KU LEUVEN

Efficient analysis of mobile eye tracker data using Deep Learning

Jesus Garcia Ramirez

Thesis submitted for the degree of Master of Science in Artificial Intelligence, option Engineering and Computer Science

Thesis supervisor:
Prof. dr. Johan Wagemans

Assessor:
Dr. Robin De Croon, Dr. Lore Goetschalckx

Mentor:
Dr. Christophe Bossens

Academic year 2019 – 2020

Preface

To my parents, Jesus and Manuela
For their unwavering support and endless love.

Jesus Garcia Ramirez

Contents

Abstract

Understanding how people navigate and interact with their environment is an important research question for various disciplines (e.g. psychology, cognitive sciences, social sciences). One important mechanism linked to a person's experience is attention. One of the possible strategies for capturing the attentional processes of a person is by studying her eye gaze patterns. Recently, the appearance of mobile eye trackers (MET) has been presented as a possible step forward towards bridging research with the "real-world". Through the utilization of these devices, researchers can record the eye gaze patterns and the point of view of an individual while exploring freely the environment. Although such devices might hold considerable potential, their adoption in the research community has been hindered by some practical limitations. One of the main practical issues is the cost of manually labelling the eye-tracking obtained with employment of MET.

In our work, we wish to provide some automatic labelling tools that aim to alleviate, as much as possible, the overall manual workload needed to process eye-tracking data. In particular, we have worked with a set of mobile eye-tracking recordings that captured how a series of participants navigated through a prestigious art exhibition. We can consider diverse aspects of interest when analysing the obtained recordings. Hence, the first contribution in our work has been to translate these different focuses of analysis into a structured data pipeline. Our presented pipeline consists of three stages: *Video Segmentation*, *Behaviour Segmentation* and *Image Registration*. To automate each stage, we have translated their respective requirements into machine learning problems. To solve such machine learning problems, we have employed deep learning techniques due to their flexibility and lack of domain knowledge.

Finally, we have provided practical approaches for the Video Segmentation and Behaviour Segmentation tasks. For the former stage, we have proposed a video classification model to predict the item seen by the participant in any given segment. We see how the results obtained with our approach are satisfactory, except for a couple of minor limitations. In the Behaviour segmentation stage, we have proposed two different types of models: one using video input and another using time-series data (accelerometer, gaze and gyroscope data) to predict the behaviour taken by a participant in any given segment. Finally, we will see in the results how this task is more cumbersome than the video segmentation task and we will discuss the strengths and limitations of the proposed models.

List of Figures and Tables

List of Figures

List of Tables

Chapter 1

Introduction

Understanding how people navigate and interact with their environment is an important research question for different disciplines (e.g. psychology, cognitive sciences, social sciences). People can interact with their environment in diverse and complex ways. One important mechanism linked to a person's experience is attention. In short, attention acts as a selective mechanism that links the individual's current incoming stimuli with her current and future actions. One of the possible strategies for capturing the attentional processes of a person is by studying her eye gaze patterns. To put it another way, through monitoring what and where a person is looking at a certain period of time we can infer some information about the person's attentional focus during such interval of time.

Traditionally, researchers conducted experiments in which they simulated one aspect of interest of a human-environment interaction under laboratory conditions. Take the experimental set-up in [29] as illustration. In this study, the authors wanted to study how people enjoy art by studying the eye-gaze patterns of a number of participants when looking to different art-pieces. To answer such research question, the authors presented the different pieces in a computer screen and recorded the gaze patterns by means of a stationary eye tracker. The results provided by experiments such as this one might yield valuable insights to understand specific interactions between an individual and her environment under constraint situations. However, the changing and complex characteristics of the world we inhabit might limit the generalization capabilities of such works. Recently, the appearance of mobile eye trackers (MET) has been presented as a possible step forward towards bridging research with the "real-world". The standard mobile eye tracker device consists of a set of cameras embedded into eyeglasses that record both the individual's point of view and his eye gaze coordinates through time.

Although mobile eye trackers might seem to offer researchers the possibility of studying "real-life" behaviours, there are some practical limitations that limit the use of MET devices in research (see [17] for a detailed overview). One of the main issues when applying mobile eye trackers in research is the cost that such devices add to the experimental set-up. The cost of purchasing a mobile eye tracker can be considered low (between a few hundred euros for an open-source system and around €20k for a commercial system). However, the cost of purchasing one of these devices can end up being negligible in comparison with how much it costs to manually label the eye-tracking data. Unfortunately, the lack of reliable

automatic annotation tools force researchers to have to manually analyse the majority of eye-tracking data obtained. The time needed to process an eye-tracking recording of, for example, 10 minutes, could easily be on the order of multiple hours depending on the specific research question (e.g. see [3]).

In our work, we wish to provide with some automatic labelling tools that aim to alleviate, as much as possible, the overall manual workload needed to process eye-tracking data. In particular, we have worked with a set of mobile eye-tracking recordings that captured how a set of participants navigated through a prestigious overview exhibition at the city museum M in Leuven of the art work by Pieter Vermeersch (°1973, Kortrijk, Belgium). In this exhibition, the artist tastefully experimented with the relationships between the classical artworks (marbles, paintings, murals, ...) and the architectural components (additional walls, windows, mirrors, ...) to create a powerful spatial experience where the distinctions between 2D and 3D, material and immaterial, and time and space become blurred.

We can consider diverse aspects of interest when analysing the obtained recordings. For instance, we can focus on processing what the participant is viewing at any particular moment. Moreover, we could also analyse how the participant is interacting with the presented pieces inside the exhibition. Hence, the first contribution in our work has been to translate these different focuses of analysis into a structured data pipeline (discussed in Chapter 2). This pipeline consists of three stages: *Video Segmentation*, *Behaviour Segmentation* and *Image Registration*. Each of the presented stages will aim to provide processed data to answer different questions about the way each participant navigates the exhibition. While the proposed pipeline is specific to our project, we have modeled each stage in such a way that they could be used for similar eye-tracking projects.

Finally, we will provide practical approaches for the Video Segmentation (Chapter 3) and Behaviour Segmentation (Chapter 4) tasks. In the former stage, we have formulated the problem of segmenting each recording depending on the particular item the individual is viewing at any given moment in time into a video classification problem. Then, we have implemented a video classification model to solve the task. We will show in the results section how the results obtained with our approach are satisfactory, except for a couple of minor limitations. In the Behaviour segmentation stage, we have formulated the problem of detecting the different behavioural patterns by each participant into a classification problem. We have proposed two different types of models: one using video input and another using time-series data (accelerometer, gaze and gyroscope data). Finally, we will see in the results how this task is more cumbersome than the video segmentation task and we will discuss the strengths and limitations of the proposed models.

Chapter 2

Pipeline Overview

Mobile eye trackers provide us with a vast quantity of unstructured data (e.g. gaze, video, accelerometer, etc). While a trained annotator can make sense of it in its raw format, this is far more difficult for any hypothetical computer model. To make the problem feasible for a computer we propose a structured data pipeline (see Figure 2.1).

In this pipeline, each of its steps filter the given data with a double goal. First, each step processes the incoming data in a way that make it suitable for the next step. Second, the output of each step will be a meaningful piece of information by itself. Next, we present a brief overview of each of the steps of the pipeline:

- **Video Segmentation**: The goal of this step is to identify the segments of each recording where a participant is viewing a particular piece of artwork. Each of the segments gives us a coarse view of the time spent by a participant on the different parts of the exhibition. Recall from the introduction the discussion about the fact that the artist played in the exhibition both with classical pieces of art and architectural elements. In the course of this work, we will put our focus on analysing the behaviours that the participants perform when viewing any of the classical pieces of art. From now on, we will categorize such segments as art-segments. Therefore, this step will also serve to filter out for the next stage the clips where the participant is not looking at any particular piece of art (e.g. is looking to other person, to a particular wall or mirror and so on). Note, that while not considered in this work, some of the examples previously discussed are also considered to be an essential part of the exhibition.

- **Behavior Segmentation**: The goal of this step is to detect different behavioural patterns inside each of the discussed art-segments. In this work, we have a taxonomy of possible behaviours that an individual can manifest during the exhibition (see Appendix A, note that this Appendix also contains the non-art gaze behaviours). These behaviours offer a more fine-grained overview of the individual's interaction with the environment.

  If we take a close look to the proposed taxonomy, we can make a distinction amongst behaviours where the participant is either looking actively to a piece or not. Through an analysis over the gaze patterns of the former group, we can obtain the visual

attention response of the individual towards a particular piece. This analysis will be conducted in the image registration step.

- **Image Registration**: Mobile eye trackers provide us with the gaze coordinates over the individual's coordinate system (with respect to the participant's point of view). This coordinate system is changing continuously with every movement of the person wearing the glasses. If we want to analyse the gaze patterns with respect to a particular piece, we need to have the gaze coordinates with respect to a static coordinate system. Hence, the goal of this step is to provide for this mapping between the gaze direction in the participant's coordinate system to the piece's coordinate system.

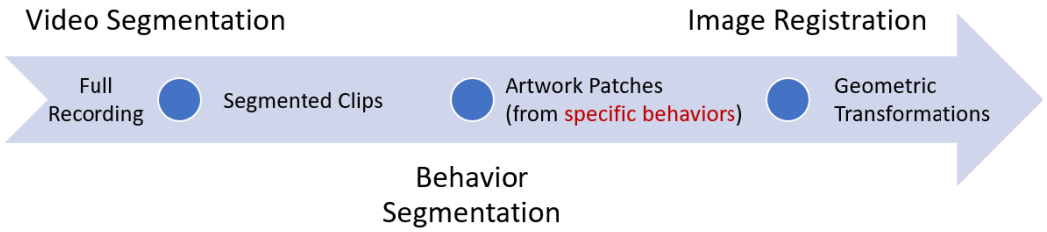

Figure 2.1: Information flow for the proposed pipeline

Although we present the idea of a three-stages pipeline, the focus of this work is to model and implement just the first two stages. While the technical details for each implementation will be discussed in Chapters 3 and 4, we need to have the intuition behind the implemented models. Both video and behaviour segmentation can be seen as classification problems over raw signals. When working with raw signals, we need to extract features from them before feeding to a classifier to obtain a decent performance. The feature extraction part can be solved using either hand-engineered approaches or deep learning approaches:

- In **hand-engineered** approaches we apply domain knowledge to extract informative features from raw signals using data mining techniques. While there are numerous successful applications of hand-engineered features in the literature, we need to start from domain knowledge and we have to manually change the approach depending on the problem at hand.

- In **deep learning** approaches we use artificial neural networks that learn to extract features depending on the data we feed them. The advantage of this approach is the generality of the models used. If we use the same data format, we can use the same model to predict different kind of outputs. For example, we can use the video recordings to segment both the artworks seen and the behavioural patterns. This flexibility has motivated the choice of the use of deep learning for the proposed models in this work.

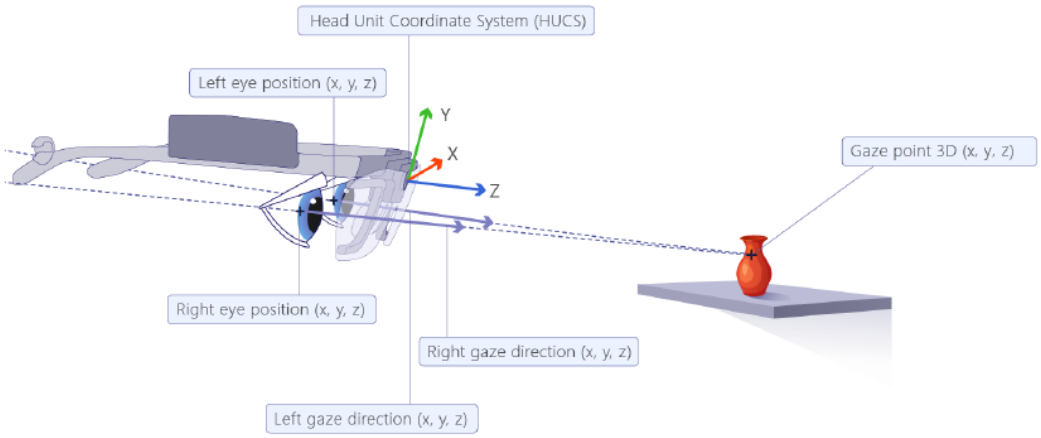

Figure 2.2: Glasses' coordinate system. The gaze point is calculated as the intersection point between the two gaze vectors (taken from Tobii's user manual)

Chapter 3

Video Segmentation

The goal of this stage is to segment each recording depending on the particular item the individual is viewing at any given moment in time. When solved by a human annotator, this task can be performed using the full recording at once. Furthermore, we can obtain annotated segments of arbitrary length within a margin of error of less than seconds. Although this would be the ideal segmentation case, working with a continuous stream complicates greatly the task at hand. Hence, a first step to make the task manageable will be to split the recordings into fixed-length clips. Then, we will try to classify the object seen inside of each clip. In this scenario, we reduce a continuous (segmentation) problem into a discrete (classification) problem.

Note that the idea of fixed-length clips may seem unnatural at first, but this step makes solving the task at hand feasible for a computer model. The idea of fixing the length of the processed clips is given by the fact that we do not have any information beforehand about the amount of time each participant spends viewing a given item at the exhibition. Moreover, this length involves a trade-off between how fine-grained we are able to segment the full recordings and the computational power needed. Finally, after having classified every clip, we would convert such clips into continuous segments.

3.1 Literature Review

Video classification has been an active focus on research for the last ten years or more. Videos are collections of frames stacked one after the other in the temporal axis. Each of the constitutive frames has their own semantic information which can be obtained by exploiting the spatial relationships over their pixels. Hence, individual frames constitute an important source of information by themselves. However, we would miss the whole story if we did not take into account the temporal relationships among the collection of frames. We can make a division of the proposed models in the literature based on whether they use only individual frames or they consider the video as a whole.

3.1.1 Single-frame models

The fact that individual frames of a video already carry descriptive information together with the fact that there exists a plethora of high-performance image classification models in the literature, incited researchers to employ image classification models for the video classification task providing the minimal changes necessary to make such models suitable for working with video. The predominant architectures for the image classification task are convolutional neural networks (CNNs; e.g. ResNet [15], Inception [38]) which have convolutional filters as their basic block. A convolution filter constitutes a window by which we examine a subset of the image. In a convolutional layer, we have a number of such filters that we scan over the entire image to generate feature maps. Each point inside these feature maps retain spatial information over an specific region of the previous layer. This characteristic, make convolutions ideal for working with images.

Convolutional Layers + LSTM

The most straightforward way to employ an image classifier to work with videos is to consecutively run a number of frames into the same classifier and pooling the predictions [21]. In other words, we select as the video label, the majority class from the individual frames' predictions. While this strategy is easy to implement and computationally-effective, it completely ignores the temporal structure of the video.

To address possible temporal relationships inside each video when adopting an image classifier model, several authors have proposed to add a recurrent layer to the model (e.g. [7]. In this case, we change the final layer of the classifier (classification layer) for a recurrent layer. Therefore, with this modified model, the convolutional layers extract spatial features from each frame and the recurrent layer processes such features across time. The most employed kind of recurrent layer in the literature is the Long Short-Term Memory (LSTM) layer [18]. This layer has a hidden internal stage that encodes temporal relationships. This hidden stage is updated every time-step with information from the current inputs and the state from the previous time-step.

Finally, in the training stage, we obtain the most probable label at each time-step with which we can compute a cross-entropy loss. The cross-entropy loss function compares how similar the output distribution of our model is with respect to the real labels. The feedback provided by this loss will serve as information to update the parameters of our model. The sum of all the cross-entropy losses across time will provide the model feedback to update its parameters. During testing time, we would randomly sample a number of frames and will use the most probable class for the last LSTM cell as the predicted label. We use the last LSTM cell as its hidden states has encoded the information for all the fed frames.

Two-Stream Networks

Another approach was introduced by Simonyan and Zisserman [31] using a two-stream architecture (see Figure 3.1). Here, they augment the spatial information of every RGB frame with motion information provided by 10 optical flow frames. Optical flow refers to the visible motion of an object in an image. It is obtained by comparing the pixels from two consecutive frames and computing the displacements of each pixel from one frame to

another. While there is a branch in the literature that employs optical flow information, we have decided not to explore further this branch since computing and storing the optical flow for every frame of our clips will impose a high computational cost.

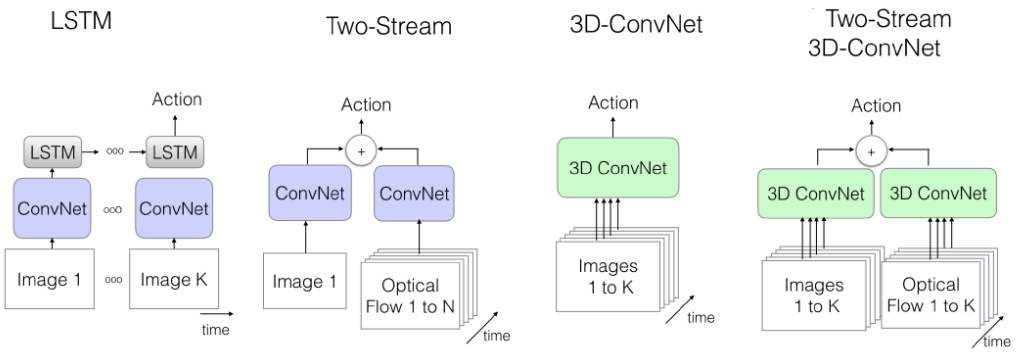

Figure 3.1: Summary of the main video architectures presented in the literature. Figure modified from [4].

3.1.2 Video models

Although single frames from a clip contain some information, this information is normally not enough for correctly predicting every possible clip. Therefore, the natural step for this field was to build models that could explicitly process a video as a set of connected frames in time and not a number of independent frames.

3D Convolutions

3-Dimensional convolutional filters are presented as a natural alternative to process a set of frames. In this case, we add one dimension to the standard convolution that filters the input along the temporal axis. One important aspect of 3D Convolutions is that they are able to learn hierarchical representations of spatio-temporal data [4]. They have been explored in various occasions ([39],[40]), but they were not further explored for a while since the extra dimension of the convolutional filters increments considerably the number of parameters of such models. This high number of parameters made them unfeasible to train from scratch without an extensive enough dataset (which were not available until the appearance of Kinetics-400 [22] counting with over 400k labeled clips).

Inflated 3D Convolutions

Carreira and Zisserman proposed in [4] a clever way to train 3D-Convnets from scratch. First, they "inflated" 2D architectures into 3D models. They did so by adding every convolutional filter for this architecture with an additional temporal dimension. In this way, the square filters of K x K were augmented into cubic 3D filters of K x K x K. Moreover, it is standard practice for 2D CNNs to pretrain their weights on ImageNet dataset [6] and

transfer the obtained weights to train in the final dataset. Given the size of the ImageNet dataset (over 1M training images), the pretraining process offers a considerable boost of performance even for small datasets. Therefore, the authors wanted to use the pretrained weights of ImageNet for their 3D convolutions. To do so, note that an image can be converted into a video just by stacking a number of copies of the same frame. In this case, we can see that the obtained video would be the video that we would obtain if we film the same still scene from the same exact point of view. Therefore, in order to bootstrap the ImageNet parameters, the authors proposed to repeat the weights of the 2D convolutions N times along the temporal dimension and to scale the obtained weights just dividing by N. In this work, the authors propose various ways of combining the inflated 3-D convolutions (I3D) depending on the input data (RGB frames, optical flow frames or a combination of both).

SlowFast

Feichtenhofer et al. proposed a new architecture, SlowFast [8], inspired by some biological studies about the human visual system. It has been shown that our visual system is composed of two pathways that process the incoming visual input with different purposes: Ventral and Dorsal. The ventral stream is believed to mainly process visual shapes and objects, whereas the dorsal stream has been primarily associated with locating and guiding how to use the seen objects. Based on this intuition, they proposed a two-stream architectures with different temporal resolution. The slow path is designed to process semantic information that change slowly with time. In contrast, the fast pathway is designed to process rapidly changing motions. For example, if we had a clip of a person throwing an object, the information related with the objects would be processed through the slow path while the information about the motion of the objects would be processed through the fast pathway. Finally, the authors added lateral connections from the fast pathway to the slow pathway to provide the slow pathway with feedback from the motion of the seen objects.

3.2 Proposed Approach

Given the number of available models in the literature, the task of choosing a suitable model for our task may seem daunting. Nevertheless, the task can be simplified by transforming the project requirements into specific design constraints.

The first constraint to bear in mind is the format of the annotations we will be working with. While a human annotator could label the frames individually, this is not the most optimal scenario. Instead, it is more straightforward for the annotator to simply specify when the individual started and ceased viewing a particular piece. Although we could assume that all the frames contained in a labeled segment correspond to the same label of this segment, this is a crude approximation in reality. For example, it often happens that a participant turns his head to look at a companion or his mobile phone while looking at a piece. These events correspond to a non negligible number of frames within a fragment labeled to a specific piece. Note that although the same argument holds for the case of the label of each fixed-length clips, we might encounter three different artworks in the same clip at most. On the other hand, there are 300 frames in a 10s clip. Thus, we can

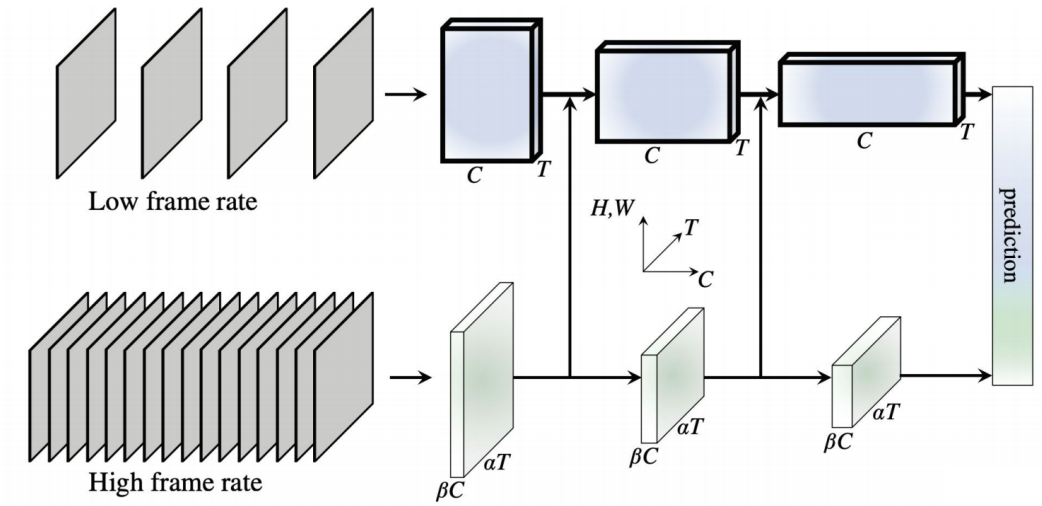

Figure 3.2: Slowfast architecture. Figure taken from [8]

have frames in the order of ten misslabeled (e.g. in case of spending 2s looking at other person we would end with 60 misslabeled frames). Hence, while the scenario is not ideal for either single-frame and video models, we consider the use of a video model to be a better approximation to reality than single-frame models.

One of the main goals of this work is to implement a model that provides reliable annotations in a faster and cheaper way than a human annotator. Hence, the computational needs of the chosen model is a determinant factor. As we discussed in section 3.1, the classification performance of a video model is often related to its complexity. Often, the more complex the model, the more computational power it needs. Based on the trade-off between classification performance and computational budget discussed, we have selected the I3D video model with a Resnet-50 backbone as our implemented model.

3.2.1 Implementation Details

Given the relatively small size of our dataset ($\sim 10k$ samples) it would be unfeasible to obtain high performances if we trained our models from scratch. Hence, we have decided to initialize the weights of our models using the ones obtained for the same architecture when trained on the Kinetics-400 dataset. All experiments were run using the Gluon-CV [12] framework. Next, we will present the specific configurations used to train and evaluate the proposed models.

- We trained our models using Google Colab platform (see technical specifications on Appendix C). Using this platform, we had free access to GPUs for training.

- We trained each model for 100 epochs with a mini-batch size of 8. We had to limit the mini-batch size given the memory capacity of the used GPU and how much memory we require to store each clip during training.

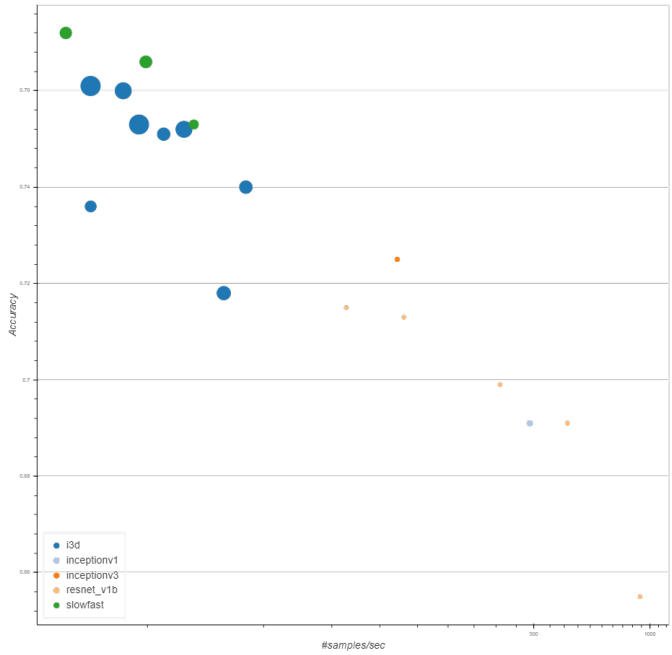

Figure 3.3: Comparison of trade-off of accuracy versus inference speed for different video classification models in Kinetics-400 dataset. Note that the diameter of each circle corresponds to the memory size of its correspondent model.

- We optimized our models using Stochastic Gradient Descent (SGD) with momentum. SGD is an iterative optimization method, that will update the parameters of our model to minimize the value of the loss function. In short, this method starts from the initial conditions of the loss function (given by the initial values of the models' weights) and travels down its slope in small steps until it reaches the optimal (lowest) point of this function.

  We set the momentum term to 0.9 and weight decay to $1e-4$. To help convergence, we used the warm restarts schedule for the learning rate (Lr) described on [26]. Using this schedule, the learning rate at the n-th iteration is $\eta * 0.5 \left[\cos(\frac{T_{cur}}{T_0}\pi) + 1\right]$ with η as the base learning rate and T_{cur} as the number of epochs performed since the last restart. Moreover, we multiply the basic period (T_0) at each restart by a factor of T_{mult}. In our experiments, we have set T_0 to ten and T_{mult} to two given the results observed in [26]. Finally, we have trained the model using three different base learning rates $(0.01, 5e-3, 1e-4)$.

- We fed our model with 32 randomly frames from each clip both at training and inference times.

- Data augmentation is known to improve the training of deep architectures while also reducing overfitting. For the *training* stage, we have employed the data augmentation strategy proposed in [41]. Following this strategy, we fix the input image size as

224x224 and randomly sample the cropping width and height from 224,196,168,148. Next, we crop the four corners and the center of each frame. After that, we resize the cropped regions to the original size. Finally, we perform the standard per channel normalization. During *inference* time, we just crop the center of each frame and perform per channel normalization.

3.3 Dataset

In this section, we will give a comprehensive overview of the dataset we have used to train and test our models. We will describe the process of building the dataset, its statistics and possible characteristics and limitations. This overview will provide a more fine-grained understanding of the task at hand. Furthermore, it will serve as useful context to better understand the results of next section.

3.4 Dataset Building

We counted with full recordings of 74 participants. The duration of each recording lies between 10 to 30 minutes on average. Transforming the annotated recordings into a structured dataset suitable for this stage is a fundamental step which we describe below.

Firstly, we have transformed each recording into a series of non-overlapping fixed-length clips. We have set the length of each clip to 10s as it is the length of the videos used to train on Kinetics dataset. Next, we had converted the temporal annotations into individual labels for each clip. We have used a simple max pooling approach for this step. With this approach, we will label a clip as the class that appears the longest during this clip. For example, if we have a clip where the participant looked at a piece for 7s and then looked 3s at a different piece, we set the first piece as the label for the clip.

The goal of this work sets a special requirement on how to arrange the processed clips. In this work, we wish to implement a model that has an appropriate classification performance using a reduced number of recordings. Hence, we have divided the full dataset into two separate subsets:

- **Benchmark subset**: Subset containing the recordings from the first 30 participants. We will use this subset to explore the performance of the proposed models with a reduced number of participants, as it would happen in a real-life scenario. We will further partition this subset into three disjoint splits: Train, validation and test. The **train** split will be used to train the different models presented. The **validation** split will be employed to select the best hyperparameter combination for the selected models. Finally, the **test** split will be used in the end of the process to assess the final models' predictive performance.

  It is important to note that each split will be done with respect to the participants and not the clips. In other words, all the samples of a participant can only belong to one of the three splits. In this way we simulate the operation in reality, where a human annotator would annotate the recordings of certain participants and then our model would try to annotate the rest of unseen participants.

- **Active Learning subset**: In situations as our own, in which unlabeled data is abundant but manual labeling is expensive, learning algorithms can actively query the user for labels. This type of learning is known as active learning. To simulate an active learning situation, we have divided the remaining 48 participants into 6 groups of 8 participants each. We will detail how we employed the obtained groups to simulate such scenario and the results obtained in section **??**.

3.5 Classes Overview

We want to identify the object primarily being looked at by the participant for each clip. Since the participants are walking inside an art exhibition, one group of classes could correspond to a particular piece of artwork. On the other hand, there are other groups of clips where the participant is not looking at any particular piece of artwork (e.g. looking at his mobile phone, at other person, etc).

We have sixteen different labels in the first group, one for each different piece of artwork. The possible artworks can be either two-dimensional paintings or three-dimensional sculptures. We can assume that the clips in this group will share some properties. First, we will encounter low intraclass variance. In other words, different clips where the same piece is displayed will not look too dissimilar. Moreover, we can assume the interparticipant variance to be low for this group. This means that samples from different participants looking at the same painting will also look similar. This is a very desirable property in our case, since it will help the generalization capabilities of our models with respect to unseen participants.

Participants can look to a large number of objects apart from the discussed pieces of art inside the gallery. However, the only relevant information from these clips is that the participant is not looking to any particular piece. Hence, we will group all these clips inside the same category: "Null". In this case, the properties shared by the samples inside this group are different from the previous group. First, we cannot assume that we have low intraclass variance since we have too many different possibilities inside the same group. On the other hand, we can still assume low interparticipant variance since the array of possible objects that a participant can watch is similar for all participants.

3.6 Handling Imbalance

Since there is an array of different visual stimuli inside the gallery it would be natural that not all the possible stimuli are viewed for the same amount of time. This fact introduces an added level of difficulty to our problem: *Class Imbalance*. We encounter class imbalance when there are one or more classes seriously over-represented with respect of the classes. This imbalance is detrimental for any possible machine learning model, since it biases the classifier towards the majority classes. Hence, we have to address this problem before training any possible model. This topic has been widely studied in the literature (see for example [23]). The possible approaches can be grouped in two big groups: Data level and Algorithmic level approaches.

Figure 3.4: Comparison frames from an example artwork (top row) and null class (bottom row). We can see graphically how diverse the samples from the null class can be with respect to samples from an artwork class.

- In the **data level** approaches, we modify the distribution of the dataset by sampling methods. Using sampling methods, we can either have more samples from the unrepresented classes (upsampling), or we can reduce the number of samples from the majority classes (undersampling).

  Undersampling techniques may cause a loss of information on the distribution of the majority class since we discard a considerable number of examples of them.

  On the other hand, upsampling techniques are more computationally expensive and more prone to overfitting [23]. One of the standard upsampling methods used in the literature is SMOTE [5]. In SMOTE we generate synthetic samples of the minority classes by sampling from a randomly chosen point and its nearest neighbors in the feature space. While this technique is widely used in the literature and provides good results, we would need a generative model that is able to generate synthetic images from features (e.g. if we had a Generative Adversarial Network (GAN) [10] model). This requirement is an added level of difficulty that could only be justified if simpler methods did not yield satisfactory results.

- In the **algorithmic level** approaches, we modify the classification models to take into account the imbalance distribution of the dataset directly. We will give a brief overview in this chapter. The reader can find a more detailed discussion of this approach in section 4.3 to tackle the imbalance on the behaviours segmentation dataset. One classical example is cost-sensitive learning where we modify the weights of the loss function to take into account the imbalance of the dataset directly. We

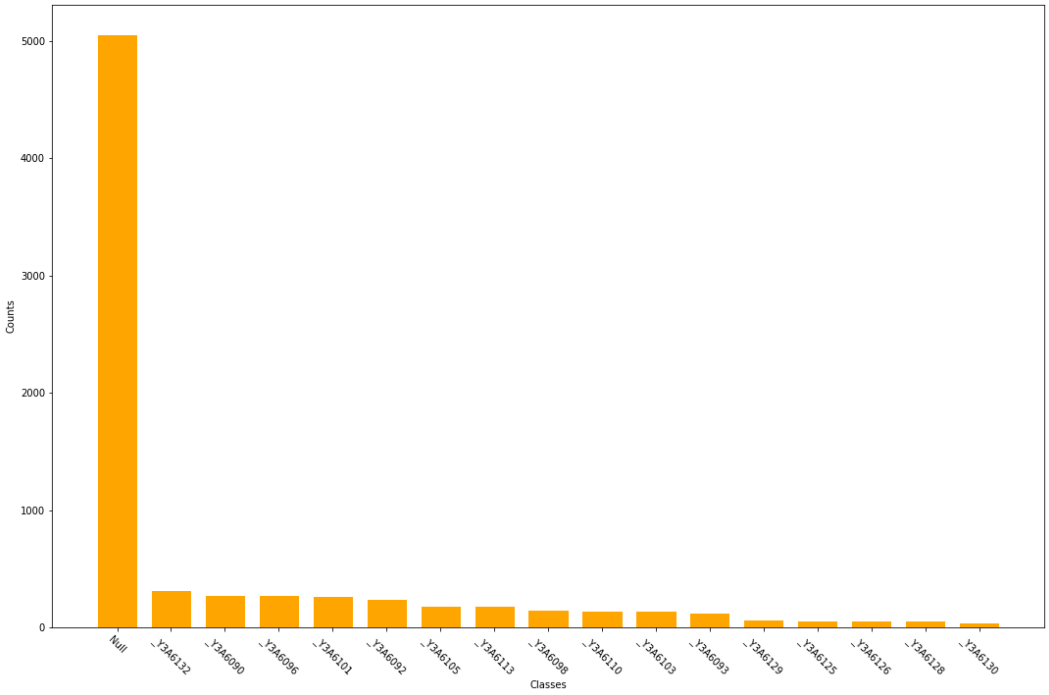

Figure 3.5: Overview of total number of samples per class in the original video segmentation dataset

penalize each misclassification mistake according to the representation of the class to which each mistake belongs. A more sophisticated strategy is metric learning [2], where we train a model to convert raw inputs into features that share a semantic structure. Since the features share a semantic structure, it makes the samples from the same class easier to separate by a classification model.

We have chosen to apply a simple random undersampling over the Null class to balance the dataset. In this approach, for each of the participants, we have chosen randomly as many samples for the Null class as for the majority class in the art group for this particular participant. The choice of this strategy for handling imbalance is motivated by two factors. First, the computational efficiency of the proposed approach is a determinant factor in our work. Therefore, using a downsampled dataset is the best option in terms of efficiency. Finally, we have chosen to only downsample the number of samples from the null class since it is the class that presents the most considerable imbalance (see Figure 3.5). Note that, in theory, discarding samples from the "Null" class could suppose in a considerable loss of information as this class is the most heterogeneous. However, we will show in the results section how this "naive" approach offers competitive results.

3.7 Results

In this section, we will compare and analyze the obtained results for the different models and scenarios proposed. If we want to obtain informative results we first need to choose suitable performance metrics to assess the performance of the implemented models. The choice of such metrics will be guided by the kind of task solved and the requirements of such task. Therefore, we will first provide a comprehensive overview of the metrics chosen and then we will report and analyze the obtained results.

3.7.1 Performance metrics

We are solving a classification problem in this stage of the pipeline. However, our problem is slightly different from the standard classification set-up. In the standard scenario, for each input sample, the model will output an array of confidences where the elements of the array correspond to the possible labels in our dataset. To convert an array of confidences into a prediction, we select the element of the array with the highest confidence and we assign its correspondent label to the sample. On the other hand, our approach differs from the standard in the fact that we only label a sample in case the element of the array with the highest confidence for that sample is above a confidence threshold previously imposed. Such confidence threshold establishes a trade-off between how many samples the model annotates and the quality of such annotations. In theory, the higher the threshold the more accurate the predictions should be but the fewer samples that the model actually labels.

There exist a large number of classification performance metrics, each one summarizing a different aspect of the performance of a model. Hence, to narrow the possible choices, we need to work with the specific requirements of our classification problem. Next, we will describe the chosen metrics together with the specific requirement aspect they summarize:

- **Coverage**: Fraction of number of samples labeled by the model with respect to the total samples presented. It gives a general overview of how confident our model is. If the model is highly confident, it will output a high number of confidence arrays where its highest confident element will lay above the threshold. Note that this metric does not provide any information about the quality of the model predictions. In order to have this information, we should compute the next two classification metrics.

- **Precision**: Precision answers the question: Out of the items that the classifier predicted to be from one class, how many are actually from that class?. This metric can be interpreted as the probability that a sample labeled as one class by the classifier is actually from that class.

  Note that in the multiclass scenario, we have as many precision scores as possible classes exist in our dataset. Therefore, to compute the final precision value, we have computed a weighted average of the scores using the number of samples per class as weights.

- **Matthews correlation coefficient** (MCC): We will present the intuition for this metric in the binary classification problem. The extension to the multiclass scenario

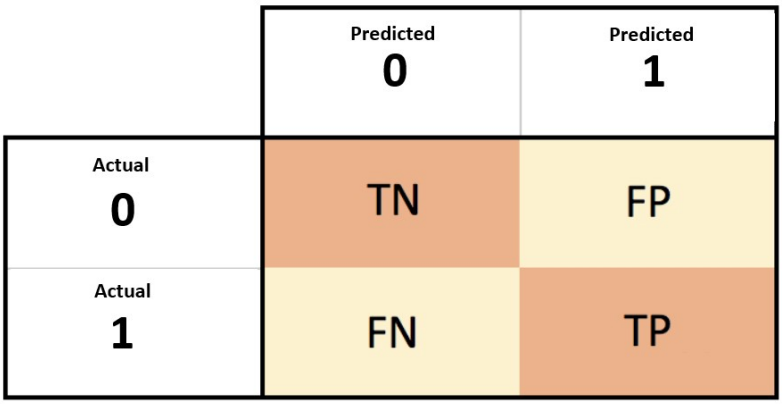

Figure 3.6: Example of a confusion matrix for a binary classification problem

will correspond to a mathematical extension with the same reasoning as in the binary case. To compute this metric, we treat the true class and the predicted class as two (binary) variables. Then, we compute their correlation coefficient. The higher the correlation between true and predicted values, the better the prediction. The attractive quality of this metric is that it takes into account all of the values of the confusion matrix (see Figure 3.6). Hence, this metric will only yield high values when the performance of the classifier is high for all the possible classes. Finally, we present the mathematical formulation for the binary case:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.1}$$

The model will output different predictions depending on the threshold set. Hence, we can compute a different value the proposed metrics for each of the possible threshold values. With the purpose of reporting the evolution of such metrics for the different thresholds we propose two curves: MCC-treshold curve and Precision-Coverage curve. The former will inform us about the overall classification performance of our model for the different possible thresholds. The latter will report on the already discussed trade-off between quantity (*coverage*) and quality (*precision*) of the model's predictions.

In order to have a quantitative comparison among the different models proposed, we could summarize each of the discussed curves into a number. The most straightforward way to summarize a curve is to compute the Area Under the Curve (AUC). The AUC value of a curve gives a general overview of the performance of the model regardless of the threshold used. This will be useful in the case that the same model will be employed in different tasks with different constraints.

3.7.2 Benchmark results

In this section, we will provide an structured summary of the results obtained for the different experiments performed. We will start comparing the performance of the model for the different learning rates proposed. Using this information, we will select the best model

and we will assess its definitive performance using the test set. Finally, we will discuss how the performance of the trained model is affected when presenting with a simulated active learning scenario.

Learning Rate selection

We compare the results for the three configurations discussed in section 3.2.1 both numerically (Table 3.1) and graphically (Figure 3.7). We can make some observations with the presented data. First, the performance of the model is rather similar for the three learning rates presented. Moreover, we can see how the presented models achieve considerable performance. Finally, we have chosen the model with Learning Rate of 0.1 as our definitive implementation given its slightly better AUC scores for the two curves studied.

Analysing in more depth the results shown in Figure 3.7 can provide us with interesting insights about the possibilities of the presented approach:

- **Precision-Coverage curve**: Two noteworthy mentions can be made in this case. First, we can see how the proposed models achieve a worst-case scenario precision (when we label every sample) higher than 80%. In other words, in case we let our model label every possible sample, we would obtain correct predictions in 80% of the cases. Second, this curve does not present a considerable slope. This is preferable to a considerable slope since the less slope this curve presents the larger number of samples can be labeled with high precision.

- **MCC Curve**: We can see in this graph how the overall quality of output predictions is related to the threshold set. For the chosen model, we can see a worst-case scenario score of 0.78 (i.e. the predictions of the model and the actual labels are highly correlated). Finally, we can observe lower values for this score than for the precision score found in the PC-curve. This was predictable, since as we already discussed in section 3.7.1, obtaining high scores for this metric is more difficult as it also take into account the class imbalance.

Learning Rate (Lr)	AUC_{MCC}	AUC_{PC}
1e-4	0.792	0.895
5e-3	0.776	0.898
0.01	**0.806**	**0.904**

Table 3.1: Comparison classification performance for the different learning rates tested. Best results presented in bold.

Final Results

In this section, we will discuss the results for the selected model when employed to predict the samples from the test set. These results are the most relevant in our case as the model did not receive any feedback during training from the test set. On the other hand, although

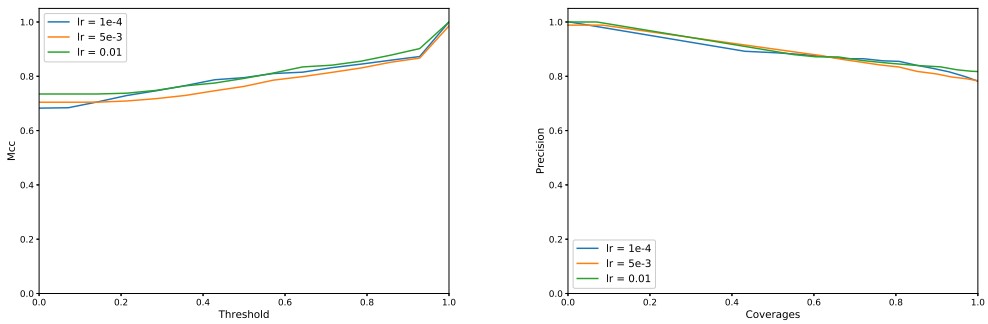

Figure 3.7: **Mcc-threshold** (left) and **Precision-Coverage** (right) curves evaluated on the validation split for the three different learning rates tested.

the validation set was not use to train the model, it was used to provide feedback of the model's performance during training. Hence, the results for the former set provide a less biased report of the model performance than the latter. The AUC scores for this experiment correspond to the *Base* iteration shown in Figure 4.14. We can see how the model provides almost the same performance for both validation and test sets. This result could be expected given the discussed low interparticipant variance that this dataset presented.

Figure 3.8 presents two confusion matrices for the test set. On the left, we can see the confusion matrix for the case when we set the threshold to zero. On the right, we show a confusion matrix for the case when we use a high threshold (0.8). We can extract some interesting insights from these graphs.

First, we can see graphically the overall good performance of the model since both confusion matrices are quasi-diagonal. In the case the trained system was optimal, we would have seen a diagonal confusion matrix (every sample correctly labeled). On the other hand, we would have seen a homogeneous confusion matrix with zeros on the diagonal in the case of a poor classifier.

Moreover, notice how the model struggles to classify correctly all the samples from the *"Null"* category. This could be explained by two factors. First, the high intraclass variance of this group makes it more difficult for the classifier to understand which are the common determinant factors that the samples from this group possess. Second, the manner used to transform the human annotations into labels could have also played a role. Since we used a max pool approach to create the labels, it is quite likely that clips from the *"Null"* group also had a reduce number of frames for the other possible groups. This problem could have also occurred for the rest of the classes. However, the *"Null"* group was likely to be more affected since it should have been easier for the model to differentiate between the frames correctly labeled and those not as the frames belonging to any of the other possible classes are more similar among them than the frames for the *"Null"* group.

Finally, note the inability of the model to differentiate between the samples from two classes (*Y6129* and *Y6130*) regardless of the threshold imposed. While this could seem anecdotal at first glance, it is more easily explained by looking at Figure 3.9. We can see in this figure one picture for each of these artworks side by side and how they are almost

identical. Moreover, the quality of the presented pictures is high enough to make the few differences between the artworks visible. However, this is not the case for the quality of the recordings obtained with the used eye tracker. With this scenario, it might seem difficult for us to expect any model proposed to be able to correctly differentiate between the two classes.

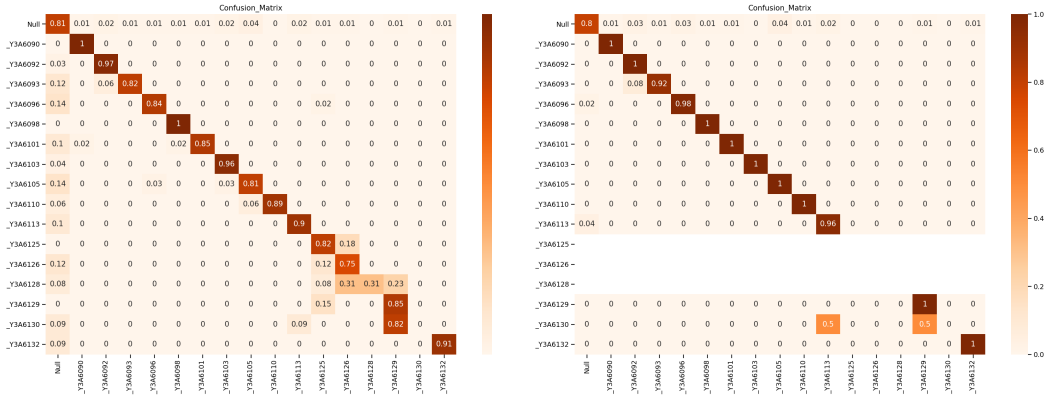

Figure 3.8: Confusion matrices on test set for the case of zero threshold (left) and the case of a high threshold of 0.8 (Right).

3.7.3 Active Learning

In this section, we wanted to simulate a real-life scenario. In this scenario, the model would be trained on a reduced set of annotated data and would try to label new recordings. Since the performance of the model is not perfect, we would include a human annotator in the labelling loop to provide feedback of possible misclassifications. In order to augment the abilities of our model, we could use the new incoming correct annotations to fine-tune our model. We have simulated this process by splitting the recordings not included in the benchmark set as 6 different subgroups of 8 recordings each.

The simulated process took form of a training loop were we finetune the trained model from each previous iteration using the data from the subset of the current iteration. Since we are finetuning in this stage, we have used shorter training parameters for each training loop iteration. Concretely, we have trained for 20 epochs with SGDR optimizer with following parameters ($Lr = 5e-3, T_0 = 1, T_{mult} = 2$). Note that we have reduced the learning rate for this case as we do not want to greatly change the weights of the already trained model. Finally, the warm restarts' parameters (T_0, T_{mult}) have been reduced to allow for a couple of restart cycles in the short number of epochs.

Figure 4.14 presents the evolution of the metrics' AUCs for the different training iterations when using the different set of data. The main observation is that the classification performance of our system remains almost static over the course of the active learning training loop. This could be due to a few reasons. First, the AUC scores were already considerably high for the base model. Hence, a priori, it would have become difficult to

Figure 3.9: **Could you distinguish these paintings?** Two different painting from the exhibition side by side. We can see a picture for the painting **Y6130** (left) and **Y6129** (right). The level of similarity between both artworks will suppose a great challenge for the proposed classifiers.

improve the performance of the system. Moreover, the procedure and the data used to finetune the system at each iteration could be the major reason. Given that we have low interparticipant variance, the samples fed to the model from the active learning subsets may have added little new information to the model. Hence, since we had already trained long enough for the base model, the new training iterations were useful to fit the new training data. On the other hand, this better fitting in the training data did not improve the generalization capabilities of the system due to the low level of diversity provided by the new training data.

3.8 Conclusions

In this chapter, we have formulated a human annotation task as a computer vision problem. We have converted the task of segmenting a video depending on the object a participant is watching as a video classification problem. Moreover, we have discussed the pros and cons for possible approaches to solve this task to motivate the selected model. Next, we have provided an overview of the dataset building process and an overview of the characteristics of the possible labels. This discussion was provided to help the reader to have a better understanding of the results section. Finally, we have provided the results of our approach for two different scenarios, the benchmark and the active learning scenarios.

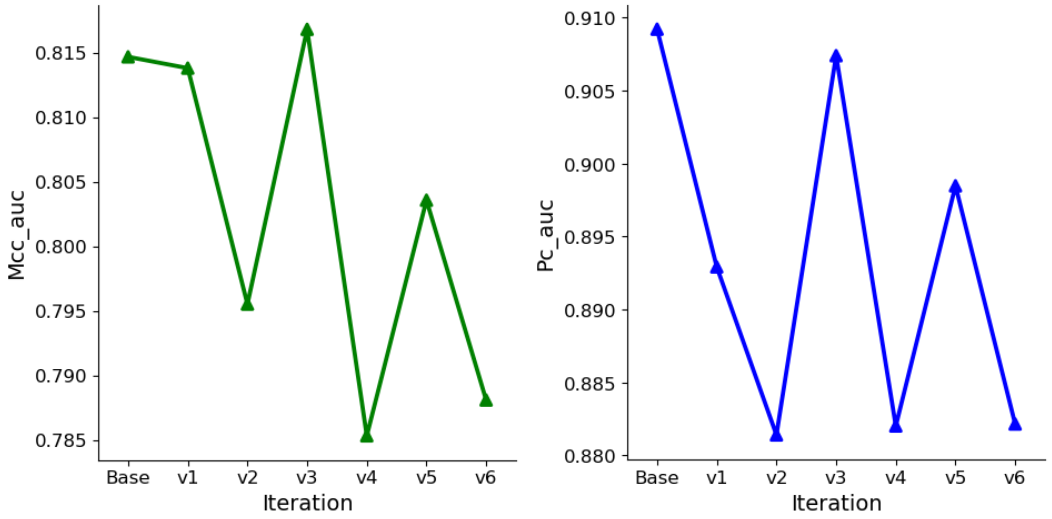

Figure 3.10: Evolution of AUC_{MCC} (left) and AUC_{PC} (right) over the course of the Active Learning training loop.

In the *benchmark* case, we have used a reduced subset of the dataset to train the proposed model from scratch. We have shown how the proposed model achieves considerable classification performance while being able to classify a large amount of the samples presented. The main limitation for the model in this case was the *"Null"* category as it was the one with the highest intraclass variance. Finally, we show in Figure 3.9 a case of limitation for our video model. Distinguishing between samples from these two different artworks is a difficult task for a human even when provided with high resolution pictures. Hence, in case we wanted a video model to be able to distinguish between these two classes, we would need to provide our model with high resolution videos. This is an issue for two reasons. First, it may be the case that the camera of the mobile eye tracker used in a project similar to ours does not have enough image quality. Moreover, even when having high-resolution videos, training a model with these videos with suppose a computational cost unbearable for most cases.

In the *active learning* case, we have simulated a real-life scenario where we finetune the already trained model using the rest of the dataset. We have seen how the performance of the model remains almost static in this case. We suggest that the already high performance and low variance between the samples of this dataset are the factors behind this result. Hence, we hypothesise that this strategy could be beneficial only when the available dataset has more variance amongst its samples. In this way, the system could be able to improve its generalization capabilities as it gets new unseen data.

Chapter 4

Behaviour Segmentation

When an individual is looking at any of the possible artworks in the exhibition, she can interact in diverse ways with such piece. For example, she can either view the piece from a close or a far distance, she can either fixate on one specific detail of the piece or just scan the whole piece and so on. There exists a considerable number of possible behaviours as the ones presented above. Thus, one of the first steps to take in this stage is to set a fixed taxonomy of possible behaviours and their correspondent descriptions. Such task was solved by the research group involved in this work and the resultant taxonomy can be found in Appendix A. Once we have a set of possible behaviours, we can treat this stage analogously as the video segmentation stage.

4.1 Dataset

Given the nature of the data available in this task, is seems a more natural choice to start this chapter providing a comprehensive overview of the dataset used before presenting our approach. We will see how the characteristics of the data will have a fundamental influence on the approach taken to solve the task.

4.1.1 Classes Overview

The first important characteristic is the classes themselves that we wish to classify. While in the previous section we wanted to classify objects seen by a participant, in this section we want to classify behaviours. Behaviors present various added layers of complexity compared to objects.

In principle, we should be able to distinguish whether we are seeing a particular object or not with low or any ambiguity given the appropriate frame. On the other hand, while it may be the same case for certain behaviours, the majority of the proposed behaviours require dynamic information in order to give a correct prediction.

The behaviours used in our work correspond to possible ways for a participant to engage with the art pieces inside the exhibition. The level of detail when describing these possible interactions can range from simple interactions (e.g. *"Stand in front of artwork"*) to really fine-grained iterations (e.g. Centring: *"Participant begins at a shallow-moderate*

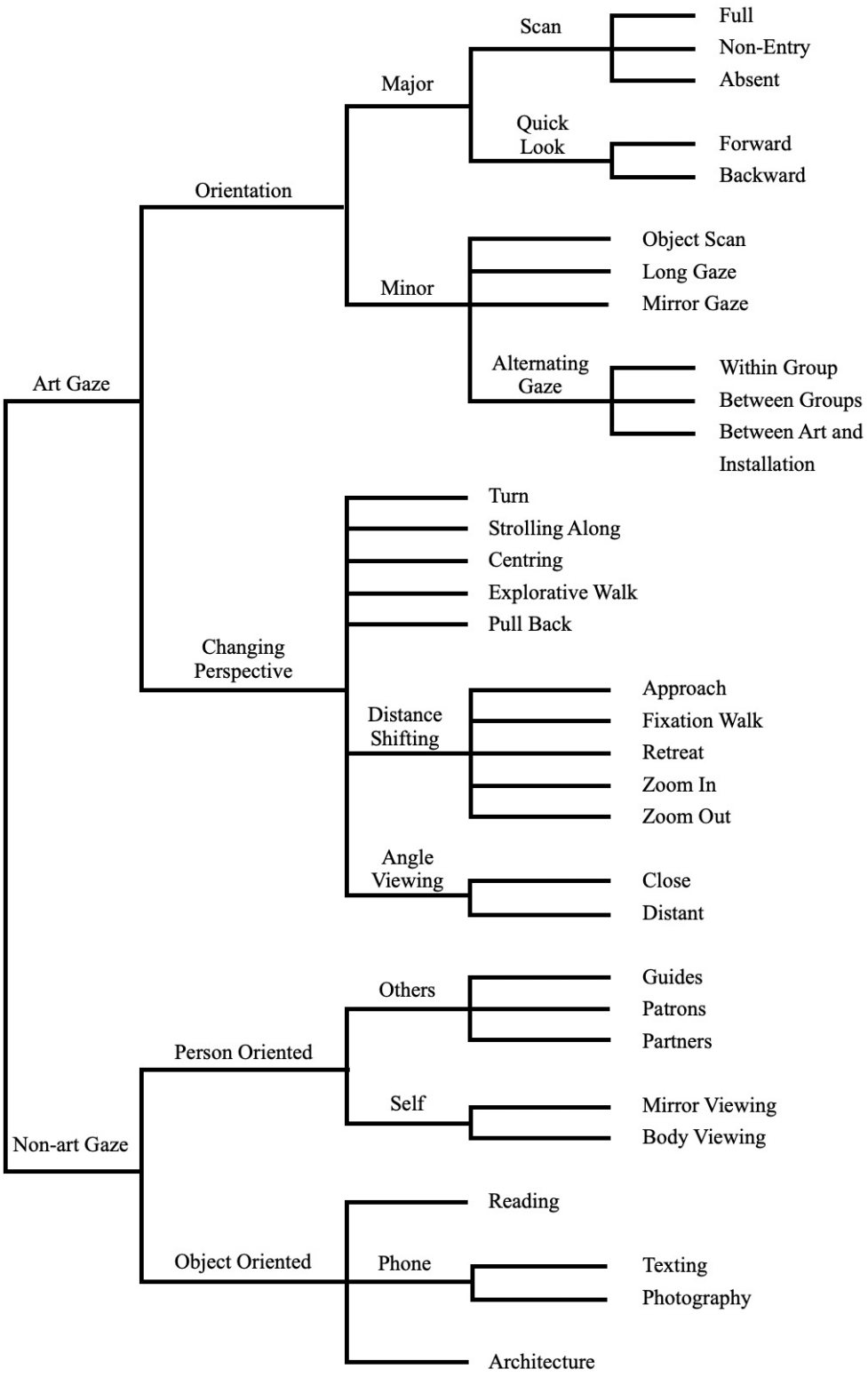

Figure 4.1: Taxonomy of the proposed behaviours consider for our behaviour segmentation task. The definitions for each of the given behaviours is presented in Appendix A

Figure 4.2: Example of ambiguity of static information in case of behaviour classification. Imagine we wished to predict whether the participant is getting closer or retreating from the piece. In this case, it is almost impossible to predict which of the actions is being performed with the available information. On the other hand, there is no need to provide more information if we just wanted to predict whether the participant is looking at a painting or not.

($0° - 70°$) angle relative to a piece and moves body to stand at ideal viewing distance and view the piece at a perpendicular angle. The majority of fixations during this behaviour remain on the piece towards which the participant is centring"). The majority of the target behaviours correspond to the latter group. Hence, we hypothesize that any proposed model will encounter great difficulties when trying to distinguish amongst the target behaviours.

We have just discussed why providing exact distinctions amongst the possible behaviours will turn out to be difficult. However, we consider that for the most part of the description of each behaviour can be covered as a combination of different factors. We believe that these factors will be of great help in choosing both which information will be useful (section 4.1.2) and which models will be most appropriate to solve the task at hand (section 4.5). We present the possible factors below:

- **Movement**: One discriminatory factor is how much the participant is moving. For example, the participant can be either still observing the piece, or moving towards it to get a closer look.

- **Orientation**: It informs us about the relative orientation of the participant with respect to the piece over time. For instance, the participant can be situated perpendicular to the piece or at a shallower angle depending on the parts of the piece the participant wants to appreciate.

- **Gaze**: The final factor involves information about the gaze patterns of the participant with respect to the environment. The participant can either focus on a specific region of an artwork or she could be glancing across different part of the artwork or between the artwork and other parts of the exhibition.

To close this section, we will provide a similar overview for the different possible groups of behaviours as the one provided in section 3.5 for the video segmentation stage. We can encounter two big behavioral groups according to whether the participant is actively engaging with some artwork (Art-Gaze behaviours) or not (Non-art Gaze behaviours).

We have twenty-one different labels in the first group, one for each different art-related behaviour. We can assume that the segments in this group will share some properties. First, we will encounter high intraclass variance. To demonstrate this fact, we will illustrate the possible sources of variance by analysing one of the behaviours from our dataset. We will use the same example as in before (**Centring**):

- **Artwork seen**: The most straightforward source of variation is the fact that the same behaviour can be performed with an array of different pieces inside the exhibition. Hence, if we use visual information (e.g. video clips) to try to classify this behaviour, this will suppose an extra degree of variation.

- **Orientation**: In the analysed behaviour, the participant can start centring from both sides of the piece, right or left. Moreover, the range of possible angles from which the participant can start centring ($0° - 70°$) is rather considerable.

- **Movement**: The possible degrees of diversity with respect to orientation affect directly the correspondent degrees of diversity in the case of movement. First, we would have opposite movement patterns depending on whether the participant started the behaviour from the left or the right side of the piece. Moreover, this behaviour will have the label of centring regardless of the speed with which is performed.

- **Gaze**: Gaze patterns are a rather informative source of information for the majority of the possible art-related behaviours. However, there are other behaviours, such as centring, where the gaze patterns are instead an extra layer of variation.

Moreover, we can assume that high interparticipant variance also exists for this group. This can be mainly due to the specific gaze and gait patterns that each individual could present. This is a rather undesirable property in our case, since it will limit the generalization capabilities of our models with respect to unseen participants.

Similar to the case of video segmentation stage, we have a group of behaviours where there is not any piece of artwork involved. This group includes a number of different possible behaviours. However, since we are only working with art-related behaviours in this work, the only relevant information these behaviours provide is the fact that the participant is not actively engaging with any piece of artwork. Therefore, we have chosen to group this set of behaviours into a single behaviour with the label of *"Null"*. In this case, the same argument as in the art-related behaviours can be made with respect to the

possible levels of variance. Hence, the samples from this group will likely present high intraclass and interparticipant variances.

4.1.2 Sources of Information

To tackle the problem of the video segmentation stage it was sufficient to use the video recordings of the participants. However, given the different variation factors explained in section 4.1.1, we can assume that there are different kinds of data that can be used to address this stage. The mobile eye tracker device used in our project offers some different type of data that could be informative for this stage. Next, we will present the different types of data selected and the information that they provide. The selected source of data used can be divided in two separate groups:

- **Time-Series data**: A time-series consists of multiple measurements of a quantity of relevance at regularly spaced time intervals. This group is formed by three different set of time-series:
  - **Accelerations**: This variable inform us of how much (*movement*) and in which direction (*orientation*) the participant is moving at any point in time. If we take a look back to Figure 2.2 we can see the coordinate system of the participant (HUCS). This coordinate system is formed by three perpendicular axes converging on the top-center of the glasses. The accelerometer inside our eye tracker will record three different set of accelerations, one for each of the discussed axes (X,Y,Z).
  - **Angular velocity**: Defined as the rate of rotation along one axis. The gyroscope (device to measure angular velocities) included in the MET device will record three angular velocities, one for each different axis of the participant's coordinate system. These quantities will provide information about the *orientation* factor.
  - **Gaze**: The most straightforward source of information. The eye gaze will provide an indirect measurement of the visual attentional focus of the participant at any point in time. In this stage, it will be a determinant factor to classify some of the behaviours proposed. Finally, note again Figure 2.2 where we see that we will record the gaze point in the three spatial axes.
- **Visual data**: The other possible source of information available are the raw video clips from the recordings. Rightly processed, videos can provide information about orientation and movement. On the other hand, videos could only provide a limited report of the individual's gaze. While we could extract a general overview of where the participant is looking by analysing the video clips, it is intractable to specify exactly the gaze patterns of the participant with the information that a raw video provides.

4.2 Dataset Building

We have implemented an analogous procedure for building the behaviour segmentation dataset as in the case of the video segmentation stage (see section 3.4). However, the already discussed differences between both tasks will also be reflected on the dataset building process. Hence, we will provide an overview of how these differences affect the dataset building for this stage.

First, we have two different groups of data (time-series and visual) to tackle this task. While the possibility existed to fuse both group of data and have a multi-modal classification system, we have decided to work with both groups separately. This decision was motivated for by idea of having a computationally efficient model. We wanted to test the achievable performance of a model which just uses time-series data as it would be considerably cheaper in terms of computation needed. Hence, we have created two separate groups of datasets, one for each group of data. We will use these two groups to compare them in terms of classification and computation performance. We will discuss in the conclusions section how we could fuse both groups of data in a multi-modal system.

Moreover, we have observed that the average duration of the behaviours considered is shorter than the 10s used to create the video segmentation clips. Therefore, in order to spot the different possible behaviours we have reduced the length used to split the data into separate segments. We have chosen 2s segments in this case as it sets a good trade-off between the possible minimal duration of a behaviour and the amount of information needed to classify such behaviour.

Finally, we have discarded the samples from a few behaviours due to two separate reasons. First, we have discarded the samples from the *"Approach"* behaviour as it was created mid-project and the first 30 participants don't have any annotation with this behaviour. This constituted a problem since the models trained for the benchmark split didn't encounter this behaviour. Hence, the possible number of classes is different between the benchmark and the active learning split. While we could have added the approach label as a possible behaviour in the benchmark split, the model wouldn't have any sample of it and it would have deteriorated the training process. Second, if we take a look at Figure 4.3 we can see the original number of samples available for the different behaviours for the whole set of participants. This figure shows a few behaviours with almost no representation. We have chosen to discard those behaviours for which the total number of samples didn't exceed ten units (*"Alternating Gaze Between Group"*, *"Scans Absent"* and *"Explorative Walk"*). We have used this threshold as behaviours with samples below it don't have one sample per possible training and test splits inside the Benchmark and Active Learning datasets.

4.3 Handling Imbalance

Given the different number of possible behaviours that we can encounter, it may seem natural that the frequency upon which they are performed is not equal for all the cases. Therefore, we encounter the same problem as in the case of the video segmentation stage (class imbalance). However, the specifics of how this problem is presented in this case is different for this stage. Moreover, the specifications of the proposed models are also

different. Hence, we will discuss in this section this differences and how they affect the choices made to deal with class imbalance for this task.

First, we can see in Figure 4.3 the distribution of the number of samples for the different possible behaviours in our dataset. In contrast to the presented imbalance for the video segmentation dataset, the problem is more severe in this dataset. We have a class that is considerably over-represented as in the video segmentation case (*"Object Scan"*). However, we can see how the distribution over the rest of the classes is not balanced either in this case.

The high intraclass and interparticipant variances that the classes of this dataset present constrains the choice of possible approaches to deal with imbalance. This is so because we will lose more information if we were to discard samples for the over-represented categories. Hence, we consider that the random undersampling approach would not be the best fit for this case.

Finally, we have two distinct sources of data to solve this task. As discussed, we will employ two different set of models for each of the possible groups of data. In this case, the discrepancies between both set of models will affect also the choice of possible approaches to tackle the class imbalance problem:

- **Video models**: The problem tackled in this stage is known in the literature as Action Recognition. Several video classification models and large benchmark datasets have been proposed to try to solve this problem in the literature. We presented a brief overview of the most well-known models in section 3.1. One of the main advantages of these models is that we can find implementations of them already trained on some of the available benchmark datasets. Hence, we hypothesise that the imbalance problem will be less determinant in this case as the used model has not been trained from scratch. Moreover, we have already discussed the high computational costs of the available video models. This make the use of complex procedures to alleviate the class imbalance problem (e.g. oversampling, SMOTE, metric learning) something to avoid. For the discussed reasons, we have chosen a cost-sensitive loss (balanced cross-entropy) as the measure to deal with the class imbalance problem. Using this loss, we penalize each misclassification mistake according to the representation of the class to which each mistake belong. Hence, we address the class imbalance problem without any computational overhead.

- **Time-series models**: While there are some pretrained models for working with time series, they are not appropriate for our task. The reason for this is that the majority of available models work with one-dimensional time series for other tasks (e.g. stock prices, item prices, energy consumption and so on). Moreover, there are no publicly available large benchmark datasets containing data from wearable sensors for activity recognition. Therefore, the models trained using this group of data will be trained from scratch. For this reason, we hypothesise that the effect of imbalance will be more acute in this case. However, the main advantage of the Time-series models over the video models is their low computational costs. Therefore, we can use more complex approaches to tackle the imbalance problem in this case. We have chosen two different approaches in this case. First, we have employed the same balanced

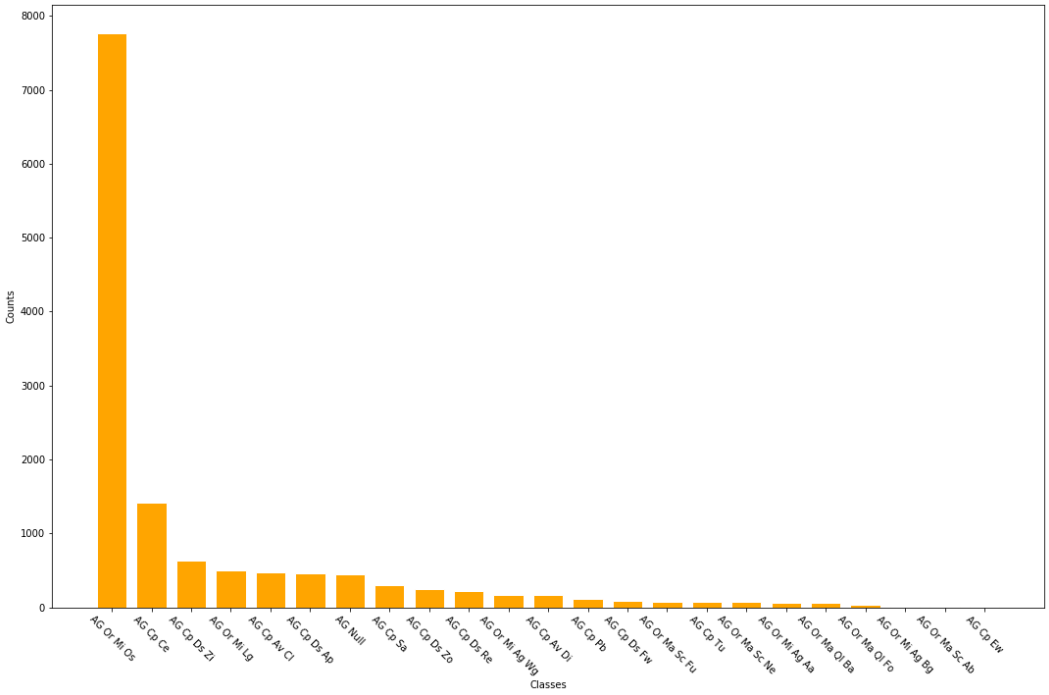

Figure 4.3: Overview of total number of samples per class in the original behaviour segmentation dataset

cross-entropy as the loss function. Second, we have made use of the metric learning literature to ensemble a more robust model against imbalance. Since we only gave a glance over the basic idea of metric learning in section 3.1, we will provide a more comprehensive review of this topic in the present section.

4.3.1 Metric Learning

Metric learning attempts to embed raw data into a feature space in such a way that semantically similar samples are contained in the same region of the space whereas dissimilar samples lay far apart. This technique does not address the class imbalance problem directly. However, metric learning can indirectly address this problem by improving the separability of the samples in the obtained feature space. In other words, every sample from each class would be clustered together in their own regions. This fact would help greatly the task of the classifier (see Figure 4.4 for a graphic example). Hence, in theory, the better mapping we achieve, the less influence of the imbalance problem on the classification performance of our model.

We are going to give an overview on how metric learning works by comparing this process with the classification process. In the standard classification scenario, we train our model by presenting it with a set of labeled samples. The model evaluates one sample at a time and it changes its internal parameters using the feedback from a classification loss

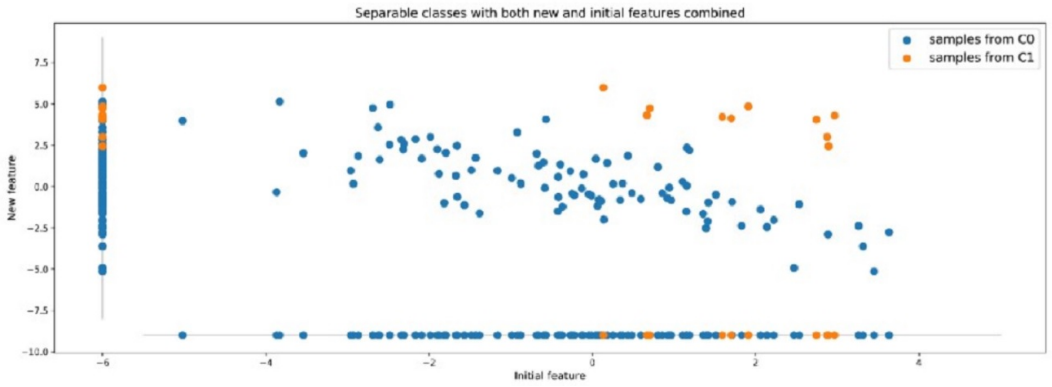

Figure 4.4: Illustration of separability as means to address the class imbalance problem. In this example, we have two different classes ($C0, C1$) with uneven representation. We can see how we would be unable to correctly classify all the presented samples with the first descriptive feature given (horizontal axis). On the other hand, if we are able to obtain a second different feature (vertical axis), we can achieve perfect classification performance with a simple linear classifier by combining both features. Finally, we can see how, in the 2D-case, the effect of the imbalance is eliminated.

function. Such classification loss, whichever it may be, will serve as a quantitative measure of the ability of the classifier to correctly label all the training samples.

On the other hand, the basic metric learning scenario present some remarkable differences. First, the model will be presented with a tuple of samples (e.g. pairs, triplets...) at a time. These tuples will be *weekly-labeled*. That is, the label for any sample in each tuple will not be the class of such sample but the relative level of similarity with the rest of the samples from the tuple. For example, if we have a pair of samples from different categories we would set the value of one label to one and the other to zero to represent the difference in categories. Finally, the model will update its parameters according to the feedback of an embedding loss function. Embedding loss functions will not report the ability of the model to correctly classify the class of each sample but they will report the quality of the generated embeddings in terms of the already discussed separability.

Embedding loss

Training the model with an appropriate embedding loss is fundamental to obtain the best possible embeddings. Pair and triplet losses provide the foundation for the two most employed metric learning approaches. The quintessential pair based loss function is the contrastive loss [13]. This loss tries to force the distance between similar pairs (d_p) below a given threshold (m_{pos}) while maintaining the distance of the dissimilar samples (d_n) larger than some other threshold (d_n).

$$L_{contrastive} = max(0, d_p - m_{pos}) + max(0, m_{neg} - d_n) \tag{4.1}$$

The theoretical downside of this method is that it applies the same distance thresholds to every pair while they may differ largely in how similar and dissimilar the different

possible pairs are.

The triplet margin loss [45] addresses this issue in theory. A triplet is formed by an anchor, positive and negative sample, where the anchor is more similar to the positive than the negative. The triplet margin loss attempts to have embeddings where the distance between anchor and positive (d_{ap}) is smaller than the distance between anchor and negative (d_{an}) by a given margin (m).

$$L_{triplet} = max(0, d_{ap} - d_{an} + m) \tag{4.2}$$

Since the proposal of the discussed loss functions, a plethora of new alternatives have been proposed in the literature based on this concepts (e.g. [47], [35], [34]). However, a recent paper [27] compared the most relevant embedding losses under the same conditions. The main conclusion of this work can be summarized as follows: given a correct optimization of the losses' hyperparameters (e.g m_{pos} and m_{neg} in the contrastive loss) the difference in performance for the losses under study was marginal. Furthermore, the results obtained with the contrastive loss were slightly better than in the case of the triplet margin loss. Based on this work, we will perform every metric learning experiment using a contrastive loss. Moreover, we will carry an extensive hyperparameter optimization (detailed in section 4.6) in order to tune the loss function according to our dataset.

Miner

Mining is the process of selecting the best possible pairs or triplets used to train at each iteration. This step is fundamental to any metric learning approach for two main reasons. First, while we could use all the possible pairs or triplets, this is extremely memory and computationally consuming. Moreover, using every possible tuple we would train with a high number of easy negatives and positives which would not serve as informative feedback to update our model. There exists two different types of mining processes: *offline* mining and *online* mining.

In the offline mining, we select the best possible pairs or triplets to include in each batch before its construction. This might be accomplished through the use of different procedures. For example, we could have a list with hard negatives [32]. Also, we could perform a nearest neighbors search before the start of each epoch [14] or before each training iteration [37] to obtain the best possible tuples.

On the other hand, online mining finds the best tuples inside each randomly selected batch. While we could select all the combinations possible inside each batch, we would face the same drawbacks as in the case of using every possible tuple to train (memory consumption and easy negatives and positives). The most straightforward idea would be to select only the hardest positive and negative samples [16]. However, this idea leads to noisy gradients and reaching bad local optima [42]. To alleviate these issues, a softer version [30] was proposed where we mine semi-hard negatives. In this case, we mine negatives that are close to an anchor but they are still far away from other positives. However, it was also found [42] that the effectiveness of this alternative decayed rapidly as the number of possible semi-hard examples dropped as effect of training. Recently, a simpler strategy [42] was proposed to address this issue with satisfactory results. In this approach we select a negative if they are closer to an anchor than its hardest positive counterpart, and positives

are selected if they are found further to an anchor than its corespondent hardest negative. We will use this mining strategy in our metric learning experiments given its simplicity and documented results.

4.4 Literature Review

In this section we will address the literature related to the models that use time series as input data, since the literature review corresponding to the video models has already been properly detailed in section 3.1. As for the problem of classifying behaviors with time series data (accelerometer, gyroscope, gaze), we see that this problem can be formalized as a time-series classification (TSC) problem [1]. This problem has been considered one of the most difficult problems in the field of data mining [46] and has been widely treated in the literature. We can group the possible approaches in two groups: hand-engineered approaches and deep learning approaches. As discussed in chapter 2, we will focus our literature on deep learning approaches because of their flexible nature and unnecessary need to use domain knowledge.

Figure 4.5 shows schematically the framework used by the models considered in this overview. In this framework, the raw data are first converted into features and are then fed through a classifier that maps these features to a probability distribution of confidences over the possible classes in our dataset. In this way, the models considered train in an end-to-end fashion both the classifier and the feature extractor using the training set.

The simplest approach to address the problem would be to employ an MLP with several fully connected (FC) layers to model the extractor feature and the classifier [43]. However, the problem with this type of architectures is that the temporal information is lost and the features obtained are not invariant in time. On the other hand, we see how most of the architectures used in the literature are CNNs [20]. The reason for this is that convolutional operation is able to learn spatial-invariant relationships from raw time-series. This is because the same convolutional filter will be slid across each time series with the same trainable parameters. This way, each filter will be activated when encountering the same time pattern regardless of where in the time series did this pattern appear. Next, we will give an overview of the different convolutional models presented in the literature.

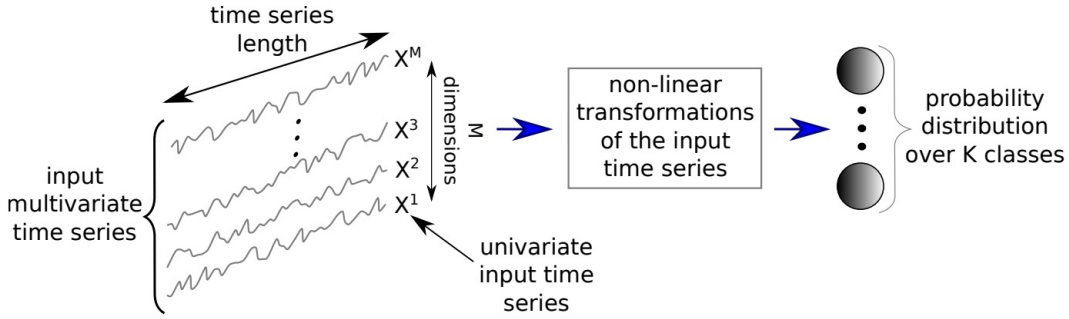

Figure 4.5: Overview general deep learning framework for time series classification [20]

4.4.1 Fully Convolutional Neural Network

Fully Convolutional Neural Networks (FCNs) were proposed for the first time to classify time series in [43]. These networks are mainly composed of convolutional layers. In addition, they do not introduce any pooling operation which keeps the length of each time series the same throughout the run of the model. Finally, one of the main improvements offered by these architectures is the change of the last FC layer for a Global Average Pooling (GAP) which drastically reduces the number of trainable parameters of these models. The specific version proposed in [43] is composed of three identical blocks in which each block performs the following operations: first a convolution operation, followed by a batch normalization layer [19] that finishes with a ReLu activation function. Each activation map obtained after the last of the described blocks, will go through the GAP layer in which the values of each activation map will be averaged over the length of the time-series. Finally, the output of the GAP layer will be fed to a conventional softmax layer which will provide an array of class confidences.

4.4.2 Time Le-Net

Time Le-Net was proposed in [24] inspired by the success of one of the first CNN architectures (LeNet [25]). This architecture has the basic structure of the CNNs in which we have two convolutional layers followed by a FC layer and a softmax layer. This architecture has two differences compared to the discussed FCNs. First, it uses a FC layer instead of a GAP layer, which increases the number of parameters needed. Second, it adds local max pooling operations between each convolutional layer. Local max pooling operations reduce the length of each activation map by repeating the same operation to each small group of points in our activation map. For example, a max pooling operation with a length of two would take every two consecutive points of the map, and take the maximal of the two values and add this value to its correspondent point into the new activation map. In this way, local pooling operations add spatial invariance to small disturbances (equal to the length of the pooling operation).

4.4.3 Time Convolutional Neural Network

The Time-CNN proposed in [48] add three modifications with respect to the previous architectures. First, it employs a Mean Squared Error (MSE) loss function to provide feedback to update the networks' parameters instead of the standard cross-entropy loss. Second, the authors propose to use *average* local pooling layers instead of *max* local pooling layers. Finally, the only difference in terms of modelling in this case is the use of a balanced cross-entropy loss function to address the class imbalance problem. Finally, in the case of this architecture, the classifier layer is directly connected to the last convolutional layer without neither adding a GAP nor a FC layer.

4.4.4 Residual Network

Residual Networks (ResNet) have been the building block for many state-of-the-art computer vision architectures. Wang et al. [43], proposed a modified version of a relatively

shallow ResNet in order to process time-series data. The main characteristic of Residual Networks is the employment of shortcuts connections between convolutional blocks (see Figure 4.6). Such residual connections enhance the gradient flow of the network, making it easier to train and thus assembling deeper architectures [15]. In the case of the specific version proposed in [43], it is composed by three convolutional blocks followed by a GAP layer and a final softmax layer. Each convolutional block repeats the same group of operations (Convolution, Batch-Normalization, ReLU) three times. In a comparative study of deep learning classification models conducted by Hassan et al. [20] the authors tested over 8730 deep learning models on 97 time series datasets, the temporal ResNet yields state-of-the-art results for almost every time-series dataset tested.

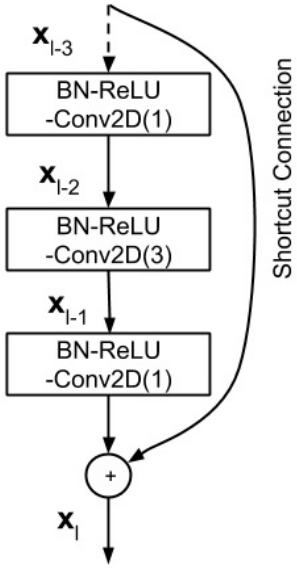

Figure 4.6: Residual block for the discussed ResNet architecture

4.5 Proposed Approaches

In this section, we provide a high-level overview of the different strategies employed in our work with the purpose of tackling the behaviour segmentation task. Since the different groups of data available have their own design constraints and specifications, we have employed distinct approaches for the two data groups.

4.5.1 Visual Data

In this case, both the task (classification) and the design constraints (computation efficiency) are analogous as in the case of the video segmentation task. Hence, we have decided to use the same architecture (I3D video model with a Resnet-50 backbone) as the one employed to solve the video segmentation stage. In this case, we argue that use of a video model was more justified in this case than in the video segmentation case. Indeed, any single

frame model would be unable to correctly classify the samples in which movement is a discriminant factor. Finally, the only difference in terms of modelling in this case is the use of a balanced cross-entropy loss function to address the class imbalance problem.

4.5.2 Time-Series Data

For this group, we will work with the same architecture but we will propose two types of strategies to train such architecture. With respect to the architecture, we have chosen to adapt a convolutional model. Convolutional neural networks are able to automatically find informative relationships amongst the time-series of a sample by successively applying convolutional filters. The choice of a convolutional model has been motivated by a few factors. First, we wanted to use a deep learning approach since we wanted a flexible approach. In theory, if we had another segmentation dataset with different behaviours from another mobile eye tracker project, we could use the same model as in our project with the weights obtained after training on our dataset and get a boost on performance for the new task. Secondly, we have chosen a convolutional approach since it has been the modality of deep learning architectures with the best performance on time-series classification problems in the literature. Specifically, we have employed an adapted version of a ResNet architecture as in [44] in our work. ResNet has been chosen over other convolutional architectures due to its superior performance [20] and the simplicity of the changes needed for this architecture to be able to process multi-dimensional time-series.

With respect to the training strategies employed, we have two big groups: baselines and metric learning approaches. The **baseline** strategy simply consists of training the proposed architecture using the traditional classification scenario. This will consist of a single-stage training process where we feed our model with the labeled samples and we use either cross-entropy or balanced cross-entropy as loss function. On the other hand, in the **metric learning** scenario we will have a two-stage training process. First, we will train the modules of the architecture that map raw data into embeddings (noted as trunk and embedder in Figure 4.7) using the discussed contrastive loss and selecting the pairs in each batch with the multi-similarity miner proposed in [42]. Next, we will add a classification module to the already trained modules with the purpose of mapping the embeddings into class confidences. The whole architecture will be fine-tuned using the labeled dataset and a classification loss function.

4.6 Implementation Details

In this section, we will only describe the implementation details for the time-series models as we have used the same implementation as in section 3.2.1 for the video model. For the time-series models, we have conducted every experiment using the PyTorch [28] framework. As previously discussed, we will use the same model architecture with two different training procedures: **Baselines** and **Metric Learning**.

We need to provide the details for the employed architecture before explaining the proposed training procedures. First, while the proposed architecture is a single model, we will distinguish between three different modules within it. This has been done because each

module has its own topology and function. We can see the presented modules together with their topologies in Figure 4.7:

- The **trunk** module process the raw time-series to convert them into meaningful features for our task. We have used an adapted version of the ResNet18 in this stage. The modification implemented with respect to the original model has been to change the original 2D convolutions (appropriate for images) into 1D convolutions (suitable for time-series).

- The **embedder** module maps the output features from the trunk module into a embedding space. The dimensions of this space are reduced with respect to the ones from the original features (512). The architecture used for this module is a simple multi-layer perceptron (MLP) with an input layer and an output layer. The first layer has the same number of units as the dimensionality of the input features. The output layer has as many units as the dimension of the embedding space (EmbDim). Note that this is an important hyperparameter as it involves a trade-off between representativeness of the embeddings and the curse of dimensionality. We face the curse of dimensionality in the case where we have lots of features, samples become harder to cluster. The reason for this phenomena is that in high-dimensional spaces every observation appear equidistant from all the others. Therefore, if the distances amongst every possible sample appear to be approximately equal, it becomes harder to distinguish amongst the possible classes in our dataset. Finally, we have applied dropout [36] between the input and the output layers with a probability of 0.5 to reduce overfitting.

- The **classifier** module transforms the embeddings into an array of confidences for the possible classes. We will use the obtained confidences to provide the label of each sample (providing the class corespondent to the element of the array with highest confidence as label). We have used the same MLP architecture for this stage. The only change in this case is the number of units per layer. For the input layer we have as many units as embedding dimensions from the previous stage. Finally, for the output layer we will need as many units as possible classes we can find in our dataset.

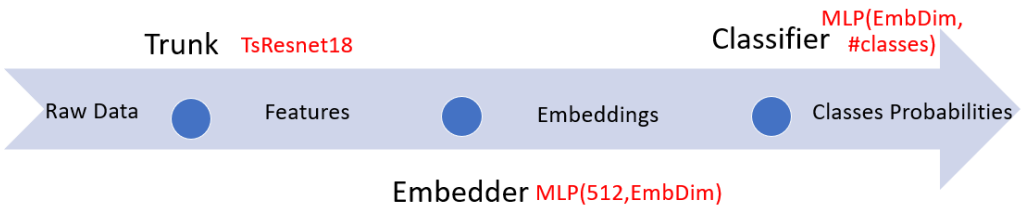

Figure 4.7: Overview of the different modules that compose the time-series architectures. Each label in red corresponds to the topology of the module specified to the right of that label.

4.6.1 Metric Learning

We will discuss this strategy first as it will share some of its training hyperparameters with the baseline approach. This strategy consists of a two-stage training process where we first run a metric learning training process and then we finetune the obtained models in the first stage through a classification training procedure.

First stage

In this stage, we train the first two modules of our architecture (trunk and embedder) with a contrastive loss function and we sample the training pairs inside each mini-batch with the multi-similarity miner. Note that both, the loss and the miner, require that we choose the value for their correspondent hyperparameters. As we discussed in section 4.3.1, if we want to obtain embeddings of high quality we have to correctly tune such hyperparameters. With this purpose, we have run an extensive hyperparameter optimization procedure. Since the details of this procedure are not an essential piece of this work, we have decided to discuss them in the Appendix B. The results of this process can be found in Table 4.1.

We have trained our models for 100 epochs with a mini-batch size of 64. We have optimized our networks using SGD with momentum. The momentum term has been set to 0.9 and the weight decay to $5e-5$. For the learning rate schedule, we have utilized the same warm restarts schedule as in section 3.2.1. The hyperparameters related to the learning rate schedule (η, T_0, T_{mult}) have been tuned and their optimal values can be found in Table 4.1.

Second stage

Once we have finished the first stage, we can "plug" the classifier module into the already trained trunk and embedder. Then, we finetune the complete architecture using a classification set-up. Note that we have to make a design choice before starting the finetuning process. Since we have two modules already trained that should produce meaningful embeddings, we could chose to either update or not update their parameters in the second stage. In the case we do not update the parameters of a module during this stage, we say that such module is **frozen**. Given the possible number of combinations in this case, we have decided to perform the procedure for the second stage for every possible combination of states (frozen or not frozen). The different combinations tested can be found In last the last six rows of Table 4.2.

For the classification set-up, we have trained our model with either a cross-entropy or a balanced cross-entropy as loss function. The optimization parameters have been kept the same as in the first stage with the exception of the number of training epochs. In this case, we have trained for a shorter period of time (50 epochs).

4.6.2 Baseline

This approach could be considered as the most straightforward in order to train a classifier model. We train our architecture in an end-to-end fashion with a classification training set (i.e. the label for each sample corresponds to the class of such sample). In this case,

Function	Parameter	Optimal Value
Contrastive loss	m_{pos}	0.0613
Contrastive loss	m_{neg}	0.6860
Multi-similarity miner	ϵ	0.2366
Optimization	η	$5.12e-3$
Optimization	T_0	10
Optimization	T_{mult}	1
Embedder	EmbDim	256

Table 4.1: Summary of the hyperparameter optimization process for the time-series models. We present the parameters considered, their function and the found optimal values.

we have trained our complete architecture using a standard cross-entropy or a balanced cross-entropy as loss functions. Each model has been trained for 150 epochs and the mini-batch size was set to 64. We have used the same optimization recipe as in section 3.2.1. In this recipe, we have optimized our networks using SGD with warm restarts. Since we want to compare the performance of this strategy with the metric learning process, we have used the same hyperparameter for the optimization process. The values for these hyperparameters are specified on the optimization rows in Table 4.1.

Model Name	Frozen Module(s)	Loss Function
Baseline Base	-	Cross-Entropy
Baseline Balanced	-	Balanced Cross-Entropy
Frozen Trunk Base	Trunk	Cross-Entropy
Frozen Trunk Balanced	Trunk	Balanced Cross-Entropy
Frozen Base	Trunk, Embedder	Cross-Entropy
Frozen Balanced	Trunk, Embedder	Balanced Cross-Entropy
Finetuned Base	-	Cross-Entropy
Finetuned Balanced	-	Balanced Cross-Entropy

Table 4.2: Summary of the different time-series models proposed for the behaviour segmentation task.

4.7 Results

In this section, we will summarize the results for the various experiments carried out. We will first discuss the results for the models inside the two different sources of data (Time-series, Video) separately and then we will compare their respective performances. Since we have tested various configurations for the time-series models, we will compare their performances on the validation split to finally choose a best model of each time-series categories (Baselines, Metric Learning). Finally, given the analogies between the video segmentation task and the behaviour segmentation task, we will give an analogous

treatment to this section as in section 3.7. However, after running our initial tests with the benchmark dataset, we discovered that the performance of the proposed models was lower than expected. We hypothesized that one of the major causes for this fact is the high interparticipant variance. Hence, to test the validity of this hypothesis, we assembled another dataset (Random dataset). The Random dataset was created by creating splits with the same sample proportion as in the first dataset (Participants dataset) but selecting the samples for each split by randomly sampling from the complete dataset.

4.7.1 Time-Series Models

In this section, we will evaluate the respective validation results for the proposed models on the two discussed datasets. The validation results for the time-series models have been summarized in Table 4.3. We can group the obtained results in diverse ways depending on the focus of analysis. For example, we can compare the overall performance of the models when trained on the participants dataset versus when trained on the random dataset. Therefore, we present next a more detailed analysis of the results depending on the possible comparisons that can be made.

Participants dataset vs Random dataset

We can see how the previously discussed hypothesis holds in this case. We found that the performance of each model is always higher when trained in the random training set than when trained in the participants' training set. More precisely, we can see how the difference in performance is more pronounced in the case of AUC_{MCC} scores than in the case of AUC_{PC} curves. In fact, the AUC_{MCC} scores of the models trained on the participants dataset is fairly zero in every case. This fact indicates fairly poor performance of models when trained on the participants dataset. We have illustrated graphically in Figure 4.8 why the MCC scores are more informative than the Precision scores for the case of imbalanced classification. Finally, while there are differences in performance along the models trained on the participants dataset, we can see how we are not able to generalize across participants in this case. On the other hand, while the performances of the models trained on the random dataset is also not optimal, we can see how there are models that show some degree of generalization.

Standard cross-entropy vs Balanced cross-entropy

One of the taken measurements to alleviate the class imbalanced problem was to train our models using the balanced cross-entropy. In order to study the effectiveness of this measure, we trained each possible model with both cross-entropy loss functions (standard, balanced). In general terms, we see how the use of the balanced version has two effects. First, the AUC_{MCC} is higher for the models trained with the balanced loss function for the models trained on the random dataset. Note, however, that the differences with respect to this score are almost negligible for both loss functions for the models trained on the participants dataset. This indicates that the models trained on the participants dataset have almost zero discriminative abilities regardless of the loss function employed. Second, the AUC_{PC} score is lower for the models trained with the balanced cross-entropy with their

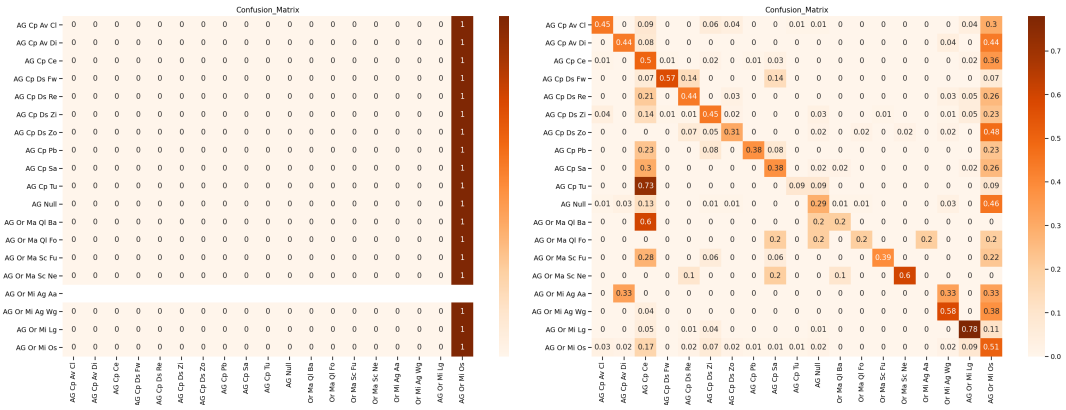

Figure 4.8: **Illustration of the relevance of the MCC score for the case of imbalanced classification**. We can see in the left the confusion matrix for the frozen trunk base trained on the participants dataset for the case of zero threshold. In such case, we obtain a MCC score of zero and a Precision score of 0.613. On the other hand, we show in the right the confusion matrix for the baseline balanced model trained on the random dataset for the case of zero threshold. In this case, the Precision score is lower (0.495), however the MCC is higher (0.295). We can see at first glance how the first model always predicts the majority class while the second model is able to offer some level of distinction amongst the possible classes. The precision score is higher in the first case since we have been able to classify correctly more samples of the majority class.

standard cross-entropy trained counterparts. We could expect this effect, since the models trained with the balanced loss function are going to label less samples as the majority class in favour of the rest of the classes. Therefore, the total number of true positives for the majority class is going to be, lowering the average precision score in this case.

Baseline vs Metric Learning

The second strategy used to address the class imbalance problem was to train the models using a metric learning set-up. However, the results reveal that the use of this technique does not bring about considerable improvements in performance. We can even see that the highest AUC_{MCC} scores -both for models trained in the random dataset and participants- are obtained with models trained with the baseline procedure. On the other hand, we can see that the highest AUC_{PC} are obtained for models trained with a metric learning set-up. While this was not the expected outcome initially, we believe that several factors may have influenced this outcome. First, the difference caused by training the possible models using one stage (baseline) or two stages (metric learning).

In the case of having trained a trunk and embedder that generated optimal embeddings in the first stage of the metric learning process, we would have obtained performance improvements with respect to the baseline models. On the contrary, in the case that modules trained in the first stage were not able to generate separable embeddings, this would detract from the performance of the final model. This is so, since the second training stage starts

with the weights obtained in the first stage. In this case, this makes the model start the optimization process at an initial point influenced by the already trained weights. In this way, in the second stage, the model would have to move further away from the initial point as worse embeddings generate the modules of the previous stage. This intuition can be seen in the table where we see that the performance of the metric learning models increases as we allow the model to move away from the starting point. This happens as we allow each module to modify its parameters in the second stage.

	AUC_{MCC}		AUC_{PC}	
Model	Participants	Random	Participants	Random
Baseline Base	**0.090**	0.300	0.526	0.671
Baseline Balanced	0.048	**0.419**	0.071	0.574
Frozen Trunk Base	0.00	0.019	**0.635**	0.616
Frozen Trunk Balanced	0.001	0.334	0.028	0.156
Frozen Base	0.000	0.000	0.627	0.603
Frozen Balanced	0.000	0.245	0.016	0.182
Finetuned Base	0.084	0.287	0.609	**0.726**
Finetuned Balanced	0.000	0.307	0.058	0.280

Table 4.3: Summary of AUC scores in the validation for the trained time-series models over the two proposed datasets.

Test Results

In this section, we will discuss in more depth the test performance of the best model trained on the participant dataset and the best model trained on the random dataset. First, we must choose the best models from among the possible ones. To do this, we will choose the best model from those trained in the participants dataset and the best model from those trained in the random dataset. To choose each model we have used the AUC_{MCC} score as a metric given its informative character in our problem. In this way, the models chosen are the base baseline trained in the dataset participants and the balanced baseline trained in the random dataset. In Figure 4.9 we show the comparative performance of both models using the MCC-threshold and Precision-Coverage curves:

- **MCC-treshold curve**: This curve perfectly reflects the inability of the model trained in the participants dataset to distinguish between the different classes. We see how the MCC value remains flat regardless of the imposed threshold. This fact means that the quality of the predictions does not depend on the model's confidence in such predictions. On the other hand, we see how the model trained in the random dataset is able to improve the quality of its predictions as we raise the classification threshold. However, we see that the MCC score grows slowly in this case, which means that if we want high quality predictions (high MCC) we will have to impose a very restrictive threshold, which will result in a poor coverage.

- **PC-curve**: In this curve we can see how misleading the precision score is in our problem. In the baseline model we see how the precision remains almost constant around 60%. This occurs since this model almost always classifies all samples as the majority class. Therefore, this curve is, approximately, representing the percentage of samples that belong to the majority class for the samples that the model predicts for the different thresholds proposed. On the other hand, the baseline balanced model also maintains a stable precision around 60%, this metric is only improved when we enter a coverage area of less than 20%. In this sense, we get the same conclusion as with the MCC-threshold curve, if we want high quality predictions, the model will only be able to label a reduced proportion of samples with respect to the total. For example, if we would like to obtain a precision of 80%, we would only be able to label 10% of all the samples.

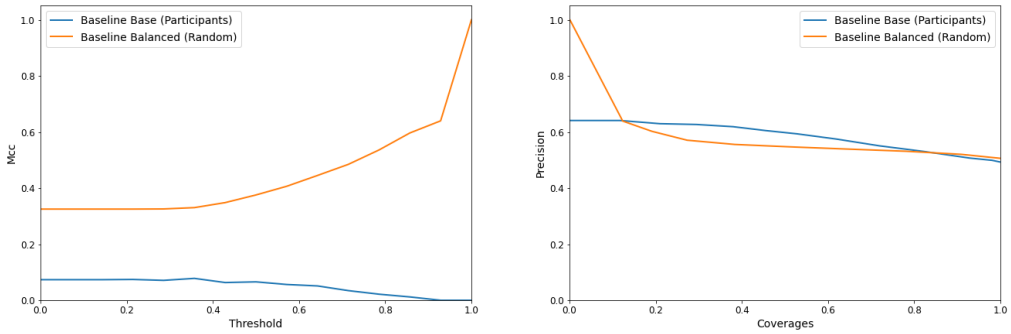

Figure 4.9: Test results for the best time-series models trained on the participants and the random datasets, respectively. We show the MCC-threshold curve (left) and the Precision-Coverage curve (right).

4.7.2 Video Models

The results for the different proposed video models can be found in Table 4.4. As we mentioned in the previous sections, the video models have been implemented with the same parameters as those used for the video segmentation task. Nevertheless, according to the results of section 3.7 where we observed that the final performance of the models did not differ too much with the learning rate adopted, we have decided to train both models with a learning rate equal to 0.01.

The results shown in this section can be applied to contrast with those obtained with time-series models. First, we observe that in general terms there is not a major discrepancy in terms of the best performances obtained. We see that the best AUC_{MCC} among all the time-series models is 0.419 while the best AUC_{MCC} among the video models is 0.317. Second, we see that the differences in performance given by the use of different loss functions is maintained in this case (higher AUC_{MCC} for the case of training with balanced cross-entropy). However, we see that in this case, the difference in performance

between the models trained in the dataset participants and in the random dataset is not present. Concretely, we see how in the case of video models, we obtain superior results in the case of models trained in the participants dataset than those trained in the random dataset.

The difference in performance between the different datasets may be due to a series of factors. First, the interparticipant variation factors discussed in section 4.1.1 (*gaze, gait*) should not have such a pronounced effect when using video data. In terms of gaze, we have already discussed that it may be almost impossible to infer gaze patterns solely based on raw video. In terms of gait, it might be also the case that the differences of the participants' gaits would be almost imperceptible in a point-of-view clip of a few seconds. However, these differences are easily visible using time-series data, both in the form and in the magnitude of the time-series. This intuition would serve to explain similar results in the case of the models trained in the different datasets, but not to see better results in the case of the dataset participants. Although this fact would be more difficult to explain, we believe that it is due to a combination of several factors. First, it could be due to a greater difficulty in the test set of the random dataset. Second, it could be due to a less informative training set in the case of the random dataset used.

	AUC_{MCC}		AUC_{PC}	
Model	Participants	Random	Participants	Random
I3D Base	0.248	0.000	**0.707**	**0.566**
I3D Balanced	**0.317**	**0.233**	0.441	0.408

Table 4.4: Summary of AUC scores in the validation for the trained video models over the two proposed datasets.

Test Results

The test results for the best video models are presented in Figure 4.10. We can see how the models trained with the balanced cross-entropy loss offer better performance regardless on the dataset used to train them. In general terms, we can observe how the performance for the model trained on the participant dataset is slightly superior for the considered metrics. However, for both models, we can notice an anomalous response on the models' performance when we increase the classification threshold imposed. In theory, if we increase the thresholds, the quality of the predictions for the respective models should increase or, at least, stay the same. In such case, the MCC should present higher or equal values for higher thresholds than for smaller thresholds. Nevertheless, we can see how neither of the models discussed present the described behaviour.

The described behaviour is detrimental for our case, as we wish to have a direct relationship between the number of samples that the model can actually label and the quality of such predictions. In this case, we see that this behaviour corresponds to what is known in the machine learning literature as a poorly calibrated model. We say that a model is *poorly calibrated* when the probability associated with the predicted class label does not reflect its ground truth correctness likelihood. Therefore, if we wanted to deploy

this model in a real-life scenario, we should calibrate our model. Confidence calibration [11] is a post-processing step in which we change the output confidences of the model for each class in order to reflect, as much as possible, the real likelihood of such confidence.

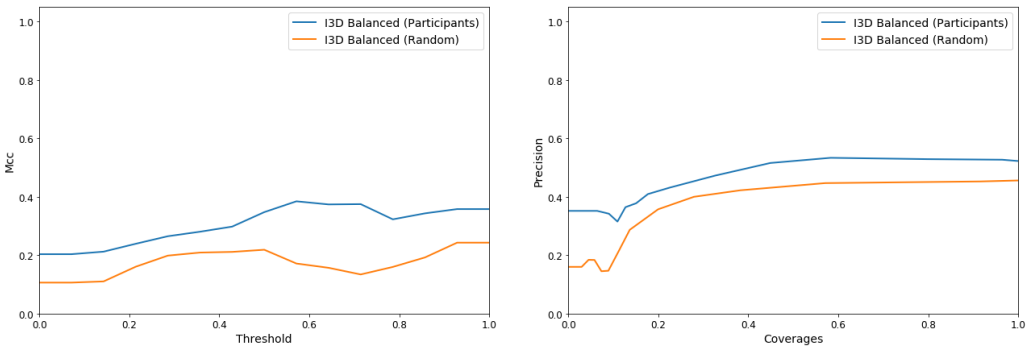

Figure 4.10: Test results for the best video models trained on the participants and the random datasets, respectively. We show the MCC-threshold curve (left) and the Precision-Coverage curve (right).

4.7.3 Active Learning

For the final section, we have performed an analogous active learning training loop as presented in section 3.7.3. We have tested the effectiveness of such process for the best four models presented in the previous sections. Note that we have made use of the same sampling techniques to construct the two different active learning datasets. In other words, for the participants' active learning splits, the samples of each participant only appear in one of such splits. On the other hand, we have selected randomly the samples for each split in the random active learning splits. We can see how in the case of the time-series models (see Figures 4.11, 4.12) this is detrimental, especially in the case of the baseline balanced model trained on the random dataset.

With respect to the video models, we can see how this process is also negative for the case of the model trained on the participants dataset. However, we observe how beneficial this training loop was for the model trained on the random dataset (the model is able to double its initial AUC_{MCC}). We argue that this difference in effectiveness for the case of the video models could be explained through the intuition discussed in section 4.7.2. In this section, we argued that one of the causes for the discrepancy in performance between the models with respect to the dataset set employed (participant, random) was how informative each of their respective training sets were. Hence, we could see how "lucky" we were in the case of the participants dataset, where its training set was highly informative. In contrast, the training set for the random dataset was considerable less informative than in the other case. Therefore, the more active learning training loops we perform with the model trained on the random dataset, the more informative samples we feed to the model. Consequently, explaining the overall increase in performance for the model trained on the random dataset. Finally, note how the peak AUC_{MCC} scores are approximately equal for

both video models which could indirectly prove our hypothesis about how the effect of the factor of variance diminished when using video data.

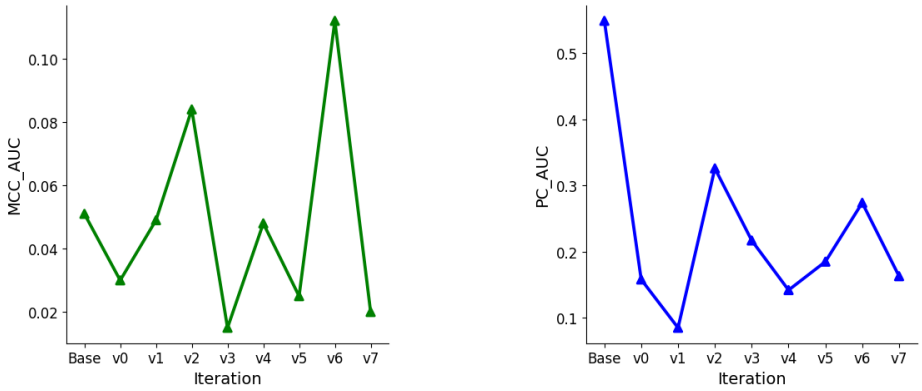

Figure 4.11: Evolution of AUC_{MCC} (left) and AUC_{PC} (right) over the course of the Active Learning training loop for the baseline base (participants dataset)

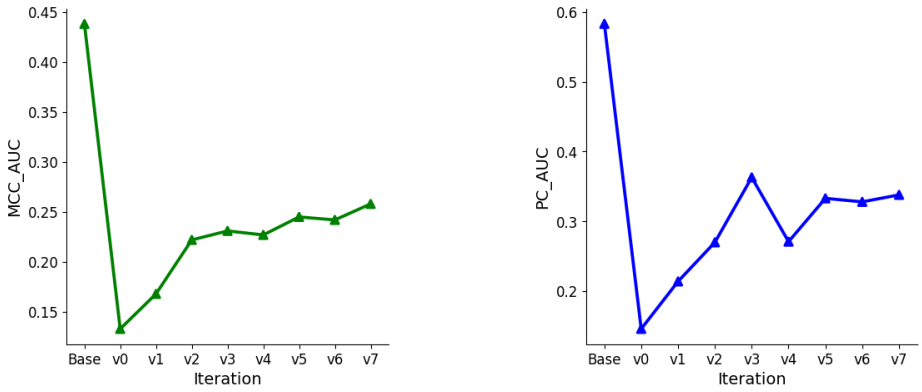

Figure 4.12: Evolution of AUC_{MCC} (left) and AUC_{PC} (right) over the course of the Active Learning training loop for the baseline balanced (random dataset)

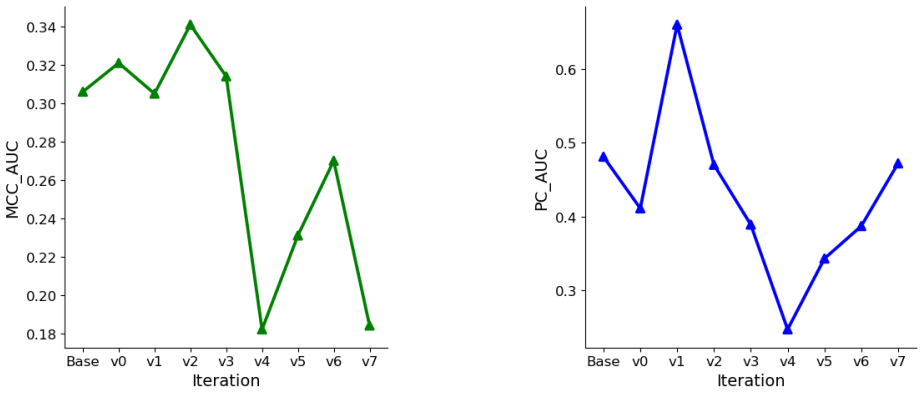

Figure 4.13: Evolution of AUC_{MCC} (left) and AUC_{PC} (right) over the course of the Active Learning training loop for the I3D balanced model (participant dataset).

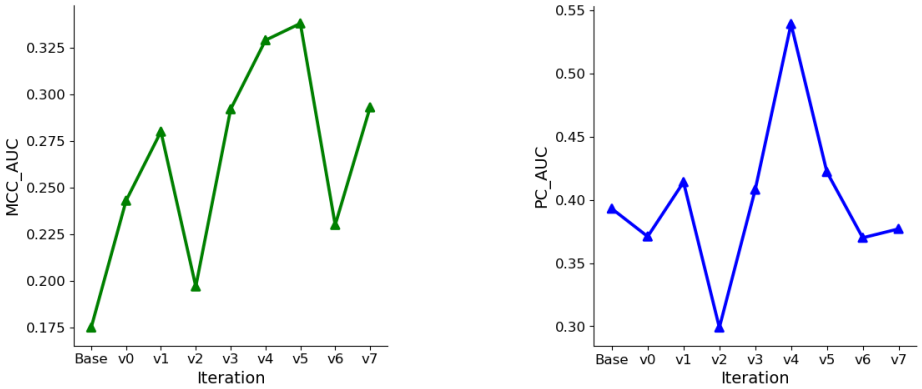

Figure 4.14: Evolution of AUC_{MCC} (left) and AUC_{PC} (right) over the course of the Active Learning training loop for the I3D balanced model (random dataset).

4.8 Conclusions

In this chapter, we have formulated a human annotation task as a machine learning problem. We have converted the task of segmenting a data stream (video, sensor data) into a series of separate behaviours into a classification task. Moreover, we have provided an overview of the dataset building process and an overview of the characteristics of the possible behaviours. This discussion has been the foundation for the remaining of the chapter. Firstly, we have seen the possible limitations of this dataset (high variance, class imbalance) and how we could try to mitigate them. Moreover, this discussion has served to provide the practical intuitions needed to select the final approaches to solve the task at hand. Lastly, this overview served to propose two different ways of arrange the different samples available into two separate datasets: Participants dataset and Random dataset.

In the participants dataset, we select the samples for each possible split in such a way that the samples for each participant could only belong to one possible split. On the other hand, the random dataset was arranged by employing the standard dataset building

approach, we selected each sample randomly for each split regardless of which participant it belonged.

We have observed in this chapter how the proposed task was more cumbersome than the video segmentation task tackled in Chapter 3. The reason for this fact could be explained as a combination of several factors:

- The **complexity of the classes** we wished to predict. We have seen how the definitions for the behaviours in our dataset have been created with such fine-grained detail that we might expect difficulties when trying to classify them.

- The **amount of labeled data** available to train the different models proposed. We have decided to employ deep learning approaches in our work for three reasons: flexibility, lack of domain knowledge and ability to extract meaningful features from raw data. However, the performance of any deep learning model relies greatly on the amount of data available. In our work, we have counted with a reduced number of recordings (less than one hundred) and one of the main goals was to be able to utilize as little as possible from these data to train our models. Therefore, this factor has been a negative point in our research. We argue that some of the models proposed could benefit greatly in similar projects where the available training data was more abundant.

- The diverse **sources of variance** for the available data. We have discussed how completely different raw data could correspond to the same behaviour given the nature of the possible behaviours. Moreover, we have seen (especially for the models that employ time-series data) how the difference in gaze patterns and gait across the different participants limits the generalization capabilities of the proposed models. To mitigate this problem, we believe that we could opt for a *hybrid labeling* scheme. In the natural scheme, a human annotator would label the data from a set of participants and our model would classify the remaining participants. In contrast, in the hybrid approach, we could split each participants' recordings in half. Then, the human annotator would label the first halves of each recording and our model would be trained using these annotated samples. In this way, the new input data would contain information of every participant which would make easier for the hypothetical model to generalize better.

- The **class imbalance** for the available samples in our dataset. It is natural that, given an array of possible behaviours that a person can display during an art exhibition, some behaviors would be more frequent than others. However, we have encountered a predominant behaviour (*Object Scan*) that capitalize 60% of the total samples in our dataset. Moreover, there are behaviours in the other extreme, which count with almost negligible representation. We proposed a metric learning training set-up to alleviate this problem. We based the election of this approach on the notion of separability. We hypothesized that utilizing this training approach we would be able to separate the samples from each class in the feature space. However, we have seen how the effectiveness of this approach has been less than expected. In future work, we believe that the utilization of metric learning to tackle class imbalance would

be beneficial given two assumptions. First, in case of having more training data available, the model might be able to learn to create more separable embeddings. Moreover, the metric learning approach employed in our work could be considered relatively basic. Therefore, the use of more advanced techniques should be explored.

Appendix A

Appendix A: Behaviours Taxonomy

Art Gaze

- **Orientation:** Viewing the exhibition with limited bodily movement
  - **Major Orientation (Room related):** Orienting oneself with respect to the room as a whole
    - **Scans:** Moving eyes/head in order to gain a full view of the gallery and the art on display
      - **Full Scan:** Entering room and looking at more than half of the pieces of artwork on display before approaching any specific artwork
      - **Non Entry Scan:** Looking around the room to see what artwork is on display after already having spent time in the room
      - **Absent:** Entering room and moving to first fixated object without looking for other artwork on display
    - **Quick Looks:** Briefly glancing toward and away from unattended area of the room
      - **Forward:** Glancing toward area of room which has not yet been attended
      - **Backward:** Glancing toward area of room which has previously attended
  - **Minor Orientation (Art related):** Orienting oneself with respect to an individual painting or group of paintings
    - **Long Gaze:** Maintaining fixation on one feature of the piece for 3 seconds or longer
    - **Object Scan:** Moving eyes over the artwork without maintaining fixation on any particular feature
    - **Mirror Gaze:** Viewing a piece indirectly by scanning its reflection in one of the mirrors which function as installation elements
    - **Alternating Gaze:** Moving eyes back and forth between two disparate items in the gallery
      - **Within Group:** Alternating gaze between two paintings within a particular grouping
      - **Between Group:** Alternating gaze between two distinct painting groups
      - **Between Art/Architecture:** Alternating gaze between paintings and architectural features
- **Changing Perspective:** Moving body to view art from a new angle/perspective
  - **Turn:** Physically turning body 90 - 270 degrees to face a new direction
  - **Strolling Along:** Scanning artwork while walking beside it without pausing to view the artwork in more detail
  - **Centring:** Moving body directly in front of artwork to view from a 90 degree angle
  - **Explorative Walk:** After viewing a freestanding installation piece from one side, physically moves to view elements of the same piece which were obscured from view
  - **Pull Back:** Returning to view a particular piece after having begun to move away from it
  - **Distance Shift:** Changing viewing distance while viewing the same piece of artwork
    - **Approach:** Moving from a distant view of the artwork to view it from ideal viewing distance*
    - **Fixation Walk:** Approaching a particular piece of art while maintaining fixation on one salient feature, moving from a distant view to ideal viewing distance
    - **Retreat:** Moving away from the artwork, starting at ideal viewing distance, to view the artwork from a far distance
    - **Zoom In:** Approaching the artwork to view the details from a very close distance
    - **Zoom Out:** Returning to ideal viewing distance to view the piece as a whole after zooming in
  - **Angle Viewing:** Viewing a particular piece of art from a shallow angle to observe the depth of the piece
    - **Close:** Angle viewing while zoomed in/directly beside the piece
    - **Distant:** Angle viewing while not zoomed in on a particular piece

**ideal viewing distance is defined as the distance at which a particular piece can be viewed as a whole while appreciating detail, i.e. when the painting fills 80-100% of the eye-tracking video*

Non-art Gaze

- **Person Oriented:** Looking at people within the exhibit
  - **Social:** Looking at other individuals in the gallery
    - **Guides:** Speaking with/receiving explanations from guides in the exhibition
    - **Other Patrons:** Briefly looking at other visitors in the exhibition
    - **Visiting Partners:** Looking at/conversing with friends while exploring the exhibition
  - **Self:** Looking at oneself while in the gallery
    - **Mirror viewing:** Looking at oneself in the mirror installation (Room 2)
    - **Body Viewing:** Looking down at one's own body
- **Object Oriented:** Looking at non-art objects in the gallery
  - **Reading:** Reading the information provided about the exhibit
    - **Pamphlet:** Reading the paper guide provided at the start of the exhibit
  - **Phone:** Looking at/using one's phone
    - **Texting:** Looking down at one's phone to text/use apps
    - **Photography:** Taking one's phone out to take photos of the exhibition
  - **Architecture:** Looking at non-installation architectural features (floors, ceilings, windows)

Orientation	Behaviour	Criteria
		All orientation behaviours occur without moving the body (or moving minimally, ex. walking slowly while conducting Scan)
Major	Full Scan	Upon entry to a new room/area, participant moves eyes across room to fixate on more than half of the pieces on display.
	Non Entry Scan	Participant moves eyes across room to fixate on more than half of the pieces on display after approaching at least one piece.
	Absent Scan	Upon entry to a new room/area, participant does not move eyes across the room and instead moves directly toward the first piece upon which s/he fixates.
	Quick Look Forward	Participant briefly (fewer than 4 seconds) looks toward an area of the room or piece of artwork which s/he has not yet approached.
	Quick Look Backward	Participant briefly (fewer than 4 seconds) looks toward an area of the room or piece of artwork which s/he has previously approached and/or viewed in detail.
Minor	Long Gaze	Participant maintains fixation for 3 seconds or more.
	Object Scan	Participant moves eyes across single piece of art with no fixations lasting longer than 3 seconds.
	Mirror Gaze	Participant conducts an object scan or long gaze while viewing the reflection of a piece of art or installation architectural feature.
	Alternating Gaze Within Group	Participant moves eyes between two or three pieces within a set of grouped paintings. Participant must view each piece at least twice in succession without shifting her/his gaze to another piece.
	Alternating Gaze Between Groups	Participant moves eyes between two or three non-grouped pieces. Participant must view each piece at least twice in succession without shifting her/his gaze to another piece.
	Alternating Gaze Between Art & Architecture	Participant moves eyes between at least one painting and one installation architectural feature (bricks, wall gradient, mirror, etc.). Participant must view each piece at least twice in succession without shifting her/his gaze to another piece.

Changing Perspective	Behaviour	Criteria
	Turn	Participant moves body 90º to 270º to face/walk in a new direction.
	Strolling Along	Participant walks alongside a painting, series of paintings, or installation architectural feature while viewing it/them without stopping to view the piece in more detail.
	Centring	Participant begins at a shallow-moderate (0º-70º) angle relative to a piece and moves body to stand at ideal viewing distance and view the piece at a perpendicular (90º) angle. The majority of fixations during this behaviour remain on the piece toward which the participant is centring.
	Explorative Walk	While viewing a piece of installation architecture, participant walks around the structure to view an element of it which was obscured from his/her original vantage point (interior wall, hidden gradient, etc.). The majority of fixations during this behaviour remain on the piece which the participant is exploring.
	Pull Back	While looking at one piece, participant starts to move away to briefly view another piece (fewer than 10 fixations and fewer than 5 seconds), then returns gaze to examine the original piece in greater detail. This behaviour includes the fixations on the interrupting piece and the next scan of the original piece until another clear behaviour is exhibited.
Distance Shift	Approach	Participant begins at a far distance and moves to approach the piece and view at ideal viewing distance.
	Fixation Walk	Participant approaches a piece of art while maintaining fixation on one feature of the art. The participant begins fixation at a large distance and moves toward the piece, stopping the approach when ideal viewing distance has been achieved.
	Retreat	Participant begins at ideal viewing distance and moves further away from the piece to view from afar.
	Zoom In	Participant begins at ideal viewing distance and moves to approach the piece and view details at a close distance.
	Zoom Out	Participant begins at a close distance and moves to return to ideal viewing distance while maintaining a view of the piece (i.e. walks backward while facing the piece).
Angle View	Close Angle View	Participant views the piece from a shallow (0º-45º) angle at a close distance (while zoomed in). This is primarily used to examine the sides/depth of the marble pieces.
	Distant Angle View	Participant views the piece from a shallow (0º-45º) angle at ideal viewing distance or further away.

Non-Art	Behaviour	Criteria
Person	Social Guides	Participant looks at one of the museum guides while speaking to/receiving information about the exhibit from the guide.
	Social Partners	Participant looks at the individual with whom s/he is exploring the gallery. This can, but need not, occur while speaking with the visiting partner.
	Social Other Patrons	Participant briefly looks at other patrons who are exploring the gallery. This includes museum guides if the participant is not directly interacting with them.
	Self Mirror	Participant looks at her/himself in the mirrors installed in the exhibit.
	Self Body	Participant looks down at her/his own body directly.
Object	Reading Pamphlet	Participant looks at/reads the informational guide about the exhibit.
	Phone Photography	Participant looks at her/his phone while taking photos of the exhibit or her/himself.
	Phone Texting	Participant looks at her/his phone for purposes other than photography (texting, social media, other apps).
	Architecture	Participant looks at non-installation architectural features, such as windows, floors, roofs, non-gradient walls, doorways, etc.

Appendix B

Appendix B: Metric Learning Hyperparameter Optimization

To find the best hyperparameters for the time-series model, we have run an optimization process consisting of three stages: *Metric Learning*, *Optimization* and *Embedder*. For each stage, we maintain the rest of the model's hyperparameters as specified in section 4.6 and we try to find the optimal values for the group of hyperparameters of each stage. Once we have found the optimal values of one stage, we use these values when running the next stages. Table B.1 is provided as summary of the global optimization process. We show the hyperparameters optimized in each stage, the values considered and their optimal values.

Note that we could have run a global hyperparameter optimization scheme considering all the hyperparameters at the same time. While feasible in theory, the possible combinations for the hyperparameters' values grow exponentially with the number of hyperparameters considered. Hence, the computational cost of this process would have resulted much larger in the case of considering all the hyperparameters at once. On the other hand, if we find groups of hyperparameters which affects the performance of the model independently, we can tune the hyperparameters of each group by separate.

In general terms, the idea of any hyperparameter optimization process could be described as follows: we have an objective function which value we wish to maximize or minimize depending on the process and we have a set of constraints that we have to satisfy when computing the objective function. In our case, we want to maximize the quality of the embeddings computed and the constraints are the architectural relationships of the weights of our specific model, the loss function, the training time and the data fed to the model. Finally, we have measured the quality of the embeddings with the metric proposed on [27]. In this work, they proposed the Mean Average Precision at R (MAP@R), which combines the ideas of Mean Average Precision and R-precision.

We have set the number of epochs of each training loop to 100. For the two first stages, we have tuned the hyperparameters running 20 iterations of Bayesian Optimization [33] with the possible values specified in Table B.1. The basic idea of Bayesian optimization is to build a probability model of the objective function to optimize and utilize it to select the most promising hyperparameters to evaluate in the true objective function. Finally, for the dimension of the embeddings, we have run a grid search with the values in [64,128,256,512].

Stage	Parameter	Type	Values Range	Optimal Value
Metric Learning	m_{pos}	Range	[0.01,0.3]	0.0613
Metric Learning	m_{neg}	Range	[0.5,1.5]	0.6860
Metric Learning	ϵ	Range	[0.1,0.3]	0.2366
Optimization	η	LogScale	[1e-6,1e-3]	$5.12e-3$
Optimization	T_0	Choice	[5,10,15,20]	10
Optimization	T_{mult}	Choice	[1,2,4]	1
Embedder	EmbDim	Choice	[64,128,256,512]	256

Table B.1: Summary of the hyperparameter optimization process for the time-series models. We present the parameters considered, the stage where they where optimized and the found optimal values.

Appendix C

Appendix C: Google Colab (GPU Virtual Machine) Specifications

Parameter	Value
GPU	Nvidia K80 / T4
GPU Memory	12GB / 16GB
GPU Memory Clock	0.82 GHz /1.59 GHz
Performance	4.1 TFLOPS / 8.1 TFLOPS
Support Mixed Precision	4.1 No / Yes
No. CPU Cores	2
Available RAM	12 GB
Disk Space	358 GB
Max execution time	12 hours
Max idle time	90 min

Bibliography

[1] A. J. Bagnall, A. Bostrom, J. Large, and J. Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *CoRR*, abs/1602.01711, 2016.

[2] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *CoRR*, abs/1306.6709, 2013.

[3] J. S. Benjamins, R. S. Hessels, and I. T. C. Hooge. Gazecode: Open-source software for manual mapping of mobile eye-tracking data. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research Applications*, ETRA 18, New York, NY, USA, 2018. Association for Computing Machinery.

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321357, June 2002.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017.

[8] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *CoRR*, abs/1812.03982, 2018.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS14, page 26722680, Cambridge, MA, USA, 2014. MIT Press.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017.

[12] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha, A. Zhang, H. Zhang, Z. Zhang, Z. Zhang, S. Zheng, and Y. Zhu. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020.

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR 06, page 17351742, USA, 2006. IEEE Computer Society.

[14] B. Harwood, V. Kumar B.G., G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2840–2848, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[16] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017.

[17] R. S. Hessels, D. C. Niehorster, G. A. Holleman, J. S. Benjamins, and I. T. C. Hooge. Wearable technology for real-world research: Realistic or not? *Perception*, 49(6):611–615, 2020. PMID: 32552490.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):17351780, Nov. 1997.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[20] F. G. W. J. e. a. Ismail Fawaz, H. Deep learning for time series classification: a review. *Data Min Knowl Disc*, 33:917963, 2019.

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[22] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[23] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Prog Artif Intell*, 5:221232, 2016.

[24] A. Le Guennec, S. Malinowski, and R. Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy, Sept. 2016.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[27] K. Musgrave, S. Belongie, and S.-N. Lim. A metric learning reality check, 2020.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] R. Quian Quiroga and C. Pedreira. How do we see art: An eye-tracker study. *Frontiers in Human Neuroscience*, 5:98, 2011.

[30] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[31] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[32] E. Smirnov, A. Melnikov, S. Novoselov, E. Luckyanets, and G. Lavrentyeva. Doppelganger mining for face representation learning. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1916–1923, 2017.

[33] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[34] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1857–1865. Curran Associates, Inc., 2016.

[35] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. *CoRR*, abs/1511.06452, 2015.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):19291958, Jan. 2014.

[37] Y. Suh, B. Han, W. Kim, and K. M. Lee. Stochastic class-based hard example mining for deep metric learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7244–7252, 2019.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[39] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 140–153, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[40] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[41] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, abs/1507.02159, 2015.

[42] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. *CoRR*, abs/1904.06627, 2019.

[43] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline, 2016.

[44] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR*, abs/1611.06455, 2016.

[45] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207244, June 2009.

[46] Q. Yang and X. Wu. 10 challenging problems in data mining research. *Int. J. Inf. Technol. Decis. Mak.*, 5:597–604, 2006.

[47] T. Yuan, W. Deng, J. Tang, Y. Tang, and B. Chen. Signal-to-noise ratio: A robust distance metric for deep metric learning. *CoRR*, abs/1904.02616, 2019.

[48] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

Master's thesis filing card

Student: Jesus Garcia Ramirez

Title: Efficient analysis of mobile eye tracker data using Deep Learning

UDC: 621.3

Abstract:
Understanding how people navigate and interact with their environment is an important research question for various disciplines (e.g. psychology, cognitive sciences, social sciences). One important mechanism linked to a person's experience is attention. One of the possible strategies for capturing the attentional processes of a person is by studying her eye gaze patterns. Recently, the appearance of mobile eye trackers (MET) has been presented as a possible step forward towards bridging research with the "real-world". Through the utilization of these devices, researchers can record the eye gaze patterns and the point of view of an individual while exploring freely the environment. Although such devices might hold considerable potential, their adoption in the research community has been hindered by some practical limitations. One of the main practical issues is the cost of manually labelling the eye-tracking obtained with the employment of MET.

In our work, we wish to provide some automatic labelling tools that aim to alleviate, as much as possible, the overall manual workload needed to process eye-tracking data. In particular, we have worked with a set of mobile eye-tracking recordings that captured how a series of participants navigated through a prestigious art exhibition. We can consider diverse aspects of interest when analysing the obtained recordings. Hence, the first contribution in our work has been to translate these different focuses of analysis into a structured data pipeline. This pipeline consists of three stages: *Video Segmentation*, *Behaviour Segmentation* and *Image Registration*.

Finally, we have provided practical approaches for the Video Segmentation and Behaviour Segmentation tasks. For the former stage, we have proposed a video classification model to predict the item seen by the participant in any given segment. We see how the results obtained with our approach are satisfactory, except for a couple of minor limitations. In the Behaviour segmentation stage, we have proposed two different types of models: one using video input and another using time-series data (accelerometer, gaze and gyroscope data) to predict the behaviour taken by a participant in any given segment. Finally, we will see in the results how this task is more cumbersome than the video segmentation task and we will discuss the strengths and limitations of the proposed models.

Thesis submitted for the degree of Master of Science in Artificial Intelligence, option Engineering and Computer Science

Thesis supervisor: Prof. dr. Johan Wagemans

Assessor: Dr. Robin De Croon, Dr. Lore Goetschalckx

Mentor: Dr. Christophe Bossens