Video Binary Classification using deep learning techniques

by

Sotirios Panopoulos

Supervisor: Dr. Theodoros Giannakopoulos

Submitted in partial fulfilment of the requirements for the degree of Masters of Science in **Artificial Intelligence**

Faculty of Informatics
University of Piraeus and IIT of NCSR
Athens, Attica, Greece

Submitted to the II-MSc “Artificial Intelligence” on February 09, 2024, in partial fulfillment of the requirements for the MSc degree

Abstract

In the video summarization domain it is needed to efficiently differentiate between informative and non-informative video segments to create concise summaries that encapsulate essential content. Utilizing advanced deep learning methods for feature extraction from both audio and visual data, the study employs a diverse array of optimized classification algorithms and novel LSTM, alongside Attention-based models and Transformers. An early fusion approach integrates audio-visual data to enhance classification accuracy. Despite notable successes, particularly with visual data, challenges in audio feature extraction and certain model performances indicate areas for future improvement. The thesis contributes to the field by demonstrating the potential of combining aural and visual features using deep learning techniques for video binary classification, setting a solid groundwork for advancements in achieving more accurate video summarizations.

Keywords: Video Summarization; Binary Classification; Deep Learning; Audio Feature Extraction; Visual Feature Extraction

Dedication

To my grandfather, Sotirios Sotirios Panopoulos

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Theodoros Giannakopoulos for his guidance throughout this long journey of completing this dissertation. Additionally, I extend my appreciation to the other two members of the examination committee, Professor George Vouros and Dr. Georgios Giannakaopoulos for their patience,insightful feedback and advice.

Last but not least I would like to thank Vlasis, Maria, Sofia, Zoe, Elias, Ilias, George and George for their constant support and patience.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In the ancient Greek mythology, Talos was a giant, thirty-meter height, automaton made of bronze used to protect the island of Crete from invaders. It is often described as one of the earliest concepts of a robot. This mythological figure, embodying the ancient Greeks' imagination of automated guardians, prefigures today's world where Artificial Intelligence (AI) and robotics have begun to play a critical role in our everyday life. Among the domains of AI, Computer Vision stands out as a particularly breakthrough technology. It gives the ability to see and understand the world, enabling a wide range of applications like autonomous vehicles navigating the streets, surveillance systems that can identify and respond to threats and healthcare assistance for making better decisions regarding the treatments of patients.

According to the latest statistics provided by YouTube, Viewers globally watch more than one billion hours on average of YouTube content on their TVs every day and more than five hundred hours of content are uploaded to the platform every minute [31]. The exponential rise in the last years of platforms where short videos are uploaded, like TikTok, YouTube Shorts and Instagram Reels, is evidence that people want to consume more content in less time. Existing audiovisual content, that is lengthy and contains repetitive scenes, could be useful when transformed into more concise versions, such as highlights or summaries. Creators can maximize the reach and engagement of their content, meeting the audience's demand for quick, yet informative and entertaining viewing experiences in the fast-paced digital age we live on.

To meet the increasing demands mentioned before, this thesis digs into the realm of Computer Vision and Computer Audition, originally motivated by the challenges in the video summarization domain, where distinguishing between informative and non-informative video segments is crucial for creating concise summaries that retain all critical content. It aims to leverage the capabilities of deep learning methods to interpret and analyze video data comprehensively. The focus is on developing AI systems capable of making binary distinctions within video segments, determining what constitutes an informative or non-informative one-second clip, so to enhance the utility and efficiency of video summarization processes.

1.2 Related Work

In [22] is addressed the growing need for efficient video summarization techniques due to the exponential increase in user-generated content. Traditional methods often overlook the importance of aural features and are primarily designed for commercial/professional videos. This study presents a novel approach that utilizes both aural and visual features to create dynamic summaries of user-generated videos. These summaries include the most important parts of the original video while preserving their temporal order. The approach involves a supervised binary classifier trained on audio, video, and fused feature representations. A unique user-generated dataset, comprising videos from various categories, is introduced for training and evaluation.

In this paper[23] is proposed an approach focusing on the fusion of audio and visual features and the use of data augmentation. The goal is to create dynamic video summaries that include the most essential segments of the original video while maintaining the original temporal sequence. The approach is based on supervised classification, using deep features from pretrained models and augmented training datasets.

In this work[32], a novel supervised learning technique for video summarization is introduced, utilizing Long Short-Term Memory (LSTM) to model temporal dependencies among video frames for selecting keyframes or key subshots. This approach aims to generate representative and compact summaries by acknowledging the sequential structure of videos, achieving superior results on benchmark datasets. Additionally, it addresses the challenge of requiring extensive annotated data by employing domain adaptation techniques on auxiliary datasets, despite their diversity, to mitigate statistical discrepancies and enhance the summarization process.

A video summarization technique introduced in [19] aimed at quickly overviewing Internet video content, addressing the challenge of identifying important segments within diverse video types without relying on prior knowledge. A deep neural network is designed to encode content semantics—objects, actions, and scenes—into deep video features by mapping videos and descriptions to a common semantic space through joint training. Summaries are generated by extracting these deep features from video segments and applying a clustering-based technique. Evaluated using the SumMe dataset and compared against baseline methods, the results highlight the benefits of incorporating deep semantic features into video summarization processes.

In [30] a novel video summarization technique is presented, that focuses on preserving semantic information, especially long-term temporal semantics. The proposed technique, named Semantic Attended Video Summarization Network (SASUM), utilizes a frame selector and video descriptor to extract semantically relevant video segments for a comprehensive summary. By aiming to minimize the discrepancy between the generated description of the summarized video and human-annotated text of the original video, SASUM ensures the retention of crucial semantic content.

By addressing the challenge of maintaining inherent relationships between a video and its summary while minimizing semantic loss, video summarization is approached in [13]. It is highlighted that while supervised deep learning methods have been effective, they often focus on one challenge without adequately addressing both. The proposed solution introduces an encoder-decoder attention mechanism and semantic preserving loss within a deep Sequence to Sequence framework, aiming to closely monitor and preserve semantic integrity.

Unlike previous attention-based models that observe entire frame sequences, this approach [1] leverages both global and local multi-head attention mechanisms to understand frame dependencies at varying granularity levels. Additionally, it incorporates a component for encoding the temporal positions of frames, crucial for generating coherent summaries.

This paper [17] introduces VISCOM, a novel video summarization approach utilizing color co-occurrence matrices to analyze and condense large volumes of video content across diverse categories. By characterizing video frames through color patterns, VISCOM aims to create synopses that capture the most representative moments.

In [20] a video summarization method leveraging a Generative Adversarial Network (GAN) model pre-trained with human eye fixations to tackle the growing volume of video data from surveillance, medical, and telecommunication systems is presented. The novel contribution of this method is its ability to generate perceptually compatible video summaries by integrating both color perception and spatiotemporal visual attention cues in an unsupervised manner.

Addressing the challenge of managing the vast volume of online videos, this approach [24] enhances video search, retrieval, and browsing. It's contribution lies in formulating video summarization as a sequence labeling problem and introducing fully convolutional sequence models, diverging from traditional recurrent model strategies. By drawing a unique parallel between semantic segmentation and video summarization, the paper adapts semantic segmentation networks to summarize videos effectively.

This study [12] develops a computational model for automatic video summarization, targeting digital videos from television archives. The proposed model draws inspiration from the human visual system and employs computer vision techniques such as face detection, motion estimation, and saliency map computation to generate static video collections of salient images or key frames from the original videos.

1.3 Methodology Overview

The methodology for video binary classification in this study integrates advanced deep learning techniques for both audio and visual feature extraction, alongside a diverse array of optimized classification algorithms. For feature extraction in the audio modality pre-trained CNN are used to extract features via Mel Spectrograms. In the visual modality a VGG19 pre-trained network is deployed and adapted through transfer learning to analyze video frames.

The study exploits various classification algorithms, including Logistic Regression, KNN, Gaussian Naive Bayes, Decision Tree, Random Forest, XGBoost. Novel LSTM, Attention-based models, and Transformer binary classifiers are developed to introduce deep learning techniques in the classification problem, each attempted to enhance classification accuracy and the evaluation metrics used.

It is adopted an early fusion approach for integrating audio and visual data, with experiments conducted on Google Colab for computational efficiency. The study focuses on binary classification of video segments into informative or non-informative categories, leveraging multimodal data and extensive computational resources to optimize model performance.

While we achieved notable success, especially with visual data, the limitations around audio feature extraction effectiveness and specific deep learning model performances warrant further investigation and optimization."

1.4 Structure

The rest of the thesis is organized in five main chapters:

In Chapter 2 the foundational concepts and theories that used in the research are addressed. This chapter is divided into several sections, covering machine learning, deep learning, transfer learning, and feature extraction methods for both visual and audio data. It provides the necessary theoretical background of the research.

Chapter 3 describes the data used in the study, including the source of the dataset, data transformations, and explains the selection and preparation of the data

In Chapter 4 the methodology employed in the study is presented in detail including feature extraction techniques for audio and visual data and the binary classification algorithms used. This chapter also outlines the evaluation metrics and the experimental design, providing an explanation of how the research was conducted.

The experiments conducted in this work are presented in Chapter 5. The experimental setup, the results obtained from the experiments, and a discussion of these results are presented.

The final chapter, Chapter 6, summarizes the findings of the research and discusses the conclusions that can be drawn. It also identifies the limitations of the current work and suggests directions for future research.

Chapter 2

Theoretical Background

Machine Learning (ML) and Deep Learning (DL) are subsets of artificial intelligence (AI) that have revolutionized the way we interact with data and technology. Machine Learning is a method of data analysis that automates analytical model building. It enables computers to learn from and make decisions based on data. This learning process is not explicitly programmed but achieved through algorithms that iteratively learn from data, thus allowing computers to find hidden insights.

Deep Learning, a subset of ML, takes inspiration from the workings of the human brain in processing data and creating patterns for use in decision making. It utilizes artificial neural networks, which are algorithms modeled after the human brain, consisting of layers of nodes, or ”neurons”. Each layer can learn to transform its input data in a different way, making DL particularly powerful for complex, large-scale tasks like image recognition, natural language processing, and speech recognition.

The key difference between ML and DL is in how they learn. Traditional ML algorithms become better at their tasks as they are exposed to more data over time. However, they still require some guidance. In contrast, DL algorithms try to learn high-level features from data in a layered manner, and this can be achieved with little or no human intervention.

As technology advances, the applications of ML and DL are becoming increasingly widespread, impacting various sectors including healthcare, finance, transportation, and more. These technologies are not just transforming the way we interact with machines, but they're also profoundly altering how we understand and utilize data.

2.1 Machine Learning

The invention of computers ignited the interest about their potential to learn. Mastering computer learning could unlock new applications and enhance customization, potentially offering insights into human learning abilities and challenges.

While computers still don't learn as adeptly as humans, algorithms for specific learning tasks have been developed, and a theoretical understanding is emerging.

2.1.1 Types of learning

In terms of its types, learning in ML can be categorized into these forms: supervised, unsupervised, semi-supervised, and reinforcement learning. Each of these types represents a different approach and methodology in the learning process, made for specific input and learning outcomes.

Supervised Learning is perhaps the most frequent form of machine learning. In supervised learning, the algorithm is trained on a labeled dataset. This means that the data is already accompanied by the groundtruth, and the model learns to predict outcomes based on this input-output mapping. Applications where supervised learning is used are image and speech recognition, as well as regression and classification tasks.

In supervised learning, we work with a dataset comprising labeled examples and try to find a relationship between data and output:

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Here, each x_i is referred to as a feature vector, representing an individual data point with multiple dimensions. Each dimension j (where $j = 1, ..., D)$) in this vector holds a specific value that characterizes the data point in some way, known as a feature and represented as $x^{(j)}$. The label in the dataset can vary in form – it might be a class in a finite set (e.g., 1, 2, ..., C), a real number, or a more complex structure like a vector, matrix, tree, or graph. Often, especially in this context, y_i is either a class label or a real number. Class labels categorize each example.

The primary objective of a supervised learning algorithm is to develop a model that predicts the label of a new, unseen feature vector. The model learns from the dataset to infer the label based on the input feature vector.

Supervised learning is typically categorized into three types: binary, multiclass and regression. In binary classification, the objective is to categorize data points into one of two distinct classes. The algorithm is trained on a dataset where each input is labeled with one of these two classes, and it learns to predict which class a new input belongs to.Multiclass classification extends the concept of binary classification to scenarios where there are more than two classes. These two types are classification tasks. Regression deals with predicting a continuous value.

Based on the methodology and the nature of the output supervised learning could be also categorized either as instance-based and model-based or as single-label and multi-label tasks.

In unsupervised learning, the data used to train the model is not labeled, meaning that the model has to identify patterns and relationships in the data on its own. This type of learning is useful for exploratory analysis, as it can uncover hidden structures in data. Unsupervised learning tasks include clustering, dimensionality reduction and outlier detection.

Semi-Supervised Learning falls between supervised and unsupervised learning. In semi-supervised learning, the algorithm is trained on a dataset that includes both labeled and unlabeled data. This approach is useful when acquiring a fully labeled dataset is expensive or time-consuming.

Reinforcement learning, as described in [27], focuses on identifying the appropriate actions in specific situations to maximize rewards. Unlike supervised learning, where the learning algorithm is provided with examples of optimal outputs, reinforcement learning requires the algorithm to learn through trial and error. It involves a series of states and actions where the algorithm interacts with its environment. In this setting, the choice of action influences not only the immediate reward but also affects future rewards at subsequent time steps.

2.1.2 Binary Classification

Since the task in hand is a binary classification task we will provide a quick overview of the algorithms used.

Logistic Regression

Logistic Regression is a statistical method used in machine learning for binary classification tasks, where the goal is to predict a binary outcome (such as yes/no, true/false, or 0/1). It's a type of regression analysis that is suited for situations where the dependent variable is categorical.

Unlike linear regression, which predicts continuous outcomes, Logistic Regression estimates the probability that a given input point belongs to a certain class. The core concept is to transform the output of a linear equation (using the standard logistic function, also known as the sigmoid function) to a probability value ranging between 0 and 1. This function outputs a smooth curve that can classify the data points as belonging to one of the two categories.

The sigmoid function takes the form:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where e is the Euler's number and is the base of the natural logarithm.

Logistic Regression is widely used due to its simplicity, efficiency, and interpretability. It works well for problems where the boundary between classes is linearly separable.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric learning algorithm. In contrast to other learning algorithms that may deallocate the training data after the model is built, kNN keeps all training examples in memory. The core idea of KNN is to predict the label of a new data point based on the labels of its 'k' nearest neighbors in the training dataset.

The process involves calculating the distance between the new data point and all points in the training set. The algorithm then identifies the 'k' closest points, or 'neighbors', and determines the output label based on the majority label among these neighbors.

The hyperparameters of KNN is the choice of 'k' which is the number of neighbors to consider and the distance metric. A smaller value of 'k' makes the model sensitive to noise in the data, while a larger 'k' value makes it computationally expensive and possibly less precise in defining the locality of the data point. Popular distance metrics are Euclidean distance, Manhattan distance, Mahalanobis distance, Hamming distance and cosine similarity.

KNN is favored where the relationship between features is complex and other models are hard to train. However, its performance can degrade with high-dimensional data (the curse of dimensionality) and it can be computationally intensive with large datasets.

Bayes

The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' Theorem, widely used for classification tasks. It's called "naive" because it makes a strong assumption that the features used to make the prediction are mutually independent given the target variable. Despite this simplification, Naive Bayes classifiers often perform remarkably well and are particularly popular in text classification tasks.

Bayes' Theorem provides a way of calculating the probability of a class label based on prior knowledge. In classification it computes the probability of a class given a set of features. The Bayes' rule is:

$$P(y|x) = \frac{p(x|y)P(y)}{p(x)}$$

Where $P(y|x)$ is the posterior probability of the class given predictors. $P(y)$ is the a priori probabilities of the class. $p(x|y)$ is the likelihood of the predictor given the class and $p(x)$ is the probability density function of x.[26]

Naive Bayes classifiers work efficiently with large datasets, are easy to implement and can be used with high-dimensional data. The assumption of feature independence simplifies the computation, allowing the model to operate by considering each feature's contribution to the probability independently.

There are different types of Naive Bayes models depending on the nature of the features in the data, like Gaussian Naive Bayes or Multinomial Naive Bayes, where each type makes different assumptions about the distribution of data so it suitable for different types of datasets. Naive Bayes can outperform more complex models, especially when the assumption of independent features holds true or nearly true.

Decision Tree Learning

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.[16] It is an acyclic graph used to make decisions.[3] The model predicts the value of a target variable by learning simple decision rules inferred from the data features. It is called a ”tree” because the model takes the form of a tree structure, comprising branches, nodes, and leaves.

In a Decision Tree, each internal node examine a specific feature or attribute, each branch represents one of the possible values for this feature, and each leaf node represents a class label, a decision taken after computing all attributes. The paths from root to leaf represent classification rules.

The most well-known decision tree algorithms include ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3), CART (Classification and Regression Trees), and CHAID (Chi-squared Automatic Interaction Detector).

One key advantage of Decision Trees is their interpretability and simplicity. They are easy to understand and visualize, making them useful for explaining the decision-making process to non-technical stakeholders. However, they tend to overfit, especially when the tree becomes too complex. This can be handled through techniques like pruning, setting a maximum depth for the tree, or requiring a minimum number of samples to split a node.

Overfitting occurs when our model tries to capture minor changes in the dataset, which only represents a limited sample of all possible instances of the phenomenon we are attempting to model.

Decision Trees form the building blocks for more advanced methods like Random Forests and Gradient Boosting, where multiple trees are combined to produce more accurate and robust models.

Ensemble Learning

Ensemble learning is a machine learning paradigm where multiple low accuracy models referred as ”weak learners” are trained to solve the same problem and combined to get better results instead of big accurate model.

The most frequently used weak learner is a decision tree learning algorithm. The obtained trees are shallow and not particularly accurate, but if the trees are at least slightly better than random guessing, then we can obtain high accuracy by combining a large number of such trees.

The weak learners can be trained simultaneously (as in bagging) or sequentially (as in boosting), and their predictions are combined using methods like averaging, weighted averaging, or voting.The two most widely used examples of ensemble learning methods are Random Forest and Gradient Boosting.

Random Forest is an application of the bagging technique. It creates an ensemble of decision trees, typically constructed using a method called bootstrap aggregation, or bagging. Each tree is built from a random sample of the training dataset, and the final prediction is typically made by averaging the predictions of each individual tree for regression problems or by a majority vote for classification problems. Random Forest is effective because it reduces the variance of the model, without increasing the bias. This means it is less likely in overfitting situations.

Gradient Boosting and it's variation XGBoost (eXtreme Gradient Boosting)[4] is a type of boosting technique where new models are added sequentially to correct the errors made by existing models. XGBoost builds one model on top of another, iteratively improving the model's accuracy. It incorporates regularisation (L1 and L2), which improves model generalization capabilities and reduces overfitting.

In both Random Forest and XGBoost, the ensemble approach combines multiple individual models to produce a more powerful and reliable prediction model compared to a single model. This is because ensemble methods can capture a variety of simple patterns in the data by different models, and then blend these patterns to achieve greater predictive performance.

2.1.3 Classifier Evaluation Train-Validation-Tests Splits

In machine learning a dataset typically is split into three different subsets: training, validation, and test datasets, each serving a distinct function in the development and evaluation of a classifier.

The training dataset is used to train the classifier. This subset is usually the largest and is kept separate from the examples used in the validation and test datasets. The purpose of this separation is to ensure that the model learns to generalize from the training data to new, unseen data.

The validation dataset, often smaller than the training set and usually about the same size as the test set, is used to validate the performance of the trained model against various hyperparameters and learning methods. This validation is based on specific performance metrics, and the results are used to decide on the best model and hyperparameters.

Finally, there is the test dataset, which is used for the final evaluation of the model before it is deployed or put into production. This step is crucial to assess how well the model will perform in real-world scenarios.

The need for these separate datasets arises from the potential issue of overfitting, where a model performs well on the training data but poorly on unseen data. While the training dataset helps in building the model, the validation dataset aids in tuning it without compromising its ability to generalize. However, there's a risk of overfitting on the validation dataset too, especially in models with many parameters or in cases of small datasets. Hence, the test dataset becomes essential as it ensures the final model evaluation is based on completely unseen data.

In essence, dividing data into training, validation, and test sets is a fundamental approach in machine learning to develop a model that not only learns effectively from the training data but also generalizes well to new data and performs reliably in practical applications.

2.1.4 Performance Metrics

Evaluating the performance of classifiers is a fundamental aspect of machine learning, as it provides insights into how well a model performs and aids in comparing different models. Several key metrics are commonly used for this purpose.

Confusion Matrix is used to describe the performance of a classification model on a set of test data for which the true values are known. It shows true positives (TP), false negatives (FN), true negatives (TN) and false positives (FP). It reveals all possible misclassifications between the different classes.

Precision is the ratio of correctly predicted positive predictions to the total number of positive predictions. Confusion matrices can be used to calculate precision and recall.

$$Precision = (\frac{TP}{TP+FP})$$

Recall is the ratio of correctly predicted positive predictions to all observations in actual class.

$$Recall = (\frac{TP}{TP+FN})$$

Accuracy represents the ratio of correctly predicted observations to the total observations.

$$Accuracy = (\frac{TP+TN}{TP+TN+FP+FN})$$

F1 Score

The F1 Score is a harmonic mean of precision and recall, offering a balance between these two metrics. The F1 Score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F1 = (\frac{2 \times Precision \times Recall}{Recall + Precision})$$

Since F1 Score is a per class metric most of the times used as “overall F1”. To calculate overall F1 there are: macro - F1 where we compute F1 of each class and average, micro - f1 same with accuracy for single-label classification tasks and weighted - f1 same as macro but weighted by (true) number of samples in class.

This score is particularly useful when dealing with imbalanced datasets where one class is significantly more prevalent than the other. In such scenarios, a model might have a high accuracy by simply predicting the majority class, but this doesn't necessarily indicate a good model. The F1 Score helps to mitigate this by considering both false positives (precision) and false negatives (recall).

ROC AUC Score

ROC (Receiver Operating Characteristic) AUC (Area Under the Curve) Score measures the ability of a classifier to differentiate between classes and is used for binary classification problems. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The TPR is the same as recall, while the FPR is the ratio of false positive results to all actual negatives.

$$TPR = (\frac{TP}{TP + FN})$$

and

$$FPR = (\frac{FP}{FP + TN})$$

The AUC represents the degree to which the model is capable of distinguishing between the two classes. An AUC of 1 indicates a perfect model that makes no mistakes in classification, while an AUC of 0.5 suggests a model that is no better than random chance.

The ROC AUC Score is particularly valuable when evaluating a model's performance across a range of thresholds, which is crucial when the cost of false positives and false negatives varies. Unlike the F1 Score, which is a single value derived at a particular threshold, the ROC AUC provides a consolidated measure of performance across all possible thresholds.

The choice of metrics depends on the specific requirements and context of the task at hand.

2.2 Deep Learning

Deep Learning, a subset of Machine Learning in Artificial Intelligence (AI), is a rapidly evolving field that has gained immense popularity due to its ability to process and learn from large amounts of data. Deep Learning models, particularly neural networks, are inspired by the structure and function of the human brain and are designed to mimic the way humans learn.

Deep Learning models involve layers of algorithms called neural networks. Each layer consists of units, or 'neurons', that transform incoming data and pass the output to the next layer. The "deep" in Deep Learning refers to the number of layers through which data is transformed. More layers allow the network to learn complex, abstract representations of data. Some of the key components of Deep Learning are the neural networks, the activation functions, the backpropagation and the gradient descent.

The center piece in Deep Learning are neural networks. These are structured in layers: input, hidden (one or more), and output. Each neuron in a layer is connected to neurons in the next layer, and these connections hold weights that adjust during learning.

The activation functions introduce non-linear properties to the network, allowing it to learn more complex relationships in the data. Examples of activation functions include Sigmoid, Tanh, and ReLU (Rectified Linear Unit).

Backpropagation is a method used in training neural networks. There we adjust the weights of neurons based on the error, difference between predicted and actual output, calculated at the output layer.

Gradient Descent is an iterative optimization algorithm that minimizes the cost or error function, which measures how far the network's prediction is from the actual result. The model's parameters are adjusted iteratively to find the minimum value of the cost function using the chain rule to backpropagate from output to input.

Some of the advantages of Deep Learning are the superiority of their models as the amount of data increases and the feature extraction where deep learning algorithms automatically detect and prioritize the most relevant features.

Deep Learning represents a significant step forward in the capability of AI systems to learn from and make sense of large-scale and complex datasets. As computational capabilities continue to advance and we gain better insights from neural network operations, DL is currently playing a transformative role in technology and society.

2.2.1 CNN

A Convolutional Neural Network (CNN) is a type of deep learning algorithm which has shown great success in computer vision problems like image classification and object detection. CNNs are essential in computer vision due to their ability to learn complex patterns in data, while requiring relatively little pre-processing compared to other image classification algorithms.

At the core of CNNs are layers that perform convolutions. A convolution is a mathematical operation that involves sliding a filter (or kernel) over the input data (such as an image) to produce a feature map. Kernel or mask is a matrix that defines the filter that is convolved with the image. This process involves multiplication of the kernel with the input, followed by summing the results into a single output pixel. The convolution operation helps the network remove noise, detect features like edges, textures, or specific objects in the image.

In convolution the result is a new shrank image where the image's edges are trimmed.

An essential feature of CNN is the zero padding where it allows us to control the kernel width and the size of the output independently.[8]

To reduce computational demands, it might be beneficial to overlook certain positions of the kernel, effectively reducing the detail in feature extraction. This approach can be seen as a form of downsampling the output of the convolution and is called stride s.[8]

A typical CNN architecture comprises several types of layers, each with a specific function. Convolutional Layer: This is the core building block of a CNN. The layer's filters convolve across the width and height of the input volume, computing the dot product between the filter and input, producing a 2D activation map that represents the response of that filter at every spatial position.

Activation Layer (ReLU or similar): After each convolution operation, an activation function like ReLU (Rectified Linear Unit) or a similar non-linear function is applied. This introduces non-linear properties to the system, allowing the network to learn more complex representations.

Pooling Layer: Pooling (usually max pooling) is used to reduce the spatial dimensions (width and height) of the input volume for the next convolutional layer. It is done to decrease the computational power required to process the data through dimensionality reduction. It also helps in extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer. These layers are typically placed before the output layer and are used to flatten the input into a one-dimensional array for classification.

Output Layer is the final layer where a softmax activation function is used for classification tasks and it returns probabilities of each class.

This architecture that is presenented with the name "LeNet5" in [15] is typical in

CNNs.

The learning process in CNNs involves adjusting the weights of the filters to minimize the difference between the predicted output and the actual output (ground truth). This is typically done using backpropagation and an optimization algorithm like Stochastic Gradient Descent (SGD). During training, the network is exposed to a large number of input images. The network adjusts its weights to try and correctly classify the images by minimizing the loss function.

Unlike traditional algorithms, CNNs can automatically and adaptively learn spatial hierarchies of features from input images. This feature learning is a significant advantage over traditional algorithms where manual feature extraction was required. In CNNs, the same kernel (weights) is applied across all pixels in a layer. This common use of the same weights throughout the layer makes the CNN identify an object regardless of its position in the image.

Connection sparsity is a fundamental characteristic of CNNs. Each output from a convolutional activation is determined by only a limited number of inputs. This aspect is crucial as it allows the network to focus on specific features within a localized area of the input image, enhancing its ability to recognize patterns with high precision. Translation invariance property of CNNs means that the network can recognize objects regardless of their location in the input field. This is one of the reasons why CNNs are doing well in handling image datasets. By being less sensitive to the exact location of features within an input image, CNNs can more reliably detect and classify objects in varied positions and orientations. This makes them particularly suited for tasks like image and video recognition, where the subject of interest could appear in any part of the frame.

Due to parameter sharing and pooling, CNNs are efficient in terms of computation and memory.

Examples of CNNs algorithms are the AlexNet [14] which significantly improved upon LeNet-5 by being deeper, with more filters per layer and used ReLU as the activation functions. VGG Networks in [25] with up to 19 weight layers were the first to use a very deep network. Residual Networks (ResNet) [10] introduced the skip or shortcut connections that allow the flow of gradients directly through these connections. It enabled the training of networks with over 100 layers, solving the vanishing gradient problem.

The vanishing gradient problem emerges when errors backpropagated from deep layers to layers closer to the input become progressively smaller, making these early layers difficult to train in very deep networks, so that increased depth does not always result in improved performance.

CNNs represent a powerful tool in the field of AI and deep learning, capable of handling complex image recognition tasks with high accuracy.

2.2.2 Sequential Modeling- RNN - LSTM - Attention - Transformers

In deep learning, sequential models are designed to recognize the patterns and dependencies in sequences of data. These models are adept at handling data with temporal dependencies. They can capture and utilize information from previous time steps to make predictions or understand the current data point.

Recurrent Neural Networks (RNNs) are a cornerstone of sequential modeling in deep learning. They are designed to process sequences by maintaining a form of 'memory' of previous inputs. This memory helps in understanding the context and making informed predictions.

RNN architectures are versatile, enabling single fixed-sized inputs with multiple outputs, variably sized inputs with a single fixed-sized output, and variably sized inputs with variably sized outputs, which do not necessarily have to be of the same size.

A "loop" mechanism plays a crucial role by transferring information across different steps within the same network. This process involves saving the output of a particular layer and then feeding it back as input to the network. This output, known as the "hidden state," acts as a representation or memory of previous outputs. As a sequence of input vectors (x) is introduced to the network, a recurrence formula is applied at each step, effectively allowing the network to process and remember information from previous inputs in the sequence.

However, traditional RNNs often struggle with long-term dependencies due to issues like the vanishing gradient problem. Training a model becomes impractical with sequences longer than ten when using traditional Stochastic Gradient Descent.

To address the limitations of RNNs, architectures like Gated Recurrent Units (GRU) [5] and Long Short-Term Memory (LSTM)[11] have been developed. These models incorporate mechanisms named gates that regulate the flow of information and can retain long-term dependencies in sequences.

A gate employs an additional hidden unit, rather than relying on the weighting of historical and new data. These gates are composed of a sigmoid neural network layer, coupled with a pointwise multiplication operation.

LSTM are designed to manage and control the flow of information. Unlike standard RNNs, which consist of a single network layer, LSTMs consist of four interacting layers that enable them to effectively retain information for long periods. These units within LSTMs are often referred to as "memory cells" or simply "cells."

The operation of an LSTM can be broken down into several key steps. The first step in the LSTM's process is to determine how much of the past information to keep or forget. This decision is made by combining the current input x with the previous hidden state $h(t-1)$ and applying a sigmoid function. The output of this function, known as the forget gate and decides which information is no longer important and should be discarded. Next, the LSTM needs to assess which parts of the new information are important to keep. Again, it concatenates the input x with the previous hidden state $h(t-1)$ and applies the sigmoid function to filter the values. Additionally, the tanh function is used to weigh the importance of this new information, resulting in two outputs: the input gate i and the tanh gate g. The final step is to decide which parts of the current memory cell state are crucial for the output. The LSTM applies a sigmoid function to make this decision, determining what portions of the cell state should contribute to the output. It also uses the tanh function to scale the cell state values within the -1 to 1 range. By multiplying these two outputs element-wise, the LSTM's output gate o is obtained.

LSTM networks, while powerful for handling long sequences and retaining information over time, have some disadvantages. They are computationally complex and resource-intensive, making them difficult to train, especially with very long sequences. LSTMs process data sequentially, limiting the potential for parallel computation. This architecture poses challenges for transfer learning, which is more successfully applied in CNNs, necessitating new data for each new application. Consequently, their practicality in real-world scenarios, where transfer learning is often crucial, is limited.

2.2.3 Attention - Transformers

More recent advancements in sequential modeling include attention mechanisms and transformers. The rise in popularity of attention mechanisms within deep learning is attributed to multiple factors. Primarily, integrating attention into models has led to state-of-the-art performance across a wide range of tasks. These mechanisms can be effectively trained alongside foundational models, such as recurrent neural networks or convolutional neural networks, through standard backpropagation techniques[2]. The introduction of the Transformer model [29] marked a significant boost in the use of attention mechanisms, showcasing their effectiveness. Disadvantages in RNNs like the parallelizing can be overcome. Originating in machine translation attention approaches found applications in other areas including video summarization [1]

Attention allows the model to focus on different parts of the input sequence when predicting each part of the output sequence, thereby capturing context more effectively. It assigns different weights to different parts of the input, indicating the importance of each part in the context of the entire sequence. In its core the model computes a score for each input token to determine its relevance. The scores are then normalized using the softmax function, converting them into probabilities. The output is a weighted sum of these probabilities and the input features, giving more emphasis to the important tokens.

The Transformer model is primarily used for handling sequences without relying on recurrence, as in traditional RNNs. Different than attention mechanisms (previously mentioned) that map inputs to outputs, self-attention allows inputs to interact with each other. Each token looks at every other token to better understand their context within the sequence. Since transformers do not use recurrence, they incorporate positional encodings to maintain the order of the sequence. The Transformer model typically consists of an encoder stack and a decoder stack. Each consists of multiple identical layers that use self-attention and feed-forward networks.

Some of advantages of Transformers are parallelization, scalability, and adaptability for transfer learning. The lack of recurrence in their design allows them to process entire sequences simultaneously, which results in more efficient and quicker training. Additionally, their ability to handle longer sequences with a greater number of parameters makes them exceptionally effective for large datasets. Their ability to be trained on huge data makes them well-suited for transfer learning applications.

2.3 Transfer Learning

Transfer learning is a powerful technique in machine learning where a model developed for one task (Task A) is repurposed for another related task (Task B). Instead of building a model from scratch for Task B, transfer learning leverages the knowledge gained from Task A, making it particularly useful when Task B has limited annotated data. Tasks A and B should have similar inputs (type and size), although this isn't always a strict requirement

Strategies for Transfer Learning involve using a Pretrained Source Model trained on Task A and using it as a starting point for Task B. The model is then fine-tuned based on the output values of Task B. Another approach, if there is a related dataset large enough, is to develop a source model for Task A. This model then serves as the foundation for Task B, where it is again fine-tuned according to requirements.

To implement transfer learning we could utilize the pretrained Model A, freezing some of its layers to retain the knowledge from the source model. Then, add new fully connected layers and train these on the target dataset. These new layers act as a classifier, using the source model as a feature extractor. Another approach is to use Model A and fine-tune it on the target data, typically with a lower learning rate. This option can be combined with previous one, starting with freezing layers and then progressively retraining the entire network.

In summary, transfer learning is a technique that boosts efficiency and effectiveness in machine learning. By leveraging pre-trained models or related datasets, we can use complex models to new tasks with relatively less effort and resources.

2.4 Visual feature extraction

Visual feature extraction is a critical process in the field of computer vision and image processing, where specific characteristics or attributes are identified and extracted from visual data. This process is essential for understanding and interpreting the content of images and videos, and is widely used in various applications such as image recognition, video surveillance, and automated vehicle systems.

The essence of visual feature extraction lies in transforming raw visual data into a set of features that are more meaningful and informative for specific tasks. These features might include edges, corners, textures, colors, or shapes within an image. For instance, in facial recognition software, features such as the eyes, nose, and mouth are identified and analyzed to differentiate between individuals.

Advanced techniques in visual feature extraction often involve the use of algorithms and machine learning models. Convolutional Neural Networks (CNNs), for example, have become a standard in extracting hierarchical features from images for deep learning tasks. These networks automatically learn to identify and abstract features at multiple levels, leading to more accurate and sophisticated interpretation of visual data.

Another crucial aspect of visual feature extraction is dimensionality reduction. High-dimensional data can be complex and computationally expensive to process. Techniques like Principal Component Analysis (PCA) are used to reduce the number of variables while preserving the most important visual information.

Overall, visual feature extraction plays a pivotal role in enabling machines to perceive and understand the visual world, much like human vision. Its continuous evolution drives forward the capabilities of computer vision systems, making them more efficient, accurate, and applicable to a broader range of real-world scenarios.

2.5 Audio feature extraction

Audio feature extraction is a fundamental process in audio analysis where specific characteristics of sound are identified and isolated. This process involves analyzing audio signals to extract meaningful data, such as tempo, pitch, timbre, and rhythm, which are crucial for various applications like speech recognition, music classification, and sound event detection.

Techniques commonly used include Fast implementation of the Discrete Fourier Transform (DFT) for frequency analysis, Mel Spectrogram, Mel-Frequency Cepstral Coefficients (MFCCs) for representing the short-term power spectrum and chroma vector for capturing harmonic pitch class profiles.

The Mel Spectrogram represents a signal in both time and frequency dimensions,

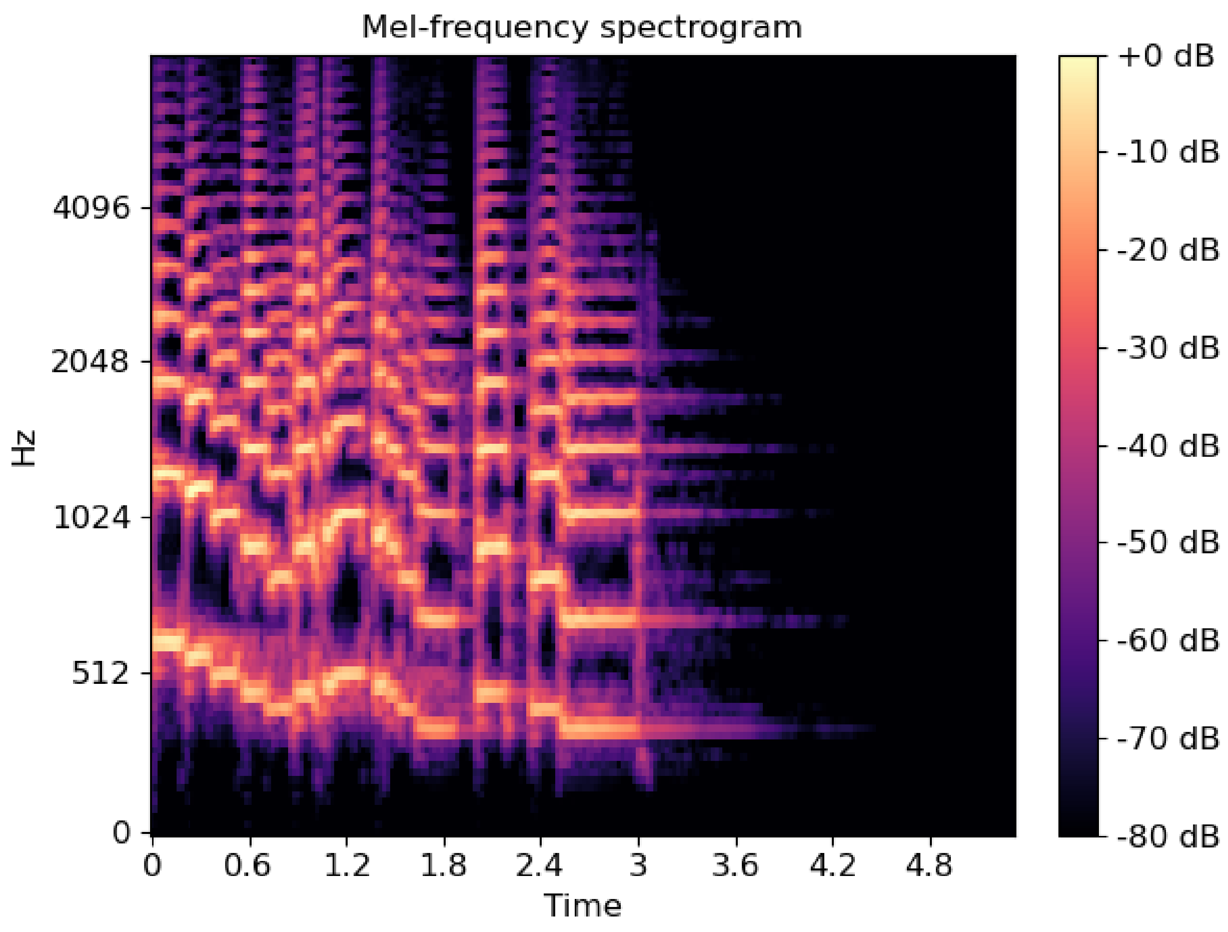

Figure 2.1: A Mel Spectrogram

but on the Mel frequency scale. To construct this, the initial step involves creating the Short Time Fourier Transform (STFT) of the audio signal. This is done by dividing the signal into segments, each defined by a chosen window size. When the window length is sufficiently small, it's assumed that the frequencies within each segment remain constant, allowing for the computation of the Fast Fourier Transform (FFT). However, a smaller window size leads to better temporal resolution but less accurate frequency resolution, and vice versa. This trade-off is due to the fixed resolution nature of the STFT, where the product of time and frequency deviations is constrained. The final step in this process is to concatenate all these window segments to form the spectrogram.

Subsequently, the spectrogram is transformed from the Hertz scale to the Mel frequency scale. This conversion is needed because the Mel scale mimics the way a human ear reacts to a sound, by being more sensitive at the lower frequency and less so to higher ones.

The Mel filter bank consisting of overlapping triangular filters is constructed, each overlapping halfway with the adjacent triangle. These triangular filters peak at a value of 1 at the center frequency. Finally, by applying the Mel filter bank on the power spectrum of each frame, the Mel spectrogram is obtained.[28]

The Mel scale (mel) is the result of the frequency (f) scale that is transformed in a non-linear way using the following formula [18]:

$$mel = 2595 lg(1 + \frac{f}{700})$$

The Mel Spectrogram is widely used as an audio representation and due to its 2D structure is compatible with the CNNs. In that way complex audio data are transformed into a more manageable form, enabling algorithms to perform tasks like genre classification, emotion recognition in speech, or identifying specific sounds within a noisy environment, to bridge the gap between raw audio data and actionable insights.

Chapter 3

Dataset

3.1 Source data set

The dataset comprises user-generated videos, categorized into fourteen distinct themes: building, cars, climbing, downhill, fishing, kayaking, moto, mountain biking, roller coasters, skydiving, snacks, spearfishing, survival, and theme parks. These videos, primarily first-person footage, are sourced from the online platform YouTube and are predominantly captured using action cameras.

In [22] which provided the dataset, an annotation process was performed. The aim of this phase was to generate ground truth video summaries to aid in the training and testing of the video summarization method. Specifically, a group of 22 individuals was tasked with watching and marking segments of interest in various videos to help establish these ground truth summaries. This task was facilitated by a web application created specifically for this annotation workflow. The application allowed users to sequentially access the videos, providing the flexibility to view the entire content, navigate through it, and identify and mark segments they found noteworthy. Users were instructed to mark the beginning and end timestamps of these segments, with no limit on the number of segments they could identify, thus ensuring the process remained highly subjective. A total of 1430 videos received annotations. While most videos received annotations from three to four individuals, some videos annotated from up to eight annotators.

A process of aggregating the annotated data is conducted, combining individual annotations to form a comprehensive ground truth summary. As highlighted in the earlier process, the annotated video summaries varied in the number of annotators per video, affecting the dataset robustness. Consequently, to enhance the dataset's integrity, videos annotated by fewer than three individuals were removed from the original collection. The resulting dataset comprised 336 videos, each annotated by at least three individuals. Ground truth was established by a majority rule, considering a segment informative if agreed upon by 60% of annotators. Annotations were processed into binary arrays for each second of video, with the average agreement calculated to determine the final ground truth, using a threshold of 0.6 for consensus. The average agreement as a macro averaged F1 metric was equal to 72.8% [22].

Consequently, the source dataset utilized in this thesis comprises two primary components: firstly, a collection of 336 videos spanning across 14 distinct categories, and secondly, the corresponding ground truth for each video.

3.2 Data transformations

In order to facilitate the experiments carried out in chapter 5 there are some data transformation made in the source video dataset. The videos were required to be split into two modalities: audio and images, in order to facilitate their processing and handling.

For the audio modality is used the "FFmpeg" python library[6]. "FFmpeg" is a widely-used, powerful tool that can handle video and audio processing tasks such as conversion, compression, and alteration of media files. To summarize, our process begins by taking a video file as input, which is then processed using "FFmpeg" to convert it into a mono audio file with a sample rate of 8000 Hz. The resulting output is subsequently saved in the WAV file format.

Concerning the visual modality, it was necessary to treat it as a sequence of images. This approach allowed for the application of algorithms and the utilization of sophisticated feature extraction methods, as outlined in Chapter 4. Therefore, each video was divided into image frames extracted from the source video file, with a sampling rate of one frame per second.

Chapter 4

Methodology

4.1 Feature Extraction

4.1.1 Audio Feature Extraction Methodology

In this work 'deep_audio_features' [7], a python library, is used, designed for exploiting CNNs on audio classification tasks.

The first input parameter is the model which is a pre-trained four class model trained to distinguish the classes 'music', 'other','silence' and 'speech'. The $file_path$indicating the path of the audio file to be tested, $layers\_dropped = 1$ this argument specifies that one layer from the end of the neural network model should be dropped before making predictions in order to extract the audio features. The feature extraction method is the Mel Spectrogram with a spectrogram size of $(128, 51)$ which is later resized to $(21, 128)$

The process runs for each audio file in the dataset and then it stored in tensor file (.npy).

4.1.2 Visual Feature Extraction Methodology

In this work 'deep_video_extraction' [21] is used to extract the visual features. Visual feature extraction using a VGG19 convolutional network [25], particularly one trained on the ImageNet dataset, represents a sophisticated approach in the field of computer vision. The VGG19 architecture, known for its depth and robustness, consists of 19 layers including 16 convolutional layers, and is highly effective in extracting complex features from images.

Trained on the vast and diverse ImageNet dataset, which contains over a million images categorized into thousands of classes, the VGG19 network has learned to identify and interpret a wide array of visual features. This pre-training enables the network to have a deep understanding of various visual patterns and textures, making it capable of handling different kinds of image recognition tasks.

This process involves transfer learning 2.3, which is beneficial for tasks like classification. For the feature extraction, the first frame of each 1-second segment of the video is analyzed, based on the frames per second (fps) information available in the video metadata.

In this process, the frame vector from each second of the video 3.2 is fed into the VGG19 pre-trained model. VGG19 model architecture with a total of 19 layers, including 16 convolution layers, 3 fully connected layers, 5 max pooling layers, and 1 SoftMax layer. In [23] the last four layers of the VGG19 model are omitted to lessen the model's domain specificity. The intermediate output from this modified network serves as the feature representation for the visual modality, yielding a total of 4096 features.

4.2 Classifications Algorithms

We have utilized libraries from scikit-learn, a powerful open-source machine learning library for Python, to implement the classification algorithms. Scikit-learn's comprehensive collection of algorithms has enabled us to experiment with various classification models efficiently. By leveraging the library's model evaluation modules, we were also able to evaluate our models' performance.

For the deep learning part we used PyTorch a dynamic, open-source machine learning library. PyTorch offers a comprehensive ecosystem with a wide array of tools and libraries to facilitate the modeling, training, and evaluation processes in, making it a powerful tool for deep learning projects. The library's integration with Python and support for GPU acceleration ensures that both development and training processes are efficient and streamlined.

In our project, key modules such as torch, torch.nn, torch.optim, and torch.utils.data play crucial roles in developing and training neural network models. It provides a wide array of complex multi-dimensional arrays known as tensors with added GPU acceleration. The 'torch.nn' module helps in building neural networks providing the building blocks needed to create and train neural networks, such as layers, activation functions, and parameters. There is a module that provides various optimization methods like SGD, Adam and other utility modules for loading and preprocessing data efficiently.

In this logistic regression model several parameters are configured to optimize its performance. The solver parameter is set to 'lbfgs', which stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm and is the default choice for logistic regression in scikit-learn.The *max_iter* parameter, set to 100,000, defines the maximum number of iterations taken for the solvers to converge. A higher number of iterations allows the algorithm more opportunity to find the optimal solution. The *random_state* parameter is fixed at 42, which ensures that the same sequence of random events occurs each time the code is run, making the results consistent and comparable across different runs. Lastly, the *C_param_range* is a set of values $0.001, 0.01, 0.1, 1, 10, 100, 1000$ that represent the inverse of regularization strength. Regularization is a technique used to prevent overfitting by discouraging overly complex models in logistic regression. The 'C' parameter controls the degree of regularization (smaller values specify stronger regularization). A range of values is provided so that the optimal balance between bias and variance can be found so an equilibrium between accuracy and generalization is achieved.

In the configuration of the KNN model, the primary parameter set is the nearest neighbors, which is assigned the value of 5.

The 'GaussianNB' classifier is part of the Naive Bayes family of classifiers and it assumes that the continuous values associated with each feature are distributed according to a Gaussian distribution. The Gaussian distribution is defined by two parameters: the mean and the variance which are automatically calculated from the training data for each class and feature.

In the 'DecisionTreeClassifier' the criterion parameter is set to 'entropy' where it chooses splits that maximize the reduction in entropy. In that way a tree that captures the most information about the dataset at each split is created. Limiting the depth of the tree $max\_depth = 6$ helps prevent overfitting while allowing the tree to learn from the data sufficiently and keeping its complexity in check.

The 'BalancedRandomForestClassifier' is used which is an extension of the standard Random Forest classifier designed to handle imbalanced datasets. The $criterion =' gini'$ parameter specifies the function used to measure the quality of a split in the decision trees of the forest. The 'gini' criterion refers to the Gini impurity, a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. It is used for binary splits. The $n\_estimators = 400$ defines the number of trees in the forest. More trees generally improve the performance and make the model more robust. This parameter $class\_weight = "balanced\_subsample"$ deals with class imbalance by adjusting weights inversely proportional to class frequencies in the input data. Like in Logistic Regression $random\_state$ parameter is fixed at 42.

The 'XGBClassifier' implements the XGBoost algorithm. The $n\_estimators = 400$ is the same as in 'BalancedRandomForestClassifier'. The $scale\_pos\_weight = imbalance\_rate$ parameter helps the model to pay more attention to the underrepresented classes. In the parameter $tree\_method =' gpu\_hist'$ is indicated that the model should use the histogrambased algorithm optimized for GPU computation.

LSTM

The LSTM-based classifier ('LSTMClassifier') is designed to handle sequential data. To use the classifier the data are reshaped to fit the LSTM's input requirements.

The model architecture contains a Long Short-Term Memory (LSTM) layer to process the input sequences which are effective in capturing long-term dependencies in sequence data. A fully connected linear layer (self.fc) that maps the output of the LSTM layer to the output dimension. In the forward method hidden state (h0) and cell state (c0) are initialized to zeros for each batch. Then the input is processed through the LSTM layer. The output of the last time step is passed through the fully connected layer (self.fc) to produce the final output.

Model parameters $input\_dim = 419$ is the number of features in the input data. $hidden\_dim = 100$: is the number of hidden states, $layer_dim$ is the number of stacked LSTM layers and is set to 2, $output\_dim = 2$ is the number of output classes for the classification task (informative or non-informative). In training the loss function used is cross-entropy loss $criterion = nn.CrossEntropyLoss()$ and the Adam optimizer with a learning rate of 0.001. Adam is popular due to its effectiveness in handling sparse gradients and adaptive learning rates. The model is configured to use a GPU if available, to speed up training. Training occurs over 18 epochs and within each epoch, the training data is processed in batches.

The 'LSTMClassifier' is designed for the binary classification task with sequential input data. The use of LSTM units allows the model to capture dependencies across different time steps in the data, making it suitable for complex sequence modeling tasks.

Attention

A neural network model named 'ClassifierWithAttention' is defined using PyTorch. It incorporates an attention mechanism and is designed for a classification task. The model has two main components the attention mechanism('Attention') and the classifier with attention('ClassifierWithAttention').

In the first the attention mechanism is used to weigh the importance of different features in the input data. There is one linear layer that transforms the input features and a second one that helps in deriving the attention weights from the transformed features.It applies a linear transformation, followed by a tanh activation and another linear transformation. The result is then passed through a softmax function to generate attention weights, which are then used to create a weighted sum of the input features. The parameters are $feature\_dim$ which is the number of features in the input and is equal to 419. The *step_dim* which is also set to the size of each input sample and *bias* type of boolean indicating whether or not to include a bias term in the attention calculation layers.

The second component is the classifier with the attention mechanism created before. The layers are: $attention_layer$ an instance of the 'Attention' class, a fully connected (dense) layer with ReLU activation function (fc1), another fully connected layer that outputs the final classification results (fc2). First, the input data is passed through the attention layer. The output of the attention layer, which represents the weighted features, is then passed through two fully connected layers with a ReLU activation function in between.

Model parameters $input\_dim = 419$ is the number of features in the input data. $hidden\_dim = 100$: is the number of hidden layer in fc1, $output\_dim = 2$ is the number of output classes for the classification task. In training the loss function used is cross-entropy loss $criterion = nn.CrossEntropyLoss()$ and the Adam optimizer with a learning rate of 0.001.

The model is configured to use a GPU if available, to speed up training. Training occurs over 20 epochs and within each epoch, the training data is processed in batches. (as provided by train_loader). During training the gradients are reset between batches.

The 'ClassifierWithAttention' model is a sophisticated neural network that utilizes an attention mechanism to focus on the most informative parts of the input features for classification. The model's structure, consisting of the attention layer and two fully connected layers, is suitable for complex classification tasks where the relationship between different features significantly impacts the output. The training loop is standard, with gradient descent optimization and a straightforward batch processing approach.

Transformers

A 'TransformerClassifier' is designed to handle classification tasks using the Transformer architecture and its ability to capture complex dependencies within the data in sequence modeling tasks.

The architecture of the model is the following. The model first projects the input features to a specified dimension (project_dim) using a linear layer (self.project_layer). This is crucial as it prepares the input data for processing by the Transformer encoder. The core of the model is the Transformer encoder, which is composed of multiple layers of Transformer Encoder Layers. Each Transformer Encoder Layer consists of multi-head attention mechanisms $nhead = num\_heads$ and a feedforward neural network $dim\_feedforward = hidden\_dim$, with dropout $dropout\_rate$ applied to prevent overfitting.The Transformer encoder processes the input data in a way that captures both local and global dependencies within the sequence. The output of the Transformer encoder is then passed through a fully connected linear layer (self.fc) to produce the final classification output. In the forward pass, the input data is first projected, then permuted to match the input requirements of the Transformer encoder. After being processed by the Transformer encoder, the output is passed through the fully connected layer to generate the final predictions.

The model's input parameters are $input_dim = 419$which is the number of features in each input sequence, $project\_dim = 420$ the dimension to which the input features are projected. The number of heads in the multi-head attention mechanism $num\_heads = 4$, the number of layers in the Transformer encoder $num\_layers = 2$. The size of the feedforward network within each Transformer encoder layer is $hidden\_dim = 2048$, the output classes are 2, indicating a binary classification and last the $dropout\_rate = 0.1$ is used in the Transformer encoder to prevent overfitting.

The model is trained using the Adam optimizer with a learning rate of 0.0001 and CrossEntropyLoss function is used, suitable for binary classification tasks. The training occurs over 40 epochs, with each epoch processing the data in batches. For each batch, a standard training procedure is followed: performing a forward pass, calculating the loss, conducting a backward pass for gradient computation, and updating the model parameters. The model is configured to use a GPU if available, facilitating faster training.

4.3 Evaluation Metrics Used

Based on the discussed performance metrics in chapter 2, the F1 macro average and overall accuracy offer a comprehensive assessment of the classification task being analyzed. The F1 score is particularly relevant as it considers the class imbalance inherent in the task. Positive class recall and precision serve as indicative metrics for the classifier's chosen operational point. For instance, a precision of 50% and a recall of 60% in the positive class implies that half of the identified 1-second segments are truly informative, and 60% of the actual informative segments are correctly identified. The ROC AUC score is also valuable, as it measures the classifier's overall ability to distinguish between the two classes, independent of the selected probabilistic threshold.

4.4 Experimental Design

Throughout the experiments we utilize all the features generated as described in Section 4.1 as input. We employ an early fusion technique, integrating the two modalities (audio and visual) before inserting them into the models, as outlined in [22]. For machine learning algorithms, we calculate features and metrics separately for audio and visual modalities, as well as in their fused form. However, for deep learning classifiers, we exclusively use and evaluate the fused features.

Regarding the training process, we established a training set comprising 80% of the videos from the initial dataset. Detailed discussions about the parameters of the algorithms used as classifiers can be found in the section 4.2.

All experimental procedures and code executions for this project are conducted using Google Colab, a cloud-based platform provided by Google [9]. It provides access to powerful computational resources, including GPUs in order to handle large datasets and complex algorithms for running machine learning and deep learning experiments. The choice of Google Colab and in later stages the Pro paid version enabled the execution of very demanding tasks using the GPUs in a fraction of the time needed in the free tier or local desktop computers.

Chapter 5

Experiments

5.1 Results

The performance of a random classifier serves as a baseline when evaluating the effectiveness of machine learning models. In the context of our binary classification task, the random classifier achieved an F1 score of approximately 45.21% and an ROC AUC score of 49.87%.

In table 5.1, the scores and outcomes of the experiments conducted using our methodology are presented.

The results indicate that the audio modality has not been effectively utilized, suggesting potential issues with the audio feature extraction method. This is evident in the analysis of the fused features, where the audio contributes minimally to the results, with the visual features predominantly driving the outcomes.

When comparing our results with the reference data presented in [22] (see table 5.2), it is observed that visual features in our work returned improved F1 scores in half of the tested algorithms. This improvement also influenced the corresponding Fused F1 scores. Additionally, the ROC AUC Score in our study outperforms the reference in two out of six cases for visual features and in one case for the fused features.

	F1 Audio	ROC AUC Audio	F1 Visual	ROC AUC Visual	F1 Fused	ROC AUC Fused
Logistic Regression	**43.87**	49.80	**51.74**	64.78	**51.88**	64.80
KNN	49.53	51.90	54.59	59.26	54.60	59.38
Naive Bayes	47.38	51.83	**58.85**	**66.74**	**58.81**	**66.75**
Decision Tree	**43.91**	54.77	44.07	62.11	43.99	62.02
Random Forest	46.15	51.46	**62.51**	**70.18**	**61.92**	70.23
XGBoost	47.66	52.73	53.01	65.49	52.94	67.04

Table 5.1: ML algorithms scores

	F1 Audio	ROC AUC Audio	F1 Visual	ROC AUC Visual	F1 Fused	ROC AUC Fused
Logistic Regression	41.40	62.76	44.62	67.18	49.42	67.38
KNN	54.56	59.28	56.26	60.71	57.65	62.55
Bayes	51.71	59.47	48.32	64.02	51.62	63.35
Decision Tree	41.81	60.58	45.60	66.32	45.57	66.48
Random Forest	57.77	66.72	60.41	69.86	60.58	71.82
XGBoost	59.77	65.32	60.41	66.75	62.34	69.61

Table 5.2: ML algorithms scores in [22]

In table 5.3, we present the results of the original deep learning classifiers developed specifically for this research. The results indicate satisfactory performance for the LSTM and Attention classifiers. However, the results for the Transformer classifier are less encouraging. This underperformance in the Transformer model may be attributed to the training loss, which at its best was at best around 0.4 and did not show signs of convergence throughout the training process.

Further experimentation with the hyperparameters of the LSTM and Attention classifiers, as well as potential modifications to their core architecture, might lead to enhanced results.

	Accuracy	F1 Fused	ROC AUC Fused
LSTM	72.72	54.67	54.45
Attention	71.59	53.68	53.55
Transformer	78.17	43.87	50.00

Table 5.3: DL algorithms scores

Chapter 6

Conclusions and Future Work

This thesis has explored the application of deep learning techniques for the binary classification of video segments into informative and non-informative categories, employing a comprehensive methodology that integrates both audio and visual feature extraction with a variety of classification algorithms. Our experiments, as detailed in the preceding sections, have yielded insightful findings on the effectiveness of different modalities and algorithms in addressing the challenges of video summarization.

6.1 Conclusions

Our results indicate a significant reliance on visual features for classification success, with audio features contributing minimally to the overall outcomes. This suggests that while visual information provides a robust basis for classifying video segments, the potential of audio information has not been fully realized, likely due to limitations in our current audio feature extraction method. Despite this, the improvement in F1 scores for visual features and their contribution to the fused F1 scores highlights the strength of visual data in our methodology.

Notably, the comparative analysis with reference data demonstrates that our approach to visual feature extraction and classification has led to improved performance in half of the tested algorithms. This is further evidenced by the superior ROC AUC scores achieved in our study for visual features and in one instance for fused features, underscoring the effectiveness of our visual data processing techniques.

The evaluation of deep learning classifiers developed for this research, LSTM, Attention-based models, and the Transformer classifier—reveals a promising performance for the LSTM and Attention classifiers. However, the Transformer classifier's performance was notably less effective, with training loss indicating issues with model convergence. This aspect of our study highlights the challenges inherent in adapting complex models like Transformers to specific tasks such as video segment classification.

6.2 Limitations

Our investigation acknowledges several limitations. The underutilization of audio features points to a need for improved extraction methods that can capture the sophistication of audio data more effectively. Additionally, the Transformer model's underperformance suggests that our current implementation may require further tuning or architectural adjustments to fully harness its potential for this application.

6.3 Future Work

Building on the findings of this study, future research should aim to enhance the audio feature extraction process, exploring advanced techniques that could provide a more meaningful representation of audio information for classification tasks. Further experimentation with the hyperparameters and core architecture of the LSTM and Attention classifiers is also recommended to refine their performance.

Moreover, addressing the convergence issues of the Transformer classifier will be crucial. This may involve exploring alternative training strategies, adjusting model architecture, or incorporating additional data sources to improve the model's learning capacity.

In conclusion, this thesis contributes to the field of video summarization by demonstrating the feasibility and challenges of using deep learning for binary video segment classification. By highlighting the strengths and limitations of our approach, we set the stage for future improvements in automated video analysis, potentially making video summarization much more efficient and effective.

Bibliography

[1] Evlampios Apostolidis, Georgios Balaouras, Vasileios Mezaris, and Ioannis Patras. Combining global and local attention with positional encoding for video summarization. In *2021 IEEE International Symposium on Multimedia (ISM)*, pages 226–234, 2021.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[3] Andriy Burkov. *The 100-Page Machine Learning Book*. Andriy Burkov, 2019.

[4] 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*, KDD '16. ACM, August 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[6] FFmpeg. FFmpeg. `https://ffmpeg.org/ffmpeg.html`.

[7] Theodoros Giannakopoulos. deep_audio_features. `https://github.com/tyiannak/deep_audio_features`.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] Google. Google Colab. `https://colab.research.google.com`.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[12] Hugo Jacob, Flávio LC Pádua, Anisio Lacerda, and Adriano CM Pereira. A video summarization approach based on the emulation of bottom-up mechanisms of visual attention. *Journal of Intelligent Information Systems*, 49:193–211, 2017.

[13] Zhong Ji, Fang Jiao, Yanwei Pang, and Ling Shao. Deep attentive and semantic preserving video summarization. *Neurocomputing*, 405:200–207, 2020.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Thomas Mitchell. *Machine Learning.* McGraw-Hill Education, 1997.

[17] Marcos Vinicius Mussel Cirne and Helio Pedrini. Viscom: A robust video summarization approach using color co-occurrence matrices. *Multimedia Tools and Applications*, 77:857–875, 2018.

[18] Douglas O'Shaughnessy. *Hearing.* Wiley-IEEE Press, 2000.

[19] Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. Video summarization using deep semantic features. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*, pages 361–377. Springer, 2017.

[20] George Pantazis, George Dimas, and Dimitris K Iakovidis. Salsum: Saliency-based video summarization using generative adversarial networks. *arXiv preprint arXiv:2011.10432*, 2020.

[21] Theodoros Psallidas. deep_video_extraction. `https://github.com/theopsall/deep_video_extraction`.

[22] Theodoros Psallidas, Panagiotis Koromilas, Theodoros Giannakopoulos, and Evaggelos Spyrou. Multimodal Summarization of User-Generated Videos. *Applied Sciences*, 11(11), 2021.

[23] Theodoros Psallidas and Evaggelos Spyrou. Video summarization based on feature fusion and data augmentation. *Computers*, 12(9), 2023.

[24] Mrigank Rochan, Linwei Ye, and Yang Wang. Video summarization using fully convolutional sequence networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 347–363, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] S.Theodoridis and K.Koutroumbas. *Pattern Recognition 4th edition.* BROKEN HILL PUBLISHERS LTD, 2012.

[27] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[28] Thanh Tran and Jan Lundgren. Drill fault diagnosis based on the scalogram and mel spectrogram of sound signals using artificial intelligence. *IEEE Access*, 8:203655–203666, 2020.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[30] Huawei Wei, Bingbing Ni, Yichao Yan, Huanyu Yu, Xiaokang Yang, and Chen Yao. Video summarization via semantic attended networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[31] YouTube. YouTube press. `https://blog.youtube/press/`.

[32] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 766–782. Springer, 2016.