used for audio extraction from videos and conversion to a unified audio format.
decord	Used for efficient video frame reading and sampling during visual processing.
soundfile	Used for reading and validating extracted audio files.
PIL / Pillow	Used for RGB image preparation before visual encoding.
Requests	Used to access or download video resources and communicate with remote URLs when needed.
Paramiko	Used as an optional SFTP-based mechanism for remote video access when required.
tqdm	Used to display progress bars during long processing, training, and evaluation stages.
Supabase	Backend service used for structured data management, including videos, users, labels, auxiliary flags, and annotation progress.
Cloudflare Pages	Used to deploy the web-based labeling interface.
HTML, CSS, and JavaScript	Used for the web-based labeling interface and browser extension user interfaces.
Microsoft Edge Extension Platform	Used to deploy the *Reels Guard* browser extension.

6.4 Model evaluation metrics

During model training and evaluation, several classification metrics were used to assess prediction quality. These metrics helped compare model performance across the three target classes and evaluate whether the classifier performed consistently beyond overall accuracy.

Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. For multiclass classification, these quantities can be computed separately for each class using a one-versus-rest interpretation. The main metrics used in this work are summarized in Table 6.3.

Table 6.3: Model evaluation metrics used in this work.

Metric	Meaning	Formula
Accuracy	The proportion of correctly classified samples among all evaluated samples [106].	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	The proportion of samples predicted as a class that are actually correct for that class [107].	$\frac{TP}{TP+FP}$
Recall	The proportion of true samples of a class that the model successfully detects [108].	$\frac{TP}{TP+FN}$
F1-score	The harmonic mean of precision and recall, useful when both false positives and false negatives matter [109].	$2\frac{\text{PrecisionRecall}}{\text{Precision}+\text{Recall}}$
Macro-F1	The average of per-class F1-scores, giving equal importance to all classes [110].	$\frac{1}{K}\sum_{i=1}^{K}\text{F1}_i$ where K is the number of classes and F1_i is the F1-score of class i.
Confusion matrix	A table that compares true labels with predicted labels and shows which classes are confused with each other [111].	$C_{i,j} = \lvert\{\, n : y_n = i,\ \hat{y}_n = j \,\}\rvert$

The scalar metrics in Table 6.3, including accuracy, precision, recall, F1-score, and Macro-F1, are unitless ratios usually reported between 0 and 1, or as percentages. In contrast, the confusion matrix entries are counts, so their unit is the number of samples.

Macro-F1 was especially important in this work because the three classes were not equally difficult to classify. It gives equal importance to each class and therefore provides a more balanced view of model performance than accuracy alone [110, 112].

6.5 Labeling platform and data management

Figure 6.1 presents the overall workflow of the web-based labeling system, from Facebook Reels collection to the construction of the final labeled dataset. It shows how the collection script, video storage, Supabase backend, and annotation frontend interact

to support the complete dataset-building process.

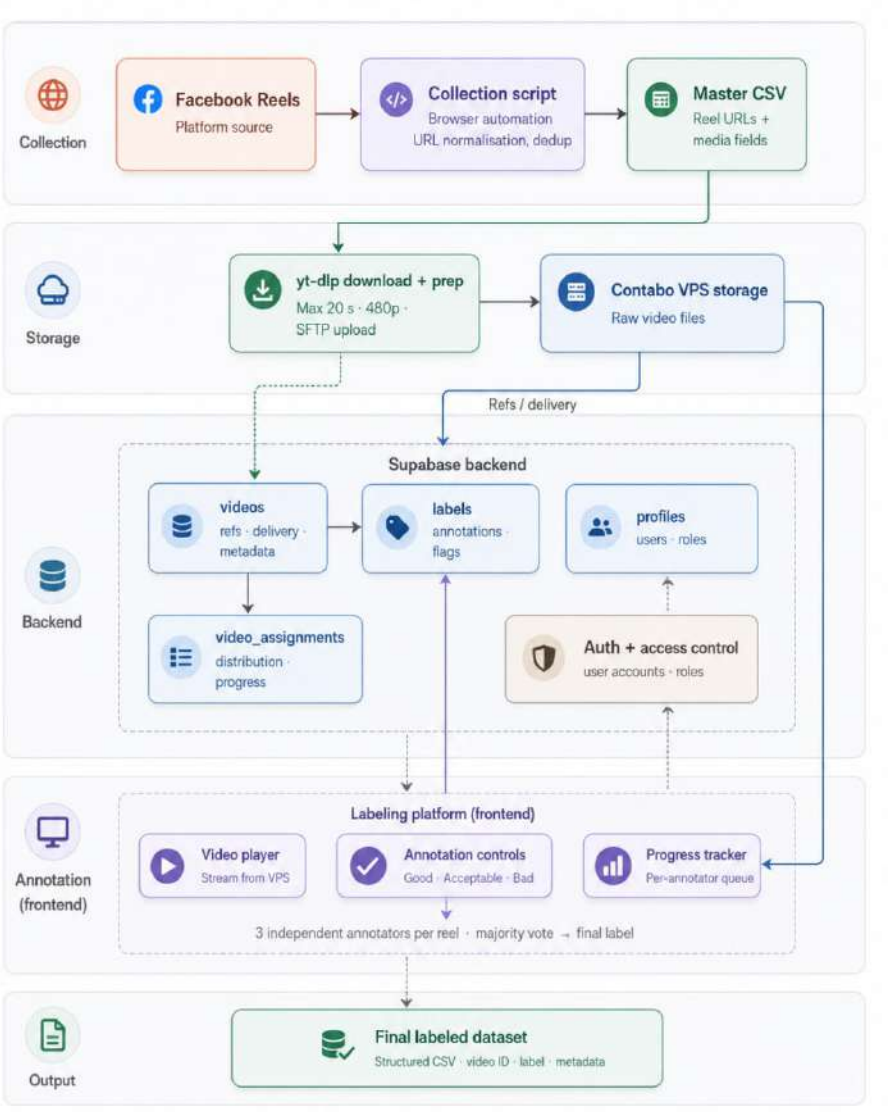


Figure 6.1: Overall workflow of the web-based labeling and dataset construction system.

6.5.1 Collection script implementation

The collection script was implemented as a lightweight browser-automation script. At the code level, its main functional components were the following:

- `find_browser_exe()`: searches for an available local Chrome or Edge executable before starting the browsing session,
- `launch_persistent_context()`: starts the browser with a saved persistent profile so that the script can continue from an already authenticated session,
- `normalize()`: cleans URLs by removing query strings and fragments so that the same reel is stored in a consistent form,

- `is_reel_url()` and `is_blocked_or_login_url()`: distinguish valid reel pages from login, verification, or blocked pages,
- `load_seen()` and `append_row()`: load previously saved rows from the master CSV and append newly discovered rows while preserving persistence across runs,
- `classify_video_url()`: checks whether a captured media response corresponds to a usable video URL and classifies it as SD, HD, or unknown,
- `move_next()`: advances the browsing session to the next reel by simulating navigation actions,
- `on_response()`: monitors browser responses during navigation and captures video-related URLs when available,
- `flush_current()`: finalizes the currently observed reel, merges newly captured media fields with any existing record, and writes the result to the master CSV when needed.

In simplified form, the script repeatedly opened the saved browsing session, navigated through reels, collected valid links and media fields, and updated the master CSV.

6.5.2 Video storage and delivery

Video files were stored on an external server rather than inside the backend storage layer. The final setup used a *Contabo Storage VPS 30*. The selected storage server was suitable for storing and serving a large reel corpus.

The backend stored the references to the videos, while the actual files were served from the server to the platform. The frontend then used these references to request and display the videos during annotation. For consistent delivery, videos were prepared in a display-friendly format with a maximum duration of 20 seconds and a target resolution of 480p.

6.5.3 Backend implementation

Figure 6.2 presents the main Supabase database schema used to organize videos, annotators, assignments, and labeling records.

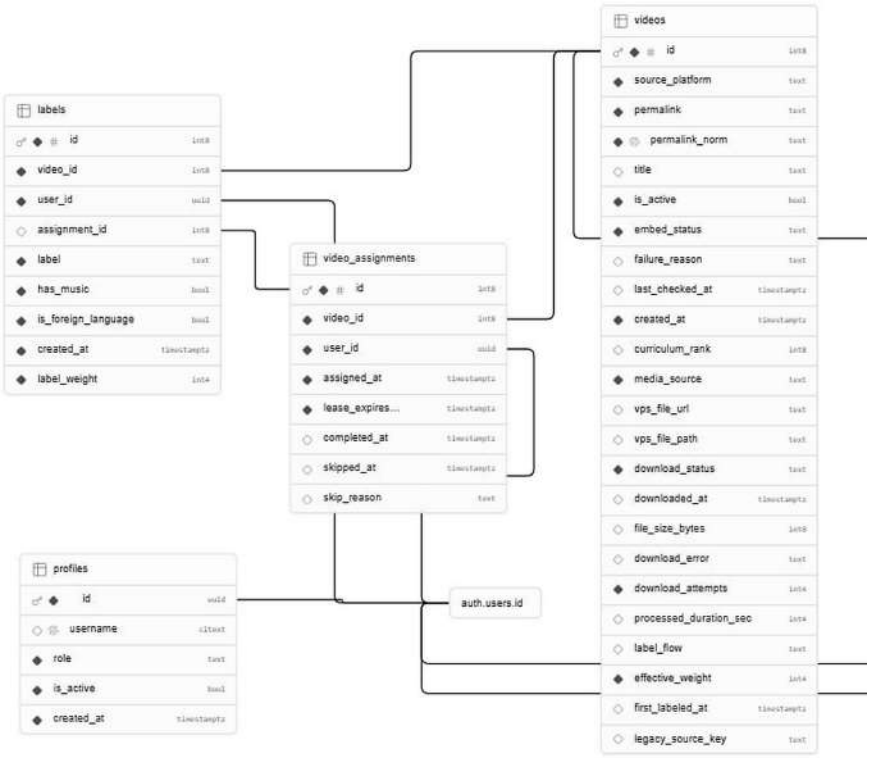


Figure 6.2: Supabase backend database schema

The backend was implemented with *Supabase*. It was used to manage the database, user accounts, access control, and annotation records.

The core database tables were the following:

- **videos**: stores video references, delivery fields, and dataset-level metadata,
- **labels**: stores human annotations and auxiliary flags,
- **profiles**: stores user identities and access roles,
- **video_assignments**: manages video distribution across annotators and tracks annotation progress.

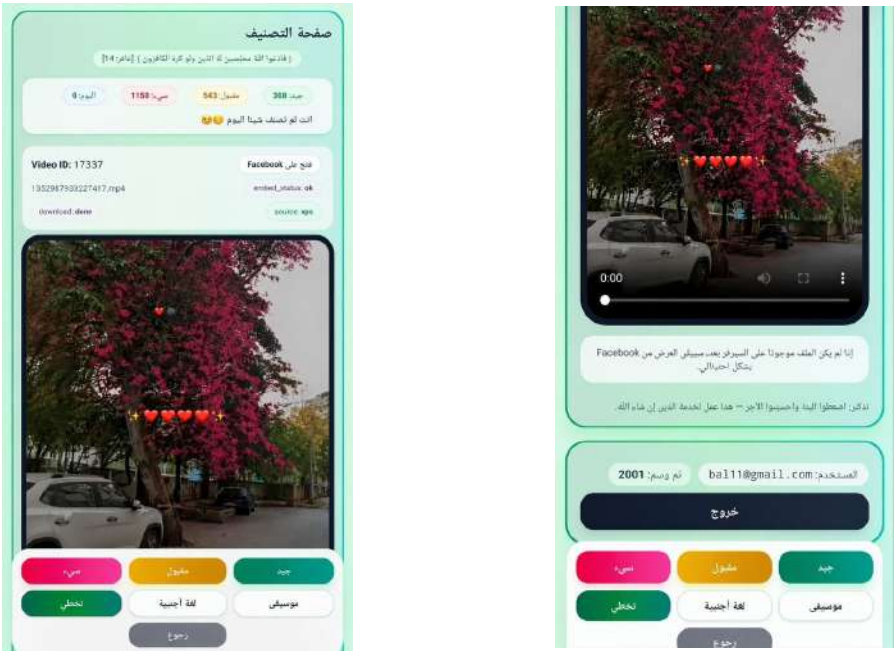
Together, these tables provided the minimum backend structure required to organize the annotation workflow in a controlled way.

6.5.4 Frontend and interface screens

The frontend provided the main user-facing interfaces of the platform. The desktop view of the labeling interface is shown in Figure 6.3, while the mobile views are shown in Figure 6.4.



Figure 6.3: Desktop view of the web-based labeling interface.



Mobile view -1- Mobile view -2-

Figure 6.4: Mobile views of the labeling interface.

After the reels were collected, stored, and manually labeled, the resulting data became ready for the model-side implementation. The following section therefore focuses on the final model components and their practical memory cost, rather than repeating the training methodology already discussed in previous chapters.

6.6 Model implementation and runtime cost

Figure 6.5 compares the original teacher-based pipeline with the final KD package. It shows how knowledge distillation reduced the loading cost while preserving the audio-visual structure of the system.

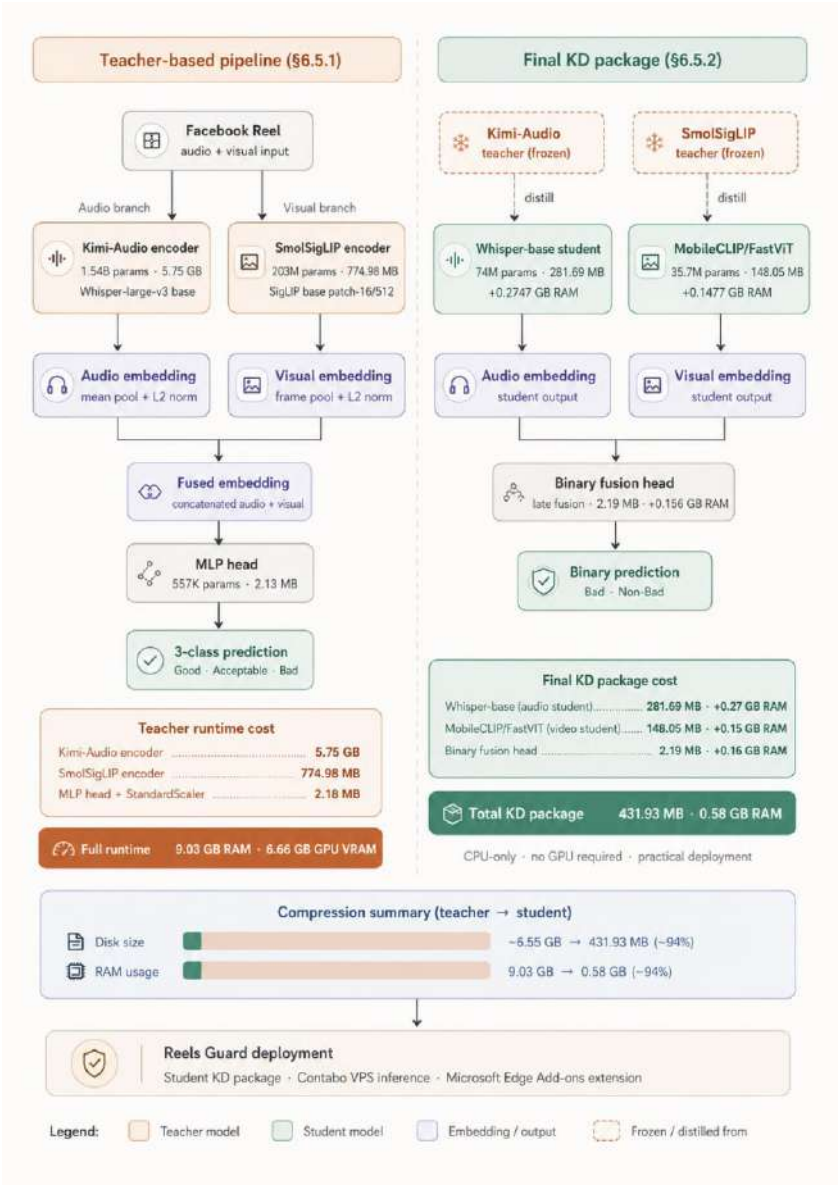


Figure 6.5: Comparison between the teacher-based pipeline and the final KD package.

6.6.1 Teacher-Based pipeline size and memory usage

After selecting the final classifier, the size of the main teacher-based pipeline components was measured. This measurement was used to distinguish between the size of the lightweight classifier head and the actual memory cost of running the full audio-visual pipeline with the pretrained encoders.

Table 6.4: Final teacher-based pipeline size and runtime memory usage.

Component	Loaded weight size	Parameters / Runtime cost
Video encoder / Smol-SigLIP	774.98 MB	203,155,970 parameters
Audio encoder / Kimi	5.75 GB	1,543,490,560 parameters
MLP head	2.13 MB	557,827 parameters
StandardScaler	48.45 KB	–
Full runtime usage	**9.03 GB RAM**	**6.66 GB GPU VRAM**

The results show that the MLP classifier head was very small compared with the pretrained encoders. Most of the memory cost came from the audio and visual encoders, especially the Kimi audio encoder. Therefore, the full pipeline memory usage should be reported separately from the classifier head size.

These measurements quantify the computational cost of the original teacher-based pipeline. Since this cost is mainly caused by the large pretrained encoders, knowledge distillation was used to prepare a lighter final package for practical execution.

6.6.2 Final KD package size and memory usage

After applying knowledge distillation, the same type of measurement was performed on the final lightweight package. This package contained the video student extractor, the audio student extractor, and the binary fusion head.

Table 6.5: Final KD package size and loading memory cost.

Final component	File size	RAM increase after loading
Video student extractor	148.05 MB	0.1477 GB
Audio student extractor	281.69 MB	0.2747 GB
Binary fusion head	2.19 MB	0.1560 GB
Total final KD package	**431.93 MB**	**0.5785 GB**

The complete final KD package required only **431.93 MB** on disk and increased RAM usage by approximately **0.5785 GB** after loading. In comparison, the earlier teacher-based runtime reached about **9.03 GB RAM** and **6.66 GB GPU VRAM**. This confirms the practical benefit of the student-based design: the final inference package is much lighter while preserving the audio-visual structure of the original system.

After preparing the lighter final inference package, the system was connected to a user-facing browser extension. The extension represents the practical layer through which watched Facebook Reels are detected, sent to the backend, and summarized in a daily report.

6.7 Reels Guard browser extension implementation

This section presents the implementation of the browser extension used in the final system. As shown in Figure 6.6, the extension connects the user-facing browser layer with backend communication, VPS-based model inference, and report visualization. The following subsections explain this workflow in detail, starting from reel detection and ending with the displayed classification report.

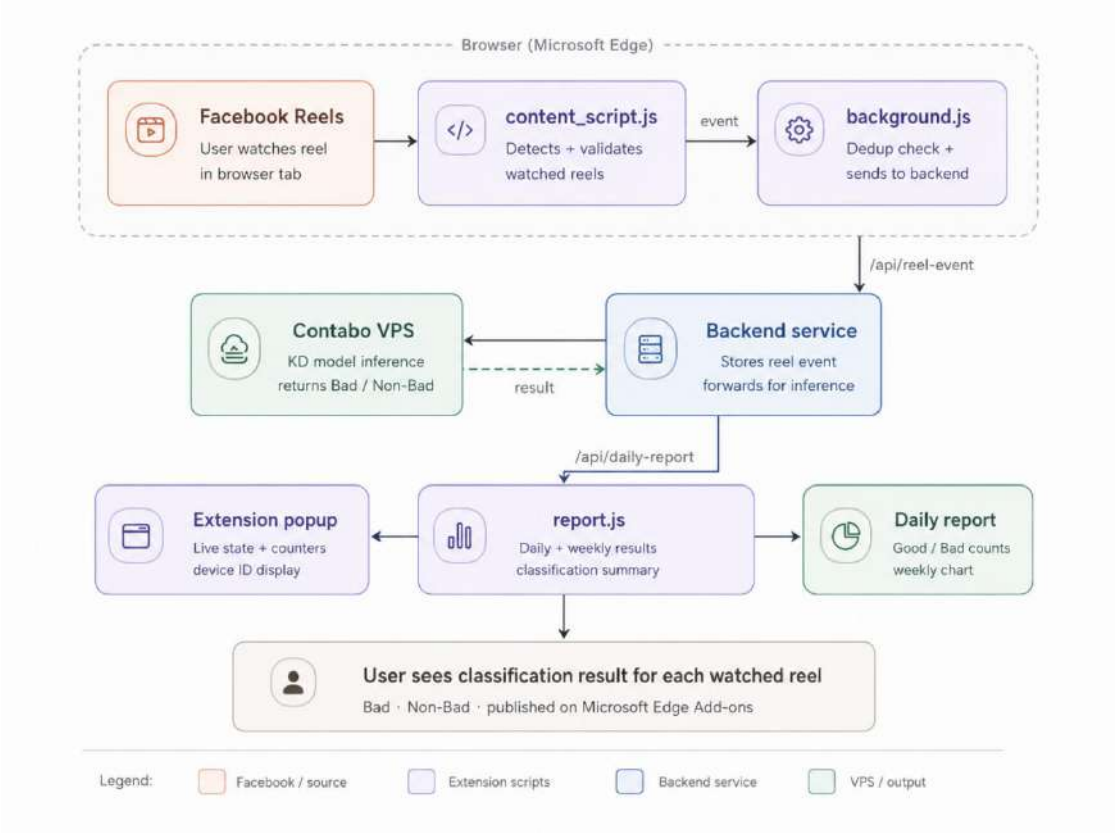


Figure 6.6: Overall workflow of the Reels Guard browser extension.

The *Reels Guard* extension was also published on the Microsoft Edge Add-ons platform, making it accessible as a browser extension for practical use.[1]

6.7.1 Extension interface and monitoring

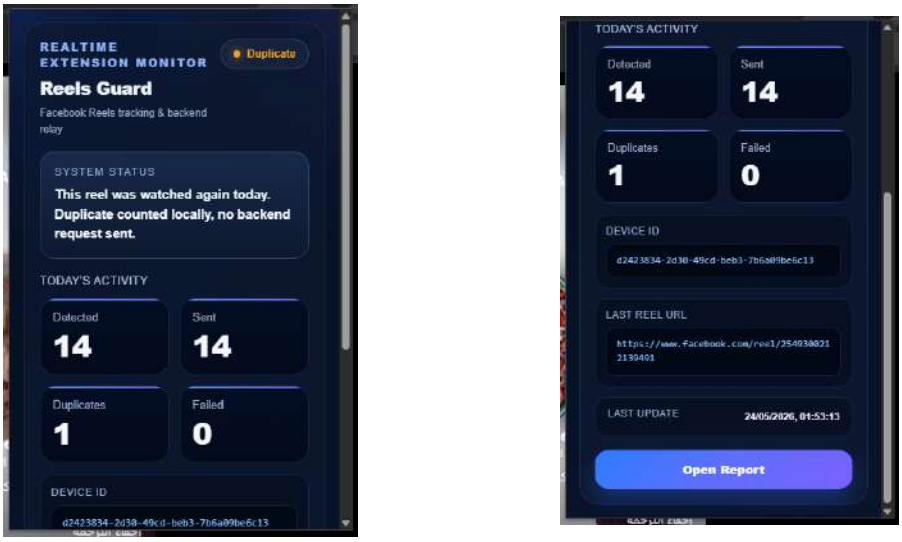
The extension includes a lightweight monitoring interface that allows the user to check whether the extension is active and whether reel detection is working correctly.

- **Current state:** the popup displays the operational state of the extension, such as whether it is ready to detect reels.

[1] https://microsoftedge.microsoft.com/addons/detail/reels-guard/fpdplihcdhdphgloohgjmhhhlaekdfnd

- **Activity counters:** the interface shows the number of detected reels, successfully sent reels, duplicate detections, and failed requests.
- **Device identifier:** a local device ID is used to associate the detected activity with the current browser installation.
- **Report access:** the extension provides access to a daily report page summarizing the analyzed reels.

The extension monitoring interface is shown in Figure 6.7.



Extension monitor -1- Extension monitor -2-

Figure 6.7: Reels Guard extension monitoring interfaces.

6.7.2 Extension workflow and model integration

6.7.2.1 Reel detection

The reel detection process starts in `content_script.js`. It monitors Facebook pages, detects valid Reels, and ensures that the video is visible and being watched before sending it to the backend.

- `normalizeFacebookReelUrl()` converts different Facebook reel or video links into a consistent format.
- `getMainVisibleVideo()` selects the main visible video on the page.
- `isVideoActuallyWatchable()` checks that the video is visible, loaded, playing, and not ended.
- `evaluateCurrentReel()` periodically checks whether the reel has been watched long enough.

The extension uses a local watch session to track the reel URL, watch duration, and sending status. Once the required watch time is reached, it sends a `FACEBOOK_REEL_DETECTED` message to the background service worker.

6.7.2.2 Backend communication

The backend communication is handled by `background.js`. After receiving a detected reel message, the background script checks whether the reel was already sent during the same day. If it is new, it prepares a payload and sends it to the backend endpoint.

- `getOrCreateDeviceId()` creates or retrieves the local device identifier.
- `getDailyState()` manages daily counters such as detected, sent, duplicate, and failed reels.
- `sendReelToBackend()` sends the reel event to the backend using the endpoint `/api/reel-event`.

6.7.2.3 Model integration

The classification model is not embedded inside the browser extension. When a new reel is detected, the extension sends the event to the backend service. The backend receives and stores the event, then forwards the required processing to the *Contabo Storage VPS 30*, where the model and inference pipeline are deployed.

The *Contabo Storage VPS 30* is responsible for preparing or retrieving the reel data, executing the classification model, and storing the prediction result. The extension only detects watched reels, sends their information, and displays the final results through the report interface.

6.7.2.4 Report retrieval and visualization

The daily report is implemented in `report.js`. This part retrieves the processed classification results from the backend and presents them in a readable interface.

- `loadDailyReport()` retrieves the daily report from `/api/daily-report`.
- `loadWeeklySummary()` retrieves weekly statistics from `/api/weekly-summary`.
- `renderBackendReport()` updates the main report counters.
- `renderWeeklyChart()` displays the weekly comparison chart.

The report summarizes the total number of analyzed reels, the number of reels classified as good or bad, and the weekly distribution of detected content, as shown in Figure 6.8.



Figure 6.8: Daily report interface generated by the Reels Guard extension.

6.8 Conclusion

This chapter presented the practical implementation of the proposed system. It described the execution environment, main tools, labeling platform, backend structure, video storage, and frontend interfaces. It also reported the runtime cost of the original teacher-based pipeline and the reduced size of the final KD package.

Finally, the chapter explained the Reels Guard browser extension, including reel detection, backend communication, model integration, and report visualization. This shows how the proposed classification approach was connected to a practical user-facing system.

General Conclusion

Summary and integrated analysis

This thesis has successfully addressed the problem of classifying Arabic Facebook Reels according to Islamic Sharia-oriented criteria through three connected contributions: dataset construction, model development, and practical deployment, showing how these stages work together as one integrated system.

Contribution 1: dataset Construction and Annotation Methodology

Problem Addressed: The lack of labeled Arabic Facebook Reels datasets with Islamic Sharia-based categories limited supervised learning for this task.

Solution Developed: A collection and annotation methodology was developed using three categories: *Good*, *Acceptable*, and *Bad*, guided by technical and Islamic jurisprudential principles. Three annotators were used to improve label reliability. The final dataset included 23,672 labeled reels.

Significance: Dataset construction was a key contribution because reliable labels were essential for training and evaluation.

Contribution 2: Multimodal Audio-Visual Representation and Classification

Problem Addressed: Reels require effective audio-visual representations for accurate multimodal classification.

Solution Developed: An audio-visual pipeline was developed using Kimi/Whisper-based audio embeddings and visual embeddings extracted from the SigLIP-based visual encoder within SmolVLM.

Performance Validation:

- **Three-class classification:** 79.25% test accuracy and 0.7858 Macro-F1.
- **Binary interpretation:** 89.24% test accuracy and 0.8861 Macro-F1.
- The fused model outperformed audio-only and visual-only models.

Significance: The results confirm the value of audio-visual fusion and provide strong baselines for the knowledge distillation stage.

Contribution 3: Knowledge Distillation and Lightweight Multimodal System

Problem Addressed: The teacher-based system was computationally heavy, requiring 9.03 GB RAM and 6.66 GB GPU VRAM during inference.

Solution Developed: A knowledge distillation methodology was developed to reduce computational cost while preserving the audio-visual structure. Kimi-Audio was used as the audio teacher to train a Whisper-base student, while SmolVLM was used as the visual teacher to train a MobileCLIP/FastViT student.

Model Compression Results:

- **Final KD package:** 431.93 MB on disk and 0.5785 GB RAM loading cost.
- **Reduction:** 96% disk-size reduction and 94% RAM loading-cost reduction.

Classification Performance After Distillation:

- **Three-class classification:** 77.27% accuracy and 0.7773 Macro-F1.
- **Binary interpretation:** 88.96% accuracy and 0.8833 Macro-F1.

Significance: Knowledge distillation enabled practical deployment while preserving the benefits of audio-visual late fusion with lighter student models.

Practical deployment

The final model was integrated into the *Reels Guard* browser extension, which detects watched Facebook Reels, communicates with the backend, and displays classification results through daily and weekly reports. This shows how the proposed method was connected to a practical user-facing system.

Integrated insights across all contributions

The three contributions are closely connected. They show that dataset construction, audio-visual representation, and lightweight deployment form complementary stages of one complete system.

- **Dataset quality enables the whole pipeline:** Careful annotation produced reliable labels, which directly affected what the models could learn.
- **Multimodality is valuable and should be preserved:** Audio-visual fusion improved classification, and knowledge distillation preserved this benefit through a lighter late-fusion design.
- **Theoretical understanding guides practical design:** Fusion and distillation concepts supported the choice of late fusion and hybrid training signals.
- **Iterative refinement is essential in applied research:** The data collection strategy evolved during the work, showing the need for continuous adaptation.
- **Performance metrics must match the application context:** Three-class classification was central, while binary interpretation supported the practical focus on detecting problematic content.

Research impact and contributions to the field

Beyond the technical results, this thesis contributes to culturally grounded AI, Islamic technology, Arabic audio-visual analysis, and efficient AI deployment.

- **Contribution to culturally grounded AI:** The thesis shows that audio-visual deep learning can support culturally and religiously specific classification tasks through clear task design, careful annotation, and suitable evaluation.
- **Contribution to Islamic technology:** The work integrates Islamic jurisprudential principles into a systematic machine learning workflow, supporting technologies aligned with Islamic values.
- **Contribution to Arabic NLP and multimodal learning:** The thesis addresses Arabic short-video analysis by combining audio and visual information, going beyond text-only approaches.
- **Contribution to efficient AI:** The knowledge distillation stage shows that audio-visual benefits can be preserved under deployment constraints, achieving 96% model compression while maintaining classification quality.

Bibliography

[1] Meta. Launching facebook reels globally and new ways for creators to make money. About Meta, February 2022. URL https://about.fb.com/news/2022/02/launching-facebook-reels-globally/. Accessed 2026-05-13.

[2] StatCounter Global Stats. Social media stats algeria, 2026. URL https://gs.statcounter.com/social-media-stats/all/algeria. Accessed 2026-05-13.

[3] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[4] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, Ali Vosoughi, Chao Huang, Zeliang Zhang, Pinxin Liu, Mingqian Feng, Feng Zheng, Jianguo Zhang, Ping Luo, Jiebo Luo, and Chenliang Xu. Video understanding with large language models: A survey. *arXiv preprint arXiv:2312.17432*, 2023.

[5] M. Zha and H.-C. H. Chang. Interpreting multimodal communication at scale in short-form video: Visual, audio, and textual mental health discourse on tiktok. arXiv preprint, 2026. URL https://arxiv.org/abs/2601.15278v1.

[6] Bousbia Laiche Wahiba. Sharia-based guidance for content evaluation criteria. Unpublished written consultation, May 2026. Prepared by a doctoral researcher in Islamic Sharia.

[7] OECD. Online social networks. *Measuring the Digital Transformation: A Roadmap for the Future*, 2019. URL https://www.oecd.org/en/publications/measuring-the-digital-transformation_9789264311992-en/full-report/component-69.html.

[8] Meta. Post facebook and instagram content in meta business suite, n.d. URL https://www.facebook.com/business/help/266630520743543.

[9] P. M. Valkenburg et al. Social media use and its impact on adolescent mental health: An umbrella review of the evidence. Current Opinion in Psychology, 2021.

[10] J. M. Twenge and W. K. Campbell. Associations between screen time and lower psychological well-being among children and adolescents: Evidence from a population-based study. Preventive Medicine Reports, 2018.

[11] B. A. Primack et al. Association between social media use and self-reported anxiety and/or depression: Results from 113 countries. SAGE Publications, 2009.

[12] P. Verduyn et al. Passive facebook usage undermines affective well-being: Experimental and longitudinal evidence. Journal of Experimental Psychology: General, 2015.

[13] J. Fardouly and L. R. Vartanian. Social media and body image concerns: Current research and future directions. Current Opinion in Psychology, 2016.

[14] H. Krasnova et al. Envy on facebook: A hidden threat to users' life satisfaction? Wirtschaftsinformatik Proceedings, 2015.

[15] M. Kim and D. H. Byun. A study on the change of editing style in youtube short-form content, 2022.

[16] M. A. Goncharova. Classification and pragmatic linguistic characteristics of the short vertical video genre in german-speaking educational discourse, 2025.

[17] Kapwing. Social media video aspect ratios and sizes — the 2026 guide. Online guide, 2026. URL https://www.kapwing.com/resources/social-media-video-aspect-ratios-and-sizes-the-2026-guide/.

[18] Z. Li, Y. Xie, R. Netravali, and K. Jamieson. Dashlet: Taming swipe uncertainty for robust short video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022.

[19] USI Università della Svizzera italiana. The short video format: Conceptualizing a ubiquitous artifact of visual platforms, 2026.

[20] Tianyang Yan, Yinshan Tang, Zhihao Lu, et al. Mobile phone short video use negatively impacts attention based on self-reports and event-related potential experiments. *Frontiers in Human Neuroscience*, 18:1383913, 2024.

[21] Meta. Introducing home and feeds on facebook. About Meta, July 2022. URL https://about.fb.com/news/2022/07/home-and-feeds-on-facebook/. Accessed 2026-05-13.

[22] Meta. Facebook feed recommendations ai system. Meta Transparency Center, June 2025. URL https://transparency.meta.com/features/explaining-ranking/fb-feed-recommendations/. Accessed 2026-05-13.

[23] Katy Y. Y. Tam et al. Fast-forward to boredom: How switching behavior on digital media makes people more bored. *Journal of Experimental Psychology: General*, 2024. Advance online publication.

[24] L. Nguyen et al. Feeds, feelings, and focus: A systematic review and meta-analysis examining the cognitive and mental health correlates of short-form video use. *Psychological Bulletin*, 2025.

[25] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2 edition, 2022.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[27] L. Wang, X. Chen, and Y. Liu. Computational challenges in short-form video understanding: A survey. IEEE Transactions on Multimedia, 2022.

[28] M. Hassan, R. Ali, and S. Khan. Multimodal fusion in short video platforms: Techniques and applications. ACM International Conference on Multimedia Retrieval (ICMR), 2023.

[29] Meta. How meta prioritizes content for review. Meta Transparency Center, November 2024. URL https://transparency.meta.com/policies/improving/prioritizing-content-review/. Accessed 2026-05-13.

[30] Meta. How technology helps prioritize review. Meta Transparency Center, January 2022. URL https://transparency.meta.com/enforcement/detecting-violations/technology-helps-prioritize-review/. Accessed 2026-05-13.

[31] YouTube. How youtube reviews content. YouTube Help, 2026. URL https://support.google.com/youtube/answer/13304829?hl=en. Accessed 2026-05-13.

[32] Google. Information and content you can trust. Google Safety Center, 2026. URL https://safety.google/intl/en_ca/safety/content-safety/. Accessed 2026-05-13.

[33] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008. doi: 10.1162/coli.07-034-R2.

[34] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[35] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622, 2008.

[36] Jabatan Kemajuan Islam Malaysia. Garis panduan kandungan berunsur islam dalam media baharu. https://www.islam.gov.my/images/garis-panduan/Garis_Panduan_Kandungan_Berunsur_Islam_Dalam_Media_Baharu.pdf, 2024. First edition.

[37] Mustafa Mat Jubri Shamsuddin, Mohd Fuad Md Sawari, and Fatimah Karim. Islamic guidelines for content creators: A jurisprudential framework. *Ulum Islamiyyah*, 37(03): 46–63, 2025. doi: 10.33102/uij.vol37no03.720. URL https://uijournal.usim.edu.my/index.php/uij/article/view/720.

[38] scikit-learn developers. Supervised learning. https://scikit-learn.org/stable/supervised_learning.html, 2026. Accessed: 2026-05-21.

[39] scikit-learn developers. Multiclass and multioutput algorithms. https://scikit-learn.org/stable/modules/multiclass.html, 2026. Accessed: 2026-05-21.

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763, 2021.

[41] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4995–5004, 2016. doi: 10.1109/CVPR.2016.540.

[42] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V. Le, Alan Yuille, and Mingxing Tan. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17182–17191, 2022.

[43] IBM. What is data labeling? https://www.ibm.com/think/topics/data-labeling, 2026. Accessed: 2026-05-21.

[44] IBM. What is ground truth in machine learning? https://www.ibm.com/think/topics/ground-truth, 2026. Accessed: 2026-05-21.

[45] Google Cloud. What is data labeling? https://cloud.google.com/use-cases/data-labeling, 2026. Accessed: 2026-05-21.

[46] Amazon Web Services. Training data labeling using humans with amazon sagemaker ground truth. https://docs.aws.amazon.com/sagemaker/latest/dg/sms.html, 2026. Accessed: 2026-05-21.

[47] Amazon Web Services. Automate data labeling. https://docs.aws.amazon.com/sagemaker/latest/dg/sms-automated-labeling.html, 2026. Accessed: 2026-05-21.

[48] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979. doi: 10.2307/2346806.

[49] Daniel Braun. I beg to differ: How disagreement is handled in the annotation of legal machine learning data sets. *Artificial Intelligence and Law*, 32(3):839–862, 2024. doi: 10.1007/s10506-023-09369-4.

[50] scikit-learn developers. Preprocessing data. https://scikit-learn.org/stable/modules/preprocessing.html, 2026. Accessed: 2026-05-21.

[51] pandas developers. pandas.dataframe.drop_duplicates. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html, 2026. Accessed: 2026-05-21.

[52] pandas developers. Working with missing data. https://pandas.pydata.org/docs/user_guide/missing_data.html, 2026. Accessed: 2026-05-21.

[53] scikit-learn developers. Standardscaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2026. Accessed: 2026-05-21.

[54] scikit-learn developers. Labelencoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html, 2026. Accessed: 2026-05-21.

[55] Jenn Riley. Understanding metadata. https://www.niso.org/publications/understanding-metadata-2017, 2017. National Information Standards Organization.

[56] Amazon Web Services. Input manifest files. https://docs.aws.amazon.com/sagemaker/latest/dg/sms-input-data-input-manifest.html, 2026. Accessed: 2026-05-21.

[57] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50. URL https://arxiv.org/abs/1206.5538.

[58] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023. URL https://proceedings.mlr.press/v202/radford23a.html.

[59] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

[60] Google Developers. Embeddings. https://developers.google.com/machine-learning/crash-course/embeddings, 2026. Accessed: 2026-05-21.

[61] PyTorch. torch.mean. https://docs.pytorch.org/docs/stable/generated/torch.mean.html, 2026. Accessed: 2026-05-21.

[62] scikit-learn developers. Normalizer. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html, 2026. Accessed: 2026-05-21.

[63] scikit-learn developers. Mlpclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, 2026. Accessed: 2026-05-21.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

[65] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

[66] Hugging Face. Loading models. https://huggingface.co/docs/transformers/en/models, 2026. Accessed: 2026-05-21.

[67] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL https://arxiv.org/abs/2108.07258.

[68] PyTorch. Transfer learning for computer vision tutorial. https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, 2026. Accessed: 2026-05-21.

[69] Hugging Face. Fine-tuning. https://huggingface.co/docs/transformers/en/training, 2026. Accessed: 2026-05-21.

[70] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607, 2020. URL https://arxiv.org/abs/2002.05709.

[71] Hugging Face. Classification heads. https://huggingface.co/docs/setfit/en/how_to/classification_heads, 2026. Accessed: 2026-05-21.

[72] Kimi Team. Kimi-audio technical report. *arXiv preprint arXiv:2504.18425*, 2025. URL https://arxiv.org/abs/2504.18425.

[73] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460, 2020. URL https://arxiv.org/abs/2006.11477.

[74] Hugging Face. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025. URL https://arxiv.org/abs/2504.05299.

[75] Hugging Face. Smolvlm-500m-instruct. https://huggingface.co/HuggingFaceTB/SmolVLM-500M-Instruct, 2026. Accessed: 2026-05-21.

[76] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, volume 35, pages 10078–10093, 2022. URL https://arxiv.org/abs/2203.12602.

[77] ByteDance Research. Vidi: Large multimodal models for video understanding and editing. *arXiv preprint arXiv:2504.15681*, 2025. URL https://arxiv.org/abs/2504.15681.

[78] ByteDance Research. Vidi1.5-9b. https://huggingface.co/bytedance-research/Vidi1.5-9B, 2026. Accessed: 2026-05-21.

[79] PyTorch. torch.nn.crossentropyloss. https://docs.pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, 2026. Accessed: 2026-05-21.

[80] PyTorch. torch.nn.softmax. https://docs.pytorch.org/docs/stable/generated/torch.nn.Softmax.html, 2026. Accessed: 2026-05-21.

[81] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL https://jmlr.org/papers/v15/srivastava14a.html.

[82] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785. URL https://arxiv.org/abs/1603.02754.

[83] scikit-learn developers. Support vector machines. https://scikit-learn.org/stable/modules/svm.html, 2026. Accessed: 2026-05-21.

[84] Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S. Kankanhalli. Multimodal fusion for multimedia analysis: A survey. *Multimedia Systems*, 16: 345–379, 2010. doi: 10.1007/s00530-010-0182-0.

[85] Dhanesh Ramachandram and Graham W. Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108, 2017. doi: 10.1109/MSP.2017.2738401.

[86] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6558–6569, 2019. URL https://arxiv.org/abs/1906.00295.

[87] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL https://arxiv.org/abs/1503.02531.

[88] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021. doi: 10.1007/s11263-021-01453-z.

[89] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6550.

[90] PyTorch. torch.nn.kldivloss. https://docs.pytorch.org/docs/stable/generated/torch.nn.KLDivLoss.html, 2026. Accessed: 2026-05-21.

[91] PyTorch. torch.nn.mseloss. https://docs.pytorch.org/docs/stable/generated/torch.nn.MSELoss.html, 2026. Accessed: 2026-05-21.

[92] Ahmad Zamry et al. An islamic approach to web filtering. In *Information Systems and Modern Society*, pages 121–130. Springer, 2014.

[93] R. Masadeh et al. Religious arabic hate speech detection using machine learning. *IEEE Access*, 10:103845–103858, 2022.

[94] Mithun Das et al. Hatemm: A multimodal dataset for hate video classification. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2023.

[95] Sergio Cespedes et al. A multimodal approach for hate speech detection in social media. In *Proceedings of the 2025 International Conference on Computational Linguistics*, 2025.

[96] Wajdi Zaghouani et al. Mahed: A multimodal arabic hate and extremism dataset. In *Proceedings of the Arabic NLP Conference*, 2025.

[97] Meta. Making it easier to create videos on facebook. About Meta, June 2025. URL https://about.fb.com/news/2025/06/making-it-easier-create-videos-facebook/. Accessed 2026-05-13.

[98] Atotti. Atotti/kimi-audio-whisper-encoder: Configuration file. https://huggingface.co/Atotti/Kimi-Audio-Whisper-Encoder/blob/main/config.json, 2025. Accessed: 2026-05-21.

[99] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

[100] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[101] Hugging Face. Huggingfacetb/smolvlm-500m-base: Configuration file. https://huggingface.co/HuggingFaceTB/SmolVLM-500M-Base/blob/main/config.json, 2025. Accessed: 2026-05-21.

[102] OpenAI. openai/whisper-base: Configuration file. https://huggingface.co/openai/whisper-base/blob/main/config.json, 2023. Accessed: 2026-05-21.

[103] timm. fastvit_mci2.apple_mclip2_dfndr2b. https://huggingface.co/timm/fastvit_mci2.apple_mclip2_dfndr2b, 2025. MobileCLIP v2 image encoder for timm.

[104] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5785–5795, 2023. URL https://arxiv.org/abs/2303.14189.

[105] Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel. Mobileclip: Fast image-text models through multi-modal reinforced training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.17049.

[106] scikit-learn developers. accuracy_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html, 2026. Accessed: 2026-05-21.

[107] scikit-learn developers. precision_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html, 2026. Accessed: 2026-05-21.

[108] scikit-learn developers. recall_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html, 2026. Accessed: 2026-05-21.

[109] scikit-learn developers. f1_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html, 2026. Accessed: 2026-05-21.

[110] scikit-learn developers. precision_recall_fscore_support. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html, 2026. Accessed: 2026-05-21.

[111] scikit-learn developers. confusion_matrix. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html, 2026. Accessed: 2026-05-21.

[112] scikit-learn developers. Model evaluation: Quantifying the quality of predictions. https://scikit-learn.org/stable/modules/model_evaluation.html, 2026. Accessed: 2026-05-21.