Doctoral Dissertation
Doctoral Program in Computer and Control Enginering (36.th cycle)

# Creativity injection into AI-powered Multimedia Storyboards

**Bartolomeo Vacchetti**

* * * * *

**Supervisors**
Prof. Tania Cerquitelli, Supervisor
Prof. Elena Baralis. Co-supervisor

**Doctoral examination committee**
Dr. Genoveva Vargas Solar, CNRS, France
Prof. Barbara Catania, Università di Genova, Italy
Prof. Maribel Acosta, Technische Universität München, Germany
Prof. Paolo Garza, Politecnico di Torino, Italy
Prof. Rossano Schifanella, Università di Torino, Italy

Politecnico di Torino
October 31, 2024

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Bartolomeo Vacchetti
Turin, October, 2024

# Summary

In recent years, deep learning has revolutionized the field of computer vision, enabling machines to interpret and understand the visual world with unprecedented accuracy. This impressive technology empowers computers to recognize objects, understand scenes, and even generate realistic images and videos. The recent boost impacted many fields, from autonomous vehicles and medical diagnostics to facial recognition and photo and video editing. One area of video editing that so far has not been investigated in depth is the operation of cutting and concatenating multiple video shots to create a scene. This PhD thesis focuses on integrating machine learning and deep learning approaches to automate the task of concatenating various shots into a meaningful scene. In other words we want to study and teach algorithms what are video editing structures and elements. By video editing structures and elements, we mean the concatenation of shots used to represent a scene or a specific moment in a movie. To complete this complex task we have focused our efforts to address the following three objectives: (i) learn to extract editing sequences from videos; (ii) analyze the correlation between the editing structures and their corresponding videos; (iii) generate new editing patterns and storyboards. A storyboard is a sequence of sketched frames used to pre-visualize a movie, essentially serving as blueprints for the final video. In other words, it is the visual representation of an editing sequence.

To address the different research objectives we have made the following contributions: (i) created a shot size dataset and trained an algorithm to perform shot size classification; (ii) developed a methodology to analyze the correlation between the editing structures and their corresponding videos; (iii) developed a methodology to generate images with the shot size constraint; (iv) developed a methodology to convert textual data into editing sequences.

Shot size classification consists of labeling images into shot sizes. It is an important step because shots are the building blocks of editing sequences. Shots can be categorized by their size based on how close the camera is to the subject. To achieve this, we have created a shot-size dataset and tested different deep-learning algorithms and machine-learning techniques to develop an effective methodology. The results of our research efforts are a shot size dataset with 10 545 images and an algorithm that has an overall accuracy of 80%, which rises up 99% with an

assumption that doesn't distort the model performance. These results show that deep learning models are able to correctly classify images into shot classes. Since we are able to work with basic video elements we can use these simple elements to study more complex movie features and structures.

To analyze what type of correlation there is between editing sequences and videos we have developed a novel methodology to analyze short sequences of shots. Our approach was developed and tested on the Cinescale dataset, in which the shots are characterized only by the shot size, and the Anatomy of Video Editing (AVE) dataset, in which the shots are characterized by also other features. This methodology groups similar sequences of shots based on shot size and other features, exploiting their structural similarities to understand how they reflect in the respective videos. On the Cinescale dataset we tested our approach in different scenarios and analyzed in depth a 16 classes case, a classifier achieved 96% overall accuracy in classifying the sequences labeled by our methodology. On the AVE dataset the accuracy drops to 92.8%, but the classes considered are 50. These results show not only that there is a correlation between what is shown in the video and the sequence of shots used to represent it, but also that deep learning models can correctly identify these structures.

In order to generate new editing sequences and storyboards, we have divided this complex operation into two separate tasks. The first one focuses on image generation with the additional constraint of shot size, essentially recreating single shots from prompts. We have evaluated our approach quantitatively using CLIP-T and DINO scores, and qualitatively, with a survey on 55 subjects. The second task involves creating sequences of shots that can be used to represent movie scenes. Specifically we input textual prompt and as output our methodology gives a sequence of shots that can be used to represent it. Our methodology, which was trained with sequences extracted from the Condensed Movie Dataset, achieved a 0.9280 average cosine similarity score on the training set and 0.8140 on the test set. To sum it up, the first task focuses on recreating individual shots that compose videos, while the second aims to recreate the editing structures that form the overall video or storyboard.

# Acknowledgements

There are many people that supported me during these years and I would like to thank them all. First of all, I would like to thank my Ph.D. supervisors, especially Tania Cerquitelli, without whom this whole experience would not have been possible. Then I would like to thank my parents (Arianna and Marco) and their beloved ones (Riccardo and Lisa), for always supporting (and enduring) me. I also want to thank my siblings (Sisca, Sara, Artu), for always cheering me up without even realizing it. Maybe it's not my place to say but you are the ones that make me proud the most. A special thank to Sarah and Emily for their support in the beginning (Sarah) and at the end(Emily) of this experience. Without the both of you it would have not been the same.

I also want to thank Professor Jean François Lalonde for hosting and guiding me at Laval University in Quebec. A special thanks also to Akshaya Justine and Yannick for their friendship and advice.

A special thanks goes also to Dawit Mureja Argaw, from KAIST, for his helpful collaboration while working on the LEMMS methodology.

I also want to thank the SMARTDATA center and the DBDMG group for their contribution to this doctorate.

Last but not least I want to thank all my friends and extended family. There are too many of you to thank you personally, so I'm just going to do a list (sorry). So Filo, Moga, Gusta, Chiara, Pliz, Flavio, Dadde, Lello, Lollo, Samiro, Ila, Nonno, Nonna, Nonna Elda, Giovi, Pietro, Marghe, Otta, Cate, Ale, Ales, Paolo, Guido and whoever else I forgot thanks.

*"E poi boh..."*

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Context

Recently, deep learning has revolutionized the field of computer vision. As a result, it enabled machines to interpret and understand the visual world with unprecedented accuracy. The marriage of deep learning and computer vision has paved the way for innovations across a multitude of applications—from autonomous vehicles and medical diagnostics to facial recognition and photo and video editing.

At its core, deep learning in computer vision leverages neural networks with multiple layers—hence the term "deep"—to learn hierarchical representations of data. Convolutional Neural Networks (CNNs), a cornerstone of this approach, have demonstrated exceptional prowess in tasks such as image classification, object detection, and segmentation. These networks, which mimic the visual processing mechanisms of the human brain, are able to capture intricate patterns and features in images.

The evolution of vision models has seen significant milestones, from simple feed-forward networks to advanced architectures like VGGs[70], ResNet [79], EfficientNet [81], and Vision Transformers (ViTs)[24]. Each new model has pushed the boundaries of what is possible, achieving even more astonishing performance than its predecessors. ResNet introduced the concept of residual connections, addressing the problem of vanishing gradients in deep networks. EfficientNet optimized model scaling by balancing network depth, width, and resolution. Vision Transformers brought the attention mechanism from natural language processing to vision tasks, enabling the model to focus on relevant parts of an image more effectively.

The rise of diffusion models represents another leap forward, with the ability to generate photorealistic images and videos from textual inputs. These models have opened new frontiers in content creation, allowing for the generation of highly detailed and realistic visuals. However, one aspect that has not yet been thoroughly investigated is the video editing process. Video editing, in this context, refers to the process of cutting and concatenating single video clips with limited meaning

into a cohesive and meaningful video. While current models excel at generating content, the seamless integration of these capabilities into advanced video editing workflows remains a promising area for future research and development.

This PhD thesis focuses on integrating machine learning and deep learning to automate various tasks in video editing. To do so we analyze video editing structures and their elements, briefly described below:

- **video editing**: time consuming process that combines multiple audio and video sources to create a movie scene;
- **scene**: concatenation of different shots;
- **shot**: video captured by a camera, usually characterized by a shot size and other features;
- **shot size**: class that represent the field of view of the camera;
- **frame**: still image extracted from a shot;
- **storyboard**: concatenation of sketches that represent a movie scene;
- **sketch**: more or less refined drawing of a single frame that represents the whole shot;
- **editing structure**: layered structure with multiple different video and audio channels that combine the different sources in the final video;
- **editing sequence/pattern**: concatenation of symbols, where each symbol represent a different shot size, that represent the video;

.

Specifically we analyze the shots and their concatenation to study video editing. Storyboards, which are sequences of sketched frames used to pre-visualize a movie, effectively illustrate these structures. An example of a storyboard can be seen in Figure 1.1.

Figure 1.1 shows a simple storyboard composed of six frames characterized by a sketch. Each sketch represents what the video will show and how close the camera has to be to the subject. While some frames have more meaning than others together they give a much better idea of the final scene.

Storyboards can be more complicated and add additional information, however every storyboard has to at least show a basic sketch of how the characters are arranged. Since we have a sketch we have an indication of the shot size to use. In other words, a storyboard can be thought of as a sequence of sketched shots. More refined storyboards also include camera movements, character dialogues, real

Figure 1.1: Storyboard example

drawings instead of sketches, and so on. To show the power of editing and storyboards let's consider the following example. If we take the short storyboard shown in Figure1.1 and we substitute a single frame we can significantly change the scene. The original storyboard shows a playful guy who smiles while cooking. If we change the first frame as shown in Figure 1.2 we have a creepy guy that poisons food while smiling. Hence storyboards and their editing structures deeply impact the video narrative.

Figure 1.2: Altered storyboard example

Hence, editing structures and their elements, the shots, are the main objects of study of this thesis, since they can impact the mood and structure of a scene. In order to study them, we have divided this complex task into the three following research objectives to highlight and gain knowledge on different aspects of these structures and their elements;

- **Challenge 1**: Extract and characterize editing sequences from videos. This

step is important because the better the characterization of the sequences, the better the data modeling that can be achieved in subsequent steps;

- **Challenge 2**: Study the correlation between the editing sequences and the corresponding videos. In other words we want to see if similar editing sequences have similar videos;
- **Challenge 3**: Generate new editing patterns and storyboards. Leveraging some of the techniques and algorithms previously seen and new models we want to generate new editing sequences and storyboards;

.

To address these challenges, we have made the following contributions.

**Main contributions addressing Challenge 1** In order to extract editing sequences from videos we focused on the shot size classification, which is the task of labeling images according to the shot size used. By being able to label frames into shot sizes we can represent a movie scene as a sequence of shot sizes. To address this task we have created a new dataset with 10,545 images divided into 8 classes to train and finetune models. Machine [15] [19] [33] and deep learning [70] [6] [59] models have been used in the past to classify the shot size or other movie features [33] in the past, but usually the classes taken into account are less. We have developed a methodology that combines pretrained convolutional neural networks and ensemble learning that reached an overall accuracy of 77%. Additionally our approach reduces the severity of their predictions errors. In other words the majority of prediction errors are made between similar classes. When Vision Transformers became available we tested them reaching an overall accuracy of 81%. The results of this research are described in chapter 2.

**Main contributions addressing Challenge 2** In order to analyze the correlation between editing sequences and videos we have developed a clustering based methodology to group short editing sequences based on their similarities. We tested this methodology on 2 datasets: Cinescale and Anatomy of Video Editing dataset. After grouping the editing sequences we trained a classifier to label unseen sequences and we analyze its performance and results. Some early studies [50] [78] [8] on the topic were done in the past, however the majority of studies, such as [69] [61] [91], focus on different aspects of video editing, while the editing sequences, the concatenations of shots, have not been investigated in depth. Still many movie critics highlighted the central role of this process in the creation of a video. The research on this topic is presented in chapter 3.

**Main contributions addressing Challenge 3** To generate new editing patterns and storyboards, due to the complexity of the task, we have divided our efforts in two directions. On the one hand, we developed a methodology that combines Stable Diffusion with novel fine-tunig techniques and tools [64] [35] to generate images with the shot size constraint. On the other hand, we focused on developing a

methodology to generate new editing sequences that can be represented as storyboards, given some textual descriptions. This approach relies on textual and video data in conjunction with large language models, vision transformers, sequence to sequence models and autoencoders. This last contribution motivated the title of the thesis, "Creativity Injection into AI-powered Multimedia Storyboards". The creativity injection comes from the users that have to generate prompts describing possible movie scenes, while the Ai powered multimedia storyboard is the output of TEdit, our methodology to recreate editing sequences. In the wider context of text to image and video generation [51] [63] [71], some studies focused on similar topics, but either not considering features like the shot size [45], or implementing solid but costly solutions [60]. Instead we propose solutions that rely on light or pretrained models, reducing the computational effort.

The research is presented in Chapter 4.

Finally in chapter 5 we present the general conclusions.

To sum it up, the contributions of this thesis project are the following:

- a new shot size dataset with 10,545 images divided into 8 classes and a fine-tuned algorithm able to perform shot size classification;
- a new methodology to analyze the correlation between the editing structures and their corresponding videos;
- a new methodology to generate images from text with the constraint of the shot size;
- a new methodology to create editing sequences from textual directions.

## 1.2 Main Achievements

In this section, we present the main achievements obtained during the doctorate. Overall, I contributed to 10 research papers: 2 journal papers, 7 conference papers on different topics, and 1 paper recently submitted to the IEEE Transactions on Knowledge and Data Engineering (TKDE) (whose pre-print is available on ArXive).

### 1.2.1 Creativity Injection into AI-powered multimedia Storyboards

#### shot size classification

The research on the shot size classification resulted in two publications. In [84], published in the proceedings of the Conference on Computers, Software, and Applications (COMPSAC), we presented an early stage of our methodology and some preliminary results. Later, we published [82] the fully developed methodology on a much more complex scenario in the ELECTRONICS journal.

#### Editing sequences analysis

The research efforts devoted to the analysis of short movie sequences resulted in two publications. The first version of our methodology was presented at the Creative Video Editing and Understanding (CVEU) workshop [83] at the European Conference on Computer Vision (ECCV) 2022. I met Dawit Mureja Argaw, from KAIST, one of the paper's authors presenting the Anatomy of Video Editing (AVE) dataset[3]. We have published a follow-up work with him in which we adapted the methodology to his dataset. The results of our research efforts were presented at the CVEU workshop at the International Conference on Computer Vision (ICCV) 2023 the following year[85].

#### Towards storyboard recreation

The results obtained in the context of shot recreation with the shot size as an additional constraint were presented at the Advances in Databases and Information Systems (ADBIS) workshop at the International Conference on Extending Database Technology (ICDT/EDBT 2024) and published in [25]. As for results concerning the generation of shot sequences we are currently working on the paper to submit it at Journal of Imaging.

#### Research period abroad

During ECCV 2022 I also met professor Jean-François Lalonde, who later hosted me at Laval University in Quebec, from April 2023 up to July 2023. During these 4 months, I carried out research activities within the Professor Lalonde's computer vision group. During this period, we started a collaboration, presented in the appendix, in which we studied editing sequence generation. Even if this approach was later discarded in favor of TEdit, some of TEdit's intuitions and key concepts come from this research.

### 1.2.2 Other research contributions

#### Concept Drift

In this context, we have worked toward the creation of a framework that can detect concept drift. Concept drift refers to the phenomenon where the data's statistical properties change over time in unforeseen ways. This can cause the predictive performance of the model to degrade as the model becomes less accurate and relevant. Hence it is important to detect when the data properties change too much. Our drift detection framework, DRIFTLENS, is unsupervised and works with deep-learning models Rather than directly analyzing the data, our methodology focuses on the data embedding representation extracted from deep models. To

see if the newly arrived data is drifting, we see how the model's internal representation changes with respect to the usual distribution. Additionally, DRIFTLENS is able to characterize the drift, showing which labels were the most impacted. Since our framework works with internal data representation, it can work with different unstructured data types. We tested it on text, images, and audio data with an extensive experimental session in which we compared our framework with other drift detection algorithms, achieving or outperforming Sota performance in all scenarios.

On this topic, we have published a demo paper at the International Conference on Extending Database Technology (EDBT/ICDT) 2024, presenting a tool for drift detection based on our methodology. Furthermore, we recently submitted an article to the IEEE Transactions on Knowledge and Data Engineering (TKDE), whose pre-print is available on Arvix [29].

### Applied Machine Learning

This research branch started with a collaboration with Iveco Group. The project started back in 2021, when Iveco was part of CNH. The initial idea was to develop a machine learning methodology to predict the execution time of simulations launched on an HPC cluster. In order to train a model to predict the running time, we have characterized the simulations in terms of different features, some generic to every simulation, like the number of CPUs, and others specific to the launched simulation itself. To make the prediction we have created a hierarchical structure of XGBoost classifiers. Due to the encouraging results obtained in the first project a follow up project started with Iveco Group and Doit Systems in late 2022.

In this second phase, in addition to increasing the number of past simulations under analysis, we have integrated the prediction time algorithm in the Altair user interface to allow users to estimate the running time before launching a simulation. Additionally we have added a drift module in order to detect shifts in the data distribution and eventually retrain the model if needed. The drift solution adopted comes from the studies made on the subject on other research projects. In [11], presented at the Data Analytics solution for Real Life Applications (DARLI-AP) workshop at the International Conference on Extending Database Technology (ICDT/EDBT) 2022, we introduce the methodology developed in the first project. The results of the second collaboration were presented at the International Conference on Control, Automation and Diagnosis (ICCAD) 2024 [86].

### GINN: Gender Inclusion Neural Network

Due to many factors related mainly to gender biases or unawareness of the problem, many data collection systems are designed with a binary understanding of gender, typically offering only "male" and "female" options. This design inherently excludes individuals who do not identify with the male-female dichotomy.

Consequently, the non-binary community is underrepresented or not considered at all. The underrepresentation of non-binary individuals in data collection leads to gaps in research and the development of policies and services. To address this issue, we have focused our research efforts on developing a methodology to counter the current oversimplification of gender concepts in data-driven systems. To this end, we developed GINN: Gender Inclusion Neural Network, our initial attempt to create an equitable neural network that goes beyond the binary concept of gender and treats it as a multiclass context, including individuals whose gender identity does not conform to the binary male/female spectrum. To highlight the limitations of the current method, we have performed a comparative analysis of several fine-tuned neural network models. To further analyze our results we have integrated explainable AI techniques. The results of our research were presented at the 2023 IEEE International Conference on Big Data [9].

**ESCAPE: a self-tuning framework to analyze large text data collections**

The number and diversity of large scientific datasets have been growing steadily in recent years. Analyzing these datasets is not a trivial task, as numerous algorithms can handle large datasets, but each requires specific parameter settings. Additionally, larger datasets come with increased complexity, necessitating the development of innovative, scalable, and parameter-free solutions. In this research project, we have collaborated to create a self-tuning framework to analyze large text data collections. The developed framework, ESCAPE (enhanced self-tuning characterization of document collections after parameter evaluation), integrates two solutions for document clustering and topic modeling: the joint approach and the probabilistic approach. In both methods, specialized self-optimization strategies have been implemented to configure the algorithm parameters. Additionally, novel visualization techniques and quality metrics have been incorporated to evaluate both approaches' performance and help domain experts interpret the discovered knowledge. Both methods can accurately identify meaningful partitions of a document corpus by grouping documents according to topics.

The results of this research were published in the Applied Sciences journal [23].

# Chapter 2

# Shot Size Classification

## 2.1 Introduction

Shot classification is the task of labeling video and image data, such as movie frames, into shot categories. The shot can be defined by one or more attributes, among which there is the shot size. The shot size is determined by the portion of the subject and of the environment shown in the field of view of the camera. This type of classification can be more or less refined, with a varying number of classes. In addition to the shot size other features, like camera movements or the subject in the scene, can be taken into account to further characterize the data.

In the context of editing structures and storyboards shot constitute the building blocks of a video. The choice of a shot is not random and directors prefer a certain shot with respect to another according to what they want to show and how much they want the viewer to empathize with the subject in the scene [8]. Hence this type of classification can be used to gain ulterior knowledge on video structures.

Automating this task brings numerous advantages, spanning various sub-fields of the creative industry. Analyzing the types of shots used provides valuable insights for film analysis, encompassing both stylistic and narrative elements [6] [66]. For instance, in [78], the authors conduct an automatic authorship attribution of films by examining two features: recording time and cinematographic shot class. This approach reveals intriguing differences in the editing styles of various directors.

Shot size classification can also be used in conjunction with other techniques and algorithms. For instance it can be used to gain knowledge on video structures (see Chapter 3 and 4) or add more control on prompts for text to image tasks (Chapter 4).

In the context of image classification, this task is slightly more challenging than typical image classification because it involves classifying the style of the image rather than its subject. In other words, the classification is based on how the elements within the image are arranged and presented, rather than just the subjects themselves. This chapter presents the research done in the context of shot

size classification. In the first part of this chapter we will give an overview of other studies conducted in this field and we will briefly present a preliminary study with 4 shot size classes using machine and deep learning models. Starting from the preliminary study we have then developed a more complex methodology to address shot size classification in a 8 class scenario, which will be shown in the second part of this chapter. The complete methodology combines predictions from different convolutional neural networks (CNNs) through stacking learning. One CNN makes the prediction on the original image while other two model make prediction on altaration of the same image. Each image alteration focuses on different features.

## 2.2 Related Works

Once a relatively small branch of research, as new computer vision models and algorithms were introduced, also in this sub-field of image classification there have been significant improvements.

One of the first studies on the topic is [15]. In this study the authors gathered 3000 images divided in the following shot types: *long shots, medium shots* and *close ups*. However, instead of classifying the images directly they extracted 5 features for every image and then fed these features to two different models: Decision Trees and Support Vector Machine. The features that they extracted were the following: 2D scene geometric composition, frame color, intensity properties, motion distribution and spectral amplitude. A similar study with focus on new shots was proposed a few years later in [77] by the same team. An interesting approach was proposed in [19]. Here using the size of the subjects head and its position in relation to the rest of the image as discriminants the images are classified into the following seven shot types: *extreme long shot, long shot, medium long shot, medium shot, medium close up, close up* and *extreme close up*.

In [33] the authors introduce a nonparametric camera motion descriptor combined with support vector machine classifier for video shot classification. Instead of characterizing the shots by their size, they classify various camera motion patterns as *static, tilt, pan* or *zoom*. Other studies that focus on different features other than the shot size are [89] [12]. Like in the previous study also in [12] the authors focus on camera motion features but for a different purpose. Specifically they present a method to represent video shots based on camera motion for complex event recognition. For every shot the method generates time series features that are then used to train SVM classifiers. In [89] the authors explore motion-based indexing in films by developing a taxonomy for film directing semantics. Using a novel Markov random field-based motion segmentation algorithm with edge occlusion reasoning, they demonstrate its success by classifying shots from Hollywood movies according to the proposed taxonomy.

More recent works, like [7] [67] [84] [66] focus on shot size classification by relying on convolutional neural networks (CNN) [70] to predict the shot size of an image. With respect to previous studies there are usually more shot classes considered or extra features, like camera movements [33]. In [6] the authors improve the performance of a resnet-50 model in classyfying shot sizes into three classes, *close up*, *medium shot* and *long shot*, by integrating semantic segmentation as image preprocessing step. In another study [59] shots are classified based on the camera movement and the following shots sizes: *long shot, full shot, medium shot, close-up* and *extreme close-up*. In order to do so they propose SGNet, a learning framework for shot type recognition. SGNet utilizes separate streams for subject and background guidance, enhancing accuracy in classifying scale and movement types. In addition to the new framework they introduce MovieShots, a large dataset of 46K shots from 7K movie trailers with annotations. In [37] the authors present MovieNet, a dataset intended to address movie understanding. The dataset can be used to address different tasks, among which there is the shot size classification. It is a comprehensive dataset designed for research in video understanding that provides a rich collection of data that includes not only raw video clips but also detailed annotations and metadata. This dataset is aimed at facilitating various tasks such as scene recognition, action detection and character identification.

MovieNet and MovieShots are not the only movie dataset that are available. In the last years a certain variety of movie dataset have been released to address different tasks. Some of these datasets are for natural language processing tasks, like IMSDB, a dataset of movie scripts [42], and IMDB, a dataset of movie metadata used for sentiment analysis [14]. Other datasets instead are intended for computer vision tasks, like Cinescale [65], Anatomy of Video Editing Dataset (AVE)[3] or Movieshots[59]. Then there are those dataset that can be used for both fields or in a multimodal context, like the Condensed Movie Dataset [5] and MovieNet[37]. Unfortunately some of these dataset were released during publication or after our research on the shot size classification was conducted, however they were used for other research tasks. Hence they will be analyzed more in depth later. Our methodology is able to handle a more complex scenarios compared to most of these studies that take into cosideration fewer shot size classes. In addition thanks to the image alterations created we are able to reduce the severity of the predition errors, sice they are made among similar classes.

## 2.3 Shot Size Classification: preliminary results

This first research effort resulted in a data-driven methodology to automatically classify video frames and images into shots through a supervised model in conjunction with a visual explanator to analyze more in depth the model's predictions.

Among various supervised models, convolutional neural networks (CNNs) excel

in analyzing unstructured data like images and videos by capturing essential properties. CNNs have achieved outstanding accuracy in large-scale image and video recognition tasks, particularly with deep architectures [70]. Despite their success, practical application of CNNs is often hindered by their opaque internal workings and the substantial data requirements for training precise models.

To address these issues, we opted to fine-tune a VGG-16 model [70] pre-trained on ImageNet, a deep convolutional neural network developed by the Visual Geometry Group of Oxford. By fine-tuning the model on our data we are able to keep what the model has previously learned in pattern recognition. To gain more insight on the model's predictions we have used a local visual explanation tool, LIME [62], which clarifies the model's decision-making process.

Our approach was evaluated through various experiments, modifying dataset properties and comparing performance against existing literature. We conducted tests on two datasets containing identical images, differing only in grayscale versus RGB profiles, to assess color's impact on VGG-16's performance. Comparisons with state-of-the-art methodologies affirm the effectiveness of our fine-tuned VGG-16 model for this specific classification tasks.

In this phase work we focused on the following 4 shot size classes: close up, half torso, half figure and full figure. Instead of considering a classic scenario with just close ups, medium shots and long shots, already addressed in other research(see related works), we have chosen these types of shots because they all focus on the human figure and hence present a harder and unexplored scenario to classify.

### 2.3.1 Dataset

The initial dataset consisted of 1 500 images, that with data augmentation became 3 000. The data augmentation performed was a 180 flip on the vertical ax, to mirror the images. More sophisticated techniques like cropping cannot be applied, due to the similarity of the classes. If we crop a half torso it could turn into a close up. The final dataset had 750 full figures, 744 half figures, 758 half torsos and 750 close ups.

The 4 classes considered in this study can be described as follows:

- Full Figure(FF): a shot in which the human figure is shown entirely and occupies the whole screen in height
- Half Figure (HF): a shot in which the subject is shown from the waist above.
- Half Torso (HT): a shot in which the human figure is shown from the upper chest to the head
- Close Up (CU): a shot in which the subject is shown from the shoulders to the head

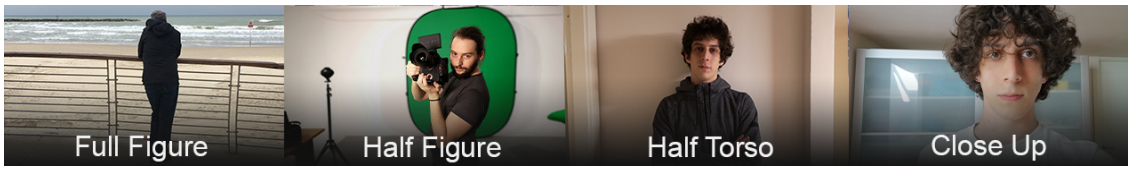



Figure 2.1: Four shots considered in this preliminary study

The original dataset was built by integrating images from multiple sources. The primary source, from which we gathered 56% of the total amount of samples, were video frames downloaded from the Internet. The remaining images were photos taken by amateur(15.67% of the total) and professional photographers (28.33% of the total). The images had a resolution of 160 x 90 pixels to have an aspect ratio of 16:9. This specific format was chosen as is it the most common one, for both images and videos. Additionally other aspect ratios can be converted into the 16:9 without altering the image too much.

### 2.3.2 Methodology

Due to our limited dataset of 3000 samples, we opted to leverage a pre-trained model and fine-tune it. Fine-tuning allows us to utilize the knowledge gained by the model on a different dataset by training specific parts of the model on our data. This approach enables us to exploit the model's previous knowledge to solve a different task effectively. In our case we have used a VGG-16 trained on the Imagenet dataset.

#### Model

The VGG-16 is a convolutional neural network architecture [70, 80, 79] proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its deep structure, consisting of 16 convolutional and fully connected layers. VGG-16 is known for its simplicity and effectiveness in image recognition tasks. It achieved state-of-the-art performance on the ImageNet[21] dataset in 2014, and its architecture has been widely adopted as a base model in various computer vision applications and transfer learning scenarios.

#### Fine-tuninig

Fine-tuning is a transfer learning technique in machine learning, particularly in deep learning, where a pretrained model is further trained on a new, often smaller, dataset to adapt it to a specific task. In our case the model was pre-trained on ImageNet.

The process began by replacing the fully connected layers, responsible for classification, at the end of the network with new ones. In this way we retain the knowledge from the earlier dataset within the remaining convolutional and pooling layers, while the new fully connected layers have to be trained. Before training on the new dataset, some of the old layers were frozen to preserve their learned information, while others were left unfrozen. During training, only the unfrozen convolutional layers and the new fully connected layers adjusted their weights via backpropagation, enabling the model to recognize both simple and complex patterns specific to the new images. The replacement of the old fully connected layers and the partial retraining of some convolutional layers was necessary because the original VGG-16 model trained on ImageNet was not tailored for shot size classification.

### Classification phase

After fine-tuning, the updated VGG-16 model demonstrates its capability to classify new images into various shot sizes. As it processes new images, the convolutional layers of the VGG-16 scan across the input, generating feature maps that are then passed to subsequent layers, which could include additional convolutional or pooling layers. The pooling layers aggregate the most significant values from each feature map and pass them forward in the network hierarchy. To manage the scale and complexity of the data, a nonlinear activation function is applied before pooling to regulate the values within a manageable range and prevent potential evaluation issues.

Once processed through the convolutional and pooling layers, the data is flattened into a one-dimensional array by the flatten layer. This transformation prepares the data for the classification performed by the fully connected layers. The output from these layers is then directed to the final layer, the output layer, which makes the ultimate predictions based on the model's classification criteria. This approach ensures that the fine-tuned VGG-16 model not only adapts to new images efficiently but also integrates learned patterns effectively to classify images into shot sizes accurately.

### Visual explanation

Then as now, machine and deep learning models make a lot of decisions that can be critical and have a social impact, using millions of parameters that are, by their own nature, black-boxes [43]. The CNNs capacity of making decisions exploiting visual information is more and more exploited, especially with the consolidation of architectures that are more complex but also more stable, such as the VGG-16. In order to provide a visual explanation to the user we decided to use LIME, although other methodologies, such as EBANO [87] could have been exploited.

*LIME* [62], in addition to being an easy and efficient approach, can work with both structured and unstructured data and it is able to, given a predictive model, produce a local explanation of a prediction. In order to compute the local explanation, a local approximation of the prediction is performed by LIME. Such approximation is performed by training a local model, which is simpler and more interpretable compared to the original, around little variations of the input data. Thanks to LIME[62],it is possible to show which parts of an image the classifier takes into account while making predictions.

### 2.3.3 Results

We tested our methodology against other classical machine and deep learning models. First we show the technical details then we move to the different experiments and scenarios.

#### Technical Details

The VGG-16 architecture conventionally comprises five blocks, each containing two or three convolutional layers followed by a pooling layer. Beyond these blocks, three fully connected layers handle classification, with the final layer employing softmax activation for predictions. In our study, we utilized a fine-tuned VGG-16 model, where the last convolutional block and subsequent fully connected layers were retrained on our specific dataset.

To assess the efficacy of our approach, we employed Stratified K-Fold Cross Validation (with K=10) and evaluated several metrics: accuracy, precision, recall, and f1-score. Accuracy provides an overall measure of model performance, while precision and recall are particularly valuable for assessing performance on individual classes in datasets with uneven class distributions. Precision is a metric that measures the accuracy of positive predictions made by a model. It is calculated as the ratio of true positive predictions to the sum of true positives and false positives. In other words, precision assesses how many of the predicted positive instances are actually positive. Recall on the other hand measures the completeness of positive predictions made by a model. It is calculated as the ratio of true positive predictions to the sum of true positives and false negatives. Recall assesses how many of the actual positive instances are correctly predicted by the model. The F1 score, or F-measure, is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, making it a useful overall measure of a model's performance when dealing with imbalanced datasets or when both precision and recall are important.

The computer used to run the simulations was a MacBook Pro from 2018 with a 2,6 GHz Intel Core i7 6 core processor.

### Comparisons with state-of-the-art approaches

Here we discuss the comparison of the proposed methodology with other types of state-of-the-art networks to better evaluate the quality of our fine-tuned VGG-16. To see if the fine-tuning approach is effective we tested it against two state-of-the art models: a multilayer perceptron network MLP and a generic CNN, with a simpler architecture compared to the VGG-16. Since the VGG-16 was fine-tuned on the dataset, while previously it was trained on imagenet it could use the previously learned knowledge to improve its performance. This leads to a sensible difference in terms of performance with respect to the other two algorithms.

**CNN )** Convolutional Neural Networks are a class of deep neural networks, very used until the rise of vision transformers, most commonly used to perform tasks like image classification , object detection and image segmentation. It is a type of deep learning model specifically designed for processing and analyzing visual data. CNNs are highly effective at recognizing patterns and features in images due to their architecture that concatenates convolutional and pooling layers.

**MLP )**Multilayer Perceptron is a type of feedforward artificial neural network designed to map sets of input data onto appropriate outputs. It consists of three or more layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node in one layer connects with a certain weight to every node in the subsequent layer, and the nodes in each layer are fully connected to the nodes in the adjacent layers. MLPs are considered one of the simplest forms of neural networks, yet they are powerful for a variety of classification and regression tasks. However while they achieve good performance in capturing complex relationships in data they struggle while handling high-dimensional data like images spatial understanding. We will see that this observation is true also in this context.

Table 2.1: VGG-16 Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Full Figure | 0.90 | 0.84 | 0.87 | 75 |
| Half Figure | 0.82 | 0.85 | 0.83 | 74 |
| Half Torso | 0.77 | 0.76 | 0.76 | 76 |
| Close Up | 0.81 | 0.81 | 0.81 | 75 |
| Accuracy | | | 0.81 | 300 |
| Macro Avg | 0.82 | 0.81 | 0.81 | 300 |
| Weighted Avg | 0.82 | 0.81 | 0.81 | 300 |

Both the MLP and CNN models underwent training using a cross-validation technique with K=50 folds. However, each fold of the CNN was trained for 30 epochs, whereas the MLP was trained for 60 epochs.

A first comparison between VGG-16 MLP and CNN first focuses on accuracy,

Table 2.2: MLP Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Full Figure | 0.60 | 0.63 | 0.61 | 75 |
| Half Figure | 0.66 | 0.57 | 0.61 | 75 |
| Half Torso | 0.48 | 0.48 | 0.48 | 74 |
| Close Up | 0.57 | 0.54 | 0.55 | 76 |
| Accuracy | | | 0.55 | 300 |
| Macro Avg | 0.58 | 0.55 | 0.56 | 300 |
| Weighted Avg | 0.58 | 0.55 | 0.56 | 300 |

Table 2.3: Generic CNN Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Full Figure | 0.74 | 0.75 | 0.75 | 75 |
| Half Figure | 0.71 | 0.62 | 0.66 | 75 |
| Half Torso | 0.55 | 0.59 | 0.57 | 74 |
| Close Up | 0.68 | 0.67 | 0.68 | 76 |
| Accuracy | | | 0.65 | 300 |
| Macro Avg | 0.67 | 0.65 | 0.66 | 300 |
| Weighted Avg | 0.67 | 0.65 | 0.66 | 300 |

where VGG-16 demonstrates superior performance. The training accuracy of VGG-16 stabilizes between 95% and 100% before the twentieth epoch, whereas MLP reaches around 85% after 60 epochs and CNN similar performances to the VGG after 25 epochs. The Test accuracies for the three models are lower than their respective training accuracies, meaning that all the models overfit, with VGG-16 that has an overall accuracy of 81.29%, the MLP model that has only 55.43% and the CNN that achieves 65.44%.

Additionally, a comparison of the loss function trends on the training set reveals significant differences between the models. All models rely on the categorical cross-entropy as their loss function. The mean training loss for VGG-16 and the CNN model is slightly below 0.6, while MLP averages around 1.6.

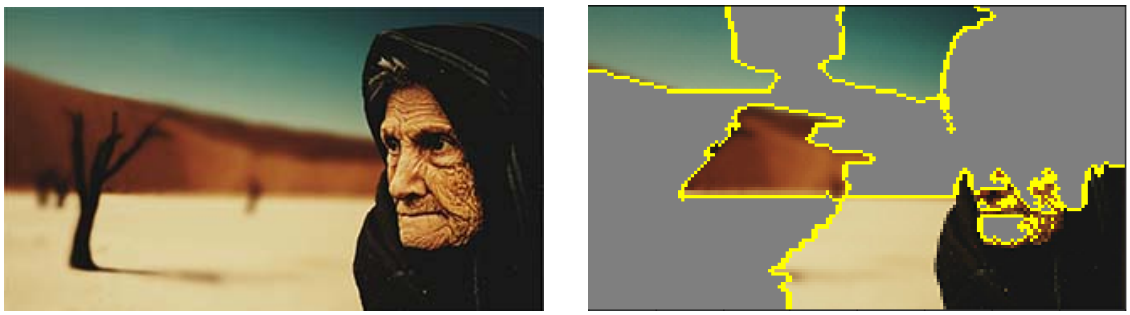
A detailed analysis using classification reports (refer to Table 2.1 for VGG-16 and Table 2.2 MLP and Table 2.3) further confirms VGG-16's superior performance across various metrics.

### RGB vs BW images

Given that colors contribute but are not definitive in discerning whether an image depicts a "close-up" or another perspective, we wanted to see how their absence impacted the model performance. To explore this aspect, we have created

a black and white version of our dataset by desaturating the images in it.

In our experiments, we observed that using DatasetRGB, the original dataset, led to a consistent increase in accuracy for VGG-16 compared to DatasetBW, the dataset with desaturated images. Without colors the model overall accuracy in classifying images into shots drops from 81.29% to 74.56%. This decrease in performance shows that for the model the color information is relevant in order to classify an image into a shot size class. This can be explained with the richer content provided by colored images compared to their grayscale counterparts. This also suggests potential gaps in the model knowledge, since ideally it should be able to discern shot sizes irrespective of color cues.



(a) Original (b) LIME's highlighted patterns

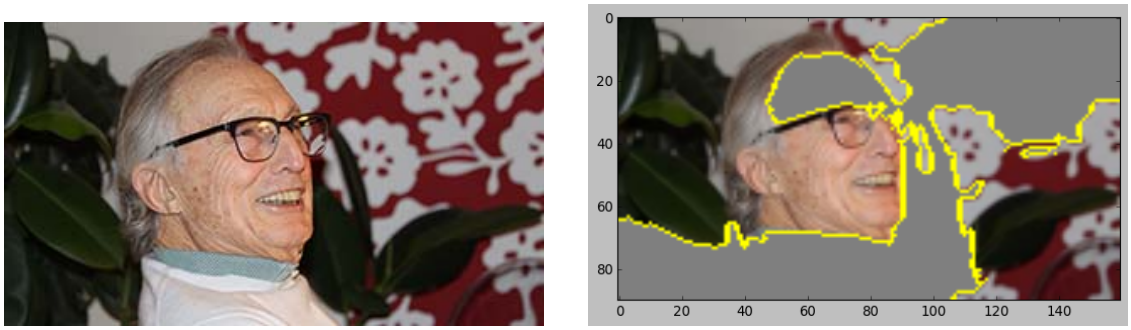
Figure 2.2: Close up shot of an elderly woman

#### Local Explanation

To further test our theory concerning the model incompleteness we used a local explanator to see which part of the images influenced the model the most in predicting its label. For example, Figures 2.2 and 2.3 show which pixels are considered relevant by the VGG-16 in order to make the prediction on DatasetRGB. If the pixels are gray they are considered not important, otherwise they are relevant. Figure 2.3a is an *close up* correctly classified. As the reader can see in 2.3b the network considered some useful elements, such as the nose, the hair and the mouth, and some useless ones, such as the wall behind. Figure 2.2a shows a *close up* classified as a *full figure*. In this case the VGG-16 misclassified the image, it did not consider at all the face of the old woman, while it considered the landscape and the dead tree, that vaguely resembles a human figure, in the background, as shown in 2.2b.

### 2.3.4 Discussion

The VGG-16 definitely outperformed other methodologies, with no surprises, since it could exploit previously learned patterns. However from the black and




(a) Caption 1 (b) Caption 2

Figure 2.3: Close up shot of an elderly man.

white test and the local explanator emerges that its knowledge is far from complete in order to correctly classify the images. Additionally the LIME visual explanator showed that while in some cases the identified pattern makes sense in other contexts the patterns that lead the model to classify an image are not meaningful or lead to misclassification errors. However, overall the performance of the fine-tuned VGG-16 is satisfying considering the size of the dataset and the similarity of the shot size classes considered.

To compensate for the small dataset size and the small amount of class considered we have expanded the dataset and we have developed a more complex methodology to achieve even better performance. This research work which will be described in the next section.

## 2.4 Enhanced Shot Size Classification

Encouraged by the preliminary results described in the previous section, we increased the number of classes from four to eight. The whole set of shot size classes is shown in 2.4. Additionally the dataset size has been increased from the initial size of 1 500 images up to 10 545 images. Due to the increased complexity of the problem, just fine-tuning a VGG-16 is not enough to obtain satisfying performance. One common approach to increase performance is to implement data augmentation techniques. However traditional data augmentation techniques, like cropping in this context can be counterproductive, as for instance it would turn a close up into an extreme close up. Hence, instead of adopting regular data augmentation techniques, we augmented the data by exploiting other neural networks. By doing so we were able to create two alterations of the original dataset. The image alterations were obtained taking into account the following considerations. As shown in [27] ImageNet pre-trained CNNs are biased toward recognizing texture patterns rather than edge patterns. Hence we tried to give more emphasis on the edge patterns without excluding the knowledge from texture patterns. To do so we have created two variations of images, one that emphasized the texture patterns and one that focused on the edge patterns. Then we finetuned three separated VGG-16 on the different types of images: normal, edge, texture. After fine-tuning the model we combine their prediction into a final one using stacking learning technique [54]. By doing so we were able to improve the overall accuracy up to. In addition the misclassified samples belong to similar classes, while before the errors were noisier.

The main contribution of this research are the following:

1. a two-stage methodology to effectively label images into shots. The methodology relies on an ensemble of VGG-16 and stacking learning to make the predictions, where each model receives a different type of image (normal, edge and texture) and a stacking learning strategy to correctly classify different classes of movie shots.

2. An intensive experimental session to validate our approach with a large amount of real movie frames belonging to 8 classes to show the effectiveness of our methodology in performing shot size classification.

3. The creation of a new large dataset of 10,545 images divided into 8 shot sizes classes.

### 2.4.1 Dataset

Here we take into consideration a more refined classification, with 8 shot size classes, that range from extreme close up to long shot, shown in figure 2.4. The complete list is:

- Long Shot (LS): a shot in which the human figure can be absent or occupy less than a third of the screen height.
- Medium Shot (MS): a shot in which the human figure occupies from a third to two thirds of the screen height
- Full Figure (FF): a shot in which the human figure occupies from two thirds to the totality of the screen height
- American Shot (AS): a shot in which the human figure is shown from above the knee above
- Half Figure (HF): a shot in which the human figure is shown from the waist above
- Half Torso (HT): a shot in which the human figure is shown from the upper chest to the head
- Close Up (CU): a shot in which the subject is shown from the shoulders above
- Extreme Close Up (ECU): a shot in which the face of the subject occupies most of the image

There are different datasets implemented in this work; however, since two of them are variations of the original dataset we will start to describe this one first.

The original dataset was created from different datasets and sources, culminating in a total of 10 545 images, distributed across the eight shot classes considered. Initially, 1,500 images were sourced from the dataset utilized in [84]. An additional 5,000 frames were taken from YouTube videos, particularly from movie trailers and reviews. To do so we have employed a straightforward script that implements the youtube-dl library, allowing us to sample frames from entire video sequences.

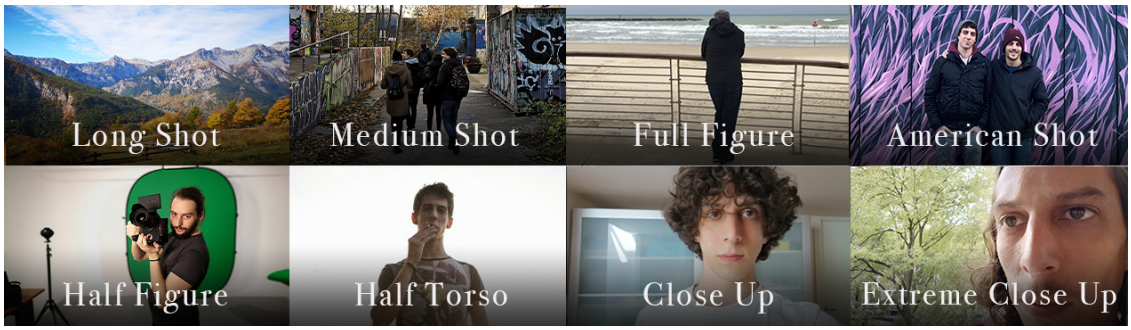



Figure 2.4: Eight shots used

Although the total number of frames could have been significantly higher, we implemented rigorous selection criteria to refine the dataset.

These criteria involved eliminating frames that were too similar to those already included and discarding any frames with blurry resolution. At the end of this first data gathering phase there was a substantial imbalance among the different classes. To address this, we supplemented the dataset with samples from other datasets originally intended for different purposes but deemed suitable for our needs. Specifically, we incorporated images from the MPII Human Pose dataset [2] and the Labeled Faces in the Wild dataset [36]. This approach helped to mitigate the imbalance and ensured a more representative distribution of images across the various shot classes.

The final composition of the dataset is show in table 2.4.

| Class | n samples | percentage |
|---|---|---|
| LS | 1 359 | 12.88% |
| MS | 1 270 | 12.04% |
| FF | 1 080 | 10.24% |
| AS | 935 | 8.86% |
| HF | 1 315 | 12.47% |
| HT | 1 673 | 15.86% |
| CU | 1 731 | 16.41% |
| ECU | 1 183 | 11.22% |

Table 2.4: number of shots and percentage per class.

Since the images in the original dataset come from different sources we have regulated their white balance, in order to reduce a bit the impact of color editing and correction applied to the different movies. An example of this operation can be seen in 2.5. As previously mentioned by fine-tuning a VGG-16 only on the original dataset we obtain discrete results. However there is room for improvement. To improve the overall classification performance we have created two alterations of

the original dataset; the Segmented dataset and the Hypercolumn dataset.


Figure 2.5: Example of white balance operation

### Segmented dataset

This dataset was obtained using semantic segmentation. In order to perform the semantic segmentation we used *DeepLabv3* [16], a model with state of the art performances, using an approach similar to the one described in [6]. DeepLab V3 is a semantic segmentation model developed by Google that is designed to handle various challenges in image segmentation, such as object boundaries and scale variations. The output of the model is an image in which the subject's edges are well separated from the background. An example is shown in 2.6. By reducing the image to such a format we help the model to focus on the edges of the shapes contained in the edited image and we completely ignore texture patterns. This dataset variation was created to counter the bias of CNN toward texture patterns [27] by creating images with only edge patterns. On the other hand relying only on this type of images can be counterproductive, since they still withhold less information than before. Additionally while texture patterns can mislead the model in classifying images, they can also be helpful to differentiate between different classes. Since texture patterns also cover an important role in classification, we have developed the next dataset variation.

### Hypercolumns Dataset

This dataset contains images where the most relevant patterns were highlighted through hypercolumn extraction [32]. A hypercolumn is a concept used in convolutional neural networks (CNNs) to represent a comprehensive feature descriptor for a specific pixel or region in an image. It is formed by concatenating the activation outputs of all the convolutional layers at a particular spatial location. A

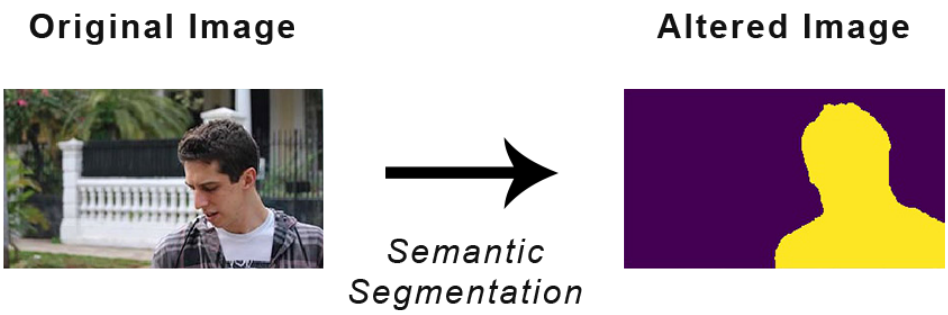



Figure 2.6: Example of image from Segmented dataset

hypercolumn is a vector that aggregates the outputs from multiple layers of a CNN at a specific spatial position. This vector includes features from different levels of abstraction, from low-level details to high-level semantics These patterns serve not only to emphasize significant texture patterns but also to diminish the visibility of irrelevant ones. After extracting the hypercolumns we recreated the image as shown in 2.7.The darker part of the image is for the less relevant patterns, while the brightest is for the most relevant ones. For the extraction of hypercolumns, we employed a VGG-16 model that had been pre-trained on the ImageNet dataset. The relevance of these patterns stems from the fact that some of the classes under consideration exhibit similar visual characteristics. In cases where shots appear similar, the presence or absence of certain elements (i.e., patterns) becomes crucial in determining the class to which a shot belongs. For example, a *half torso* shot and an *extreme close-up* shot share several patterns, such as eyes, noses, and hair texture. However, other patterns are present in a *half torso* shot but not in an *extreme close-up* shot. Therefore, by extracting and projecting the hypercolumns, we aimed to assist the model in distinguishing between these classes as effectively as possible.

### 2.4.2 Methodology

Our methodology consists of a two-stage classification based on ensemble learning. In the first stage image classifiers make predictions on new data and pass them on. At the second stage we recombine the prediction using an ensemble learning technique, stacking learning. Ensemble learning is a machine learning paradigm where multiple models are trained to solve the same problem and then combined to produce a better overall result. The main idea is that by combining the predictions from multiple models, the ensemble can often achieve better performance and generalization than any single model alone. The predictions can be combined

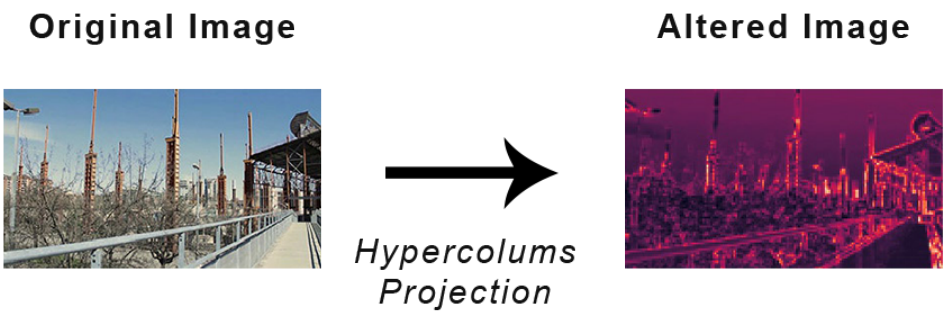



Figure 2.7: Example of image from Hypercolumn dataset

in different ways. Stacking learning involves training a "meta-model" to combine the predictions of several base models. For our methodology to work then we need to finetune image classifiers and train a meta learner, another classifier. After fine-tuning the image models on our datasets we have to train the meta classifier that combines their predictions. The overall pipeline can be seen in 2.8.

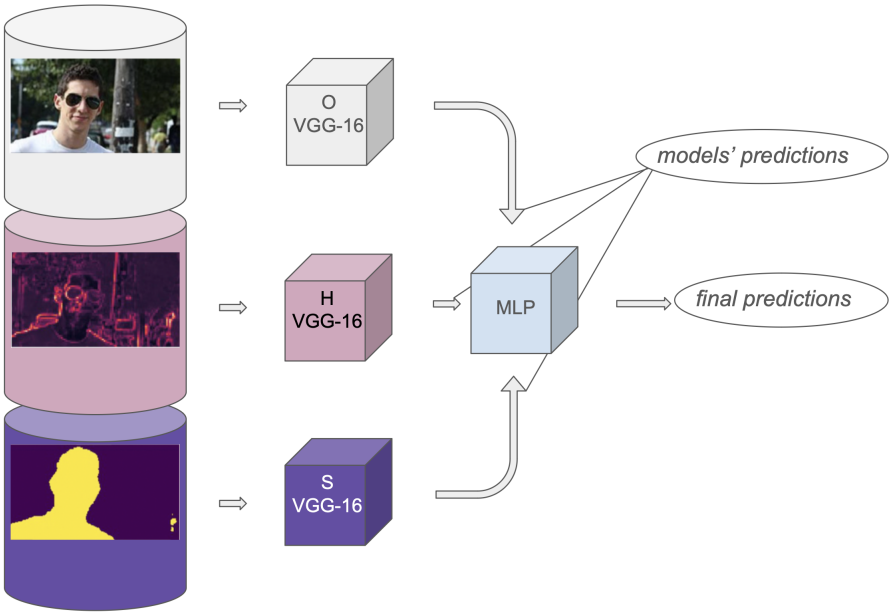



Figure 2.8: Overall methodology

Similar to the previous study, we have removed the fully connected layer and retrained the last convolutional block of the model, by leaving it unfrozen. Also in this case the three VGG-16 models were pre-trained on ImageNet. Like before

this step is necessary because what is classified is how objects are displaced in an image rather than the objects per se. Hence it is necessary to tune a bit also the convolutional layers to achieve a better performance. The blacks of convolutional layers instead do not need to be retrained, since they are the ones in charge of recognizing more simple patterns, common to every type of image.

**Ensemble learning**

There are many techniques and algorithms that fall under this category. the main approaches can be grouped as follows:

1. Bagging (Bootstrap Aggregating): Involves training multiple versions of a model on different subsets of the training data (created using bootstrap sampling) and then averaging their predictions (for regression) or taking a majority vote (for classification). Random Forest is a popular example of a bagging method.

2. Boosting: Sequentially trains models, each trying to correct the errors of the previous one. The final prediction is a weighted sum of the predictions from all models. AdaBoost and Gradient Boosting are well-known boosting methods.

3. Voting: Combines the predictions of multiple models by taking a majority vote for classification or averaging the predictions for regression. There are different types of voting such as hard voting (majority vote) and soft voting (average probabilities).

4. Stacking (or Stacked Generalization): This is a more complex method that involves training a "meta-model" to combine the predictions of several base models.

In [54], the authors demonstrated that combining predictions using stacking ensembles yields better results than using Boosting or Bagging ensembles. The integration of convolutional neural networks (CNNs) with ensemble learning techniques has been explored in various studies [49] [92]. Specifically, in [49], these methods were applied to classify Alzheimer's disease image data. Meanwhile, in [92], the focus was on classifying transportation modes from images using CNNs in combination with ensemble learning. The convolutional models employed in these studies differed in terms of size and the number of layers. In both cases, stacking ensembles achieved the highest performance, with an accuracy of 91%, outperforming Boosting and Bagging ensembles. This indicates the superior capability of stacking ensembles in effectively leveraging the strengths of various models to improve classification accuracy. Hence we have decided to rely on this approach.

### 2.4.3 Results

In this part we will show the performance of our methodology in different contexts and scenarios to validate our approach. Each section focus on different aspects with respect to the others. Specifically:

- **VGG-16s Fine-Tuning**: here we show the performance of our models separately and combined.
- **Case 1**: a comparison between our methodology and state-of-the-art models ResNet-50 and VGG-16 finetuned with augmented data;
- **Case 2**: here we present different implementations of stacking learning impact our methodology;
- **Results: updated**: contains the performance results of a Vision Transformer trained on our original dataset;

The reason why this last paragraph was not included originally in [82] is because Vision Transformers were not available when this research was first published.

#### VGG-16s Fine-Tuning

The three models were trained on 70% of the available data, while the remaining 30% was used to create the test set. To keep consistency between train and test set, and avoid performance alterations, the label distribution is the same in both datasets. Each model underwent training for 20 epochs, with a batch size set to 5. The optimization process utilized the stochastic gradient descent algorithm, and the loss function employed was categorical cross entropy.

The first model, trained on the normal images from the original dataset and referred to as O-VGG-16, achieved a training accuracy of 97.27% and a validation accuracy of 74.27%. For the model trained on the Hypercolumn dataset and referred to as H-VGG-16, the performance reached is slightly inferior than before with a training accuracy of 97.09% and a validation accuracy of 69.5%. Finally, the model trained on the Semantic dataset and referred to as S-VGG-16 achieved a training accuracy of 94.42% and a validation accuracy of 69.41%. As we can see the model trained on the original images performs better. This is no surprise, since the original images contain more information with respect to their correspective variations.

For more insight on the different models performance we show the confusion matrices of the O-VGG-16, S-VGG-16 H-VGG-16 models in Table 2.5, Table 2.6 and Table 2.7 respectively.

After fine-tuning the three VGG-16 models we extract their prediction and recombine them using a meta-learner. In our case we used a Multilayer Perceptron. Our fourth classifier was trained using as input the concatenated predictions of the

| | LS | MS | FF | AS | HF | HT | CU | ECU |
|---|---|---|---|---|---|---|---|---|
| Long Shot (LS) | 301 | 61 | 7 | 2 | 6 | 13 | 7 | 10 |
| Medium Shot (MS) | 29 | 255 | 57 | 13 | 12 | 11 | 1 | 3 |
| Full Figure (FF) | 1 | 27 | 252 | 26 | 4 | 8 | 2 | 4 |
| American Shot (AS) | 0 | 5 | 26 | 192 | 52 | 7 | 0 | 0 |
| Half Figure (HF) | 0 | 3 | 3 | 54 | 256 | 73 | 2 | 2 |
| Half Torso (HT) | 0 | 6 | 4 | 1 | 28 | 435 | 27 | 2 |
| Close Up (CU) | 0 | 2 | 1 | 2 | 1 | 108 | 376 | 29 |
| Extreme Close Up (ECU) | 0 | 1 | 1 | 0 | 1 | 3 | 66 | 283 |

Table 2.5: Confusion Matrix of a VGG-16 trained on the Original Dataset. reading it orizontally we can see how many of the samples were correctly classified and how many where mistaken for other shots

| | LS | MS | FF | AS | HF | HT | CU | ECU |
|---|---|---|---|---|---|---|---|---|
| Long Shot (LS) | 301 | 61 | 7 | 2 | 6 | 13 | 7 | 10 |
| Medium Shot (MS) | 29 | 255 | 57 | 13 | 12 | 11 | 1 | 3 |
| Full Figure (FF) | 1 | 27 | 252 | 26 | 4 | 8 | 2 | 4 |
| American Shot (AS) | 0 | 5 | 26 | 192 | 52 | 7 | 0 | 0 |
| Half Figure (HF) | 0 | 3 | 3 | 54 | 256 | 73 | 2 | 2 |
| Half Torso (HT) | 0 | 6 | 4 | 1 | 28 | 435 | 27 | 2 |
| Close Up (CU) | 0 | 2 | 1 | 2 | 1 | 108 | 376 | 29 |
| Extreme Close Up (ECU) | 0 | 1 | 1 | 0 | 1 | 3 | 66 | 283 |

Table 2.6: Confusion Matrix of a VGG-16 trained on the Segmented Dataset. Reading it orizontally we can see how many of the samples were correctly classified and how many where mistaken for other shots

| | LS | MS | FF | AS | HF | HT | CU | ECU |
|---|---|---|---|---|---|---|---|---|
| Long Shot (LS) | 312 | 63 | 20 | 0 | 7 | 1 | 2 | 2 |
| Medium Shot (MS) | 48 | 251 | 56 | 11 | 11 | 0 | 2 | 2 |
| Full Figure (FF) | 8 | 43 | 238 | 12 | 16 | 1 | 2 | 4 |
| American Shot (AS) | 3 | 15 | 49 | 142 | 67 | 3 | 2 | 1 |
| Half Figure (HF) | 2 | 15 | 20 | 38 | 277 | 34 | 1 | 6 |
| Half Torso (HT) | 8 | 7 | 20 | 5 | 92 | 235 | 44 | 2 |
| Close Up (CU) | 8 | 9 | 9 | 1 | 5 | 68 | 375 | 44 |
| Extreme Close Up (ECU) | 5 | 0 | 1 | 1 | 3 | 5 | 61 | 279 |

Table 2.7: Confusion Matrix of a VGG-16 trained on the Hypercolumn Dataset. Reading it orizontally we can see how many of the samples were correctly classified and how many where mistaken for other shots

three networks on their training data and validated on their concatenated predictions on their validation sets.

The fourth classifier achieved a training accuracy of 95.66% and a validation accuracy of 77.02%. For a detailed performance analysis, refer to Table 2.8, which presents the confusion matrix for the fourth classifier. Additionally, Table 2.9 provides an evaluation of the fourth classifier's performance using precision, recall, and f1-score metrics.

| | LS | MS | FF | AS | HF | HT | CU | ECU |
|---|---|---|---|---|---|---|---|---|
| Long Shot (LS) | 320 | 67 | 3 | 1 | 4 | 6 | 4 | 2 |
| Medium Shot (MS) | 30 | 294 | 36 | 9 | 7 | 3 | 1 | 1 |
| Full Figure (FF) | 4 | 36 | 247 | 23 | 4 | 4 | 2 | 4 |
| American Shot (AS) | 0 | 7 | 18 | 194 | 57 | 5 | 1 | 0 |
| Half Figure (HF) | 0 | 6 | 3 | 44 | 291 | 44 | 2 | 3 |
| Half Torso (HT) | 2 | 6 | 6 | 0 | 51 | 402 | 36 | 0 |
| Close Up (CU) | 1 | 6 | 1 | 1 | 3 | 75 | 409 | 23 |
| Extreme Close Up (ECU) | 1 | 0 | 0 | 0 | 3 | 3 | 66 | 282 |

Table 2.8: Confusion Matrix of the MLP trained on the VGG-16's predictions on their training sets. Reading it orizontally we can see how many of the samples were correctly classified and how many where mistaken for other shots

| shot class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Long Shot (LS) | 89% | 79% | 84% | 407 |
| Medium Shot (MS) | 70% | 77% | 73% | 381 |
| Full Figure (FF) | 79% | 76% | 77% | 324 |
| American Shot (AS) | 71% | 69% | 70% | 282 |
| Half Figure (HF) | 69% | 74% | 72% | 393 |
| Half Torso (HT) | 74% | 80% | 77% | 503 |
| Close Up (CU) | 79% | 79% | 79% | 519 |
| Extreme Close Up (ECU) | 90% | 79% | 84% | 355 |
| accuracy | | | 77% | 3164 |
| macro avg | 78% | 77% | 77% | 3164 |
| weighted avg | 78% | 77% | 77% | 3164 |

Table 2.9: precision, recall and f-1 score of the MLP classifier

Before digging deeper into the results section just by looking at the confusion matrices of the different models we can already see that the stacking learning approach has some merits. In fact the confusion matrix of the fourth classifier is less noisy than the other. On top of that the majority of mistakes are made between visually similar classes, and are less severe. In other words if the model mistakes a

close up for an extreme close up is less relevant than when it mistakes it for a long shot.

### Case 1:Us, VS VGG-16 and ResNet-50 with augmented data

In this first scenario we compare the performance of our methodology with state-of-the-art classification models. These models have been trained with regular augmented data. Back then, the ResNet-50 and the VGG-16 were the standard models to address image classification tasks and both of them have been already tested in shot size classification [66, 6]. ResNet-50 is a deep convolutional neural network that is 50 layers deep. It is part of the ResNet (Residual Network) family of architectures, introduced in [34] ResNet-50 is widely used for image classification tasks and has been highly influential due to its innovative architecture, which addresses the problem of vanishing gradients in very deep networks. Our methodology outperforms the VGG-16 model without relying on traditional data augmentation techniques, highlighting its effectiveness. Moreover, the comparison with ResNet-50 justifies our decision to use the VGG-16 architecture. The performance differences, in terms of f1-score per class and overall accuracy, are summarized in Table 2.10. The table includes various scenarios. Each scenario is identified with a different number that indicates the specific data augmentation techniques used on the dataset. Overall we have:

- Scenario 0: No data augmentation techniques used. The dataset size is its original size.
- Scenario 1: Flip images on the y-axis. The images in the train set are mirrored. The dataset size is twice the original size.
- Scenario 2: Flip images on the y-axis +zoom range = 0.1. The images in the train set are either mirrored or cropped by 10%. The dataset size is three times bigger than the original size.
- Scenario 3: Flip images on the y-axis +zoom range = 0.3. The images in the train set are either mirrored or cropped by 30%. The dataset size is three times bigger than the original size.

All the data augmentation techniques have been applied on the Original Dataset.

As it can be seen between the ResNet-50 and the VGG-16 models the latter clearly outperforms the former. If we test our approach against other VGG-16s trained with regular data, but augmented we can still see how our approach is still slightly better. In terms of data augmentation techniques we can see that in general if we augment the data the performance of the model increases. However in Scenario 3 we can see how if we crop the image too much the model performance starts to drop. The models in Scenario 0 and Scenario 3 are the same, the main difference is

Table 2.10: Comparison with VGG-16 and ResNet-50.

| **Model** | | | **f1** | **Score** | **per** | **Class** | | | **Accuracy** |
|---|---|---|---|---|---|---|---|---|---|
| | **LS** | **MS** | **FF** | **AS** | **HF** | **HT** | **CU** | **ECU** | |
| VGG-16 0 | 82% | 69% | 75% | 67% | 68% | 75% | 75% | 82% | 74.26% |
| VGG-16 1 | **86%** | 69% | **77%** | 69% | 69% | 71% | **79%** | 83% | 76.55% |
| VGG-16 2 | 85% | 70% | **77%** | **72%** | **72%** | 76% | 76% | **85%** | 76.74% |
| VGG-16 3 | 85% | 72% | 71% | 62% | 69% | 74% | 77% | 80% | 74.68% |
| ResNet-50 1 | 78% | 62% | 61% | 52% | 68% | 74% | 71% | 83% | 69.15% |
| **Us** | 84% | **73%** | **77%** | 70% | **72%** | **77%** | **79%** | 84% | **77.09%** |

the data augmentation performed. In scenario 3 the train set is three times bigger than in scenario 0 however the training accuracy on the test set is the same. This proves that our intuition on not considering the cropping operation makes sense in this context since it can impact the models performance, as we said it would. For what concerns scenario 1 and 2 we can see that if the crop operation is set to maximum 10% of the image, the model performance is not really impacted. The overall accuracy in scenario 2 is 0,2 points higher than scenario 1, but in scenario 2 the train set has 50% more images with respect to scenario 1.

| shot class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Long Shot (LS) | 89% | 79% | 84% | 407 |
| Medium Shot (MS) | 70% | 77% | 73% | 381 |
| Full Figure (FF) | 79% | 76% | 77% | 324 |
| American Shot (AS) | 71% | 69% | 70% | 282 |
| Half Figure (HF) | 69% | 74% | 72% | 393 |
| Half Torso (HT) | 74% | 80% | 77% | 503 |
| Close Up (CU) | 79% | 79% | 79% | 519 |
| Extreme Close Up (ECU) | 90% | 79% | 84% | 355 |
| accuracy | | | 77% | 3164 |
| macro avg | 76% | 76% | 76% | 3164 |
| weighted avg | 77% | 77% | 76% | 3164 |

Table 2.11: precision, recall and f-1 score of the scenario 1

### Case 2: Stacking learning variations

Here we show how different stacking learning implementation impact the performance of our methodology. Various studies have demonstrated the advantages of using stacking ensembles over other ensemble methods[92, 54]. In light of this, instead of testing our approach against bagging or majority voting, we test the performance of our approach in comparison with different implementations of the stacking learning strategy. The results of these different stacking techniques are summarized in Table 2.12.

The first different implementation involves using a Random Forest as the fourth classifier, instead of a Multilayer Perceptron (MLP). The results from this alteration show the performance of a Random Forest in this context is inferior to the MLP as meta-learner.

The second implementation is slightly different. Instead of training three separate VGG-16 models along with a fourth classifier, we combined the four models into a single network, which we named the Cerberus model. Similar to our original methodology, the meta-learner in this network is a Multilayer Perceptron. However, unlike the original approach that employs four distinct models, the Cerberus model

consists of a unified network with three "heads," each corresponding to a different dataset.

The comparison results indicate that our approach outperforms the variations achieved with different stacking learning modifications. This highlights the effectiveness and robustness of our methodology in leveraging stacking ensembles for improved performance.

Table 2.12: Stacking learning variations.

| | | **LS** | **MS** | **FF** | **AS** | **HF** | **HT** | **CU** | **ECU** |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest as meta-learner | 65% | 70% | 76% | **71%** | **72%** | 76% | **79%** | 83% | 72.56% |
| **Us** :MLP as meta-learner | **84%** | **73%** | **77%** | 70% | **72%** | **77%** | **79%** | **84%** | **77.09%** |
| Cerberus | 83% | 71% | 68% | 66% | 63% | 75% | 73% | 83% | 73.32% |

### Results: updated

When we were testing these methodologies, Vision Transformers were not developed yet. They are deep learning models that apply the principles of Transformers, from the natural language processing (NLP) field, to the domain of computer vision. A Vision Transformers (ViT) divides an image into a sequence of fixed-size patches and treats each patch as a token (similar to words in a sentence). These tokens are then processed by a Transformer encoder. We tested one of these models on our dataset. The results are shown in 2.13 and 2.14 .

| shot class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Long Shot (LS) | 91% | 95% | 93% | 394 |
| Medium Shot (MS) | 82% | 74% | 78% | 381 |
| Full Figure (FF) | 78% | 80% | 79% | 324 |
| American Shot (AS) | 72% | 78% | 75% | 281 |
| Half Figure (HF) | 75% | 74% | 74% | 394 |
| Half Torso (HT) | 79% | 83% | 81% | 502 |
| Close Up (CU) | 85% | 76% | 80% | 519 |
| Extreme Close Up (ECU) | 82% | 88% | 85% | 314 |
| accuracy | | | 81% | 3150 |
| macro avg | 81% | 81% | 81% | 3150 |
| weighted avg | 81% | 81% | 81% | 3150 |

Table 2.13: precision, recall and f-1 score of the vit model

| | LS | MS | FF | AS | HF | HT | CU | ECU |
|---|---|---|---|---|---|---|---|---|
| Long Shot (LS) | 373 | 20 | 0 | 0 | 0 | 1 | 0 | 0 |
| Medium Shot (MS) | 30 | 281 | 52 | 7 | 7 | 3 | 1 | 0 |
| Full Figure (FF) | 0 | 27 | 259 | 30 | 5 | 1 | 0 | 0 |
| American Shot (AS) | 1 | 6 | 17 | 219 | 37 | 1 | 0 | 0 |
| Half Figure (HF) | 0 | 5 | 2 | 46 | 291 | 50 | 0 | 0 |
| Half Torso (HT) | 2 | 3 | 0 | 1 | 47 | 419 | 30 | 0 |
| Close Up (CU) | 1 | 0 | 0 | 0 | 1 | 52 | 394 | 71 |
| Extreme Close Up (ECU) | 3 | 0 | 0 | 0 | 0 | 1 | 37 | 214 |

Table 2.14: Confusion Matrix of a Vit trained on the Dataset. Reading it orizontally we can see how many of the samples were correctly classified and how many where mistaken for other shots

The boost in performance, although noticeable, is not incredible, however if we look at the confusion matrix we can see that the majority of the mistakes are made between similar classes. If we ignore those mistakes the total accuracy rises up to 99%.

### 2.4.4 Discussion

These works show that it is possible to address shot size classification with satisfying performance. Especially with vision transformers the task can be considered solved. While in some contexts knowing if we are dealing with a Close Up or a medium Close Up can be relevant in our context is an acceptable approximation. The two reasons why these errors can be ignored are the following. On one hand since there are not too many in the context of editing pattern extraction they do not add excessive noise to the extracted sequences. On the other hand, some of these images are hard to classify even for people. For instance take a look at figure 2.9.

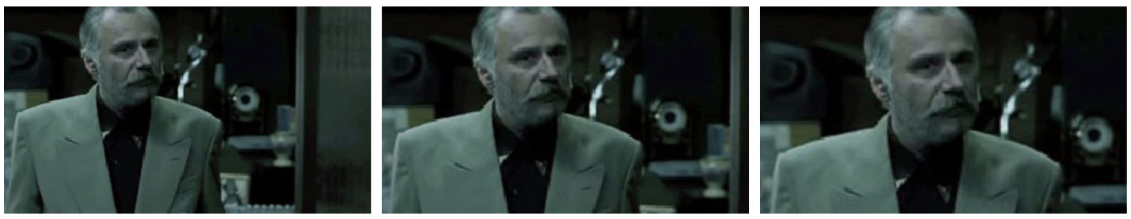


Figure 2.9: On the left medium close up, on the right close up, in the center the shot size is ambiguous

Here the first image on the left is the original image and it is a medium close up, or half torso. The other two images are the same image cropped and resized. While the image on the right is a close up the image in the center is hard to classify as it is in between the two classes. The ability to classify images into shot sizes, while it can have already useful implementations, allow us to study the next topic of our research; the editing patterns. This is because while some datasets already contain the shot size attributes, like AVE and Cinescale, some others, like CMD, do not. having a model able to label movie frames into shot sizes can be used to extract editing patterns from videos. With editing patterns we refer to the sequences of shots used to create a movie scene. These sequences of shots can be characterized with more or less features, but the main feature that they all have, directly or indirectly, is the shot size.

## 2.5 Chapter Conclusions

This research branch achieved good results. The preliminary analysis led us to adopt the fine-tuned VGG-16 approach. After expanding the dataset we ended up with 10 545 images divided into 8 classes. This dataset helped us to define a methodology with state of the art performance in classifying images into shot sizes. However the main disadvantage of the ensemble approach is that it requires 4 models in order to make predictions. This issue was overcomed later thanks to the rise of Vision Transformers. The fine-tuned ViT model achieved better performance compared to the previous approach. Additionally with the sensible assumption that error between similar classes can be ignored the overall accuracy rises up to 99%.

These achievements allow us to label with good accuracy unlabeled images into shot sizes, a feature that we will exploit later. The next step in the main research context is to analyze what type of correlation there is between the sequences of shots and what is shown in the final video. Instead of labeling new shots dataset we have exploited an already existing dataset that had been released while we were conducting our research on shot size classification. Hence we have exploited these new dataset to conduct our analysis on sequences of shots.

# Chapter 3

# Video Editing Pattern Analysis

## 3.1 Introduction

Videos and machine and deep learning approaches have been the subjects of several studies in the past. These studies range from object detection and tracking to action recognition, from video understanding and summarization to autonomous driving applications. The editing patterns and structures, however, have been studied much less. One of the reasons why could be related to their intrinsic hidden nature. During the video editing process, individual shots are joined together to create a scene. How these shots are joined together and the shots' features themselves are all elements that impact the viewer's perception. Specifically, editing helps define the narrative and mood of a film, along with other elements such as the setting, the soundtrack, VFX, and so on. While, for instance, the soundtrack can be heard, and the actor can be seen, the audience is able to notice the editing only when it is poorly done (unless they are actively looking at it). According to some movie critics, like Walter Murch, video editing is like telling a story. You can have a great plot with interesting characters, but if the narrator focuses on the wrong parts or tells it with the wrong rhythm, it has no impact. When a video is poorly edited, the viewer is less engaged in the story or won't get the story at all. This is because editing, among other elements, determines the structure of a scene, which consequently affects the mood and the narrative style of the movie.

There are different processes involved in video editing, however here we refer to the action of joining the shots together, rather than editing the shot themselves.

In other words, we focus on the process of concatenating clips that have limited meaning on their own to create a meaningful video. New algorithms are now able to generate single video clips directly from textual prompts; however, in terms of editing structures, so far, some applications have emerged with good results but with limited and standard templates, like interviews. By gaining more knowledge on video editing patterns, more powerful video editing assist tools could be developed. For instance, text-to-video models could also use basic editing patterns. Instead of

creating a video made of a single clip, it would be possible to create a video with multiple clips and coherent cuts.

Editing Pattern analysis consists of analyzing the sequences of shots that constitute the structure of any video. The shots defining the sequences can be more or less refined, taking into account a more refined shot size classification or multiple features describing the shots.

We defined a first methodology while analyzing the Cinescale dataset. This methodology groups sequences of shots, characterized by only the shot size, based on their structural similarities. We further refined our methodology while analyzing the Anatomy of Video Editing dataset. In this dataset, while the shot size classes are only 5, the shots are also characterized by other features. Hence, it was necessary to adapt the original methodology.

Figure 3.1: People interacting with objects

Figure 3.2: Shot reverse shot

In both contexts, we represent movie scenes as sequences of symbols that encode one or more features. Given the unique nature of every scene, we focus our analysis on scene segments rather than entire scenes. Our intuition is based on the observation that similar situations can occur in different scenes at various moments, yet they will rely on similar shots. For example, a dialogue can take place in different

scenes and at different points within a scene, but it is likely to be depicted using close-ups and half-torso shots. The mood of the moment influences the order and frequency of these shots. If the context changes, different shots, and editing paces are employed. This is not surprising, as the choice of shots is closely related to the director's intent.

To better illustrate this concept, consider the example in Figure 3.1. The top segment is from the movie "Thoroughbreds" directed by Cory Finley, and the second segment is from "The Birdcage" directed by Mike Nichols. In both videos, we can see people interacting with objects. Despite the different reasons behind the directors' choices to depict these interactions, the portrayal is strikingly similar. This is just one method to show people interacting with objects, and there are many others.

A more familiar example of editing patterns is the "shot-reverse shot" technique, shown in Figure 3.2. The top segment is from "The Great Lebowski" directed by Joel and Ethan Coen, while the bottom segment is from "Pulp Fiction" directed by Quentin Tarantino. Again, the same set of shots is used to describe a similar context — two people talking to each other — but the reasons for their conversation and the content of their dialogue differ. These examples should give an idea of what we mean with the correlation between sequences of shots and what is shown in the video.

The next section will show studies conducted in the same field. In Section 3.3, we show the first draft of the methodology and its performance on the Cinescale dataset, while in Section 3.4, we present the refined methodology used to identify and group similar editing structures in the AVE dataset.

## 3.2 Related Work

In recent years, videos and films have been analyzed using different techniques and for different purposes. Most of these studies are related to computer vision and image classification, but other branches of research focus on other aspects, like [14]. Here, the authors use natural language processing techniques on the IMDB dataset[14] to perform movie sentiment analysis.

One common task in this branch of computer vision, for instance, is the classification of movie frames, like[7] [67] [84] [33], as seen in the previous chapter [82]. Others instead focus on different movie and video features. In [39], the authors use machine learning algorithms to detect inappropriate language and visual content and label the movie accordingly into one of the following categories: "Universal," "Universal Adult," or "Adult". In [95], the authors perform video retrieval from large video collections. Their algorithm uses dynamic programming to measure the similarity between video sequences in terms of temporal and visual features.

The first to introduce the concept of editing pattern were the authors of [50] in a paper that dates back to 2002. This paper proposes methods to extract editing

rules from video streams using data mining, allowing for the creation of new videos with similar quality to the original by applying these extracted rules. In other words they identify some preliminary editing patterns. They also emphasized the central role of editing and its pace, like Murch, in videos. The task that they address is the next shot type prediction with three classes. A slightly younger study that focuses on video editing but from a different perspective is [53]. It investigates the evolution of cuts and transitions among shots in films over time. In addition, to prove that the rate of cuts over 4 years, sampled at the regular rate (1945, 1965, 1985, 2005), across three genres (action, comedy, drama) has increased, the authors present another discovery. Their results show that certain structures repropose themselves among groups of shots in sequences of the same length. This finding hints at the fact that there is more structure behind the placement of cuts.

We mention again (see Chapter 2.2) two studies that focus on the role of shots and sequences of shots to impact the viewer perception [8]. In [8], the authors analyze how the shot size, or shot scale in their case, distribution, and rotation of Close, Medium, and Long Shots influence the viewers' perceptions of film, particularly in violent scenes.

In [69] the authors use CNNs for movie trailer genre classification, introducing a new dataset of over 3,500 trailers with known genres and a novel classification method called CNN-MoTion. Also in [94] the authors address movie genre classification, however they rely on a different approach. They decompose movie trailers into keyframes using shot boundary analysis and then extract features from them to perform unsupervised shot categorization. Then, they use the features with a bag of words extracted from the trailers to label them into four genres: action, comedy, drama, or horror. In [78], instead, the authors focus on single shots and investigate the relationship between the movie director and the shots that he chooses in the film.

As the number of studies on video editing and machine and deep learning approaches kept increasing, more complicated tasks became feasible. For instance, new tools to automate certain tasks in the video editing process have been proposed. In [10], the authors introduce a suite of tools to assist editors in placing cuts and creating transitions in interviews. In [61], the focus is not on classifying frames or videos into recording types but on video segmentation, i.e., splitting a video into individual clips.

Another interesting study dealing with automatic video editing is presented in [91]. Here, they teach a model of how to edit a video of group meetings and conference events in a multi-camera environment.

In [20], the authors compare the performance of movie style analysis based on two different types of features. They show that high-level features (e.g., character segmentation, pose estimation, camera motion type, and pose) are better for this task compared to low-level features (e.g., color histograms and average shot lengths). In [56], the authors exploit 10,000 already edited videos to teach a model

when to place a cut between two consecutive shots. The following year, they published [57] in which the authors perform recognition of transition cuts on a cut dataset they introduce. In [90], the authors focus instead on long video understanding, subdivided into several tasks. They analyze preprocessed videos taken from the MovieClip dataset [13] and test themselves in the different tasks from the director and year prediction to scene location classification (and many others). They also release the Long Video Understanfiding dataset, built upon the MovieClip dataset.

Also in [17] the authors use movie metadata to group together similar movies and then use a contrastive approach to identify similar scenes in them. They tested their approach on the LVU dataset, MovieNet [37] and MovieCL30K, a dataset they introduced for movie metadata classification. Additionally, they introduce the Mature Content Dataset for video moderation.

The amount and detail of new movie datasets released in recent years, in addition to those already mentioned, also open the door to new types of analysis. For instance, there is the The Condensed Movie dataset [5], which contains the main scenes from different movies with metadata. Unfortunately, the shot sizes are not taken into account. We will further investigate this dataset in Chapter 4. Cinescale [65] instead is a dataset with the shot size feature. It contains 120 movie frames sampled at 1 second per frame and labeled into 8 shot size classes. The movies come from six different directors. An even more refined dataset is the Anatomy of Video Editing Dataset (AVE)[3]. In this dataset, the shots are characterized not only by the shot size but also by other multiple features. Overall, it contains scenes from 5 591 movies. We have chosen these last two datasets to perform our study: Cinescale and AVE. While previous studies focus on different aspects and interesting applications of video editing, only a few of them take video editing patterns into account. Furthermore, while some suggested more complicated structures and patterns, none had the data availability that arrived recently since those studies were published prior to the release of large movie datasets.

## 3.3 Movielens

In this early study, we proposed a novel data-driven methodology called Movielens. It is able to identify editing patterns and group editing patterns based on their similarities in terms of structures. In this early work, we focused only on editing sequences represented by shot size classes. Our goal was a preliminary analysis of the type of correlation between sequences of shots and the corresponding video. To this aim, we have used the Cinescale dataset [65]. However, the sequences obtained were not labeled. Hence, we have developed a methodology based on a joint approach that relies on K-Means and the Levenshtein distance. As a result, we were able to group similar sequences of shots based on their elements and structures.

Then, we trained and tested a classifier using the original sequences and the newly obtained labels to validate the previous approach. Finally, we present a preliminary characterization of the 23 887 labeled sequences extracted from 120 different movies.

### 3.3.1 Cinescale Dataset

To address this task, we used the Cinescale dataset[65], a dataset that contains 120 movies from 6 different directors. To be more precise, only its labels were used. Originally, the dataset was used to classify shot size. For each movie, a frame was sampled and labeled every second. The labels assigned to each frame follow a similar yet different classification system compared to the one that we have used in previous studies, shown in 3.3.


Figure 3.3: Eight shots used

The shot classes in this dataset are the following:

- Class 0) Foreground Shot (FS): a shot that contains elements of different shot classes. For instance, camera movements fall under this category (128 335 frames)
- Class 1) Extreme Close Up (ECU): a shot that focuses on details, such as the eyes of the subject, what the character is holding, and so on (3 367 frames)
- Class 2) Close Up (CU): a shot focused on the subject's face; it shows the actor from the shoulder up. It can also be used to focus the viewer's attention on some detail like objects or hands (83 682 frames)
- Class 3) Medium Close Up (MCU): the subject figure is shown from the upper half of its torso (252 639 frames)
- Class 4) Medium Shot (MS): only the upper half of a human subject is shown (78 053 frames)

- Class 5) Medium Long Shot (MLS): the human figure is shown from the knee up (89 450 frames)
- Class 6) Long Shot (LS): the human figure occupies the totality of the frame height or 2/3 (49 788 frames)
- Class 7) Extreme Long Shot (ELS): the human figure is absent or occupies less than a third of the screen height (7 118 frames).

Since we are interested in analyzing segments of the video rather than the whole scene, we have divided the movie scene into 30-second long segments. This approach also allowed us to simplify the problem on one hand while keeping a good granularity of the results. If we were to choose longer segments, we would have more complicated patterns, and we would lose local information, which is what we are focused on the most at the moment. On the other hand, if the time interval is too short, the identified patterns are not too interesting.

In addition to the aforementioned shot size classes, Cinescale has two additional classes: one for opening and closing titles and another for undefined frames. After removing these shots from the original scenes, the resulting 30-second-long editing segments are 23,887.

In the dataset, the shot sizes are represented as numbers. However, the class must be treated as a categorical value rather than a number. This is because although there is a sense of scale from close-up to long shot, representing it with numbers doesn't make much sense. For example, if a close-up is indicated with label 2 and an extreme close-up with label 1, we would have that an extreme close-up is half a close-up, which doesn't make sense. Additionally, the class foreground shot defies any scale categorization. Hence we treat them as symbols rather than values.

### 3.3.2 Methodology

Movielens, our methodology, has two main analytic building blocks, shown in 3.4: the *Label Estimation Phase* and the *Editing Patterns Analysis*. The goal of the first block is to label each short movie sequence. To achieve this, we begin by analyzing each sequence using a distance metric. This analysis measures the similarity of each sequence to a set of fixed reference sequences. The resulting distances are then used as coordinates that represent each sequence as a point in a multidimensional space. These points are then grouped using a clustering algorithm. The resulting clusters identify the different labels.

The second block's objective is to evaluate the Label Estimation Phase results by analyzing the performance of a classifier trained on the sequences and the newly obtained labels. After training the model, we test it on the remaining dataset portion. We have also involved domain experts to evaluate the quality of the classes

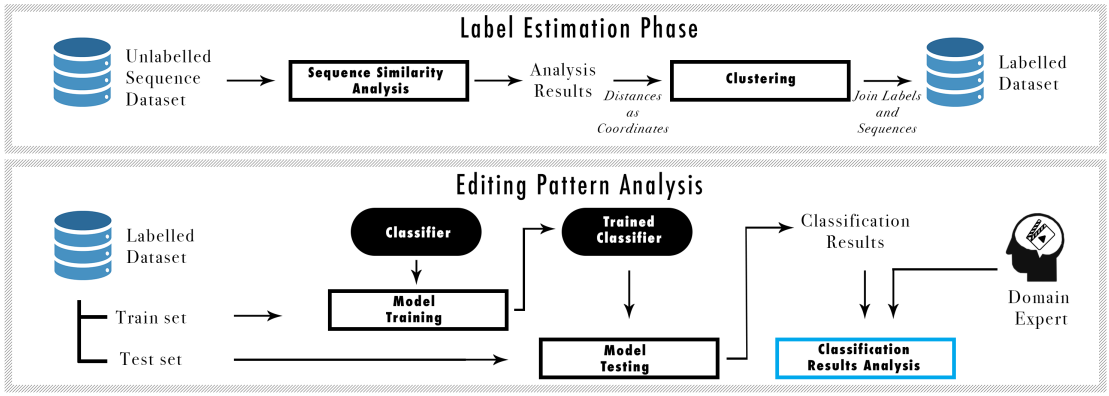



Figure 3.4: The two Movielens blocks

identified. This analysis is performed manually on a subset of short movie sequences from each class, along with the labels provided by the classifier. Specifically, the first block labels each sequence of shots based on similarity, while the second block validates the identified labels.

### Label Estimation Phase

This analytical component conducts a similarity analysis and models its output using a clustering algorithm. Specifically, we rely on the Levenshtein distance [31] to measure the similarities and differences between sequences and employ the K-Means algorithm [48] to indirectly model editing patterns. The Levenshtein usually counts the number of substitutions, additions or subtractions required to convert one sequence into another, however in this instance, since all sequences are of equal length, it only computes the number of substitutions necessary. It is important to keep in mind that changing a 2 to a 3 or a 3 to a 7 for the Levenshtein always counts as one substitution. Due to the length and the different structures the editing sequences can have, counting the number of substitutions directly among the sequences is not very effective. Thus, first, we have created 8 artificial sequences, one for each of the possible symbols contained in the sequences. These reference sequences are 30 frames long, with each frame belonging to the same shot size. Then, for each sequence, we measure the Levenshtein distance with respect to the reference sequences, and we store those distances. Finally, we use these distances as high 8-dimensional point coordinates, and we use a clustering algorithm. In this way, the clusters in which the sequences are grouped identify the editing structure labels.

Among the various clustering algorithms available, we chose to use the K-Means algorithm. This decision was based on the algorithm's ability to converge quickly

while delivering satisfactory results [1]. The only downside of this clustering algorithm is that it needs to know the number of clusters a priori. To determine a reliable value for this parameter, we implemented two methods: the elbow graph [38] and Ward's method [52]. The elbow graph method is a technique used to determine the optimal number of clusters in a dataset for K-Means clustering, while Ward's method is a hierarchical clustering approach that aims to minimize the total within-cluster variance. These strategies aim to identify the optimal number of clusters by minimizing the within-cluster variance, thereby ensuring a more accurate clustering outcome.

At the end of this phase we have the sequences grouped in labels based on their similar structure, however we do know yet if identified labels are actually useful to represent editing sequences. Additionally, we do not know precisely what these labels represent. To address these issues we have developed the second block of this methodology.

#### Editing Pattern Analysys

This second phase is intended to validate and characterize the results of the previous phase. To validate the identified labels and see if they are fit to represent the editing sequences, we have split them into two sets, one to train a sequence classifier and one to test its performance, using different metrics, in labeling new sequences. After training and validating the classifier, we can move on to analyze and characterize its classification results. In this last analysis, we, as domain experts, analyze and characterize the editing patterns that emerge from the different groups. To verify the consistency of these sequences, we have selected and analyzed manually the 5% of samples coming from each group identified by a separate label. The results and editing patterns are characterized in the results section.

### 3.3.3 Experimental results

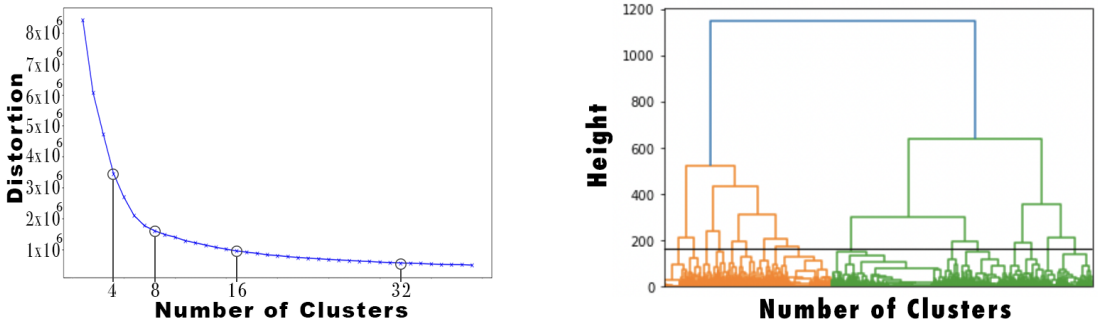
The following is a description of the experiments we conducted to evaluate the quality of our methodology and its results. At first, we focus on the quality of the results of the K-Means algorithm, and then we move to the classifier performance. Finally, we will analyze our results in terms of identified sequences.

Thanks to Ward's method and the Elbow graph, we were able to identify different configurations with various levels of granularity in the identified patterns. In Figure 3.5, we show the obtained results with these two techniques.

After selecting a different number of clusters to study more scenarios, we have to train a classifier. We trained and tested two models, a Multilayer Perceptron (MLP) classifier by adapting the model for sequence classification available at the GitHub repository [88], and a Long Short Term Memory (LSTM) classifier. The MLP classifier was the first model used to evaluate the sequence labels. Hence, an

editing analysis was run on the results. Later on, we trained and also tested the LSTM classifier.

In table 3.1, we show the performance of the MLP classifier with 4, 8, 16, and 32 classes in terms of accuracy, f1-score macro average, and f1-score weighted average, while in 3.3, we show the results achieved with the LSTM classifier.




(a) Elbow graph

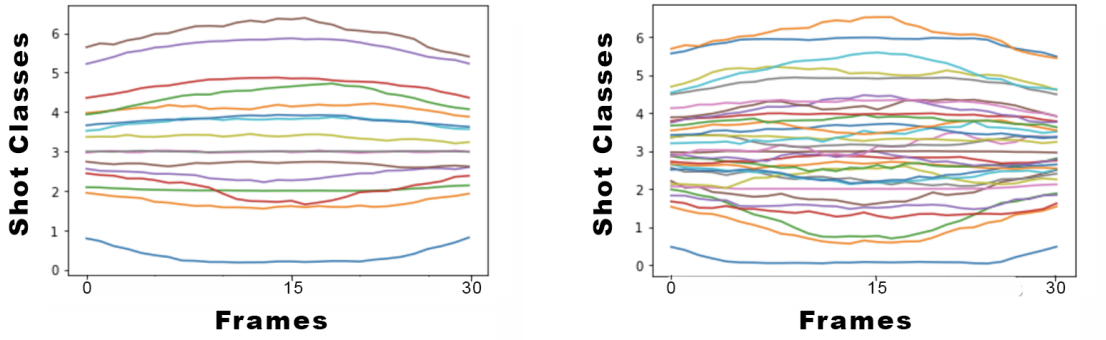
(b) Ward's dendrogram

Figure 3.5: Results of elbow graph and wards'method.

| number of classes | overall accuracy | macro average | weighted average |
|---|---|---|---|
| 4 | 93% | 93% | 93% |
| 8 | 88% | 88% | 88% |
| 16 | 81% | 79% | 81% |
| 32 | 77% | 72% | 77% |

Table 3.1: Performance of the MLP classifier with a different amount of classes considered.

In general, with no surprises, as the number of classes increases, the overall accuracy decreases. To conduct our analysis of the identified patterns, we have chosen the 16 class scenario. The scenario with 4 classes is not really interesting due to a low number of classes. With 8 classes still, it would not be a study case, since in this case, we would only see how these sequences are distant from the artificial ones that we have used to compute the Levenshtein distance. In the 32 case scenario the loss in terms of accuracy is sensible compared to the loss in accuracy of the previous scenarios. Additionally, this case presents a very complicated scenario to characterize. Hence, we have chosen the 16-class scenario, in which the classifier achieves 81% and offers a more interpretable scenario. Figures 3.6a and 3.6b shows the average pattern per class, i.e., centroid identified in these last two scenarios. By analyzing the patterns in Figure 3.6a we can see that centroids modeling editing pattern groups are well-separated with respect to the ones in Figure




(a) Patterns with 16 classes

(b) Patterns with 32 classes

Figure 3.6: Average of the values contained in the sequences to represent them.

3.6b. Thus, fine-grained partitions obtained with 32 classes are too detailed and present overlapped centroids (i.e., some types of frames in the sequences are very similar). Table 3.2 shows the precision, recall, and f-1 score of the average model for the MLP classifier.

| Labels | Semantic Label | Precision | Recall | f1-Score | Support |
|---|---|---|---|---|---|
| 0 | CER | 66% | 61% | 63% | 130 |
| 1 | CER | 73% | 79% | 76% | 109 |
| 2 | N | **99%** | **97%** | **98%** | 177 |
| 3 | CER | 76% | 72% | 74% | 101 |
| 4 | CER | 88% | 86% | 87% | 140 |
| 5 | ED | 96% | 64% | 77% | 77 |
| 6 | CCI | 85% | 82% | 83% | 197 |
| 7 | CCI | **97%** | **95%** | **96%** | 369 |
| 8 | N | 67% | 61% | 64% | 127 |
| 9 | N | 69% | 83% | 75% | 237 |
| 10 | CCI | **99%** | **95%** | **97%** | 155 |
| 11 | ED | 79% | 71% | 75% | 21 |
| 12 | ED | 87% | 92% | 89% | 118 |
| 13 | N | 56% | 77% | 65% | 119 |
| 14 | N | 67% | 57% | 62% | 136 |
| 15 | N | 77% | 75% | 76% | 175 |
| accuracy | | | | 81% | 2388 |
| macro avg | | 80% | 78% | 79% | 2388 |
| weighted avg | | 82% | 81% | 81% | 2388 |

Table 3.2: Precision, recall, and f1-score of the MLP classifier with 16 classes.

### Updated performance with LSTM

The original methodology used a MLP classifier to evaluate the labels and sequences, however later on this model was substituted with an LSTM classifier. An LSTM classifier is a type of neural network specifically designed to process and classify sequential data. LSTMs are a special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies, which makes them particularly effective for tasks involving sequences, such as time series prediction, language modeling, and speech recognition. This model, with no surprises, outperforms the MLP classifier.

The analysis of the MLP results is done in the next paragraph. However, since we also run these experiments, we show the performance of the LSTM classifier in the 32-case scenario to show its incredible performance, shown in 3.4.

| number of classes | overall accuracy | macro average | weighted average |
|---|---|---|---|
| 4 | 98% | 98% | 98% |
| 8 | 97% | 97% | 97% |
| 16 | 96% | 96% | 96% |
| 32 | 93% | 93% | 93% |

Table 3.3: Performance of the MLP classifier with a different amount of classes considered.

| Labels | Precision | Recall | f1-Score | Support |
|---|---|---|---|---|
| 0 | 95% | 93% | 94% | 42 |
| 1 | 91% | 83% | 87% | 60 |
| 2 | 100% | 100% | 100% | 47 |
| 3 | 91% | 91% | 91% | 47 |
| 4 | 95% | 98% | 96% | 169 |
| 5 | 98% | 100% | 99% | 107 |
| 6 | 100% | 100% | 100% | 40 |
| 7 | 90% | 97% | 93% | 65 |
| 8 | 88% | 78% | 73% | 77 |
| 9 | 82% | 81% | 81% | 151 |
| 10 | 100% | 100% | 100% | 28 |
| 11 | 92% | 97% | 94% | 59 |
| 12 | 88% | 88% | 88% | 143 |
| 13 | 91% | 97% | 94% | 65 |
| 14 | 100% | 84% | 91% | 43 |
| 15 | 99% | 91% | 94% | 75 |
| 16 | 99% | 100% | 99% | 87 |
| 17 | 93% | 98% | 96% | 56 |
| 18 | 93% | 94% | 94% | 54 |
| 19 | 78% | 94% | 86% | 66 |
| 20 | 100% | 100% | 100% | 243 |
| 21 | 85% | 100% | 92% | 41 |
| 22 | 78% | 70% | 80% | 20 |
| 23 | 100% | 100% | 100% | 4 |
| 24 | 85% | 89% | 87% | 63 |
| 25 | 95% | 94% | 95% | 87 |
| 26 | 91% | 77% | 84% | 83 |
| 27 | 94% | 85% | 89% | 53 |
| 28 | 94% | 96% | 95% | 118 |
| 29 | 93% | 93% | 93% | 61 |
| 30 | 96% | 95% | 95% | 55 |
| 31 | 94% | 96% | 95% | 79 |
| accuracy | | | 93% | 2388 |
| macro avg | 93% | 92% | 93% | 2388 |
| weighted avg | 93% | 93% | 93% | 2388 |

Table 3.4: Precision, recall and f1-score of the LSTM classifier with 32 classes

In the next paragraphs, we will describe the editing patterns that characterize the different groups. The identified editing labels can be grouped into three macro categories: (1) *Character-Environment Relationship (CER)*, (2) *Environment Descriptions (ED)*, and (3) *Character-Character Interaction (CCI)*

### Character-Environment Relationship

This first semantic group focuses on characters interacting with the surrounding environment in different ways. The classes in this group are 0, 1, 3, and 4, as shown in Table 3.2. Class 0, on which the MLP achieved an f1-score of 63%, the second lowest score of all classes, is characterized by the heavy presence of medium close-ups and medium-long shots. This class exhibits mainly two types of editing patterns or a mixture of both. The first type of sequence usually represents dialogues where the emphasis is not on characters interacting with each other but rather its split between the characters and the environment in which the scene is set. Also, the second set of sequences focuses on both characters and the environment, but the corresponding videos present a different aspect. In these segments, what is shown is a character moving through the scene environment from the point of view of the camera that follows them. A third smaller group of sequences that shows mixed features of the previous groups is also present. In these sequences, we can see characters talking, and then we get to another location through narrative expedients, like flashbacks, that show a different context. Also class 1 (f1-score=76%) represents segments in which the character interacts with the environment, however the results are less noisy. Also, the shots that characterize these sequences are mainly close-ups mixed with wider shots. In this class, one pattern emerges, although there are some outliers. The analyzed clips, for a large part, represented characters reacting to changes in the scene environment. The wider shots are used to present the environment and show what is happening. Instead of showing the facial reactions of the characters, there are the close-ups. This set of shots can be used to show two types of character reactions. The difference lies in what caused the change, whether it is caused by the character or the environment itself. Class 3 (f1-score=74%) contains sequences with similar shots compared to Class 1 but with a different distribution. While we are always in the context of character environment relationships in these shots, there is also a focus on specific objects that compose the scene, which changes the narrative focus and mood. The last class that belongs to the macro category Character Environment is class 4 (f1-score=87%). This class is characterized by a different pattern from what we have seen so far. In these segments, which usually contain a unique shot, like a camera movement or a fixed shot, the character and the environment are presented together. Usually, these segments are used to introduce something. The characters are usually shown in internal settings while doing something in line with the environment they are presented within. An example can be seen in Figure 3.7.

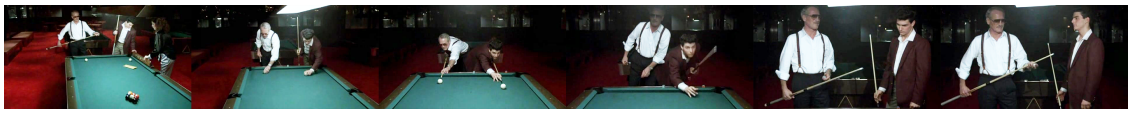


Figure 3.7: Character and environment presented together

### Environment Description

This group focuses only on the scene environment and pays little attention to the characters. The classes that belong to this group are 5, 11, and 12. What differentiates one class from another is the shot sizes used. These groups focus on the environment description: what changes mainly is the shot scale used. Class 5 (f1-score=77%) is characterized by long shots and medium long shots focused on showing the scene environment. Sometimes characters can appear on the screen but the attention to them is minimal, they are presented like pieces of the environment. Class 12 (f1-score=89%) instead has mainly segments characterized by medium-long shots. These segments are mainly characterized by medium-long shots used to represent sequence shots. A sequence shot is a filmmaking technique where an entire scene or significant part of a scene is captured in a single continuous shot without any cuts. These sequence shots either follow a subject that moves around the environment at a fixed distance or are stationary. Finally, class 11 (f1-score=75%) contains mainly segments made of long shots or extreme long shots. In this class, attention is completely focused on the scene environment. An example can be seen in Figure 3.8. It has been taken from the movie "Bande à Part" by Jean-Luc Godard.

Figure 3.8: Example of a sequence taken from class 11 with long shots and extreme long shots.

### Character-Character Interaction

This main category focuses on character-to-character interactions in the form of dialogues. Labels 6,7, 10, and 15 belong to this macro category. Class 6 (F1-score = 83%) contains many foreground and narrow shots. While still involving dialogues here, there is more emphasis not only on the characters' facial emotional reactions through the use of close-ups but also on their physical reactions, represented with what in Cinescale is labeled as a foreground shot. Class 7 (F1-score = 96%) consists primarily of medium close-up sequences and contains the highest number of sequences, all of which represent characters talking. An example of this

type of sequence is shown in Figure 3.9. Class 10 (F1-score = 97%) represents a more specific pattern characterized by many close-ups. Typically, these sequences involve one-on-one interactions between two or three characters. In contrast, Class 15 (F1-score = 76%) includes dialogues among multiple characters, where usually the main character speaks while others listen or vice versa. These interactions are one-to-many, and as a result, this class features wider shots compared to Class 10.

Figure 3.9: Charachter character interaction

### Undefined Classes

Not every group of sequences shows such specific patterns. From classes 2-8-9-13-14 (see Table 3.2) no meaningful patterns emerge. Specifically, For classes 8 (f1-score=64%), classes 9 (f1-score=75%), 13 (f1-score=65%), and 14 (f1-score=62%), no meaningful pattern emerge, and as a consequence, we have called this group undefined classes. These classes share similar shots and moments. This class contains sequences characterized mainly by foreground shots. Usually, these types of shots are camera movements that switch between different shot sizes. However, there is no indication regarding the shot sizes included in this shot. Nonetheless, it is important to know if we are looking at a zoom, a pan, or something else. Hence a further characterization of this specific class is very hard, since there are no indication of the camera movements contained in the different shots. For this reason, class 2 has been included in the undefined classes.

### Misclassified Sequences

We also analyzed the most common mistakes made by the classifier. Specifically, here we show the most common five misclassification errors. The first couple of sequences that are usually mistaken are sequences belonging to class 5 that get classified as belonging to label 13. The reason is related to the fact that class 13 exploits shots similar to those of class 5. We have identified the classes that get mistaken more commonly for other classes. A similar discussion can be made for sequences of classes 0, 6, 8, and 14 that get classified as sequences coming from classes 9, 15, 13,9. It is important to notice that most of these sequences come from classes that we have called undefined and from classes on which the classifier has some difficulties and that share similar shots.

### 3.3.4 Discussion

Our first approach gave us some interesting results while highlighting some criticisms of this first draft of the methodology. On the one hand, we have seen that there is a correlation between the sequence of shots used to represent a certain scene and what is going to be represented. However, while in some cases we were able to better characterize the editing pattern characterizing the different classes, in some cases, we were not able to extract meaningful information. This shows that it is necessary to better articulate what is happening in the scene and that the shot size alone, while giving already a good idea of what is going to be shown, is not enough. Additionally, the way we represent the sequences loses track of other important attributes, like the editing pace. To overcome these issues, we have further refined our methodology while analyzing the AVE dataset. In this new implementation, we characterize the sequences of shots with multiple features.

## 3.4 LEMMS

In this work, the focus is still on 30-second-long sequences, but instead of being characterized by only the shot size, new features are also taken into account. The dataset that we have used as a starting point is the Anatomy of Video Editing dataset(AVE), which we transformed into what we call the Sequenced AVE. Conscious of the limitations of our first methodology, which used sequences of shots characterized by only the shot size, in this new study, we consider more features. The newly added features are the shot subject, the editing trend, and the editing pace. Since we focus on movie segments characterized by multiple features, we have called our semi-unsupervised methodology LEMMS, Label Estimation of Multifeature Movie segment movements. Like before, our methodology relies on the use of the Levenshtein distance, k-means algorithm [4] and the Levenshtein distance[31] and an LSTM classifier. However, this approach is more stratified than before. The model was trained on 50 classes and tested with a 10-fold stratified validation strategy, achieving an overall accuracy of 92.8% in labeling multi feature movie segments coming from 50 different classes.

### 3.4.1 Dataset: AVE to Sequenced AVE

The Anatomy of Video Editing dataset is a comprehensive collection of video clips curated to study various aspects of video editing. Overall, it contains 5,591 scenes from different movies. In addition to the actual video, this dataset contains additional metadata on each shot that composes those clips. There are several features characterizing each shot, and we had to choose some while discarding others to avoid an excessive increase in the complexity of the editing structures. In order to choose which attributes to keep, we have made the following considerations.

The shot features can be grouped into the following macro-categories: camera attributes(*shot size, shot angle, shot type, shot motion*), set attributes (*shot location, shot subject, number of people*), and additional attributes (*sound source, start time, end time*). Then, from each group, we have chosen one feature. For the camera attributes, we have chosen the shot size since it is more meaningful than the other attributes (shot angle, shot type, and shot motion). From the set attributes, we chose the shot subject since it is more meaningful than the number of people. Additionally, this attribute already includes the class location among its subjects, while the attribute location specifies only if we are in an exterior or interior scene. Finally, from the additional attributes, we have discarded the sound source and kept the start and time. In figure 3.10, we show the different preprocessing steps we use to transform AVE into the Segmented AVE dataset.

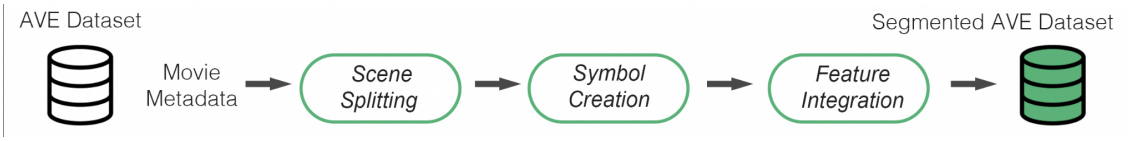



Figure 3.10: Data preprocessing steps

| Symbol-Size(abbr) | number of samples |
|---|---|
| 0 - other(O) | 1,977 |
| 1 - extreme close-up (ECU) | 405 |
| 2 - close-up (CU) | 26,692 |
| 3 - medium shot (MS) | 142,314 |
| 4 - wide shot (WS) | 22,852 |
| 5 - extreme wide shot (EWS) | 1,936 |

Table 3.5: Shot Sizes

### Spliting Sequences

The first step is to split the editing sequences in AVE made from the concatenation of shots representing the movie clips. Since we are analyzing 30-second-long segments to capture local patterns, to avoid losing the ending part of most scenes, we also resampled the last 30 seconds of every scene that has a duration different from a multiple of 30 seconds. By doing so we ended up with 24 363 movie scene segments.

| Symbol-Subject(abbr) | number of samples |
|---|---|
| 0 - other (ot) | 3 000 |
| 1 - face (f) | 5 671 |
| 2 - human (h) | 97 696 |
| 3 - location (l) | 69 959 |
| 4 - object (ob) | 19 855 |
| 5 - animal (a) | 8 995 |

Table 3.6: Shot Subjects

### Symbol Creation

In this step, we characterize the scene segments in terms of shot sizes and subjects. Specifically, we encode the textual labels of both attributes using categorical values as shown in 3.5 and 3.6. Although the shot sizes present an uneven distribution, we kept all of them since they already present a less fine-grained scenario compared to Cinescale. Instead, in the subject attribute, we merged some minor classes into the class *other*. The classes merged were *cartoon, limb, other, text.*

### Additional Features

In this last pre-processing step, we further characterize the scene segment with two additional features. The first one is the editing pace. We have defined three editing paces: slow, medium, and fast. The editing pace is defined by the number of shots present in the segment. This number can be inferred since we have the start time and end time of every shot. The different editing paces are characterized in Table 3.7. To represent this feature directly in the 30-second segment, we have encoded it in the following way. We added an additional attribute that can be either 1 or 0 to each element representing a shot at a time. If this value is 0 it means that the frame under analysis belongs to the same shot of the previous frame, 1 if it belongs to a new shot.

Finally, the last additional feature is the editing trend. Since we are representing the shots as categorical values, we lose some valuable information. As a matter of fact, the shot size roughly indicates how close the subject is to the camera's point of view. This information is important because moving closer to or further from a subject has a different impact on the viewers. Hence, the creation of the editing trend attribute and its classes is shown in Table 3.8. To represent this feature, we have also added an additional token to each element of the sequences. There are three token values: 0,1,2. When the shot size is the same as the next frame we the trend token is 0. If the token is 1 or 2 it means that the camera's next frame is closer or further respectively.

| Class | $n_{shot}$ in segment | number of samples |
|---|---|---|
| Fast | $n_{shot} > 10$ | 8,101 |
| Medium | $5 < n_{shot} < 11$ | 9,354 |
| Slow | $n_{shot} < 6$ | 6,908 |

Table 3.7: Editing paces: definition and cardinality.

| Class | trend token | number of samples |
|---|---|---|
| Stable | 0 | 8,410 |
| Mixed | 0, 1, 2 | 12,857 |
| Further | 2, 0 | 1,334 |
| Closer | 1, 0 | 1,762 |

Table 3.8: Trend classes: composition and cardinality.

As a result of all these pre-processing steps, the final segments are 30 seconds long. At each element representing a frame at time t we have the shot characterized by multiple features as follows:

$$Shot_{Size}, Shot_{Subject}, token_{SameShot}, token_{Trend}$$

The value of each separate attribute characterizes the frame in terms of shot size, shot subject, same shot token, and trend token.

### 3.4.2 LEMMS methodology

To characterize these new multi-feature segment representations of the 30-second long movie clips, we developed the LEMMS methodology. It is based on the methodology introduced in the previous section, but it has been updated to include this more complex scenario. The end goal is always the same: group and label sequences that have similarities in terms of editing patterns. The methodology overview is shown in Figure 3.11. In the coordinates extraction phase, we extract sequence coordinates using Levenshtein distance. We extract two sets of coordinates to better characterize the scenario. Then, we move to the multi-clustering phase. Conceptually similar to what was done before, this refined clustering phase allows us to identify more editing patterns with different cardinality. At the end of this phase, we have sequences grouped into labels that represent them in terms of shot sizes, shot subjects, editing pace, and editing trend. In the last phase, we train and test a classifier to validate the proposed labels.

| performance | trend and pace | no trend | no pace |
|---|---|---|---|
| Silhouette | 0.75% | 0.99% | 0.97% |
| Macro average | 88% | 76% | 84% |
| weighted average | 93 % | 95% | 94% |

Table 3.9: LSTM performance in different scenarios.

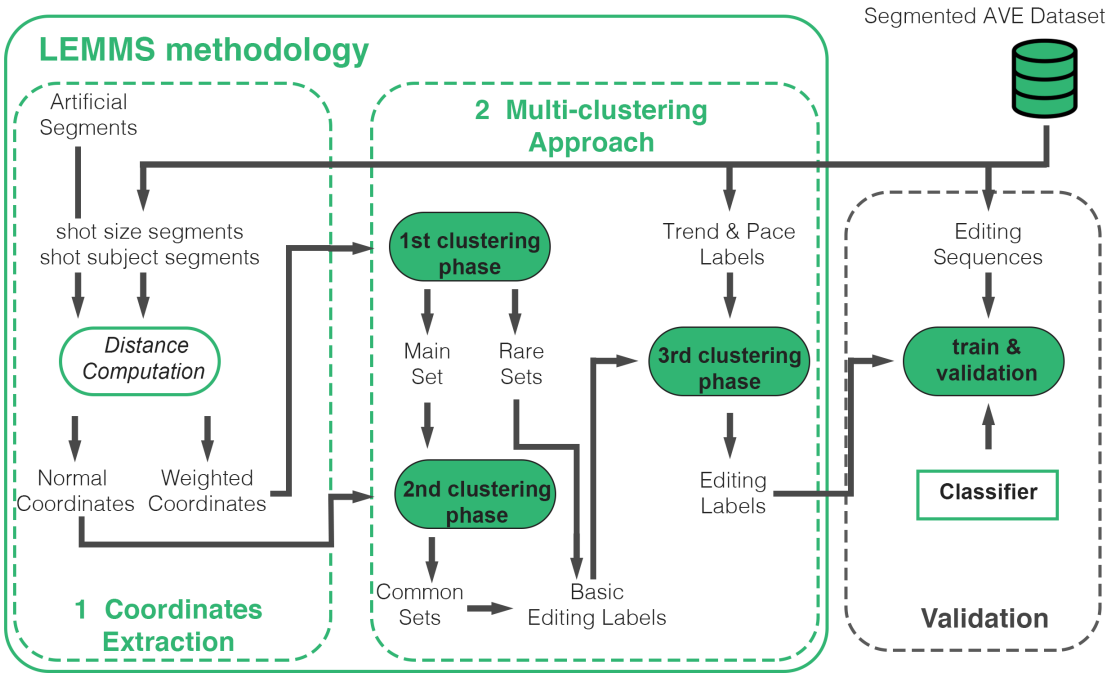



Figure 3.11: LEMMS methodology

### Coordinates Extraction

In this phase, we extract segment coordinates similar to what was done with our early methodology. However, instead of creating 8 artificial sequences, we generate only 6 artificial sequences since we have fewer shot size classes. After generating six artificial sequences with 30 elements, each set to the same symbol, one per sequence, we compute their Levenshtein Distance with respect to every segment in our dataset. Then, we store those distances, and we use them later on as multidimensional point coordinates. We repeat this process several times. The first time, we extract the coordinates from the sequence segments characterized only by the shot sizes, while the second time, the sequence segments are characterized only by the shot subjects. We then join these two sets of coordinates in the following way: **[Ds0, Ds1, Ds2, Ds3, Ds4, Ds5, Dsb0, Dsb1, Dsb2, Dsb3, Dsb4, Dsb6]**. The first 6 elements contain the coordinates that represent the shot sizes

used in the sequence, while the other 6 elements represent the subjects used. At the end of this process, we have the normal coordinates. To extract the weighted coordinates, we multiply the normal coordinates by a set of coefficients. We have as many coefficients as the dimensions that characterize our sequence points. Each coefficient is inversely proportional to the frequency of the specific class with respect to the other classes of its type. In this way, the weighted coordinates give more results to editing patterns that are consistent but less evident because they are characterized by less frequent shots.

### Multi-step clustering

In this phase, using the sets of coordinates obtained from the previous step, we deploy multiple clustering steps to characterize the sequences in different ways. The algorithm used at every clustering step is the KMeans++ [4]. To identify an ideal number of clusters, we used the Silhouette index score [68], which measures the cohesion and separation of the identified clusters.

The first clustering phase is performed on the weighted coordinates. In this way, we separate the main set, which contains the most common patterns, from the rare sets, which contain sequences with less common shots and subjects but still have consistent patterns. In the second clustering step, we used the normal coordinates of the sequences belonging to the main set. At the end of this process, we have grouped the sequences in the main set into different clusters. Those in addition to the ones defined in the rare sets define what we call the basic editing labels. These labels characterize the sequences only in terms of subjects and shot sizes used. At this point, we perform one last clustering step, taking into account also the previously defined editing pace and trend labels. This time, the clustering algorithm is applied to data samples characterized by the basic editing label, which represents the shot sizes and subjects used in the sequence, the editing pace label, which characterizes the amount of shots in the segments, and the editing trend label, that characterizes the size relationship among the different shots. The resulting clusters identify the editing labels. We will use these editing labels in conjunction with the editing segments to train and test an LSTM classifier to validate our approach.

### Validation

This last phase is straightforward. As done previously, we train and test a classifier, an LSTM model in this case, on our data, characterized by the editing segments and their newly defined editing labels. With respect to the Cinescales study case, these labels characterize the segments in multiple ways. We rely on accuracy pèrecision recall and f1-score to evaluate the performance of the classifier. We show the result of our methodology in the next section, together with an analysis of our first discoveries.

### 3.4.3 Exsperimental Validation

Here, we present our discoveries and results in a 50-class scenario.

In the first clustering step, in which we separate the main set from the rare sets, the silhouette index showed that 4 clusters were a good number, with a value of 0.89. If we further increased the number of clusters, the silhouette score would rapidly decrease. with 5 we would get 0.86, while with 6 we would get a value of 0.76. Of these 4 segments, the main set is identified by the biggest cluster, while the rare sets by the smaller ones.

In the second cluster phase, we focus on the main cluster, which has 24,010 points representing sequences. This time we use the normal coordinates. After simulating a multiplicity of different scenarios (2-60) with the help of the silhouette index, we identified 8 as a good number of clusters. This scenario is more complex than the previous one, and the best value of the silhouette index is 0.63 with 8 clusters. At the end of this second clustering phase, we have identified the basic editing patterns, characterized in Table 3.10.

| Phase | Cluster | Number of samples | main sizes | main subjects |
|---|---|---|---|---|
| 1 | 1 | 33 | ECU | Ot-F |
| 1 | 2 | 227 | O | Ot |
| 1 | 3 | 93 | ECU | Ot-Ob |
| 2 | 4 | 1498 | MS - WS | H |
| 2 | 5 | 1659 | WS - MS | l |
| 2 | 6 | 1166 | MS - CU | f-h |
| 2 | 7 | 8441 | MS | h |
| 2 | 8 | 1713 | MS - WS | h-ob |
| 2 | 9 | 1781 | CU | l |
| 2 | 10 | 922 | MS-WS | a |
| 2 | 11 | 6830 | MS | l |

Table 3.10: The different clusters identified after the first two clustering phases.

Finally, in the last step, we join the basic editing labels with the editing pace and trend labels and apply the KMeans++ algorithm once more. We stopped at 50 classes. While we could have identified more classes above a certain threshold, those classes start to have not enough samples to be properly learned.

With 50 clusters, the value achieved by the silhouette index is 0.91. We characterize the results of this analysis in Table 3.12. In 3.11 we show from which basic editing patterns the final editing labels come from. To validate our representation now, we use an LSTM classifier. using a 10-fold stratified cross-validation approach.

| Cluster | final classes (n sample) |
|---|---|
| 1 | *44(11), 8(22)* |
| 2 | *30(14), 44(81), 8(132)* |
| 3 | *14(25), 16(28), 21(23), 30(13), 42(4)* |
| 4 | *10(72), 14(510), 16(99), 21(496), 30(141), 42(180)* |
| 5 | 1(541), 31(301), 34(265), 41(163),<br>*10(189), 32(200)* |
| 6 | 20(382), 25(438), 47(138), 46(113),<br>*13(19), 22(2), 32(65), 48(4)* |
| 7 | 15(1381), 19(372), 45(131),<br>49(131), 6(2356), 7(1421),<br>*2(1421), 22(764), 13(220), 35(201), 48(96)* |
| 8 | 23(521), 3(714), 38(164), 43(133)<br>*2(1), 22(1), 29(16), 35(79), 4(83), 48(1)* |
| 9 | 17(614), 24(160), 28(289), 39(236),<br>*29(95), 37(197), 4(190)* |
| 10 | 11(301), 26(381), 36(25), 9(31),<br>*5(49), 12(19), 37(102), 40(8)* |
| 11 | 0(1207), 27(457), 33(258), 18(651),<br>*9(234), 36(321),*<br>*40(226), 5(2196), 12(1284)* |

Table 3.11: On the right are the clusters identified from the first two phases; on the right are the final labels identified for those classes.

The model achieved an overall accuracy of 92.8% in labeling sequences that it has never seen before in a 50 classes scenario.

| layer(type) | output shape | number of parameters |
|---|---|---|
| Embedding | (None, 30, 30) | 300,000 |
| LSTM | (None, 100) | 52,400 |
| Dense | (None, 50) | 5,050 |

Table 3.12: Total trainable parameters 357,450

Table 3.12 shows the composition of the *Basic Editing Sets* and which final classes are formed from them. The classes in italics are the classes that are created from different *Basic Editing Sets*. Of the 50 classes considered in this scenario, 28 of them originate from one class while the remaining 22 are shared among different basic editing labels.

### Quantitative Analysis

| | precision | recall | f1-score | support | | precision | recall | f1-score | support |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.87 | 0.93 | 0.90 | 121 | 25 | 0.90 | 0.80 | 0.84 | 44 |
| 1 | 0.90 | 0.87 | 0.89 | 54 | 26 | 0.88 | 0.92 | 0.90 | 38 |
| 2 | 0.99 | 0.99 | 0.99 | 142 | 27 | 0.95 | 0.85 | 0.90 | 46 |
| 3 | 0.93 | 0.89 | 0.91 | 71 | 28 | 0.96 | 0.93 | 0.95 | 29 |
| 4 | 0.82 | 0.85 | 0.84 | 27 | 29 | 0.77 | 0.91 | 0.83 | 11 |
| 5 | 0.98 | 1.00 | 0.99 | 224 | 30 | 0.81 | 0.76 | 0.79 | 17 |
| 6 | 0.94 | 0.97 | 0.95 | 236 | 31 | 0.87 | 0.67 | 0.75 | 30 |
| 7 | 0.93 | 0.97 | 0.95 | 137 | 32 | 0.81 | 0.81 | 0.81 | 27 |
| 8 | 0.00 | 0.00 | 0.00 | 15 | 33 | 1.00 | 0.85 | 0.92 | 26 |
| 9 | 0.92 | 0.85 | 0.88 | 27 | 34 | 0.86 | 0.67 | 0.75 | 27 |
| 10 | 0.76 | 0.85 | 0.80 | 26 | 35 | 0.86 | 0.86 | 0.86 | 28 |
| 11 | 0.89 | 0.81 | 0.85 | 31 | 36 | 0.88 | 0.80 | 0.84 | 35 |
| 12 | 0.97 | 0.98 | 0.98 | 130 | 37 | 0.84 | 0.90 | 0.87 | 30 |
| 13 | 0.86 | 1.00 | 0.92 | 24 | 38 | 0.92 | 0.69 | 0.79 | 16 |
| 14 | 0.86 | 0.80 | 0.83 | 54 | 39 | 0.96 | 1.00 | 0.98 | 24 |
| 15 | 0.97 | 1.00 | 0.99 | 138 | 40 | 0.67 | 0.96 | 0.79 | 23 |
| 16 | 0.82 | 0.69 | 0.75 | 13 | 41 | 1.00 | 1.00 | 1.00 | 16 |
| 17 | 0.90 | 0.93 | 0.92 | 61 | 42 | 0.62 | 0.89 | 0.73 | 18 |
| 18 | 0.91 | 0.95 | 0.93 | 65 | 43 | 0.93 | 1.00 | 0.96 | 13 |
| 19 | 1.00 | 0.92 | 0.96 | 37 | 44 | 0.00 | 0.00 | 0.00 | 9 |
| 20 | 0.88 | 0.97 | 0.93 | 39 | 45 | 1.00 | 0.92 | 0.96 | 13 |
| 21 | 0.88 | 0.87 | 0.87 | 52 | 46 | 0.88 | 0.64 | 0.74 | 11 |
| 22 | 1.00 | 0.99 | 0.99 | 77 | 47 | 0.61 | 1.00 | 0.76 | 14 |
| 23 | 0.90 | 0.87 | 0.88 | 52 | 48 | 0.89 | 0.80 | 0.84 | 10 |
| 24 | 1.00 | 1.00 | 1.00 | 16 | 49 | 0.92 | 0.85 | 0.88 | 13 |

Figure 3.12: Precision, recall and f1-score of the LSTM on 50 classes

| | Precision | Recall | f1-score |
|---|---|---|---|
| macro average | 85.7% | 85,8% | 85.5% |
| weighted average | 92.0% | 92.8% | 92.4% |
| Accuracy | | | 92.8% |

Table 3.13: Overall performance of the LSTM classifier on 50 classes.

In Table 3.13, we have characterized the performance of the LSTM model in terms of Precision(P), Recall(R), and f1-score(f1). In 3.12 we show these metrics per class. The macro-average computes the f1-score, which is calculated for each label independently of the class proportions, whereas the weighted average considers the class proportions. The scores reported are averages derived from the 10 different folds used in stratified cross-validation.

The general performance of the classifier is very good, however if we take a closer look we can see the on some classes it performs definitely better with respect to others. For instance, the LSTM is not able to predict one single sample coming from classes 8 and 44. This is because these classes do not have enough samples to be learned by the LSTM model. Additionally, both of these classes are made from

sequences that come from different basic editing labels. The issue with these types of sequences is that even if they have different shots and elements since they are not consistent enough, they get grouped with other sequences that have similar editing pace and trend labels. To avoid this issue while keeping a decent amount of samples inside each class would be to increase the amount of samples under analysis. Just increasing the number of classes would not suffice. We run two simulations with 70 and 90 classes, respectively. In the 70 classes scenario, we can see some degradation in the model performance but not too severe. If we increase the number of classes under analysis, the averaged f1- score drops to 75%. On the other hand, reducing the number of classes would only increase the noise inside each class.

### Qualitative analisys

Figure 3.13: 30 seconds sequence from "Star Trek: Insurrection".

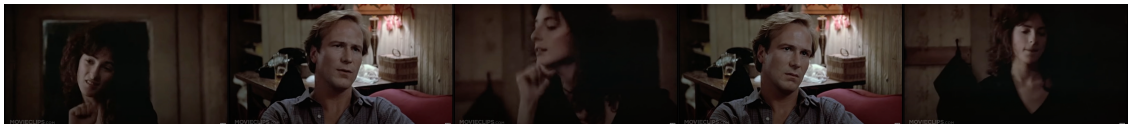


Figure 3.14: 30 seconds sequence from "Children of a Lesser God"

The identified classes delineate patterns that range from general to highly specific. When examining two segments originating from classes defined by distinct basic editing patterns, editing paces, and trends, the differences are outstanding. For example, Figure 3.13 illustrates a segment from "Star Trek: Insurrection", directed by Jonathan Frakes. This segment is characterized by a rapid editing pace and a combination of predominantly medium and wide shots, capturing scenes where characters are running and shooting across the location.

If we take a look at the segment illustrated in Figure 3.14 we can see how not only we have a slower editing pace but also different shots and concatenation of them. This second segment is taken from "Children of a Lesser God," directed by Randa Haines. In this scene, with respect to the previous one, there is much greater emphasis on the characters rather than the overall context. This segment belongs to class 46 a class that derives from a basic editing label characterized by close ups and medium shots of people.

When comparing two segments from different classes within the same basic editing pattern cluster, the differences are less pronounced. The segments depicted in

Figure 3.15: Sequence 1

Figure 3.16: Sequence 2

Figure 3.15 3.16 are taken from class 43, which is exclusively derived from basic editing set 8. Both scenes primarily feature medium shots of individuals interacting with objects. The fact that both segments conclude in a similar manner is somewhat coincidental. This indicates that while segments represented by a single class can exhibit varying structures, they generally conform to the boundaries implicitly defined by the editing pace and trend labels. On the other hand, if segments come from different classes, they are very different from each other.

### 3.4.4 Discussion

Another issue is that even if LEMMS can identify more defined labels, there are not enough data examples for the LSTM classifier to learn and thus properly validate them. Thus, if we want to identify more labels and their corresponding segments, we need to increase the amount of data to be analyzed.

In some instances, sequences belonging to the same class may appear visually similar in segment representation, yet differ significantly when the videos are compared. This discrepancy can be illustrated with a specific segment in Figure 3.17. This segment is composed entirely of medium shots, but the sizes of these shots vary. The variation arises because the medium shots represented fall at the extremities of the medium shot classification, causing some to resemble close-ups while others resemble long shots.

Figure 3.17: 30 second sequence from "Pacific Rim"

To address this issue, implementing a more detailed shot size classification system would be beneficial. Additionally, incorporating more features from the AVE dataset, such as the *number of people* and *camera movement*, would enhance the data representation. This enriched representation would enable the characterization of more complex editing patterns.

Another challenge is that, although LEMMS can identify more specific labels, there are insufficient data examples for the LSTM classifier to effectively learn and validate them. Consequently, it is necessary to increase the volume of data analyzed to identify a greater number of labels and their corresponding segments.

Finally, while the trend attribute has proven useful in characterizing editing sequences, a more sophisticated strategy should be developed to better characterize mixed trends.

## 3.5 Chapter Conclusions

This concludes this chapter on editing pattern analysis. In both case studies, we were able to identify meaningful patterns and to characterize them in such a way that a model could learn to classify them with good accuracy. On the other hand, these studies show that there is a correlation between the sequence of shots and the final video. However, there is still room for improvement in the characterization of these patterns. As we have seen, it is better to have a more fine-grained shot classification. Also, the characterization of what is happening in the scenes needs to be improved. On the other hand, now that we have seen that there is a correlation between editing sequences and videos, we can exploit this correlation to teach the model to generate new editing sequences. To this end we have devoted the second half of chapter 4. In which we use a more refined classification and characterize the editing sequences with textual data. In the Appendix instead, the results of

a preliminary study conducted at Laval University on editing sequence generation using dataset AVE are presented. Many key concepts introduced later on were assessed during this research.

# Chapter 4

# Toward Automatic Storyboard Creation

## 4.1 Introduction

In recent years, the number of points of contact between the movie and AI fields has increased intensively. With the advent of vision transformers and stable diffusion models, the quality of generated images and videos has increased to a level that was not thinkable just a few years ago. Nowadays, there are models able to create images or videos using simple textual input. While the realism of generated images and videos keeps growing every day, a key aspect has been left aside: the video editing structure, i.e., the concatenation of shots used to represent the video.

In this last chapter, we focus on the challenge of recreating storyboards and editing sequences. Due to the complexity of the challenge, we addressed it by dividing it into two tasks. The first task we address is the creation of specific shots given a textual prompt and the shot size. In this study, presented in [25], we introduce Dreamshot, a methodology to fine-tune diffusion models that we use to generate images from prompts with the shot size constraint. The second task that we want to address is the prompt to edit the sequence. It can be summed up as follows: given a textual prompt that describes a movie scene, we create an algorithm that gives back as output a sequence that represents the shots to be concatenated to represent that scene. The intuition that defined our work is that depending on what is happening on the scene certain shots will be preferred with respect to others, as shown by our previous studies. Hence, it should be possible to find a correlation between the shots used and the scene textual descriptions, that offer a more detailed representation of the scene with respect to what was done in our previous studies.

## 4.2 Related Work

In addition to the studies that focused on video editing mentioned in the previous chapters, with the rise of Transformers and diffusion models new models and challenges were addressed in the context of video editing. To give an idea, when this Phd started the first version of DALLE [58] was still to be released. Back then the state of the art methodologies relied on different Generative Adversarial Networks (GAN) [28]. The first version of DALLE itself relied on a combined approach of GAN and Variational AutoEncoders (VAE) [41]. Shortly after the first Diffusion models appeared followed by Latent diffusion models like Stable diffusion [63]. Due to the superiority in terms of performance of diffusion models compared to approaches based on GAN in general, we adopt this family of algorithms in our studies.

Text to video models follow a similar trend, although they offer even more challenges and offer unique solutions. For instance in [51] the authors present a new methodology to perform video editing. Here with video editing they mean the process of visually altering editing clips. This new approach employs a video diffusion model to blend low-resolution spatio-temporal data from the original video with high-resolution synthesized information that matches a text prompt. As a result they can turn a video into another using only a text prompt or generate a video from an input image and a text prompt. Other works [71] [40]instead proposed text to video models that use as basis the knowledge learned from text to image models. As new models were released also new transfer learning techniques were developed to finetune these models for new tasks. In [64] the paper introduces a new method for "personalizing" text-to-image diffusion models. By fine-tuning a pretrained model with a few images of a subject, the model learns to associate a unique identifier with that subject. This identifier can then be used to generate new, photorealistic images of the subject in different scenes. Another example is ControlNet [93], a technique that leverages the knowledge of diffusion models and to produce images that can be heavily edited through prompts. None of these works however take into account the shot size which could be a useful constraint to generate more scenographic images. However more tool to create storyboards are being developed,although they focus on the single shot recreation rather than the whole sequence, like [73], [74], [75], [76].

Other studies and tools focus more on video editing as the process of concatenating images together. A first example is [45]. This work introduces a new task called Story Visualization, which involves generating a sequence of images from a multi-sentence paragraph, with one image per sentence. Unlike video generation, this task emphasizes global consistency across dynamic scenes and characters rather than continuity between frames. The proposed model, StoryGAN, is based on the sequential conditional GAN framework and includes a deep Context Encoder to track the story flow dynamically. They do not keep the shot size in consideration.

Curiously they never mention the term storyboard, although it is the output of their methodology.

Another work that focuses on storyboards is [60]. Here the authors present the Virtual Dynamic Storyboard (VDS) system, which is designed to create storyboards. VDS allows users to storyboard shots in virtual environments, enabling them to test settings before actual filming. It operates on a "propose-simulate-discriminate" mode: given a formatted story script and a camera script, VDS generates character animation and camera movement proposals based on predefined rules. In this study the shot sizes used are three; Close up, Medium Shot and Long Shot. In the context of shot generation also our proposed solution focuses on three shots, however we generate photorealistic images with simple text prompts, instead of relying on entire scripts. For what concerns the creation of sequences of shots our proposed solution takes into account 10 different shot classes and is able to generate editing patterns with limited textual information compared to other studies seen so far.

## 4.3 Dreamshot

In this work we defined a simple methodology to fine-tune diffusion models to the task of shot recreation. The idea is to add the shot size in the textual prompt used to generate the image to get the corresponding image with the right shot size. The shot sizes considered in this study are Close up, Medium shot and Long shot, shown with some generated examples in figure 4.1.


Figure 4.1: Generated examples of shot sizes considered.

Current Diffusion models can create high quality, both photorealistic and not, images through conditioning, which is usually in the form of a textual prompt. In order to generate high quality images Diffusion models rely on the following two steps: the forward diffusion and the reverse generation. During the training phase the model progressively adds small amounts of Gaussian noise over several steps to the images to the point that they are indistinguishable from random noise. Then in the reverse generation, as the name suggests, we perform the opposite operation. Using a trained Variational AutoEncoder to remove noise at each step

the model gradually denoises the image. After numerous steps, the noise is sufficiently reduced, resulting in a new image that closely resembles the original training data, ensuring stability and high quality. A key feature of Stable Diffusion models is the integration of textual conditioning, which allows the generation of images based on textual descriptions. This is done by encoding text inputs into latent vectors using pre-trained language models. The main downside of these models is that they require a lot of hardware resources to be trained or even traditionally finetuned. There are however new finetuning techniques, like Dreambooth and ControlNet that allow the customization of diffusion models with a sensible lower computational load. These techniques can add new subjects or styles to the model knowledge. We leverage this feature by teaching the shots as if they were graphic style. For instance the close up can be thought of as the style of a specific painter that focuses on close portraits.

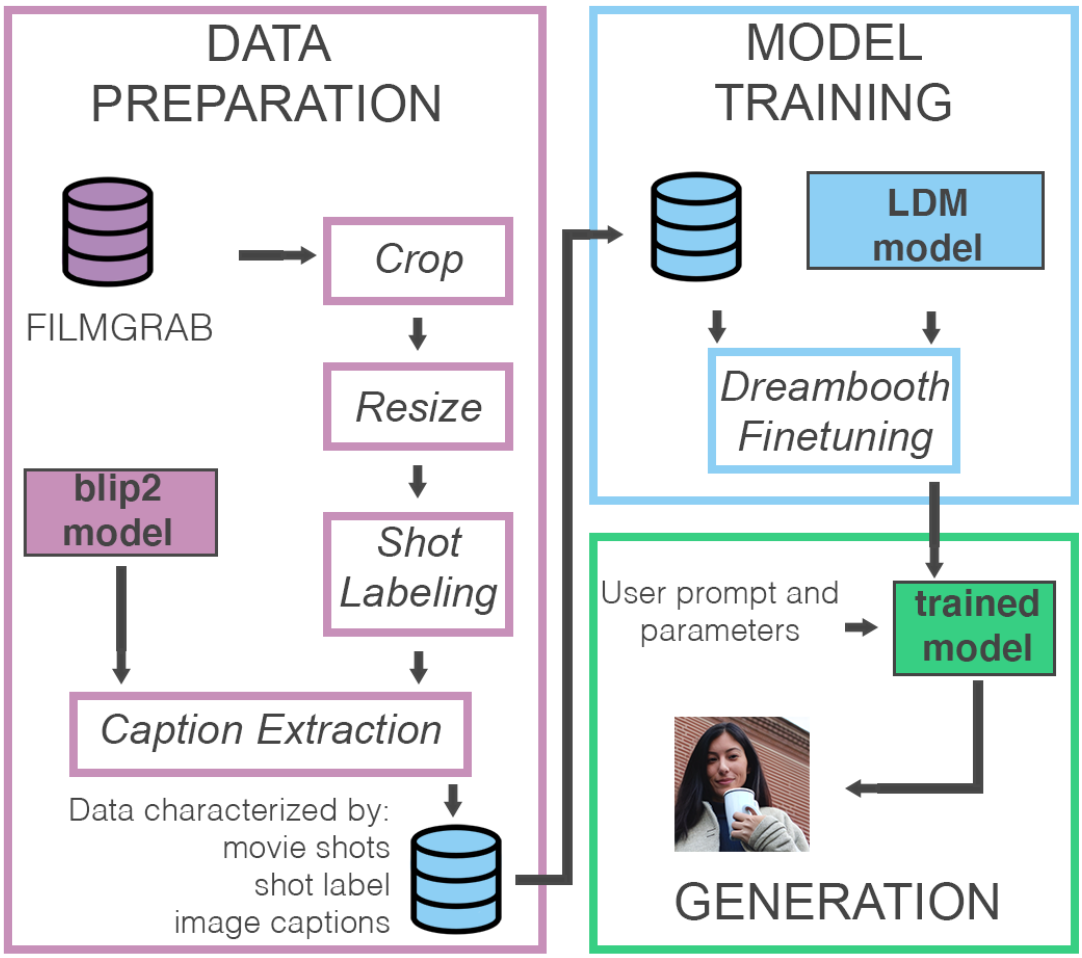



Figure 4.2: Methodology overview

In figure 4.2 we show a brief overview of the different operations that take place during data preparation, model training and generation phases. To finetune a stable diffusion model toward image generation with the shot size constraint we used a Dreambooth implementation that leverages Low Rank Adaptation [35].

### 4.3.1 Data preparation

This phase is crucial for the final output quality of the diffusion model, necessitating high-quality samples and associated captions that accurately describe the images. To achieve this, we sourced our images from the FILMGRAB movie repository, which offers high-quality frames from a diverse array of movies. This repository is available for research purposes at [26]. To prepare the images for the model, we resized and cropped them to a resolution compatible with the stable diffusion model (512x512).

After cropping, we labeled 200 images for shot type, resulting in a total of 600 images. The final preprocessing step involved extracting the image captions. For this task, we relied on the Vision-Language model Blip2 (Bootstrapping Language-Image Pre-training) [44]. Vision-language models, such as Blip2 and CLIP (Contrastive Language-Image Pre-training) [55], integrate visual and textual data to understand and generate information by associating images with their corresponding textual descriptions. These models typically consist of two neural networks: a vision encoder and a text encoder, which map inputs into a shared multimodal embedding space using techniques like contrastive learning. Pre-trained on extensive datasets of image-text pairs, vision-language models excel in zero-shot learning, image retrieval, and enhancing AI systems' contextual understanding.

At the end of this process, we have a curated dataset of 600 movie frames, all standardized to the same resolution, annotated with shot sizes, and accompanied by detailed textual descriptions. This enriched dataset serves as a robust foundation for fine-tuning our diffusion model, ensuring high-quality output by leveraging both visual and textual information.

### 4.3.2 Methodology

Now that we have an appropriate dataset for the task we can finetune a latent diffusion model using the Dreambooth technique to recreate images with the shot size constraint.

#### Model finetuning

The core concept of DreamBooth involves associating a unique identifier with a small set of input images. This identifier, when used in prompts alongside its

associated class (e.g., "A [V] dog"), combines prior class knowledge with new information to reconstruct the subject. To enhance diversity and mitigate language drift, a new autogenous class-specific prior preservation loss is introduced during training. Additionally, the model is supervised using its own generated samples to maintain and leverage both class-specific knowledge and instance-specific information for generating new samples effectively. In our case the identifier instead of being an object is a shot size.

Dreambooth alone effectively reduces computation cost. However it can be further reduced using Dreambooth in conjunction with Low RanK Adaptation.

This finetuning technique is designed to efficiently fine-tune large models by modifying only a small subset of their parameters. This method leverages the concept of low-rank approximation, which reduces the number of parameters that need to be adjusted during the adaptation process. LoRA is able to do so by decomposing the parameter matrices of a neural network (W) into matrices with smaller dimensions (A and B), in a way that AxB approximates W. This reduces the computational resources because during the finetuning process only matrices A and B are updated instead of the entire weight matrix. Additionally by fine-tuning only a small set of parameters LoRA preserves much of the original pre-trained model's knowledge. Suppose that the original model is denoted by $W$ and the fine-tuned model is defined as:

$$W' = W + \Delta W \tag{4.1}$$

What LoRA does is train the residual $\Delta W$ instead of $W$. However instead of training the whole matrix $\Delta W$ it decomposes it in two smaller matrices A and B such that:

$$\Delta W = AB^T \tag{4.2}$$

Once the finetuning phase is finished we can store the weights learned and store them. When we need to use the fine-tuned model we use the original one alongside the previously stored weights. In this study we have used stable diffusion 1.5.

In our specific case, no unique identifier was specified during training; by not binding the concept to a specific token, the model always generates in the trained style (or shot type in our case) when the $\Delta W$ model is specified in the prompt.

### Generation

Once that we have fine-tuned the model we can generate new images. Along with the textual input describing the image our finetune model requires the configuration of some parameters. The prompts that we can use can be either positive, hence moving the subject forward the specified description, or negative, hence moving it away. Using meaningful prompts is essential in order to obtain a meaningful

Figure 4.3: *Prompt*: a high-quality close_shot picture of a woman holding a cup of coffee in front of a brick building

image. In addition to the textual prompt, other parameters that heavily impact the generation of the final image are the following: sampler, number of steps, CGF scale, seed, $\alpha$. The sampler influences the amount of noise predicted and subtracted at every step of the diffusion process, while the number of steps is self explanatory. The higher the number of steps and higher the quality and generation time of the image. CGF Scale is a free guidance classifier technique that separates the generated samples from random unlabeled ones, reinforcing the adherence of the image to the provided prompt. The seed influences the initial noise map. It follows that using different seeds implies the generation of different output images.

The last parameter that we need to set is $\alpha$. It is a parameter that we use to regulate the use of the trained weights with LoRA. The value of $\alpha$ ranges from 0 to 1. When its value is 0 the model used is the original one, while when set to 1 we completely rely on the fine-tuned one. With a value in between we can use a mixture of both. Sometimes to achieve better result reducing the influence of the fine-tuned model a bit can produce better results. In 4.3 we show an example of our generated images.

### 4.3.3 Results

#### parameter setting

The dataset used for testing is composed of 1800 shots sampled from the FILM-GRAB, like it was done with the previous frames, equally divided among the shots. In addition to the images we also have generated the image captions using Blip2. Then we randomly select these captions and use them to generate two images, one generated from the original model and one generated from the finet-uned one. We repeat this procedure several times, for a total amount of 1500 pairs of generated

images, divided among three shot classes. In Table 4.1 we show the parameters that we have used to generate our images.

Table 4.1: The pararameters used for generation during testing

| | |
|---|---|
| sampler | DPM++ SDE Karras |
| steps | 16 |
| seed | random |
| cfg_scale | 6 |
| prompt | a high-quality [shot_type] picture of [caption] |
| size | 512 x 512 |

### Quantitative results

To obtain quantitative results, following the example of Dreambooth, we have used the CLIP-T and DINO metrics. CLIP-T, computes the average cosine similarity between the embeddings extracted from the textual prompt given as input and the embeddings extracted from the generated image produced as output. The DINO score on the other hand computes the average cosine similarity between the embeddings of the image generated from by model under analysis and the original image.

In Table 4.2 we can see that on average our fine-tuned model performs better, without surprises, than the baseline model. The increase in the CLIP-T score indicates that the fine-tuned model is able to interpret the shot size constraint contained in the prompt slightly more efficiently, which makes sense. The reason why the increase is so small is related to the fact that the shot constraint, while it has an impact on the generated image, does not alter significantly the semantic content of the text. Since CLIP-T measures the similarity of textual embedding and embedding of the generated images, what we indirectly measure is how little the shot size constraint impacts the semantic concepts contained in the input prompts. The more consistent increase in performance in terms of DINO metric also is to be expected. Since this metric measures the closeness of the generated image with its original counterpart, the fine-tuned model outperforms the baseline because it knows how to represent a shot size. These two matrices together show how the knowledge of the shot size has a small impact on the semantic concept of the image, but on the other hand it has a more consistent impact on the generated images.

| | **CLIP-T** | **DINO** |
|---|---|---|
| baseline | 0.3221 | 0.4163 |
| ours | **0.3269** | **0.4989** |

Table 4.2: Results for the CLIP-T and DINO metrics on the 1500 pairs test.

We performed a second study with 600 new images with textual caption, but we have removed the shot size constraint from the textual prompt. The setup is the same that was implemented previously. The results are in Table4.3. the fine-tuned model still outperform the baseline, showing that even without direct indication the model preserves an orientation toward the specific shot it has been fine-tuned on.

| | **CLIP-T** | **DINO** |
|---|---|---|
| baseline | 0.3214 | 0.4014 |
| ours | **0.3234** | **0.4803** |

Table 4.3: Results for the CLIP-T and DINO metrics on the ablation test.

### Qualitative Survey

In addition, we conducted a survey involving human subjects. The subjects were colleagues, friends and family, with different knowledge backgrounds and expertise. Each participant was shown a total of 36 pairs of images, labeled $A$ and $B$, generated using the same settings and prompts, with one image from the baseline model and the other from the fine-tuned model. The labeling of images as $A$ or $B$ was randomized. For each pair of images, presented with the shot size and input prompt, the participants were asked to answer the following three question:

- Which image do you prefer?
- Which image better corresponds to the associated shot type?
- Which image better matches the associated prompt?

Participants could choose $A$, $B$, or *neither/same* if they didn't have substantial differences. A total of 55 subjects responded to the survey, and the results are summarized in Table 4.4.

These results indicate that our fine-tuned model generates images that are more appealing and represent more accurately the shot size and the prompt in input. Also in terms of image likability, the fine-tuned model outperformed the baseline one that model scored a lower score, which suggests that the fine-tuned model produces images of equal or superior quality. The survey results align with the CLIP-T and DINO metrics, showing that higher likability and shot-type accuracy, which are closely related to DINO scores, are significantly improved compared to prompt alignment and CLIP-T scores relative to the baseline.

Table 4.4: Results collected from a survey conducted on 55 subjects. The score are expressed as percentage over the total number of answers.

| question | baseline | ours | same / neither |
|---|---|---|---|
| Which picture do you like most? | 26.18 | **57.43** | 16.4 |
| Which picture is closer to the associated shot type? | 20.46 | **56.84** | 22.7 |
| Which picture is closer to the associated prompt? | 20.35 | **49.31** | 30.34 |

### Visual results: shot creation

Here we present some visual result that we have obtained using Dreamshot. In 4.4 we show how the same image is impacted by the shot size and the value of the alpha coefficient. As the value of the coefficient increases the influence of the shot size is more influential on the image output.

These examples show that the model's ability to create images of the right shot size depends also on the input image. However even when the resulting image is not of the right shot size at least is toward the right size. The shot on which the model has more issues is the long shot. Especially with the second and fourth images the generated long shots are not really long shots, although the resulting shot size is wider than the original. With close ups and medium shots we haven't run into the same problem. This could be related to the fact that moving from a wider shot to a more narrow one is easier than the inverse since the model has to add new details to fill the image. Increasing the data samples could be one way to resolve this issue, as the model learns to generate more details related to the class long shot. In Figure 4.5 we show the difference between our generated frames and the baseline ones with an alpha value of 0.7. In the first case, our generated image is very consistent with the generated prompt, while the one generated by the baseline model gets the right details but the wrong shot size. In the second example both models get the shot size wrong, however our image is aesthetically more consistent. Finally in the third case both models got the shot size right but the details are wrong. The baseline model ignores some detail, like the motorbike. Our generated image instead gets all the details but represents them in the wrong way, like the fence that comes down the electricity cable.

### From storyboard to images

In 4.6 we started from a sketched storyboard (provided to entrants of the BBC's 'my place my space' competition), and generated the equivalent shots using


Figure 4.4: Some examples of the generation of the same subject with the three different trainings (close, medium, and long shot) with different levels of $\alpha$

Dreamshot. By using simple, effective prompting we managed to recreate similar

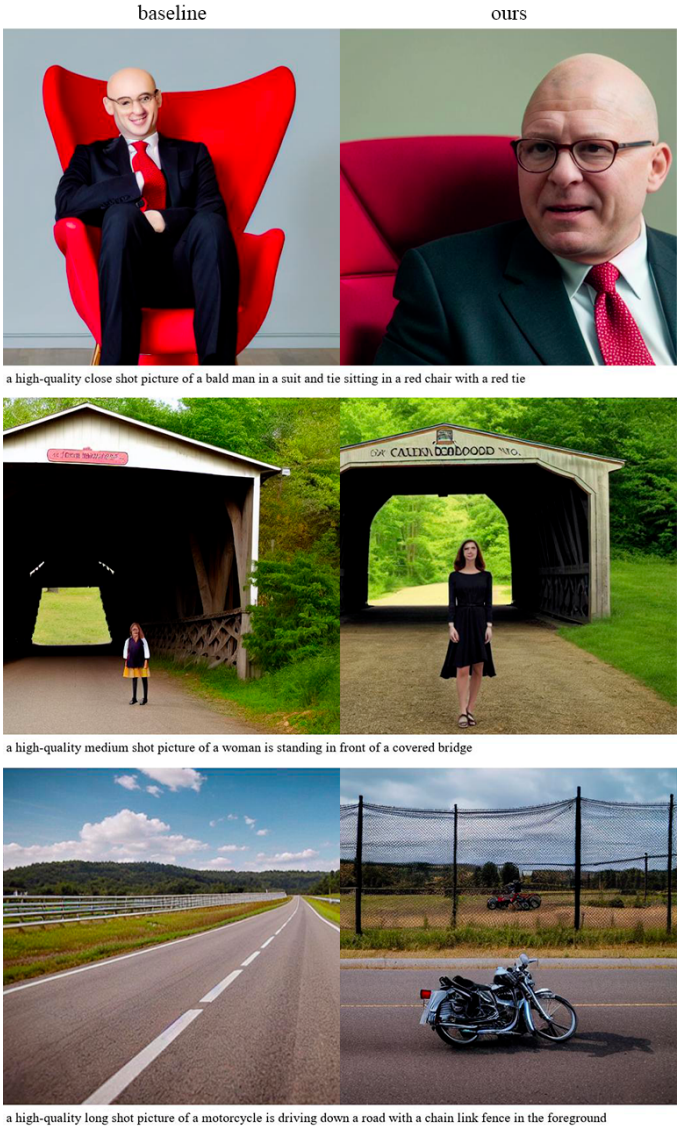
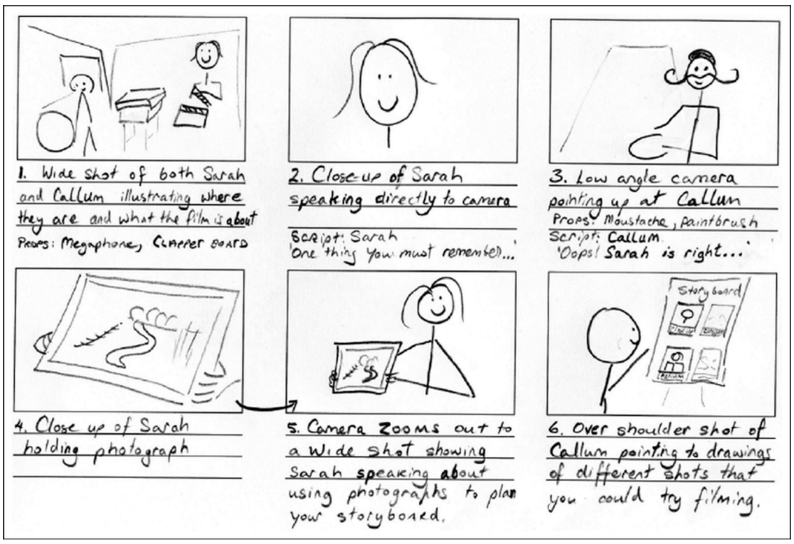



Figure 4.5: Image generation with shot constaint: baseline vs ours.

scenes with an evident cinematic style. In some cases character consistency is still an issue for diffusion models. However the generated storyboard is definitely more interpretable than the hand sketched one.

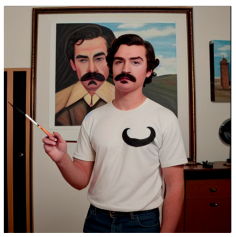
### 4.3.4 Discussion

While this approach can be further improved, it already achieves good results. We have shown that it is possible to finetune a model to encapsulate the concept of shot size. This enables the model to perform different tasks, from generating

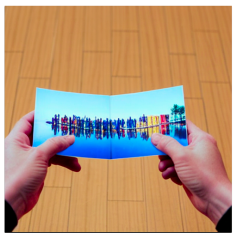



Long Shot, Sarah and Callum talking to each other in the middle of a room

Close Shot, Sarah speaking directly to the camera



Medium Shot, A front view of Callum with a mustache holding a paintbrush



Close Shot, two hands holding a photograph of something over a floor background

Long Shot, Sarah holding a small photograph in a room

Medium Shot, Callum seen from behind pointing at drawings

Figure 4.6: Example of a storyboard enhacement

single images with the shot constraint to converting a sketched storyboard into one with photorealistic images.However, several improvements can be made. On the one hand more shot size should be considered. This would allow it to add even

more control on the shot generation. Also the numerosity of the samples for the classes could be increased to refine the amount of detail in the generated images. Another open issue to address in future research is the character consistency among different shots.

Even if we are satisfied with the model's ability to generate cinematographic images, these alone do not make a storyboard. For the storyboard it is also necessary to order the shots in a meaningful sequence to represent the movie scene. To do so we have developed a novel methodology that we present in the next section, in which we investigate editing pattern generation.

## 4.4 TEdit: Text to Editing

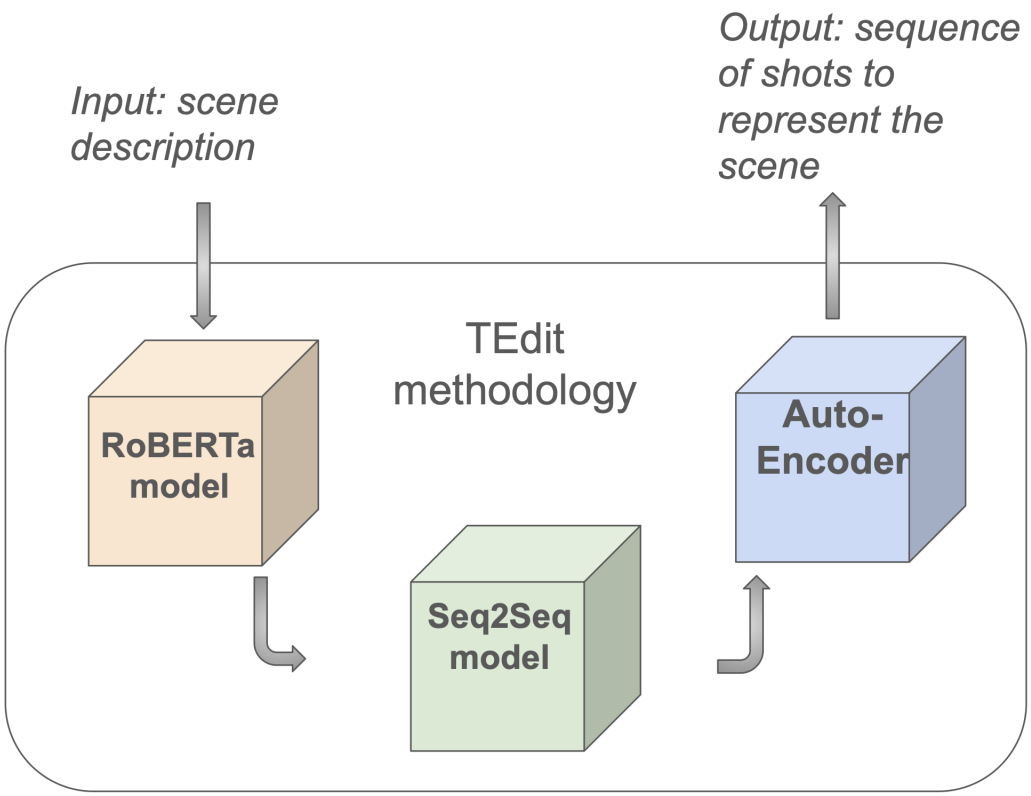



Figure 4.7: Methodology

Since our previous studies and other works showed that there is a correlation between the shots used and what is shown in the video we devoted our research efforts to develop a model that suggests a possible editing solution. In other words, given as input what will happen in the video, the model will give as output shot

sizes concatenated in a meaningful disposition. Starting from the Condensed Movie Dataset, a dataset that contains movie scenes descriptions, metadata and video urls, we have developed TEdit, a text to editing methodology, shown in 4.7. The intuition behind our methodology is that from a certain point of view they shots compose movie scenes like words compose a sentence. Shots like words do not have a lot of meaning alone, but they gain meaning when put together with other shots. However textual data has been and it is still being studied in depth editing much less. In the Condensed Movie Dataset there are movie scene descriptions and the related videos URLs. Hence to study and infer knowledge on the editing patterns extracted from the videos we used the corresponding scene description. TEdit receives as input a description of the scene and gives out as output a possible sequence of shots that can be used to represent the scene according to what is happening. To do so, Tedit exploits three models. The first model is a RoBERTa Language Model that is used to extract the sentence embedding contained in the textual scene description. Then a Sequence to Sequence model, previously trained for this purpose, is used to transform the textual embedding into its corresponding editing embedding. Finally, using an autoencoder previously trained for this task, the embedding is converted into a meaningful sequence of shots that can be used to represent the scene.

### 4.4.1 Data

We started from the Condensed Movie Dataset [5]. It is a dataset that has more than 34'000 videos. For each video there is also additional metadata.For every video in the CMD dataset, there is the youtube link to the video, the scene textual description, the cast, the movie genre from which the scene was taken, and other minor features like release country and so on.

### 4.4.2 Methodology

While the textual embedding extraction is not a difficult task, thanks to Large Language Models such as BERT [22], RoBerta [47] and others, the editing embedding extraction required some extra steps. The overall training process is shown in 4.8, while the different phases are better characterized in the following paragraphs.

In the textual features extraction phase we extract textual sentence embeddings using a pre-trained RoBERTa model. In the Editing Features Extraction we extract an embedding representation of the editing sequences extracted from the YouTube videos. Additionally in this step we also train the AutoEncoder that will be used later on in the TEdit methodology.

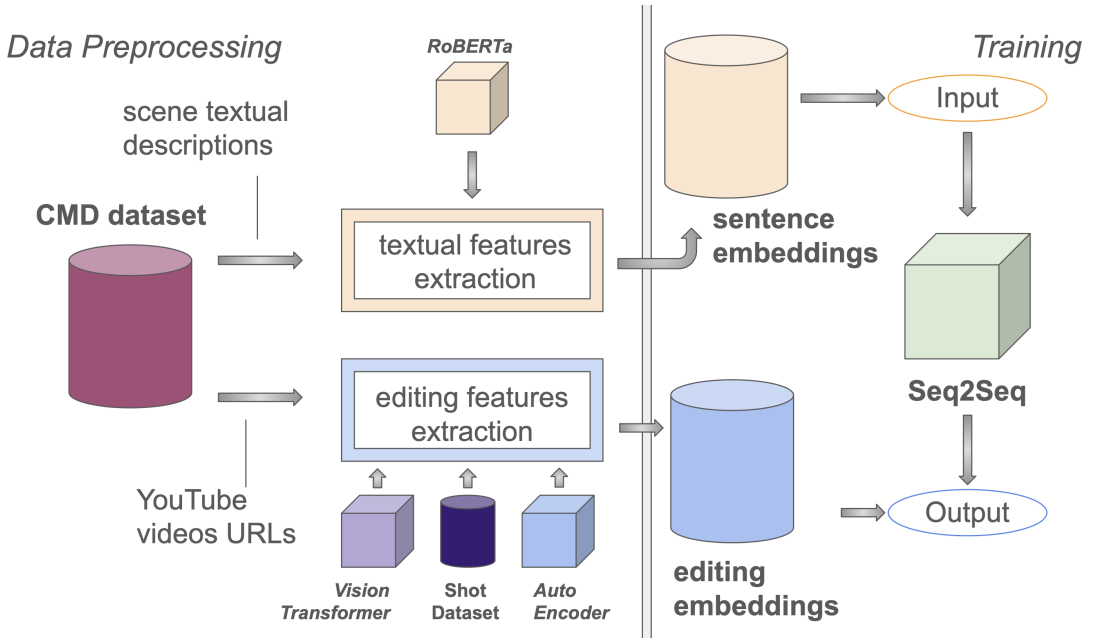



Figure 4.8: Overall training process

### Textual features extraction

To extract the sentence embedding we use a pre-trained Roberta model. Specifically we extract the embeddings after the Tokenizer layer. They have shape (768,1) and capture the semantic meaning of sentences in a continuous vector space.

### Editing Pattern Extraction and Encoding

In the Editing Pattern Extraction phase starting from the url videos and metadata we end up with editing sequences embeddings. Additionally at the end of the process we have the trained Autoencoder. The steps to do so are shown in 4.10.

The overall process can be summarized in 4 steps. The first three steps are necessary to prepare the data to feed to the Autoencoder during the final phase, the actual Autoencoder training.

### Step 1: fine-tuning a Vision Transformer

Since the CMD Dataset does not contain shot sizes we need to train a model to perform Cinematographic Shot Classification. Due to the results obtained by the Vision Transformer model we decide to fine-tune one on shot size classification. The dataset used to fine-tune the vision transformer is an alteration of the dataset presented in Chapter 2. Starting from that dataset we have included two additional classes, the trash and object classes. In the class thrash ends up all the "useless"

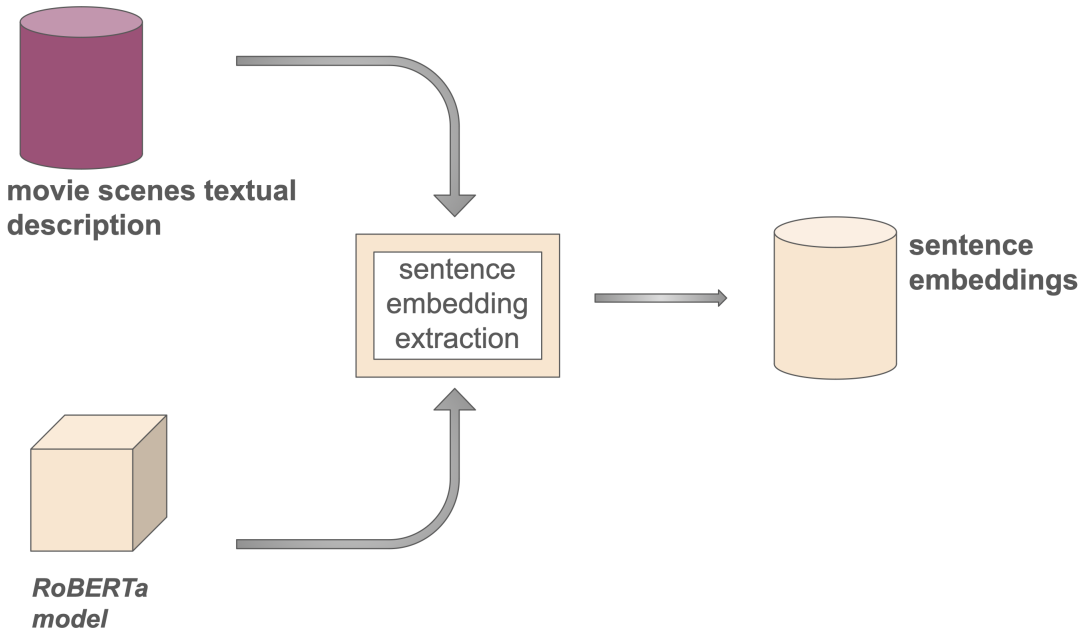



Figure 4.9: Scene description embedding extraction

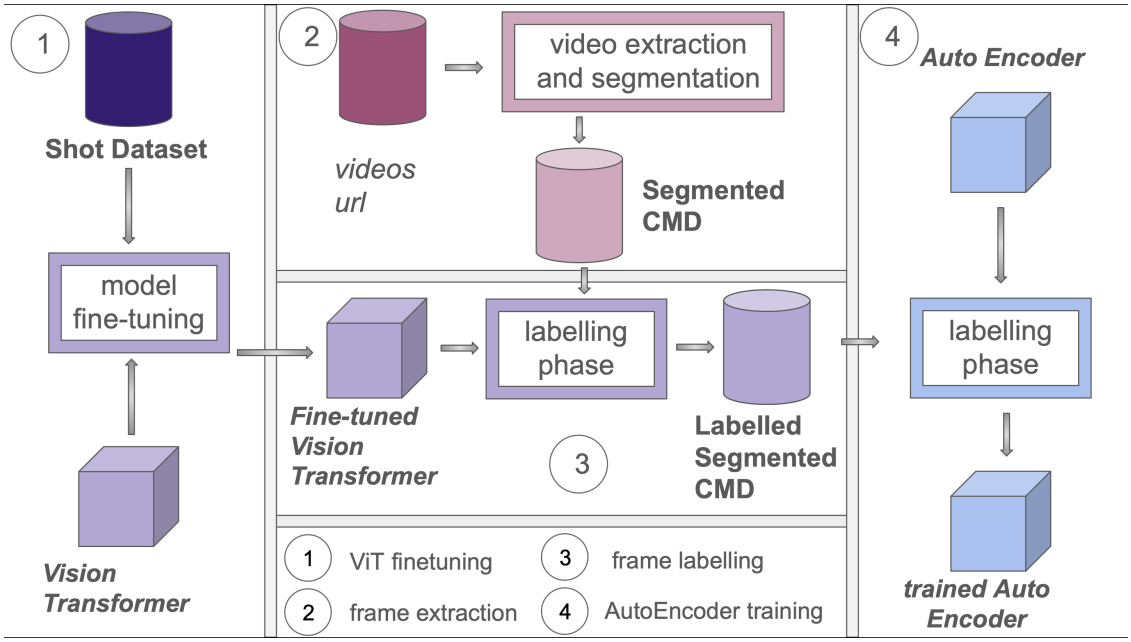



Figure 4.10: Editing embedding extraction

shots, i.e. credits while in the class object all of those images in which the subject is not the human figure but an object of some kind(gun, knife, watch...)

After including these new two classes we have fine-tuned the vision transformer

on the new dataset obtaining the performance shown in the results section.

Now that we have a fine-tuned Vision Transformer the next step is to extract the video frames that later on will be fed to it.

### Step 2: Frame Extraction

Starting from the YouTube urls we used the youtube-dl library to download the videos. Then, using the video partitioning algorithm presented in [30] we partitioned them into individual video clips. The partitioning algorithm gives back as output the duration and timing of each shot and from each of those ranges we select a frame. Visually the simplified process is shown in 4.11.

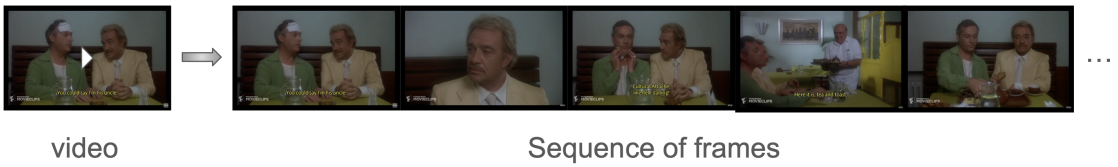



Figure 4.11: Video to sequence of frames

### Step 3: Frame Labelling

After splitting the video into clips we extracted a frame from every clip and labeled it using the fine-tuned vision transformer. The ViT model classifies every frame and then using the duration and timing of each shot we recompose the video as an editing structure characterized by the shot sizes. The simplified process is shown in 4.12

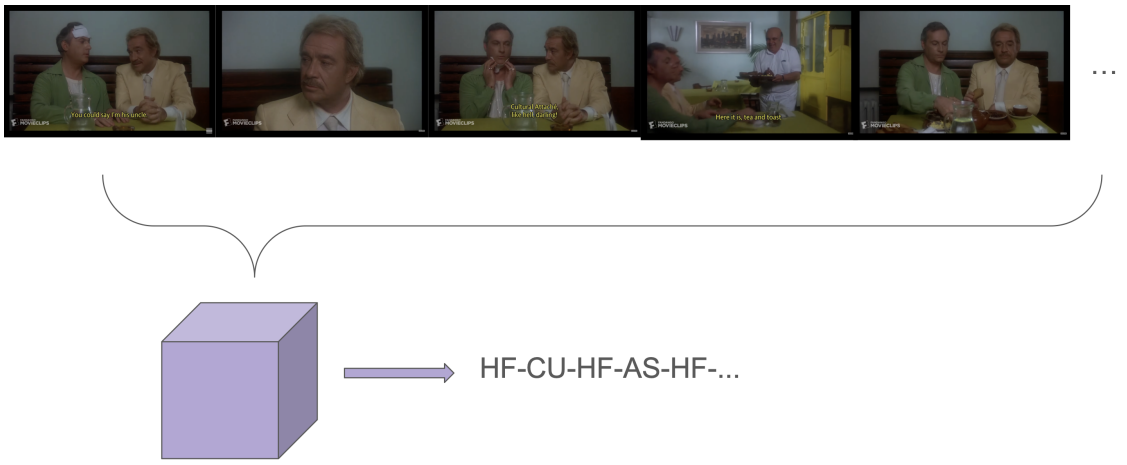



Figure 4.12: From sequence of frames to sequence of shot sizes.

These Editing Sequences have different lengths, so we have mapped them all to length 100. we excluded some videos in this way but just a minority. After padding the sequences with elements from the class trash appended at the bottom of each sequence, we one hot encoded them, obtaining sequences with shape 100,10.

After shaping the sequences into this one hot coded representation we flatten them and use them to train the autoencoder.

#### Step 4: AutoEncoder training

This type of model is often used for data preprocessing. An autoencoder is an unsupervised model that learns a compressed representation of the data. If the model is trained correctly then it should be able to recompose the original data starting from the embedded representation. In our case the autoencoder has to learn the embedded representation of the editing sequences. An initial approach consisted in keeping the multidimensionality of the data and passing from a representation of size(100,10) to an embedded representation of size (100,2). While the autoencoder is able to learn effectively this data representation it poses an issue for the Seq2Seq model that we use afterwards, hence we have discarded the approach in favor of a flat representation with size(1000,1). We have chosen this approach for its simplicity and effectiveness. On one hand sequences with only zeros and ones are easy to recompose into the original data format(100,10). On the other hand the autoencoder is able to effectively learn this data representation with the additional bonus that the embedding representation is with size(256,1). Once than the Autoencoder is trained we can use it to extract the embedding representation of the editing sequences, as shown in 4.13

#### Seq2Seq training

At this point we have the textual embeddings extracted from the scene descriptions on one hand and the editing embeddings of the corresponding video extracted from the AutoEncoder. The last step now is to map the textual embeddings to the editing embeddings. To do so we have used a simple Seq2Seq architecture described later. After training the model we now have all the necessary pieces to convert a textual prompt into meaningful editing sequences. should be able to feed to its movie scenes description and obtain a sequence of shots to represent it as output.

### 4.4.3 Results

Since different models are involved in this approach we will have a separate section for each model starting with the autoencoder, which after 15 epochs reaches a training and test accuracy of 100%. The loss used is the binary cross entropy. We tried also with a smaller embedding size, (192,1) and (128,1) but the autoencoder

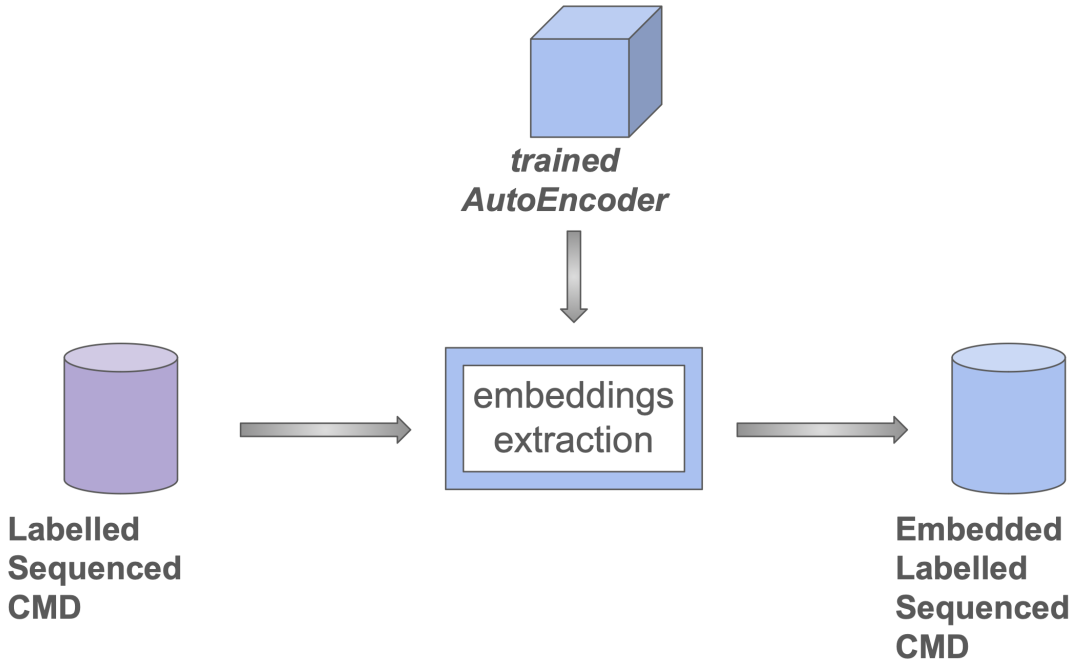



Figure 4.13: Sequence of shot embedding creation.

was not able to converge to a 100% accuracy in reconstructing the sequences. It works also with greater sizes, like (512,1). However as we well see in later having an embedding size in this context might not be ideal.

### Seq2Seq

The final model that we have used is a simple Sequence to Sequence model that maps the scene descriptions embeddings extracted from Roberta with size 768 to the corresponding editing sequence embedding extracted from the Autoencoder. Maybe to perform the task other architectures would have been more fit, like transformer, however due the the data in our possession a simpler model is more fit. The model reaches an overall average cosine similarity of 0.9280 on the training set, while on the test set it reaches an overall cosine similarity of 0.8140, which is inferior but still good. As we will show in generating new sequences or reconstruction of old ones the model overfits on the most common shot used ignoring the others. Additionally, while with some parameter tuning it is possible to improve its performance on the training set after a while its accuracy on the test set doesn't improve anymore. The seq2seq was trained on 80% of the data and tested on the remaining 20%. We have used the Adam Optimizer with a learning rate of 0.0001. The cosine similarity score is a metric used to measure how much two vectors are similar to each other.

### Vision transformer

To give an idea of the model performance we have shown below its performance after training it on 90% of the data and tested and the remaining 10%. It reached the performances shown in 4.5.

Table 4.5: precision, recall and f1-score of the Vision Transformer.

| Class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Long Shot (LS) | 88% | 90% | 89% | 136 |
| Medium Shot (MS) | 87% | 68% | 76% | 127 |
| Full Figure (FF) | 74% | 83% | 78% | 108 |
| Half Torso (HT) | 70% | 78% | 74% | 167 |
| Half Figure (HF) | 71% | 73% | 72% | 132 |
| Detail (D) | 91% | 98% | 94% | 60 |
| American Shot (AS) | 67% | 83% | 74% | 94 |
| Close Up (CU) | 87% | 62% | 72% | 173 |
| Extreme Close Up (ECU) | 83% | 89% | 86% | 118 |
| Trash (TR) | 100% | 100% | 100% | 10 |
| accuracy | | | 79% | 100 |
| macro avg | 82% | 83% | 82% | 100 |
| weighted avg | 80% | 79% | 79% | 100 |

At a first is look seems a decent performance, but far from ideal. However if we take a look at the confusion matrix 4.6we can see that the majority of the mistakes are made among similar classes, hence the performance is actually good. If every now and then a half torso get mistaken for a half figure is not too bad.

Table 4.6: Confusion matrix of the Vision Transformer.

| Class | LS | MS | FF | HT | HF | D | AS | CU | ECU | TR |
|---|---|---|---|---|---|---|---|---|---|---|
| LS | 123 | 11 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| MS | 13 | 86 | 22 | 0 | 1 | 1 | 4 | 0 | 0 | 0 |
| FF | 1 | 2 | 90 | 1 | 0 | 1 | 13 | 0 | 0 | 0 |
| HT | 0 | 0 | 0 | 131 | 32 | 0 | 1 | 3 | 0 | 0 |
| HF | 1 | 0 | 1 | 12 | 97 | 0 | 21 | 0 | 0 | 0 |
| D | 1 | 0 | 0 | 0 | 0 | 59 | 0 | 0 | 0 | 0 |
| AS | 0 | 0 | 9 | 0 | 7 | 0 | 78 | 0 | 0 | 0 |
| CU | 0 | 0 | 0 | 43 | 0 | 2 | 0 | 127 | 21 | 0 |
| ECU | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 105 | 0 |
| TR | 123 | 11 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |

The dataset used to fine-tune the model is composed as follows:

- Long Shot: 1359 samples. The humane figure occupies one less than third in height or is absent.
- Medium Shot : 1270 samples. The humane figure is framed entirely and occupies from 2/3 to 1/3 of the screen in height.
- Full Figure: 1080 samples. The humane figure is framed entirely or almost and occupies more than 2/3 in height.
- Half Torso: 1673 samples. The humane figure is framed from half the torso up.
- Half Figure: 1315 samples. The upper half of the humane figure is framed.
- Objects(Details) :609 samples. Shot that focuses on object or on hands holding objects
- American Shot: 935 samples. The humane figure is framed from above the knee to the hips.
- Close Up: 1731 samples. The humane figure is framed from above the shoulder up.
- Extreme Close Up: 1183 samples. The humane figure is framed from the chin up.
- Trash(Other/titles): 102 sample frames not belonging to the original scene. We need to recognize them in order to filter them out later

### Qualitative results

Here we analyze the quality of the editing sequences obtained under different points of view. In the first part we will see the difference between original vs reconstructed sequences. In the second paragraph we will analyze generated sequences from new scene descriptions. Finally in the last part we will show a practical example on how this methodology can be used. Since the editing sequences, reconstructed or not, are sequences of one hot categorical vector, we implement the following sketches to represent the different classes, to give a more visual understanding of the sequences. The class sketches are shown in 4.14.

### Reconstructed Sequences

Here we analyze how our methodology is able to reconstruct the original movie scene sequences. Our methodology achieved a 92% in terms of average silhouette score, which indicates that generally the sequences are close enough to the original

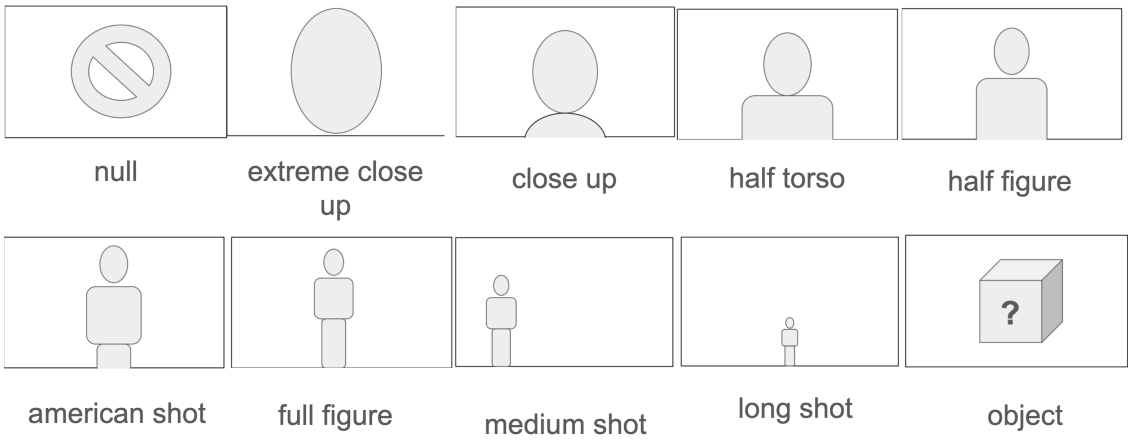



Figure 4.14: Shot icon representation

ones. In figure 4.15 we show three examples of reconstructed sequences: a short, a medium and a long one in terms of number of shots used.

As we can see the reconstructed sequences are a bit simpler than the original ones. In terms of general structure they match or come close to the original length except when the sequence is extremely long, like in the first case, in which the reconstructed sequence, while still long, is noticeably shorter than the other one. In terms of shot sizes used the reconstructed sequences manage to use the most used frames in the sequences. however when there are less common shots they usually fail to use them. This is probably due to the natural imbalance in the shot sizes distribution that the model has seen, since certain shots are more common than others. In figure 4.16 we show the first ten frames reconstructed using the sketches shown in 4.14.

The loss in granularity, although annoying, is to be expected since the scene textual descriptions, while good to get a general idea of what is happening, are not good to capture the details, and the consequent shots used to represent them. As a consequence the granularity of the results is also impacted by this factor.

### New editing sequences generation

Now we move to the generated sequences. We have created some brief textual description, then we have fed them to TEdit and have analyzed the results. In 4.17 we show the first ten shots that we receive as output after inputting the following text prompt:" Sarah bids farewell to her sister".

The complete resulting sequence, after removing the padding, is :

$$2-2-3-2-3-3-3-2-2-2-3-3-2-2-2-2-3$$

While the generated pattern per se is not particularly complicated or creative

```
tensor([3, 2, 3, 2, 2, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3, 4,
        2, 2, 3, 2, 2, 3, 3, 3, 2, 2, 3, 2, 2, 3, 3, 4, 3, 4, 3, 3, 1, 3, 2, 3,
        4, 2, 9, 4, 1, 1, 9, 2, 3, 1, 3, 2, 3, 3, 1, 3, 2, 3, 2, 3, 9, 2, 9, 3,
        3, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
tensor([3, 2, 2, 7, 2, 3, 2, 3, 1, 2, 2, 9, 2, 7, 2, 3, 2, 4, 3, 2, 4, 2, 5, 2,
        4, 3, 3, 2, 5, 3, 2, 2, 2, 3, 4, 6, 2, 5, 4, 5, 4, 4, 3, 3, 4, 2, 5, 2,
        3, 3, 3, 2, 4, 2, 3, 3, 3, 2, 3, 2, 9, 2, 3, 3, 2, 4, 3, 6, 3, 4, 2, 3,
        4, 2, 3, 3, 3, 2, 2, 2, 3, 7, 2, 3, 2, 2, 9, 3, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
tensor([2, 3, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
tensor([3, 3, 2, 3, 3, 3, 9, 2, 2, 2, 2, 9, 2, 2, 2, 3, 8, 4, 4, 9, 4, 4, 3, 3,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
tensor([3, 8, 4, 7, 9, 8, 8, 7, 7, 6, 3, 8, 8, 4, 2, 3, 8, 4, 9, 3, 3, 3, 2, 8,
        2, 2, 3, 3, 3, 4, 3, 2, 3, 2, 3, 3, 5, 3, 3, 7, 8, 7, 3, 7, 9, 2, 9, 7,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
tensor([3, 8, 1, 2, 9, 6, 8, 7, 8, 6, 8, 7, 3, 8, 2, 3, 8, 4, 9, 3, 7, 8, 2, 8,
        7, 3, 3, 5, 2, 4, 3, 2, 3, 2, 3, 3, 5, 9, 3, 3, 8, 1, 7, 1, 9, 1, 3, 7,
        9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0])
```

Figure 4.15: Examples of reconstructed sequences, on top the reconstructed on bottom the original.

the proposed shots, close ups and medium close ups, pertinent to the situation, an emotional moment. This is because ideally we would want the viewers to connect with the characters, hence shots that focus on their facial expression are a fit choice.

The quality of the generated sequences varies. In some cases it reaches a good level of detail, like in the next example. The input prompt is "the two armies battle each other in a final duel". The sequence, after removing padding, is:

9 – 8 – 9 – 9 – 8 – 8 – 9 – 8 – 2 – 8 – 8 – 8 – 8 – 2 – 8 – 8 – 8 – 8 – 2 – 9 – 2 – 9 – 2 – 9 – 2

The sketched sequence is shown in 4.18 .

Also in this case while the patterns are not particularly original the shots selected are consistent with what must be shown, since mostly is long shots and objects and some close ups to highlight key reactions. We can identify three segments in terms of type of shots used which we will visualize separately. Each segment can

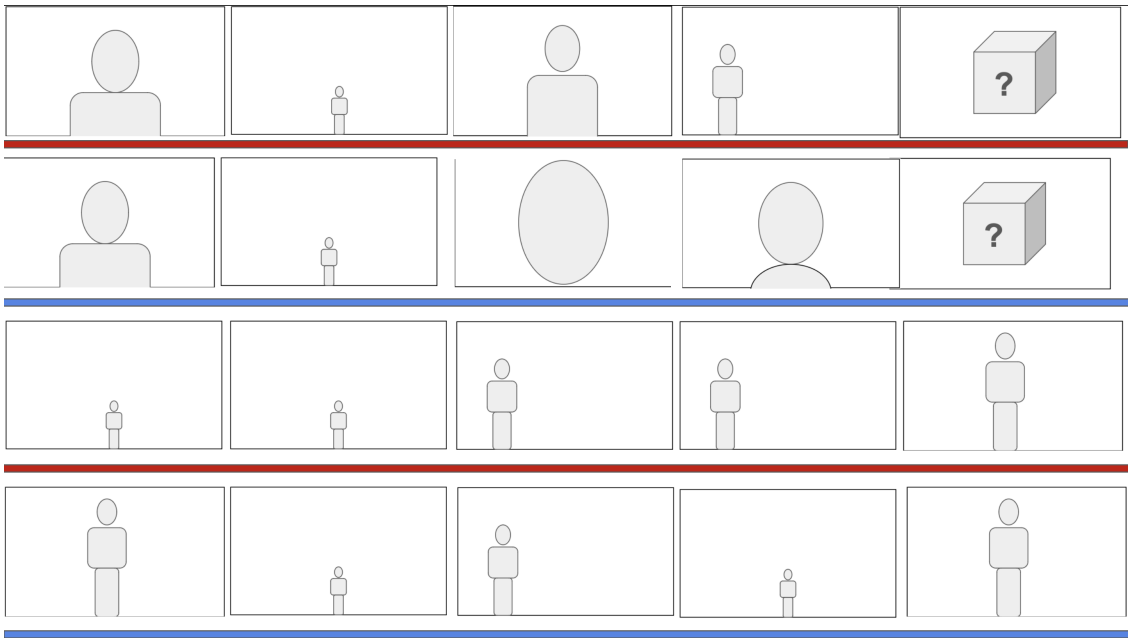


Figure 4.16: Examples of reconstructed sequences. On top the reconstructed on bottom the original.

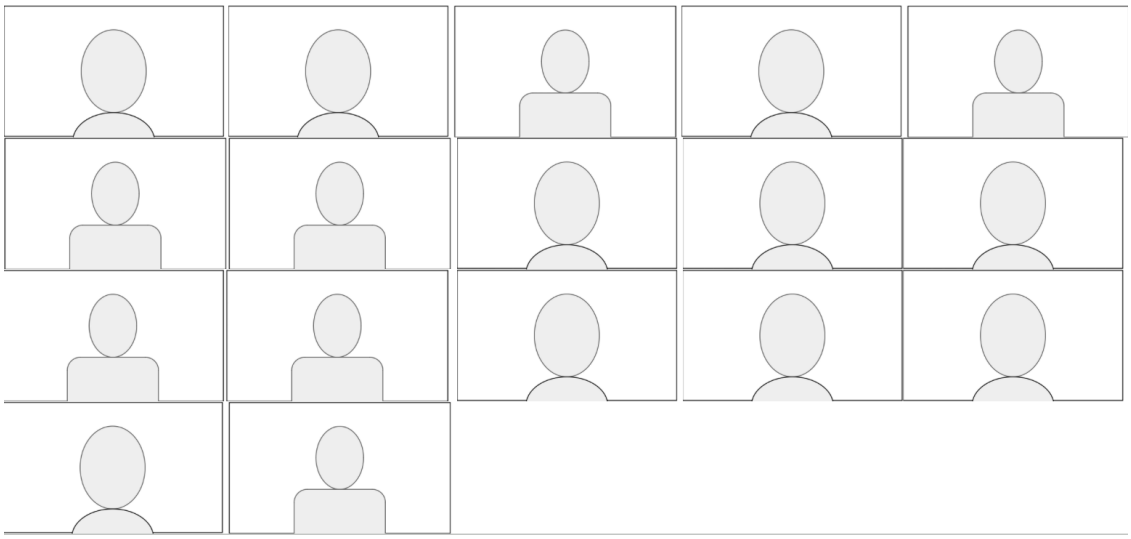


Figure 4.17: Generated editing sequence with the prompt "Sarah bids farewell to her sister"

be used to illustrate different moments of the battle. The first segment serves to introduce the environment, which can be done using long shots and objects shots to illustrate tools, weapons or people using them. The second one has mainly long shots and few close ups and can be used to show the moments in which the armies

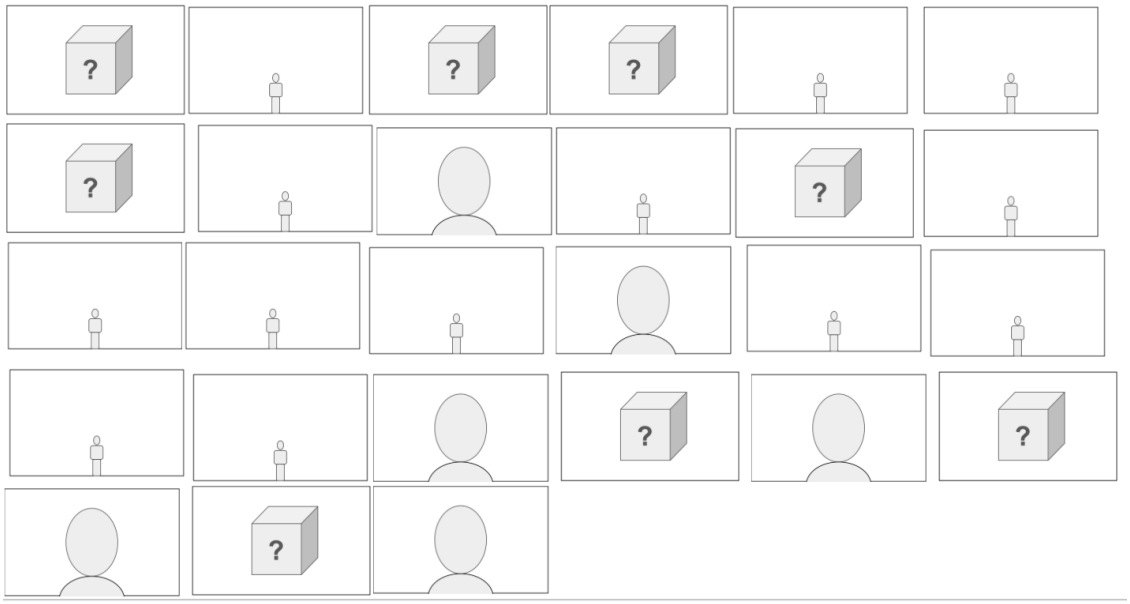


Figure 4.18: Generated editing sequence with the prompt "the two armies battle each other in a final duel".

clash. Finally in the third one we have close ups and objects, which can be used to represent a short duel between two characters.

### Specific use case example

In this last paragraph we show a potential use case for TEdit. In addition to the Generated sequence we are going to use a Stable Diffusion algorithm available at [72] to recreate the proposed sequences. The input prompt is "The hero and the antagonist fight each other in a final duel". The generated sequence is

$$7-8-9-1-8-8-1-1-8-1-1-8-1-8-1-8-4-1-1-8$$

In this sequence there are mainly long shots to show the two characters fighting and close ups to show their expressions. The sequence could be refined and use more medium range shots, however there is enough to start with. The sequence visualized in terms of frames is shown in 4.19.

Then we generated some prompts and converted them into images using the Stable Diffusion model. By substituting the images to the icons we get the sequence shown in figure 4.20.

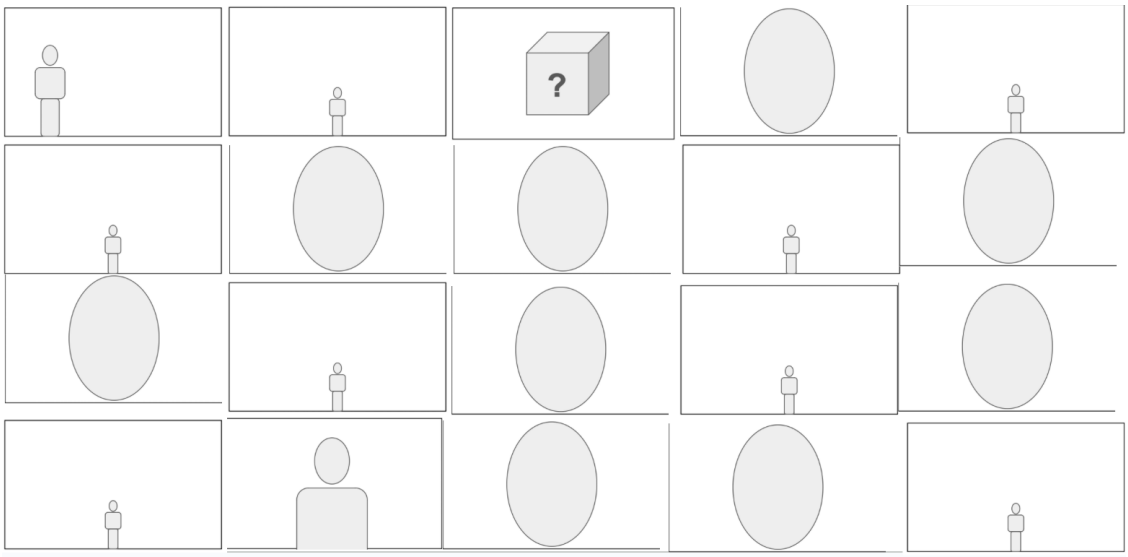



Figure 4.19: Generated editing sequence with the prompt "The hero and the antagonist fight each other in a final duel".

### 4.4.4 Discussion

Overall the results of TEdit are good, considering that it is a proof of work. The sequences both regenerated and created from new prompts overall have a solid structure, although they lose some detail in representing less common shots and present patterns that are not too complex. However several improvements can be made on the dataset and the methodology. From the methodology point of view, models more fit to deal with these data structures can be tested and integrated. From the data point we can integrate our data collection with videos coming from another dataset and extrapolate the textual scene description with Vision-Language models. One dataset that could used for the task is the newly released Panda 70 [18], a dataset with video and caption pairs. Even better, we could use movie scripts instead of textual prompts to generate more refined editing sequences. If we use more detailed textual descriptions we could be able to generate more sophisticated editing patterns. One additional improvement that can be made is to add additional shot sizes like the extreme long shot, aerial shot and others.

## 4.5 Chapter Conclusions

In this work we have addressed storyboards creation from textual data. Given a text description our methodology is able to give back a meaningful, although a bit coarse grained, editing sequence given a short textual description. The different contributions are summed up as follow:

Figure 4.20: Generated storyboard draft.

- We have a fine-tuned Vision transformer able to correctly classify images into shot classes.
- we have developed a methodology that can create new editing sequences
- we have shown how to improve the quality in recreating images with the shots constraint to recreate more cinematographic sequences.

The results provided in the last section of the previous chapter show that it is possible to generate meaningful sequences of shots. This work combined with Dreamshot allows us on one hand to recreate editing sequences from textual data, and on the other hand to have control on the generated images with the shot constraint size.

A further advantage of the methodologies presented here is that they can be integrated not only with text to image models, but also with more powerful text to video models like SoraAI or others. This is because the difference between storyboard and final video is the time dimension. Integrated with text to video model users could have a roughly edited movie from a simple prompt.

# Chapter 5

# Conclusions

The objective of this thesis was to investigate editing structures and their elements, the as shots, and to apply deep and machine learning algorithms to automate various video editing tasks. To tackle this complex research objective, we focused on addressing the following three challenges:

- **Challenge 1**: Extract editing sequences from videos;
- **Challenge 2**: Study the correlation between the editing structures and the corresponding videos;
- **Challenge 3**: Generate new editing patterns and storyboards;

.

For each challenge we have made one or more contributions.

**Main contributions addressing Challenge 1** To extract editing sequences from videos, it was essential to classify images into shot sizes, which are the fundamental components of these sequences. For this purpose, we created a shot size dataset containing 10,545 images categorized into eight shot classes. Additionally, we developed a methodology that achieved state-of-the-art performance in labeling new shots. This approach, based on VGG-16 models and ensemble learning, was later replaced by a more efficient Vision Transformer (ViT) model, which was unavailable at the start of this research. The overall accuracy of the ViT model is 80%. However, by disregarding errors between similar classes, which are inherently ambiguous and challenging to label accurately, the accuracy rises to 99

**Main contributions addressing Challenge 2** To explore the correlation between editing sequences and the corresponding videos, we initially developed a methodology that effectively grouped sequences with similar structures while analyzing the Cinsecale dataset, where editing sequences are defined solely by shot size. We later refined this methodology, achieving improved results with the AVE dataset, where shots and sequences are characterized by multiple shots. Both

methodologies yielded good results but also underscored the complexity of the task, indicating that future research will require new approaches and algorithms.

**Main contributions addressing Challenge 3** For the final research objective, we divided it into two tasks due to its complexity. The first task involved creating a methodology to generate single shots based on textual prompts and shot constraints, which outperformed the baseline model. The second task developed a methodology to generate meaningful, though somewhat coarse-grained, editing sequences from short text descriptions. These methodologies can also be integrated with text-to-image and advanced text-to-video models, such as SoraAI, enabling users to generate roughly edited movies from simple prompts.

In conclusion, this thesis has successfully addressed the complexity of automating video editing tasks by integrating deep and machine learning techniques. By achieving significant progress in extracting editing sequences, analyzing their correlations with corresponding videos, and generating new editing patterns and storyboards, we have laid a solid foundation for future research in this field.

Future research offers several options. Editing patterns can be further characterized by more shot size classes and other features, like camera movements. Other movie datasets can be integrated with our actual data collection, thanks to Vision language models and our methodology to label images into shot sizes. Among the other research directions that this field offers the most interest in is the integration of our proposed methodology to text-to-video models rather than limiting it to text-to-image models.

# Appendix A

# Short Sequence generation from AVE

## A.1 Introduction

Here we show some initial results on editing sequence generation. This research, although it showed good preliminary results, was later abandoned in favor of TEdit. The main reason is related to the fact that the AVE dataset characterizes in detail the shots used to represent a scene, but there is not much information concerning what is happening in the scene. When we decided to use the Condensed Movie Dataset, we had to discard this methodology in favor of TEdit, however few key concepts that were later integrated in TEdit come from this initial approach.

## A.2 Dataset

The starting dataset was the AVE dataset. To be more precise it was a version of sequenced AVE in which the segments were 10 seconds long instead of 30. Each shot in every editing segment is characterized by the shot size and the shot subject. We have chosen to reduce the length of the segments to reduce the problem complexity, as this was a preliminary study. Additionally instead of symbols we have decided to use a different data representation. Each shot now is a 5 dimensional vector, where each dimension represents a different shot subject. To represent the shot size we have used values from 0 to 1 to represent them. Additionally using LEMMS methodology we have labeled the segments into 8 classes. The number of reduced classes is related to the fact that with shorter sequences the variety of the sequences themselves diminishes.

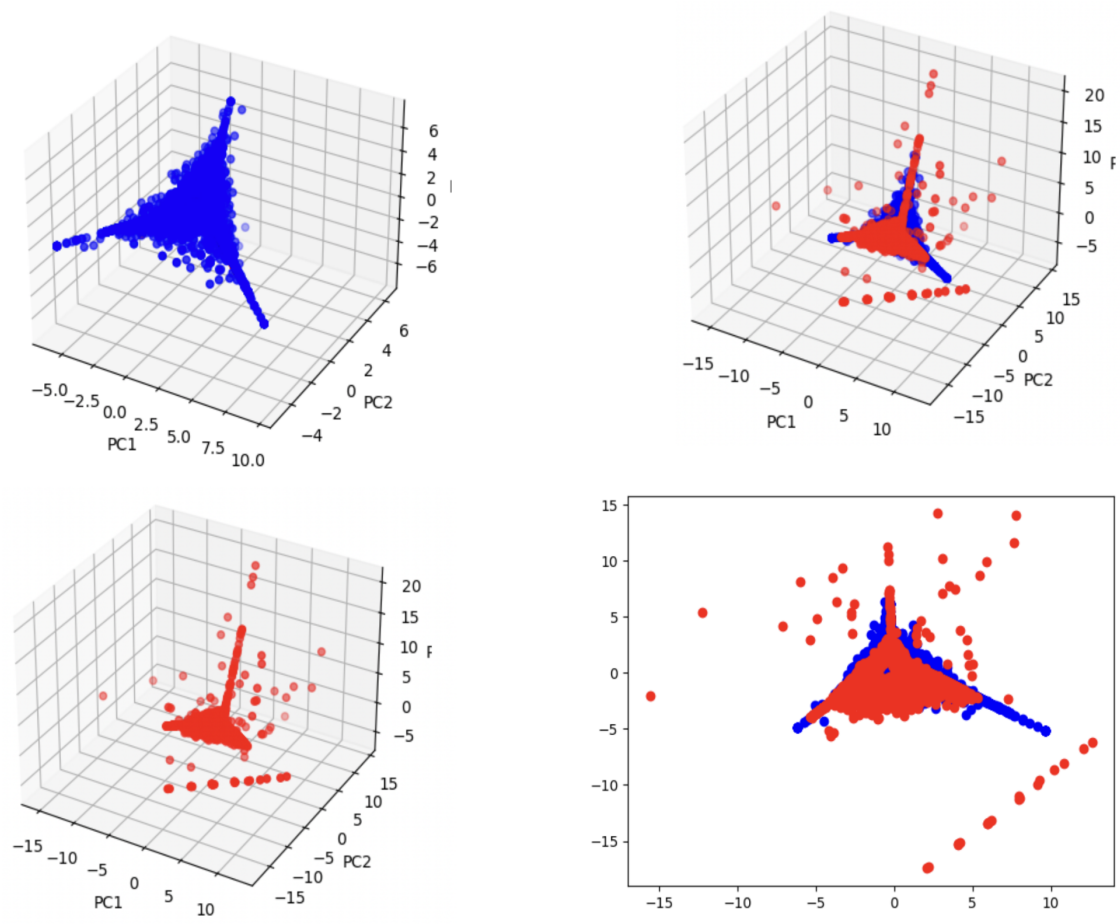



Figure A.1: Overall distribution, blu dots represent the original distribution, red dots the new one.

## A.3 Methodology

To recreate the sequences we decided to use a DoppelGANger model [46]. This model reached state of the art performance in recreating symbolic data, both synthetic and real. The model was trained on the newly formatted scene segments and their editing labels. Once that model has learned the data distribution is able to replicate it all or just parts of it, given as input to the editing label.

## A.4 Results

### Technical details

The model was trained for 500 epochs with a batch size of 32. The other parameters were the one pre-setted on the DoppelGANger model.

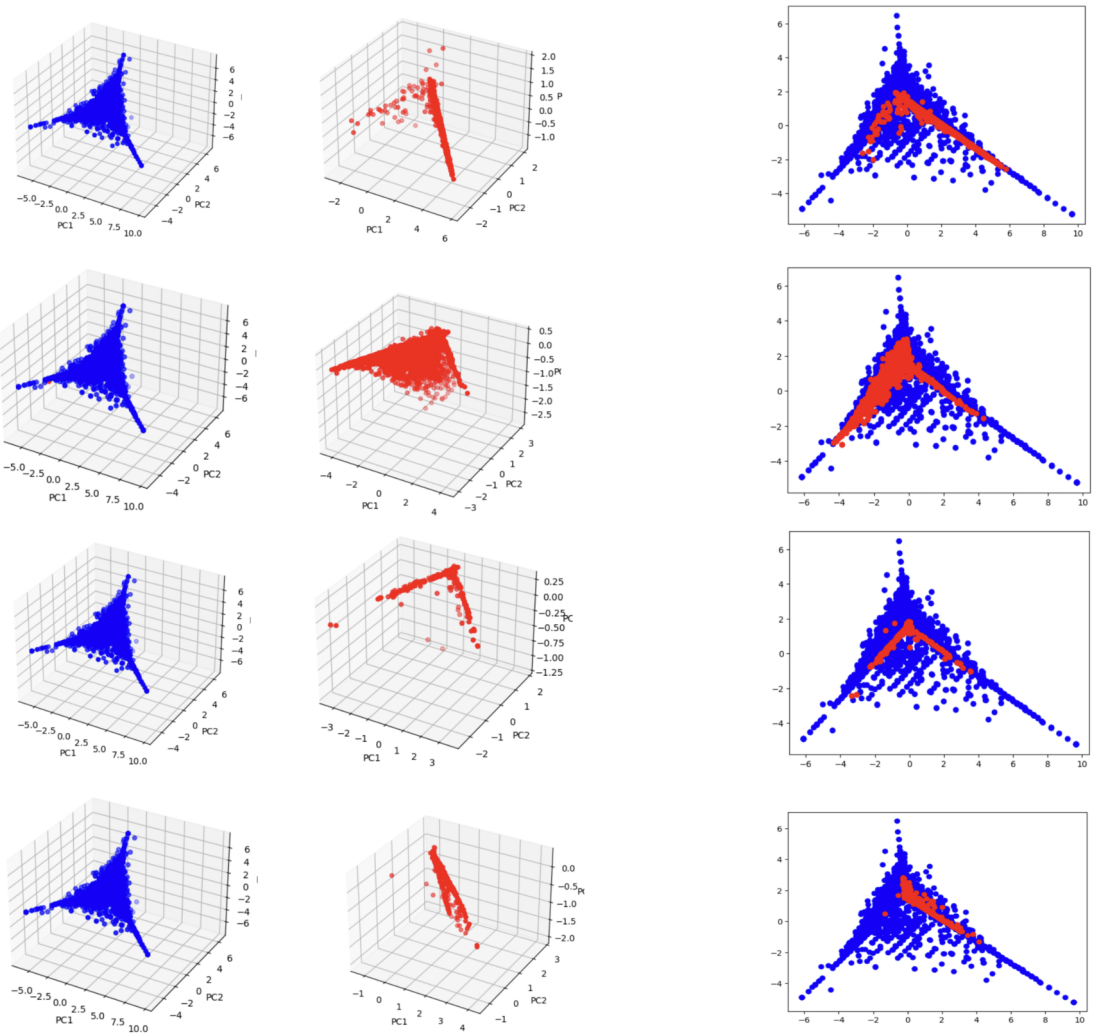



Figure A.2: 3D and 2D distribution of classes 0,1,2,3.

**performance**

To visualize how close the newly generated data distribution is to the real one we have used the PCA technique. In A.1 we show the overall distribution, while in A.2 and A.3 we show the results for the 8 different classes (grouped 4 by 4 for visualization purposes).

As we can see, depending on the granularity of the results varies depending on the amount of samples available in each class.

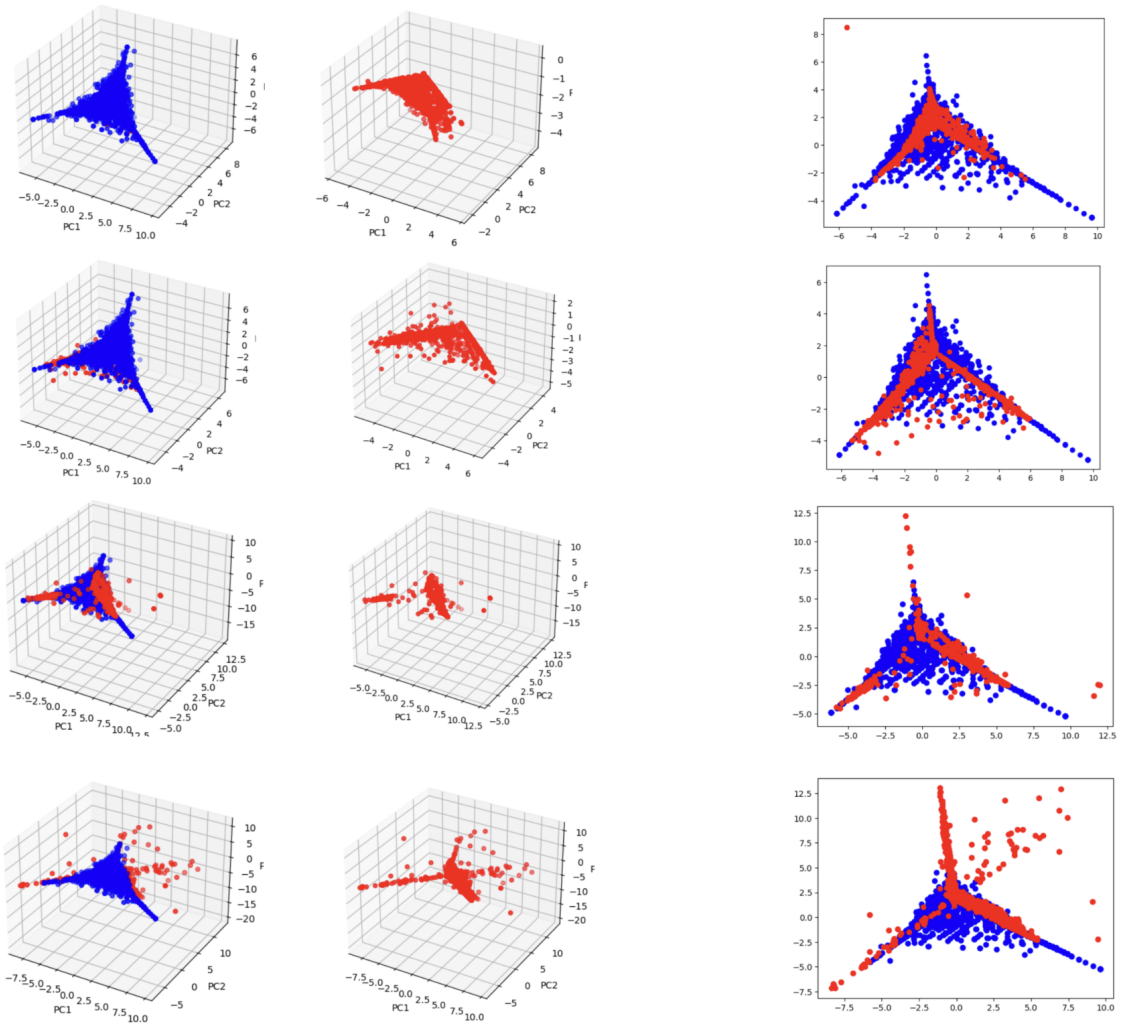



Figure A.3: 3D and 2D distribution of classes 4,5,6,7.

## A.5 Discussion

This approach, while not reaching outstanding performance, highlighted some successful strategies and criticism of the overall approach. The first useful insight comes from the data representation. Using a vector like representation instead of sequences of symbols greatly impacted the model understanding of the data representation. This also was kept to transform the sequences of shots into sequences of one hot categorical vector that TEdit uses. The second useful insight comes from the coarse granularity of the results. While one one hand it is true that the DoppelGANger model can be trained for more epochs with different parameters and so on there is only so much level of detail that can be reached with ten seconds long sequences characterized by only the shot size and the subjects if there no indication

of what is happening. These considerations on the data that we were using led us to use the Condensed Movie Dataset instead, which has the actual textual description of movie scenes, that offer much richer details to work with. For what concerns the DoppelGANger since it did not reach outstanding performance it was discarded in favor of a different approach. Additionally the model was very computationally expensive to train. This motivated us to rely instead on the interaction of simpler models.

# Bibliography


[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Islam. "The k-means Algorithm: A Comprehensive Survey and Performance Evaluation". In: *Electronics* 9 (Aug. 2020), p. 1295. DOI: 10.3390/electronics9081295.

[2] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[3] Dawit Mureja Argaw et al. "The Anatomy of Video Editing: A Dataset and Benchmark Suite for AI-Assisted Video Editing". In: *European Conference on Computer Vision*. 2022.

[4] David Arthur and Sergei Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245.

[5] Max Bain et al. *Condensed Movies: Story Based Retrieval with Contextual Embeddings*. 2020. arXiv: 2005.04208 [cs.CV].

[6] Hui-Yong Bak and Seung-Bo Park. "Comparative Study of Movie Shot Classification Based on Semantic Segmentation". In: *Applied Sciences* 10 (May 2020), p. 3390. DOI: 10.3390/app10103390.

[7] Hui-Yong Bak and Seung-Bo Park. "Comparative Study of Movie Shot Classification Based on Semantic Segmentation". In: *Applied Sciences* 10 (May 2020), p. 3390. DOI: 10.3390/app10103390.

[8] Sergio Benini et al. "On the influence of shot scale on film mood and narrative engagement in film viewers". English. In: *IEEE Transactions on Affective Computing* 13.2 (2022), pp. 592–603. ISSN: 1949-3045. DOI: 10.1109/taffc.2019.2939251.

[9] Matteo Berta, Bartolomeo Vacchetti, and Tania Cerquitelli. "GINN: Towards Gender InclusioNeural Network". In: *2023 IEEE International Conference on Big Data (BigData)*. 2023, pp. 4119–4126. DOI: 10.1109/BigData59044.2023.10386328.

[10] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. "Tools for placing cuts and transitions in interview video". In: *ACM Transactions on Graphics* 31 (July 2012), pp. 1–8. DOI: 10.1145/2185520.2335418.

[11] Paolo Bethaz et al. "Predicting job execution time on a high-performance computing cluster using a hierarchical data-driven methodology." In: *Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference*. Edinburg, UK: Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference, 2022.

[12] S. Bhattacharya et al. "Classification of Cinematographic Shots Using Lie Algebra and its Application to Complex Event Recognition". In: *IEEE Transactions on Multimedia* 16.3 (2014), pp. 686–696. DOI: 10.1109/TMM.2014.2300833.

[13] Digbalay Bose et al. "MovieCLIP: Visual Scene Recognition in Movies". In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022), pp. 2082–2091. URL: https://api.semanticscholar.org/CorpusID:253018714.

[14] Beatriz Bosques Palomo et al. "Sentiment Analysis of IMDB Movie Reviews Using Deep Learning Techniques". In: Jan. 2024, pp. 421–434. ISBN: 978-981-99-3235-1. DOI: 10.1007/978-981-99-3236-8_33.

[15] Luca Canini, Sergio Benini, and Riccardo Leonardi. "Classifying cinematographic shot types". In: *Multimedia Tools and Applications* 62 (2011), pp. 51–73.

[16] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *ArXiv* abs/1706.05587 (2017).

[17] Shixing Chen et al. "Movies2Scenes: Using Movie Metadata to Learn Scene Representation". In: 2022.

[18] Tsai-Shien Chen et al. "Panda-70M: Captioning 70M Videos with Multiple Cross-Modality Teachers". In: *ArXiv* abs/2402.19479 (2024). URL: https://api.semanticscholar.org/CorpusID:268091168.

[19] I. Cherif, V. Solachidis, and I. Pitas. "Shot type identification of movie content". In: *2007 9th International Symposium on Signal Processing and Its Applications*. 2007, pp. 1–4.

[20] Robin Courant et al. "High-Level Features for Movie Style Understanding". In: *ICCV 2021 - Workshop on AI for Creative Video Editing and Understanding*. online, France, Oct. 2021, pp. 1–5. URL: https://hal.science/hal-03381587.

[21] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[22] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

[23] Evelina Di Corso et al. “Simplifying Text Mining Activities: Scalable and Self-Tuning Methodology for Topic Detection and Characterization”. In: *Applied Sciences* 12.10 (2022). ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/12/10/5125.

[24] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[25] “DreamShot: Teaching Cinema Shots to Latent Diffusion Models”. In: *CEUR WORKSHOP PROCEEDINGS* 3651 (2024). ISSN: 1613-0073.

[26] *FILMGRAB*. https://film-grab.com.

[27] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”. In: *CoRR* abs/1811.12231 (2018). arXiv: 1811.12231. URL: http://arxiv.org/abs/1811.12231.

[28] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *Neural Information Processing Systems*. 2014. URL: https://api.semanticscholar.org/CorpusID:261560300.

[29] Salvatore Greco et al. *Unsupervised Concept Drift Detection from Deep Learning Representations in Real-time*. 2024. arXiv: 2406.17813 [cs.LG]. URL: https://arxiv.org/abs/2406.17813.

[30] Igor S. Gruzman and Anna S. Kostenkova. “Algorithm of scene change detection in a video sequence based on the threedimensional histogram of color images”. In: *2014 12th International Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*. 2014, pp. 1–1. DOI: 10.1109/APEIE.2014.7040826.

[31] Rishin Haldar and Debajyoti Mukhopadhyay. “Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach”. In: *Computing Research Repository - CORR* (Jan. 2011).

[32] Bharath Hariharan et al. “Hypercolumns for Object Segmentation and Fine-grained Localization”. In: (Nov. 2014).

[33] M. A. Hasan et al. “CAMHID: Camera Motion Histogram Descriptor and Its Application to Cinematographic Shot Classification”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.10 (2014), pp. 1682–1695. DOI: 10.1109/TCSVT.2014.2345933.

[34] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778. URL: https://api.semanticscholar.org/CorpusID:206594692.

[35] J. Edward Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *ArXiv* abs/2106.09685 (2021). URL: https://api.semanticscholar.org/CorpusID:235458009.

[36] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

[37] Qingqiu Huang et al. "MovieNet: A Holistic Dataset for Movie Understanding". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 709–727. ISBN: 978-3-030-58548-8.

[38] Hestry Humaira and Rasyidah Rasyidah. "Determining The Appropiate Cluster Number Using Elbow Method for K-Means Algorithm". In: Jan. 2020. DOI: 10.4108/eai.24-1-2018.2292388.

[39] Kunal Jani et al. "Machine learning in films: an approach towards automation in film censoring". In: *Journal of Data, Information and Management* 2 (Mar. 2020). DOI: 10.1007/s42488-019-00016-9.

[40] Levon Khachatryan et al. "Text2Video-Zero: Text-to-Image Diffusion Models are Zero-Shot Video Generators". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 15908–15918. DOI: 10.1109/ICCV51070.2023.01462.

[41] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013). URL: https://api.semanticscholar.org/CorpusID:216078090.

[42] Seong-Ho Lee, hye yeon yu hye yeon, and Yun-Gyung Cheong. "Analyzing Movie Scripts as Unstructured Text". In: Apr. 2017, pp. 249–254. DOI: 10.1109/BigDataService.2017.43.

[43] Bruno Lepri et al. "The Tyranny of Data? The Bright and Dark Sides of Data-Driven Decision-Making for Social Good". In: *Transparent Data Mining for Big and Small Data*. Ed. by Tania Cerquitelli, Daniele Quercia, and Frank Pasquale. Cham: Springer International Publishing, 2017, pp. 3–24. DOI: 10.1007/978-3-319-54024-5_1.

[44] Junnan Li et al. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models". In: *International Conference on Machine Learning*. 2023. URL: https://api.semanticscholar.org/CorpusID:256390509.

[45] Yitong Li et al. "StoryGAN: A Sequential Conditional GAN for Story Visualization". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 6322–6331. URL: https://api.semanticscholar.org/CorpusID:54457433.

[46] Zinan Lin et al. "Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions". In: *Proceedings of the ACM Internet Measurement Conference*. IMC '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 464–483. ISBN: 9781450381383. DOI: 10.1145/3419394.3423643. URL: https://doi.org/10.1145/3419394.3423643.

[47] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv* abs/1907.11692 (2019). URL: https://api.semanticscholar.org/CorpusID:198953378.

[48] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[49] Robert Logan et al. "Deep Convolutional Neural Networks With Ensemble Learning and Generative Adversarial Networks for Alzheimer's Disease Image Data Classification". In: *Frontiers in Aging Neuroscience* 13 (2021). ISSN: 1663-4365. DOI: 10.3389/fnagi.2021.720226. URL: https://www.frontiersin.org/article/10.3389/fnagi.2021.720226.

[50] Yuya Matsuo, Miki Amano, and Kuniaki Uehara. "Mining video editing rules in video streams". In: Jan. 2002, pp. 255–258. DOI: 10.1145/641007.641058.

[51] Eyal Molad et al. "Dreamix: Video Diffusion Models are General Video Editors". In: *ArXiv* abs/2302.01329 (2023).

[52] Fionn Murtagh and Pierre Legendre. "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?" In: *Journal of Classification* 31 (2011), pp. 274–295. URL: https://api.semanticscholar.org/CorpusID:7134583.

[53] Christine Nothelfer, Jordan DeLong, and James E. Cutting. "Shot Structure in Hollywood Film". In: 2009.

[54] Rising Odegua. "An Empirical Study of Ensemble Techniques (Bagging, Boosting and Stacking)". In: Mar. 2019.

[55] Xuran Pan et al. "Contrastive Language-Image Pre-Training with Knowledge Graphs". In: *ArXiv* abs/2210.08901 (2022). URL: https://api.semanticscholar.org/CorpusID:252917745.

[56] Alejandro Pardo et al. "Learning to cut by watching movies". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6858–6868.

[57] Alejandro Pardo et al. “Moviecuts: A new dataset and benchmark for cut type recognition”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 668–685.

[58] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation”. In: *ArXiv* abs/2102.12092 (2021). URL: https://api.semanticscholar.org/CorpusID:232035663.

[59] Anyi Rao et al. “A Unified Framework for Shot Type Classification Based on Subject Centric Lens”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 17–34. ISBN: 978-3-030-58621-8.

[60] Anyi Rao et al. “Dynamic Storyboard Generation in an Engine-based Virtual Environment for Video Production”. In: *ACM SIGGRAPH 2023 Posters*. SIGGRAPH ’23. Los Angeles, CA, USA: Association for Computing Machinery, 2023. ISBN: 9798400701528. DOI: 10.1145/3588028.3603647. URL: https://doi.org/10.1145/3588028.3603647.

[61] Jian Ren et al. “Best Frame Selection in a Short Video”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 3201–3210. DOI: 10.1109/WACV45572.2020.9093615.

[62] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

[63] Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 10674–10685. URL: https://api.semanticscholar.org/CorpusID:245335280.

[64] Nataniel Ruiz et al. “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 22500–22510. URL: https://api.semanticscholar.org/CorpusID:251800180.

[65] Mattia Savardi et al. “CineScale: A dataset of cinematic shot scale in movies”. In: *Data in Brief* 36 (2021).

[66] Mattia Savardi et al. “Shot Scale Analysis in Movies by Convolutional Neural Networks”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 2620–2624. DOI: 10.1109/ICIP.2018.8451474.

[67] Mattia Savardi et al. “Shot Scale Analysis in Movies by Convolutional Neural Networks”. In: Oct. 2018, pp. 2620–2624. DOI: 10.1109/ICIP.2018.8451474.

[68] Ketan Rajshekhar Shahapure and Charles Nicholas. "Cluster Quality Analysis Using Silhouette Score". In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. 2020, pp. 747–748. DOI: 10.1109/DSAA49011.2020.00096.

[69] Gabriel Simões et al. "Movie genre classification with Convolutional Neural Networks". In: July 2016, pp. 259–266. DOI: 10.1109/IJCNN.2016.7727207.

[70] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[71] Uriel Singer et al. "Make-A-Video: Text-to-Video Generation without Text-Video Data". In: *ArXiv* abs/2209.14792 (2022).

[72] *Stable Diffusion model avaliable at hugging face*. https://huggingface.co/spaces/stabilityai/stable-diffusion.

[73] *Storyboarder*. https://wonderunit.com/storyboarder/.

[74] *Storyboarder*. https://storyboarder.ai.

[75] *Storyboardthat*. https://www.storyboardthat.com/.

[76] *Studiobinder*. https://www.studiobinder.com/storyboard-creator/.

[77] M. Svanera et al. "Over-the-shoulder shot detection in art films". In: *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*. 2015, pp. 1–6.

[78] Michele Svanera et al. "Who is the Film's Director? Authorship Recognition Based on Shot Features". In: *IEEE MultiMedia* 26.4 (2019), pp. 43–54. DOI: 10.1109/MMUL.2019.2940004.

[79] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[80] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[81] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[82] Bartolomeo Vacchetti and Tania Cerquitelli. "Cinematographic Shot Classification with Deep Ensemble Learning". In: *Electronics* 11.10 (2022), p. 1570.

[83] Bartolomeo Vacchetti and Tania Cerquitelli. "Movie Lens: Discovering and Characterizing Editing Patterns in the Analysis of Short Movie Sequences". In: *European Conference on Computer Vision*. Springer. 2022, pp. 660–675.

[84] Bartolomeo Vacchetti, Tania Cerquitelli, and Riccardo Antonino. "Cinematographic Shot Classification through Deep Learning". In: *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2020, pp. 345–350. DOI: 10.1109/COMPSAC48688.2020.0-222.

[85] Bartolomeo Vacchetti, Dawit Mureja, and Tania Cerquitelli. "LEMMS: Label Estimation of Multi-feature Movie Segments". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3027–3035.

[86] Bartolomeo Vacchetti et al. "JEM: An AI-based engine workflow to predict simulation's execution time on HPC cluster". In: *2024 International Conference on Control, Automation and Diagnosis (ICCAD)*. IEEE. 2024, pp. 1–5.

[87] Francesco Ventura, Tania Cerquitelli, and Francesco Giacalone. "Black-Box Model Explained Through an Assessment of Its Interpretable Features". In: *New Trends in Databases and Information Systems - ADBIS 2018 Short Papers and Workshops, AI*QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings*. 2018, pp. 138–149.

[88] Austin Walters. *Sentence Classification*. URL: https://github.com/lettergram/sentence-classification.

[89] H. L. Wang and L. Cheong. "Taxonomy of Directing Semantics for Film Shot Classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 19.10 (2009), pp. 1529–1542. DOI: 10.1109/TCSVT.2009.2022705.

[90] Chao-Yuan Wu and Philipp Krähenbühl. "Towards Long-Form Video Understanding". In: *CoRR* abs/2106.11310 (2021). arXiv: 2106.11310. URL: https://arxiv.org/abs/2106.11310.

[91] Hui-Yin Wu et al. "Joint Attention for Automated Video Editing". In: *ACM International Conference on Interactive Media Experiences*. IMX '20. Cornella, Barcelona, Spain: Association for Computing Machinery, 2020, pp. 55–64. ISBN: 9781450379762. DOI: 10.1145/3391614.3393656. URL: https://doi.org/10.1145/3391614.3393656.

[92] Ali Yazdizadeh, Zachary Patterson, and Bilal Farooq. "Ensemble Convolutional Neural Networks for Mode Inference in Smartphone Travel Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), pp. 2232–2239.

[93] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. "Adding Conditional Control to Text-to-Image Diffusion Models". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 3813–3824. URL: https://api.semanticscholar.org/CorpusID:256827727.

[94] Howard Zhou et al. "Movie Genre Classification via Scene Categorization". In: Oct. 2010, pp. 747–750. DOI: 10.1145/1873951.1874068.

[95] Jian Zhou and Xiao-Ping Zhang. "Automatic Identification of Digital Video Based on Shot-Level Sequence Matching". In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*. MULTIMEDIA '05. Hilton, Singapore: Association for Computing Machinery, 2005, pp. 515–518. ISBN: 1595930442. DOI: 10.1145/1101149.1101265. URL: https://doi.org/10.1145/1101149.1101265.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.