Tackling automatic audience experience measurement in online environments

Abordando la medición automática de la experiencia de la audiencia en línea

Por
Pablo Villalobos Sánchez y Eduardo Rivero Rodríguez

UNIVERSIDAD COMPLUTENSE
MADRID

Trabajo de fin de grado del
Doble Grado en Ingeniería Informática y Matemáticas
Facultad de Informática

Dirigido por:
Borja Manero Iglesias
Meriem El Yamri El Khatibi

Madrid, 2020–2021

Abstract

The availability of automatic and personalized feedback is a large advantage when facing an audience. An effective way to give such feedback is to analyze the audience experience, which provides valuable information about the quality of a speech or performance. In this document, we present the design and implementation of a computer vision system to automatically measure audience experience. This includes the definition of a theoretical and practical framework grounded on the theatrical perspective to quantify this concept, the development of an artificial intelligence system which serves as a proof-of-concept of our approach, and the creation of a dataset to train our system. To facilitate the data collection step, we have also created a custom video conferencing tool. Additionally, we present the evaluation of our artificial intelligence system and the final conclusions.

Keywords – computer vision, machine learning, sentiment analysis, emotion recognition, object tracking, affective computing, WebRTC

Resumen

La disponibilidad de feedback automático y personalizado supone una gran ventaja a la hora de enfrentarse a un público. Una forma efectiva de dar este tipo de feedback es analizar la experiencia de la audiencia, que proporciona información fundamental sobre la calidad de una ponencia o actuación. En este documento exponemos el diseño e implementación de un sistema automático de medición de la experiencia de la audiencia basado en la visión por computador. Esto incluye la definición de un marco teórico y práctico fundamentado en la perspectiva del mundo del teatro para cuantificar el concepto de experiencia de la audiencia, el desarrollo de un sistema basado en inteligencia artificial que sirve como prototipo de nuestra aproximación y la recopilación un conjunto de datos para entrenar el sistema. Para facilitar este último paso hemos desarrolado una aplicación de videoconferencias personalizada. Además, en este trabajo presentamos la evaluación de nuestro sistema de inteligencia artificial y las conclusiones extraídas.

Palabras clave – visión por computador, aprendizaje automático, análisis de sentimiento, reconocimiento de emociones, seguimiento de objetos, computación afectiva, WebRTC

Acknowledgements

We woud like to thank Borja Manero Iglesias, Alejandro Romero Hernández, and Meriem El Yamri El Khatibi for their direct contributions, feedback, and continuous support throughout the entire year. Without them this would not have been possible. We also thank Scott, who helped us correct the paper; our families and friends, who have been supporting us until now; and the volunteers in our experiment, who generously lent us their time and patience.

Contents

Chapter 1

Introduction

Public speaking is a core skill in the modern world. Both educators and learners share a common desire towards better teaching methods for this elusive skill. In particular, automated feedback systems for public speaking training have sparked the interest of researchers due to their promise of objective and personalized advice at a massive scale.

There have been various proposals of this kind of system over the last years and, with the recent advances in machine learning (ML) and affective computing, the capabilities of these systems have increased. The majority of the proposed systems directly evaluate the verbal and nonverbal behavior of the speaker and they are usually trained from expert ratings of public speaking performances.

However, the true judge of public speaking skill is the audience, which collectively decides which speakers are inspiring and which are not. Needless to say, these audience evaluations are not explicit, but remain implicit in their experience of the performance. Furthermore, the process that determines the nature of that experience is poorly understood, even by the audience members themselves.

It follows that, if we had quantified and automatic access to the inner experience of audience members, this would constitute a true gold standard for measuring speaker performance. Thus, the focus of our work is achieving an automated assessment of audience experience.

How to use this document In this document, we will reference several concepts from machine learning and affective computing. While we provide a glossary for reference, we will assume the reader is familiar with these topics.

We will document our work with as much detail as possible, and will provide working code for every software component used. The code is published under the MIT license, and the reader is welcome to use it and improve upon it.

Our goals As mentioned before, our end goal is producing an automated system for accurately measuring audience experience. However, this is no small task, and a fully functional, ready to use system is out of reach for this experimental work. Therefore, we clarified our purpose by dividing it into three goals.

1. Determining a practical framework to quantify audience experience. This framework should be grounded in existing theory as well as in empirical data. Ideally,

this framework would be an already existing one. However, in our research we found that existing frameworks are not entirely adapted to our requirements, which forced us to develop one of our own.

2. Compiling a dataset that can be used to train ML models to measure audience experience using our framework.

3. Develop a proof-of-concept system to automatically measure the audience experience that is both lightweight and fast.

Our work We have achieved our three goals with different degrees of success. In particular, we have

- Designed a conceptual framework to quantify audience experience.
- Developed a custom video conferencing tool to facilitate experimental data collection.
- Created a small audience experience dataset.
- Designed a lightweight fast ML system to measure audience experience.
- Developed a tracking algorithm that improves the performance of the matching phase of existing track-by-detection systems.
- Produced an academic paper summarizing our work, titled *Tackling the design and evaluation of a theater-based intelligent system to monitor audience experience in virtual public speaking settings*, and submitted it for review and publication to the the 29th International Conference on Computers in Education (ICCE 2021). The full content of this paper can be found in Appendix B.

Document structure This document is structured in the following manner. There are six chapters, this introduction being Chapter 1. In Chapter 2, we present an overview of the current state of the art in the different fields of interest of our work. In Chapter 3, we introduce our framework, the data collection process, and our dataset. Chapter 4 and Chapter 5 detail the architecture and implementation of our ML system. Chapter 4 is focused on the tracking module while Chapter 5 describes the emotion recognition module. Finally, Chapter 6 is dedicated to presenting our conclusions.

In addition to the main chapters, we provide the code for our videoconference tool, which can be found in Appendix A, the content of our submitted paper and the submission confirmation, which can be found in Appendix B, and a glossary of terms for the convenience of the reader. These can be found at the end of the document.

1.1 Workplan

In this section we will briefly present the plan we followed from the inception of this work. Our planification was structured around a series of semi-regular meetings with our supervisors, and the plan naturally evolved as we learned more.

- **First approximation:** We identified the main functionalities that would need to be provided by the system: real-time agent detection and tracking, and emotion recognition. We quickly identified YOLO as a promising algorithm for the first of these. We also started to explore the literature on emotion detection and audience

experience evaluation. We soon had the idea of performing experiments to test our tracking algorithm and gather data for training.

- **Onset of COVID and initial design:** The start of the pandemic made clear that in-person experiments would have to be ruled out due to safety concerns. Therefore, the focus of our proposed experiment changed to the online setting and we decided to focus our experimental efforts on collecting training data. We also realized at this point that there were no satisfactory frameworks for measuring audience experience in this setting. As a consequence of all of this, we decided to focus on creating a prototype of the tracking system and experience evaluation framework, since the emotion detection module is dependent on both of them.
- **Developing an experimental methodology:** Having defined and tested a first version of the tracking module, and prepared a first theoretical approach to experience evaluation, as well as a simple self-report questionnaire to measure it, we started preparing our data collection experiment. We developed the general approach of streaming video performances and recording through webcam. Our efforts over the next month would be directed at finding a set of video performances, refining the questionnaire, and developing a detailed methodology for the experiment.
- **Video conferencing and paper:** At this stage, our progress was hindered by the lack of adequate video conferencing tools. In the process of exploring the different solutions we made a minimal prototype video conferencing tool, and realized that the benefits of having a custom tool exceeded the cost. At the same time, the opportunity of writing a paper was presented to us. Thus, our efforts were directed towards creating a usable version of the tool and coming up with our final experimental design, to be able to perform the experiment before the paper submission deadline. The development of the emotion recognition module was not prioritized at this moment.
- **Final design:** After further refinement, we finally started the search for volunteers for our experiment, and were ready to tackle the design an emotion recognition module that would be able to learn from the experiment. This was our last phase of design work.
- **Experimentation and evaluation:** After performing a pilot study, we made some final tweaks to our methodology and performed the final experiment. The remaining time was spent training and testing our system with the dataset, and documenting our conclusions.

1.2 Individual contributions

In this section, we will detail the individual contributions each one of us made to our work, as well as this document. It is important to note, however, that every one of our contributions was thoroughly reviewed by the other author to guarantee the highest quality. Furthermore, some contributions, like the experimental design or the introduction and conclusion of this document, required a higher degree of coordination and thus were partially or totally overlapping. For this type of contribution, we will focus on explaining the specific role each author played.

The main difference between our work is that the problem we are tackling requires in-

depth knowledge of very distinct areas. On one hand, communication and affective computing play a key role in the design of our framework to quantify the audience experience and in the experimental design. On the other hand, machine learning and computer vision make up the foundation of our system. This results in each one of us working for the most part either on the machine learning side or on the affective computing side of our work.

1.2.1 Eduardo Rivero Rodríguez

Initially, I was in charge of researching the different techniques and approaches used in object tracking and emotion recognition, what we had initially identified as the 2 main components of our machine learning system. As it will be highlighted in Chapter 2, these fields are very diverse and, therefore, required some narrowing down in order to focus the research on the techniques that best fit our specific needs.

After studying some of the most successful techniques in object tracking, I opted to favor track-by-detection methods because of their simplicity and ease of implementation. Furthermore, the chosen detector, YOLO, has readily available implementations in several programming languages that can be adapted to tailor application-specific needs. The rationale behind this choice is further discussed in Chapter 4.

Once the general tracking technique and the detector backbone were chosen, the matching part of the tracking algorithm had to be designed. Initially, I proposed the metric $m(\cdot, \cdot)$ described in Chapter 4 and designed a naïve algorithm that simply matched each identity to the closest detected object based on the position in the last frame. This approach did not yield the expected results and it was slowly refined up to the version described in Chapter 4. I also implemented the tracking algorithms, although the `naive_track` function was adapted from previous code written by Pablo.

Once the tracking module was functional, it did not make sense to design the emotion recognition module until we had workable data. Thus, I worked with Pablo on the experimental design. I was in charge of creating the questionnaires on Google Forms, creating the consent form, selecting the videos that would be played during the experiment, and communicating with the volunteers before the experiment. I also analyzed the data obtained from both the pilot test and the actual experiment, providing statistics, graphs, and the scores that would become the labels of the final dataset.

With a dataset to work with, I designed the emotion recognition module and chose the testing metrics as explained in Chapter 5. I also wrote the code to create the module, to work with the dataset videos, and the custom training and validation loops required to train the model in PyTorch.

In the academic paper we produced, I wrote section 2.2, which includes a general overview of our system's architecture and a description of its implementation in subsection 2.2.1. I also wrote the interpretation of the results described in subsection 3.1.3. Both the introduction and conclusion sections were written in close coordination with Pablo. My main goals on these sections was to ensure an accurate portrayal of the role of our ML system, its relationship with the rest of our work, and the rationale of our design choices.

As for my contributions to this document, I wrote the sections of Chapter 2 related to object detection, multiple object tracking, and emotion recognition. I also wrote Chapter

4 and Chapter 5. The Abstract, Introduction, Conclusion, and Glossary were also written in close coordination with Pablo with similar goals to the ones described in the previous paragraph.

1.2.2 Pablo Villalobos Sánchez

I was initially tasked with finding a suitable method for measuring audience experience. Being an outsider to this field, finding previous work on this topic took some search effort. I eventually found the main supporting work of our framework, described in Chapter 2.

After realizing the need for a new framework, I once again delved into the literature, but eventually designed our framework by combining the features of some of the existing ones and adapting it to our needs. I also came up with the different options for taking measurements, including external annotation and self-reports. After we decided to use self-reports, I designed the first version of our questionnaire, which was later refined several times by both of us. This process is detailed in Section 3.1.

After our initial tracking system was defined, I implemented a first full prototype, using the Darknet implementation of YOLO and our custom tracking algorithm. This allowed us to perform some initial tests and modify our algorithm accordingly.

Once we became aware of the poor fit of existing video conferencing software for our needs, I developed a first prototype of our custom tool as an exploratory exercise and found it less costly than expected thanks to the versatile functionality provided by the WebRTC API. From that point I developed the final version of our custom tool. The majority of this work was spent in developing a frontend web app, but tweaking the backend software for our purposes and setting up the infrastructure took a significant part of the total development effort as well. More detail can be found in Section 3.2.1.

Since I was more familiar with the tool, I was charged with conducting the experiments. This involved explaining the process to the participants, playing the videos through the app and ensuring the correct recording of all participants, as well as answering any questions or doubts.

Since I had access to better hardware, I performed the processing of the recordings after the experiment to create our dataset, ran the training and evaluation scripts for our system, trying to find the best parameters, and performed general tests of our full pipeline with the trained system.

In our paper, I wrote section 2.1 on the context of virtual settings for public speaking. The goal in this section was conveying why our framework needed to be adapted to this setting, and why we needed to develop our own video conferencing software. I also wrote sections 3.1.1 and 3.1.2, detailing our experimental design and the resulting dataset, with the goal of clarifying our methodology to facilitate replication, as well as revealing any possible bias or confounding variables. Finally, I wrote Sections 2.3 and 2.5 of this document, in addition to Chapter 3 and the code in Appendix A.

Chapter 2

State of the Art

This chapter is dedicated to surveying the most successful methods used within the different domains this project covers. It serves therefore a double purpose. Firstly, it allows us to assess the different methods and make an informed choice that fits our needs. Secondly, it allows us to highlight potential alternative approaches to the problem we are presenting.

For the problem of emotion recognition from video feed, we have two separate systems working together: an object tracking system that detects and tracks the agents and an emotion recognition module, that is able to output the recognized emotions for each of the detected agents. We can then formulate our problem as a combination of multi object tracking (MOT) and emotion recognition. We must also note that, within the domain of MOT, the most common techniques have a direct link with those for object detection. Therefore, we will also consider this domain within our survey. In the following pages, we will detail the state of the art within each of the aforementioned domains, as well as detail some of the most promising techniques.

2.1 Object detection

Object detection is a computer vision problem where we try to detect instances of objects of one or more classes. The general approach to object detection consists of extracting certain features to represent the image and a set of region proposals (regions where there may be an object) and then using the features to predict whether the proposed regions contain an object of the desired classes.

In the past, these features were largely handcrafted and relied in the extraction of robust descriptors from the images [49][5], in the sense that they would ideally be invariant to certain transformations such as scale or translation. However, in the last decade, there has been a rise in the use of deep learning to extract these features. In particular, there has been a widespread use of convolutional neural networks (CNNs) because of their useful properties when dealing with image data.

In the following sections, we will detail the most relevant of the modern developments in the field, with particular interest in methods that are realtime-capable and have relatively low ressource requirements.

2.1.1 R-CNN

Regions with CNN Features, commonly known as R-CNN, is a 3-step object recognition method that relies on CNNs to extract features from different regions in the image which are then used for prediction. Initially, the image is processed through a region proposal system, that extracts regions of the image that may contain an object. Then, the image is passed through a CNN that transforms each region into a 4096-dimensional representation. Finally, there is a Support Vector Machine (SVM) for each object class that indicates whether there is an object in a given region and, in the affirmative case, a linear regression model indicates the position and dimensions of the bounding box.

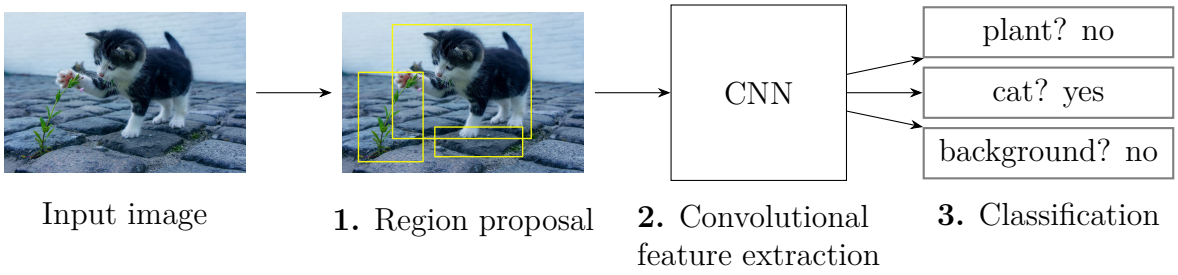

Figure 2.1: Classification process with R-CNN

Region proposal

The region proposal model relies on a particular region extraction algorithm, the most commonly used one being selective search. Selective search works in two-steps. Firstly, it makes use of Felzenszwalb and Huttenlocher's algorithm to obtain an initial set of regions. Then it proceeds to iteratively reduce the amount of such regions by merging neighboring regions with the highest degree of similarity.

Algorithm 1: Selective search [64]

Data: RGB Image
Result: Object location hypothesis $L = \{l_1, ..., l_N\}$
Obtain initial regions $R = \{r_1, ..., r_n\}$ through Felzenszwalb and Huttenlocher's algorithm
Initialise the similarity set $S = \phi$
for (r_i, r_j) *neighboring region pair* **do**
  Calculate similarity $s(r_i, r_j)$
  $S = S \cup \{s(r_i, r_j)\}$
end
while $S \neq \phi$ **do**
  Get highest similarity pair (r_i, r_j) such that $s(r_i, r_j) = max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding r_i $S = S \setminus s(r_i, r_*)$
  Remove similarities regarding r_j $S = S \setminus s(r_j, r_*)$
  Calculate similarity set S_t between r_t and its neighbors
  $S = S \cup S_t$
  $R = R \cup \{r_t\}$
end
Extract object location bounding boxes L from all regions R

The initial region proposal is calculated through Felzenswalb and Huttenlocher's algorithm, which treats images as graphs and determines a preliminary component segmentation. A monochrome (intensity) image is a graph where each pixel p_i has an associated vertex $v_i \in V$. Each one of those vertices is connected to the vertices associated with neighboring pixels. The weight of those edges is given by

$$w(e_{v_i,v_j}) = |I(p_i) - I(p_j)|$$

where $I(p_k)$ is the intensity of pixel p_k. We also need an operator to compare two separate components and, therefore, determine whether they should be merged. We consider the following functions named, respectively, the internal difference of a component $C \subset V$ and the minimum internal difference between two components

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$$

where $MST(C, E)$ is the minimum spanning tree of the component C and τ is a non-negative threshold function, usually $\tau(C) = \frac{k}{C}$ with k being a particular constant that determines (in practice) how large the final components will be. It is considered that two components should be merged when the weight of an edge joining them is smaller than their their minimum internal difference.

Algorithm 2: Felzenszwalb and Huttenlocher's algorithm [21]

Data: $G = (V, E)$
Result: Segmentation components $S = (C_1, ..., C_r)$ that partition the graph
Sort E into $\pi = (o_1, ..., o_m)$ by non-decreasing weight
Start with segmentation S^0 where each vertex is in its own component
for $q = 1, .., m$ **do**
  Let V_i, V_j denote the vertices connected by the q-th edge in the ordering
  if *V_i and V_j are in disjoint components and* $\omega(q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ **then**
    S^q is obtained by merging C_i^{q-1} and C_j^{q-1}
  else
    $S^q = S^{q-1}$
  end
end
Return $S = S^m$

Feature extraction

In the original paper [26][43], feature extraction is done through a very simple CNN architecture consisting of mean substraction and region warping (227×227) as preprocessing, 5 convolutional layers (using ReLU activation) and 2 Fully Connected (FC) layers. Each region is finally represented by a 4096-dimensional vector. There are additional max pooling and local response normalization [43] operations applied after convolutional layers $CONV1$, $CONV2$, and $CONV5$.

Prediction: classes and bounding boxes

An SVM trained for each class scores the given region feature vectors and regions having high intersection-over-union (IoU) with a higher score region (larger than a certain learned threshold) are discarded. The final version of the model also includes a linear regression model that predicts a new detection window given the max pooled results after $CONV5$, calculating new bounding boxes for the given objects [22].

2.1.2 Fast R-CNN

Fast R-CNN appears as an iteration of R-CNN that achieves a considerable speedup on its predecessor [25]. For this purpose, Fast R-CNN avoids computing the CNN features for each of the regions by extracting a global feature map from the image using a CNN. Fast R-CNN still uses Selective Search as a region proposal network before processing the image. Each region proposal, or Region of Interest (RoI), is projected onto the feature map and a specialized layer called RoI pooling extracts a lower-dimensional feature vector that will be later used for prediction. The RoI pooling layer works by applying max pooling on the feature map of the RoI. If we wished to get an $H \times W$ dimensional output and we had a RoI with top-left corner coordinates (r, c), height h, and width w; the pooling process would proceed as follows: the projected feature map (the section of the image corresponding to the RoI) is divided into a grid of $\frac{w}{W} \times \frac{h}{H}$-dimensional rectangles and then max pooling is applied on each of the rectangles.

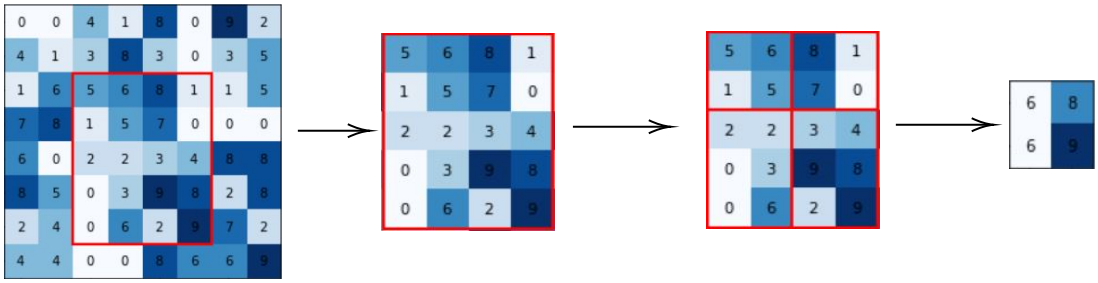

Figure 2.2: Illustration of RoI pooling ($H = W = 2$)

The last part of the architecture consists of two sibling FC layers. One of them applies softmax over $K + 1$ classes (where there are K possible object categories) and the other one is connected to class-specific bounding box regressors. It is important to note that the choice of W and H must be compatible with the size of these FC layers. The output of these two prediction layers is, respectively, $p = (p_0, ..., p_k)$ probability map over $K + 1$ categories for the object detection layer and $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$ for each of the K object classes, which indicates the bounding box regression offsets.

Another one of the key components in Fast R-CNN's proposal is the use of a multi-task loss for training, which allows for end-to-end single stage training. Each training RoI has a ground-truth class u and a ground-truth bounding-box regression target v. For such RoI, the multi-task loss is

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

where $L_{cls}(p, u)$ and $L_{loc}(t^u, v)$ are the classification and bounding-box regression losses given by

$$L_{cls}(p,u) = -\log p_u \qquad L_{loc} = \sum_{i\in\{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$$

Furthermore, $[u \geq 1]$ evaluates to 1 only when $u \geq 1$ and to 0 otherwise. The smooth L_1 loss is a variation of the L_1 loss (see Fig. 2.3) aimed at reducing sensitivity to outliers of the L_2 loss while maintaining is properties and is given by

$$smooth_{L_1}(x) = \begin{cases} \frac{1}{2}x^2 & |x| < 1 \\ |x| - \frac{1}{2} & otherwise \end{cases}$$

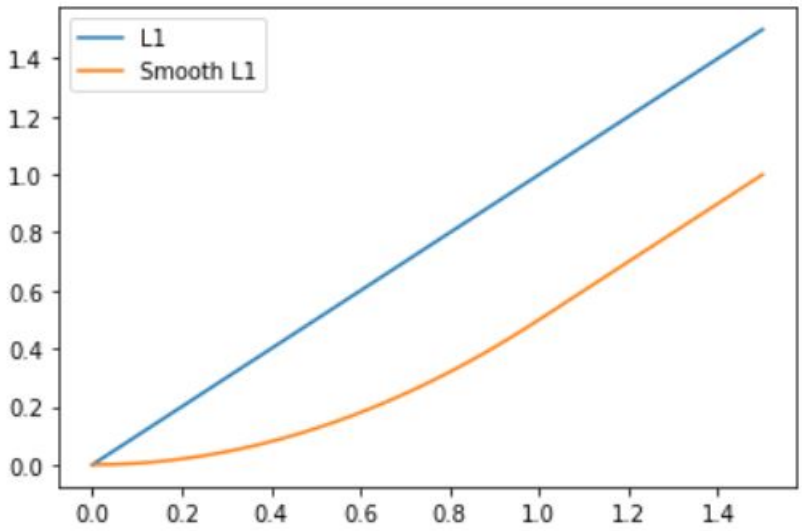

Figure 2.3: L_1 loss and $smooth_{L_1}$ loss comparison.

The above loss is a particular case of the Huber loss [37]

$$L_\delta(x) = \begin{cases} \frac{1}{2}x^2 & |x| < \delta \\ \delta(|x| - \frac{1}{2}\delta) & otherwise \end{cases}$$

with $\delta = 1$.

2.1.3 Faster R-CNN

Faster R-CNN is yet another iteration of R-CNN that addresses another one one of its problems (which it shares with Fast R-CNN): the selective search bottleneck [58]. Instead of using selective search, Faster R-CNN proposes the use of a dedicated CNN called the Region Proposal Network (RPN). An RPN receives an image as input and outputs a set of region proposals (rectangular in our case). This is achieved in practice through a CNN. Furthermore, both the RPN and the CNN for feature extraction share convolutional layers.

The RPN works in a sliding window fashion, by sliding the RPN over an $n \times n$ spatial window of the convolutional feature map outputted by the last shared convolutional layer. This window is mapped onto a smaller dimension feature vector (256-dimensional in the

original paper) which is passed onto the two prediction layers: the classification layer (*cls*) and the regression layer (*reg*). The outputs of these layers are the same as in Fast R-CNN.

Another important part of Faster R-CNN's approach is the use of anchor boxes. Anchor boxes are predefined shapes (usually rectangles) used to mimic the scale and aspect ratios of the objects of the class to be predicted. The total number of anchor boxes is $\mathcal{K} = S \cdot A$ where S is the number of scales and A the number of aspect ratios (in the original paper $S = A = 3$). Faster R-CNN uses the anchor boxes to predict $\mathcal{K}$ region proposals for each one of the sliding-window locations. This gives us the size of the outputs of the *cls* layer ($2\mathcal{K}$ scores, $p(object)$ and $p(no\ object)$) and the *reg* ($4\mathcal{K}$ scores, 4 for each predicted bounding box). This whole part of Faster R-CNNs architecture is translation invariant.

The training of Faster R-CNN uses a multi-task loss (just as Fast R-CNN) to achieve one stage learning. This function takes as input the output of the *cls* and *reg* layers for the ith anchor and is given by

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_j L_{cls}(p_j, p_j^*) + \lambda \frac{1}{N_{reg}} \sum_j p_j^* L_{reg}(t_j, t_j^*)$$

where λ is a balancing weight, N_{cls} and N_{reg} are normalization weights, p_j the $j-th$ value of $\{p_i\}$ and t_j is the $j-th$ value of $\{t_i^*\}$ where p_j^* and t_j^* are the respective ground-truth values. L_{cls} is the log loss and $L_{reg}(t_j, t_j^*) = smooth_{L_1}(t_j - t_j^*)$. As for the regression values, the coordinates are reparametrized as follows

$$t_x = \frac{x - x_a}{w_a} \qquad t_y = \frac{y - y_a}{w_a} \qquad t_w = log\left(\frac{w}{w_a}\right) \qquad t_h = log\left(\frac{h}{h_a}\right)$$

for the regression coordinates x, y, h, w denoting the midpoint coordinates (x, y), height and width respectively. The same reparametrization is applied to the ground-truth values.

The regression itself is also different from that of previous versions. A set of $\mathcal{K}$ bounding box regressors are learned, each regressor being being responsible for one scale and one aspect ratio. These regressors do not share weights and receive as input features of the same spatial size ($n \times n$).

2.1.4 YOLO

You Only Look Once (YOLO) is an object detection algorithm focused on prediction speed. While relatively accurate and very fast, it still has issues with the detection of certain objects, especially small ones [57]. YOLO's architecture is structured as follows. First, an input image is received and divided into an $S \times S$ grid, where each cell is responsible for predicting B bounding boxes. Then each bounding box is predicted by the network with a confidence score defined by

$$\text{confidence}(pred) = \text{p}(Object) \cdot \text{IoU}_{pred}^{truth},$$

where p(*Object*) is the probability of any object being in the frame and $\text{IoU}_{pred}^{truth}$ is the intersection over union of the predicted bounding box and the ground truth one (see Fig. 2.4). For each bounding box, 5 values are predicted: x, y, w, h and confidence. The first two values determine the center position of the bounding box and the last two its width and height. There are also C predictions for each bounding box (where C is the number of classes being considered) that indicate the proability of the bounding box belonging to each particular class. This prediction is achieved through a CNN and is encoded as an $S \times S \times (5B + C)$ output tensor.

Figure 2.4: YOLO confidence score calculation. On the right, predicted bounding box (red) versus ground-truth (blue) for one of the grid squares.

Once the bounding boxes have been predicted, YOLO makes use of Non-Max Suppression (NMS), where overlapping bounding boxes are removed if they predict the same type of object and have a high IoU with the bounding box that has the highest confidence score. This helps prevent duplicate detections and select the most adequate bounding box.

YOLO has undergone several iterations and improvements since it was first proposed. One of its most recent versions [9] makes use of a sophisticated feature extraction process. A pretrained or fine-tuned backbone network (VGG, ResNet, Darknet...) is connected to a neck network (FPN, Bi-FPN, PANet) in order to extract hierarchical convolutional features from the input image. These features finally feed a prediction network (RPN, YOLO, SSD, RetinaNet...) to output the bounding boxes.

2.2 Multiple Object tracking

When we talk about the Multiple Object Tracking (MOT) problem we must specify whether we are dealing with online or offline tracking. In online tracking, the system only has information about the current frame and previous ones. In offline tracking, the system has information about all frames in a recording, thus being able to use future information to adjust predictions. For the purposes of our work, we will restrict ourselves to online MOT. In the following sections we will describe some common approaches to online MOT.

Classically, object tracking schemes usually fall in one of four categories:

1. **Filtering schemes:** the name refers to the use of a Kalman filte, an algorithm used to estimate the state of dynamic systems while simultaneously keeping track of the variance of the estimate.

2. **Mean-shift methods:** a set of methods relying on the mean-shift algorithm, a mode-seeking algorithm used to find maxima in probability density functions, to perform object tracking.

3. **Template matching:** a set of techniques used to find parts of an image that match a given template.

4. **Optical flow estimation:** a set of techniques to determine apparent motion across adjacent frames. Optical flow may refer to dense optical flow, when flow vectors are calculated for the entire image, or sparse optical flow, where only the "most interesting" flow vectors are considered.

More recently, however, several alternatives have emerged with various degrees of success in either tackling object tracking by themselves or enhancing existing trackers. Some of the most relevant are:

1. **Track-by-detection methods:** they work in two steps. First, a detection module is applied to each frame in order to locate objects. Then, a tracking module associates existing object identities to the new detections. This process resembles the filtering aproach mentioned earlier, and Kalman filters are often used in track-by-detection systems [59][74][7].

2. **Particle swarm optimization (PSO):** they use a set of particles whose movements are adjusted depending on both their position and their neighbors' position to try to locate global optima. In multi object tracking, this translates to defining adequate similarity functions that are to be minimized for the objects to be tracked. Some proposed options include dividing the particle swarm into several "species" that keep track of a specific object [76] or applying PSO successively to locate each object and then associate each one of them with previous detections [45] as in track-by-detection systems. It is also common that the similarity metric compares the covariance matrix of the image patches determined by the swarm with the image templates that represent each object [38].

3. **Integration of context information:** refers to techniques that try to exploit the particular context in which the system is going to be used. Therefore, these techniques are mainly aimed at enhancing an existing tracking system by incorporating useful context information. An example of context information being used to enhance a tracking system could be an object tracking system that stores the appearance profiles of tracked objects in order to be able to reacquire targets if they are ever lost.

4. **Ensemble tracking:** ensemble tracking systems combine one or several of the above techniques (or others that have not been mentioned) to achieve a more robust system. The goal of ensemble systems is to draw on the strengths of different techniques while simultaneously covering up their weaknesses.

Out of the above methods, track-by-detection methods are the most common and can be both accurate and fast. PSO methods have seen mild success in specialized systems but

are generally outperformed by other techniques. Integration of context information has become standard and is almost always used in some way or another. Finally, ensemble tracking should be able to produce accurate results, but may be bottlenecked by the performance of each individual tracker.

For the purpose of this section, we are particularly interested in systems that achieve high precision scores while maintaining high frame rates. Therefore, we have selected 3 well-known tracking systems that achieve state of the art results both in terms of precision and inference speed: ROLO, SORT, and DeepSORT.

2.2.1 ROLO

ROLO stands for Recurrent YOLO and, as it name suggests, it make use of the YOLO network and and Long Short-Term Memory (LSTM) cells in order to perform multi-object tracking. An LSTM [33] is a specific Recurrent Neural Network (RNN) architecture that is able to capture both short and long distance dependencies in sequence data.

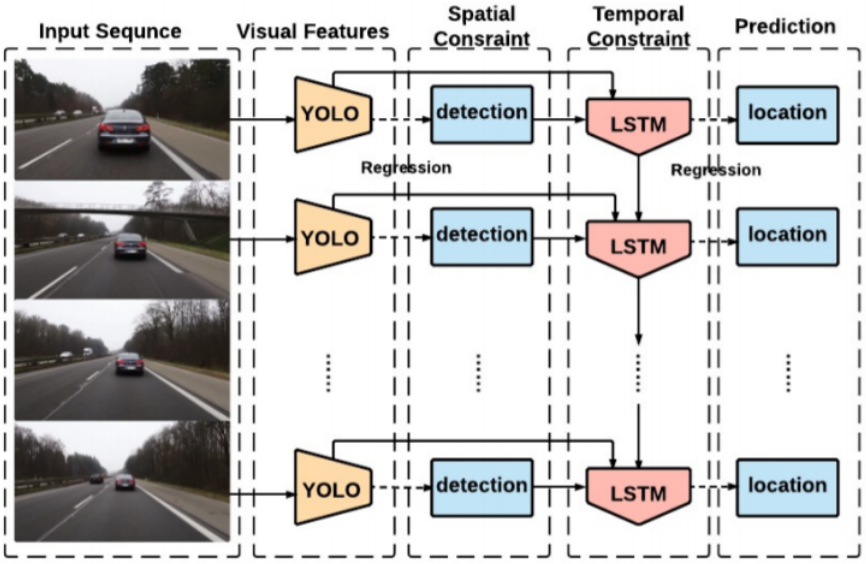

Figure 2.5: ROLO architecture [53].

ROLO adds a layer of LSTMs on top of the YOLO detection model (see Fig. 2.5). These LSTMs take as input both the features extracted by YOLO and the detection information. The tracking problem is then treated as a regression problem, and the loss function used is the Mean Squared Error (MSE).

Depending on the step size, the number of previous frames ROLO takes into account for prediction, the FPS count ranges from 270 to about 33 for step size values between 1 and 9 respectively, and the average accuracy on the OTB-30 dataset is as high as 0.45, measured in average IoU with the ground-truth, for a step size of 6.

2.2.2 SORT

Simple Online Realtime tracking (SORT) is a track-by-detection system that combines CNN feature extraction with a Kalman filter and the Hungarian algorithm in order to track objects at up to 260 FPS.

The CNN architecture used for feature extraction in the original paper is Faster R-CNN [58] and the Kalman filter, based on a linear constant velocity model, is used to

estimate inter-frame displacement. As for association between existing identities and new detections, an optimal matching is performed through the Hungarian algorithm [44] by minimizing the pairwise IoU metric.

If new objects enter the scene or existing ones leave it, two handling strategies are applied. For object detections with too low an overlap with existing objects, a new identity is generated. Upon disappearance of an object, the identity will be maintained for T_{Lost} if the object is not detected again.

An observation about SORT is that, if an object is not detected (for example, because it is occluded by another one), the linear constant velocity model will be the one that determines the de facto position of the object in the next frame. Since this model is a poor predictor (in general), the authors argue that T_{Lost} should be set to 1. Furthermore, this has the added benefit of a lower computational overhead when agents exit the scene.

2.2.3 Deep SORT

Deep SORT is simply an extension of SORT that incorporates appearance information (in the form of an image embedding) to facilitate identity tracking. The main goal of this iteration of SORT is to reduce the amount of identity switches. For this purpose, appearance information is used to re-identify objects that have been temporarily lost.

Another innovation of Deep SORT is the simultaneous use of 2 metrics: the squared Mahalanobis distance for association between Kalman states and the coordinate-wise smallest cosine distance in appearance space. The Mahalanobis distance works well when the uncertainty is low (e.g: short-term predictions) but, when this uncertainty is increased, the cosine distance offers a better similarity indicator by taking into account appearance information.

Deep SORT runs at approximately 33 FPS for 32 bounding boxes on an Nvidia GeForce GTX 1050 mobile GPU.

2.3 Public speaking and affective computing

Public speaking has been extensively studied for millenia, but only in the last decades have computerized methods been applied to this study. In particular, the field has been revitalized by the techniques of affective computing, a term which refers to the study and development of systems to recognize and interpret human emotions.

In this section, we will overview work that has been made to quantify audience experience, as well as the existing public speaking training systems and datasets that we found.

2.3.1 Public speaking training systems

The use of technological tools to improve public speaking skills has received a lot of attention from researchers. In this section, we'll briefly review some of the systems that have been proposed, including some virtual audience systems, in which an audience of varying degrees of responsiveness is simulated and presented to the speaker. While most work on virtual audiences has focused on reducing speaker anxiety, some have tackled performance quality.

Cicero

Cicero [4] is an interactive virtual audience system, where the speaker performs in front of a simulated audience that also reacts to the performance, providing real-time feedback. Using three sensors (microphone, camera, and Microsoft Kinect), the system extracts a set of descriptors of quality that are then combined to create an overall score. This score controls the members of the virtual audience, which can change posture, head orientation and eye gaze to convey different degrees of interest in the presentation.

Figure 2.6: Virtual audience snapshot [4].

Source	Assessed behavior	Behavior descriptor	Spearman's ρ	p-value
Voice	Flow of speech	Num. pauses	-.469	.09
	Clear intonation	Avg. intensity	**.805**	.002
		Breathiness	**-.615**	.033
	Interrupted speech	Num. pause fillers	**.612**	.034
	Speaks too quietly	Avg. intensity	**-.842**	$< .001$
	Vocal variety	Std. f_0	**.709**	.010
		Spectral Stationarity	**-.586**	.045
Body	Paces too much	Leg movement	**.682**	.021
	Gestures to emphasize	Arm movement	**.710**	.014
	Gestures to much	Arm movement	.437	.179
Gaze	Gazes at audience	Face gaze towards	**.621**	.030
	Avoids audience	Face gaze towards	-.548	.065

Figure 2.7: Rated behaviors and associated descriptors [4].

The ground data used to train the system is a set of expert reviews by two senior members of the public speaking organization Toastmasters. The experts watched recordings of each presentation once and were asked to rate 21 characteristics of the performance (some of them are shown in Figure 2.7), as well as to give an overall impression. The ratings use 7-point Likert scales.

Next, the authors attempt to identify automatic descriptors that correlate well with each of the rated behaviors. They use a framework called MultiSense to integrate multimodal data from their three sensors, and the result can be seen in Figure 2.7. Eight of these descriptors (five voice features, two posture features, and one gaze feature) are chosen as inputs to an SVM which is trained to approximate the overall expert rating.

The system was tested in a posterior experiment [16] with 51 participants divided in three groups: a group with no feedback (passive virtual audience), a group with direct feedback (passive virtual audience and a visual indicator of the performance score), and a group with indirect feedback (interactive audience, no visual indicator).

ROC Speak

ROC Speak [23] is a web tool that allows users to practice speaking and get feedback. The users are recorded with a webcam and microphone and then are given an automatic score based on their nonverbal behavior. They also have an option to get a crowdsourced human rating.

To produce the automatic score, the following features are extracted from the recording:

- **Smile intensity**, captured from a standard facial feature detection library.
- **Movement**, defined as normalized pixel differences.

- **Loudness and pitch**, extracted using a standard speech processing library.
- **Word prosody**, again captured by a combination of software tools.

For the crowdsourced score, a task is created in a common crowdsourcing website that asks workers to provide a 1-7 score for overall performance, bodily gestures, friendliness, and volume modulation, as well as at least one comment in one of the previous categories. The comment is associated with a specific timestamp.

The usefulness of this system was tested in three experiments, assessing separately the usefulness of automated feedback alone, crowdsourced feedback alone, and a combination of both. The results showed that the users generally find the automated feedback valuable.

Flow prediction

This system [62] displays aggregate levels of Boredom, Anxiety and Flow in the audience. For each audience member, their emotional state is classified into one of those three, and then the percentage of the audience in each state over time is displayed.

The emotional state classifier first extracts 72 facial features using a facial analysis library. Then, three different classifiers (a decision tree, an SVM, and an MLP) make a prediction and the average of those predictions is taken as the final score.

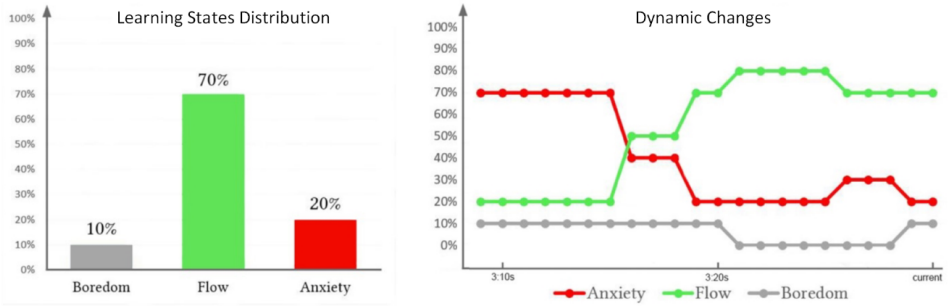

Figure 2.8: Visualization of the audience state.

The system was tested in 8 experiments, in which a speaker gives a lecture and receives feedback in 4 different conditions: No feedback, full feedback, full feedback including sound alarms and text prompts when Boredom or Anxiety reach 50%, and misleading feedback (percentages for Boredom and Anxiety are swapped, and alarms sound randomly).

The speakers found the system useful both for real-time decision making, and for identifying previously overlooked problems (like sections of the presentation that are more boring than expected). However, the system sometimes caused negative emotion in the users and increased their workload.

2.3.2 Characterizing and quantifying audience experience

While there is abundant work on quantifying communication and public speaking in particular, we find the same problem as in the previous section: most of the existing work focuses directly on the speaker's behavior, instead of the audience's. However, we did find some work on audience experience in the form of two technical reports.

Capturing the audience experience: A handbook for the theatre [13] is a report comissioned by different theatrical associations to help them understand how audiences interact with theater performances. After a series of qualitative interviews and an online survey of 2,500 individuals, they authors developed what they call the Audience Experience Framework. The framework is composed of five dimensions:

1. **Engagement and concentration:** The extent to which the performance captures and maintains the audience's attention.
2. **Learning and challenge:** The challenge can be on knowledge, expectation or attitudes.
3. **Energy and tension:** Physiological reactions to the performance, such as excitement or anxiety.
4. **Shared experience and atmosphere:** The sense of collective experience afforded by a performance.
5. **Personal resonance and emotional connection:** The extent to which member of the audience can feel empathy or identify themselves in the performance.

The report also suggests ways to assess each of the dimensions of the framework in an audience, mostly focusing on self-report questionnaires. The authors offer some examples of Likert-scale questions they've used to evaluate the five dimensions, which we incorporated into our experimental design in Chapter 3.

In *Audience Engagement in Multimedia Presentations* [69], the authors propose a conceptual similarity between engagement and the states of playfulness and flow. Playfulness is a term used in studies of human-computer interaction, while flow is a psychological state of deep pleasurable focus that can be experienced in a wide variety of situations ranging from sports to work.

These two states are often associated with situations which provide challenge, clear feedback, a feeling of control, and variety. Thus, the authors test the hypotheses that learners will experience higher engagement during presentations that exhibit those four properties. These hypothesis were separately tested in three different studies, in which students enrolled in a course were taught using different software tools, specifically Authorware and PowerPoint.

In the first study, students perceived Authorware to be more engaging, as well as higher in feedback and challenge, but not significantly different from PowerPoint in terms of control. Variety was not tested.

In the second and third studies, the authors spent much more effort trying to eliminate bias, increased the level of challenge, control, feedback and variety of the Authorware presentation using a series of techniques such as adding sound effects and buttons to control the flow of information within each screen. All of the variables of interest were measured by a self-report questionnaire.

The results of both experiments show significant differences in the four dimensions, as well as in overall engagement. In addition, factor analysis produced a single underlying factor, supporting the hypothesis that engagement is very related to these measures of flow and playfulness.

2.3.3 Public speaking datasets

We found two datasets related to audience experience. However, we were unable to get access to either of them.

CCJ

This dataset was developed by Curtis, Campbell and Jones [17], so for lack of a better name we will refer to it as CCJ. It's built from audiovisual recordings of paper presentations at an academic conference, for a grand total of 520 minutes. Both the audience and the stage (including speaker and slides) are included and the average presentation length is 17 minutes.

Apart from the raw data, a series of features were added by human annotation: emphasis, speaker ratings, audience engagement, and audience comprehension. These were crowdsourced from 40 annotators.

- **Speaker ratings and engagement:** A pre-study was used to determine an optimal window length (between 10-50 seconds) and number of engagement levels for annotations. This resulted in 30s windows and an ordinal 4-point scale. Attendance is also annotated once per talk on a 1-5 scale. Finally, speaker ratings are annotated in 30s windows using the annotator's agreement with the statement 'This is a good speaker who is able to capture the attention of the audience and bring the presentation to life.' on a 1-8 ordinal scale.

  Each segment is annotated once, annotations are cleaned from outliers and normalized by annotator. Three 30s windows are combined into a 90s window, and the mean of the annotations is taken as the annotation of the new window.

- **Emphasis:** The window size here is 5 minutes. Ten humans annotate the same segments of both speaker and audience recordings. The authors found high level of disagreement among annotators, so they used a custom algorithm to detect potential areas of emphasis, which are then judged again by humans as emphasized or not. This latter judgement turned out to be more reliable.

- **Comprehension:** Each presentation is divided into 4 to 7 video segments of 2-4min length, resulting in 172 segments. Annotators watch every segment, write a summary, and estimate on a 1-8 scale the comprehensibility of the material. Each segment is annotated three times, and the final value is the mean of each annotation.

While the dataset itself is not available, this research was very helpful in designing our own experiment.

MAHNOB-HCI

This dataset [61] contains multimodal information of 27 participants watching videos. The modalities include face videos, audio recordings, eye gaze, and physiological data, which makes this a very complete dataset for studying audience experience.

The dataset was created from two experiments. In the first one, participants watched 20 emotional videos and self-reported their emotional state, operationalized as arousal, valence, dominance, and predictability, as well as an emotional keyword that describes the state. In the second experiment, the participants watched 28 images and 14 short

videos together with a word tag, and expressed their agreement or disagreement with the given tag.

Unfortunately, the data is not available to the general public. Since replicating this setup is very hard, requiring specialized tools and a physical space, we were unable to extract a lot of value from this dataset.

2.4 Emotion recognition

Emotion recognition refers to the problem of identifying emotions, in a sense that must specified on a case by case basis, from data. Generally, this data is multi-modal [66]. That is, the data comes from several sources called modalities. An example would be the joint usage of audio and image data or even including measurements of physiological signals. For the purpose of this document, we will restrict ourselves to systems that make use of some type of image or video data. We are particularly interested in systems that deal with body language or, more generally, with audiences.

Emotion recognition is generally framed as a classification problem. A set of emotions to be recognized is defined and the data is classified accordingly. This approach is relatively simple to implement and, if an adequate labeling mechanism is in place, simplifies the data collection process. An alternative approach is treating emotion recognition as a regression problem. For this, one needs to define a set of emotional dimensions to measure and apply regression within them. This allows for a finer granularity in the information that the system infers but also increases the requirements on the data collection process, since the quantification of emotions needs to accurately reflect reality. The choice of dimensions, the magnitude of measurements, and the labeling process are all vital to obtain an adequate regression system that is capable of conveying useful information.

When it comes to the emotion recognition problem in audiences, it is common to try to estimate engagement, as it is a metric of interest in serveral fields. This has been applied in contexts as diverse as television spectators [32] and students in the classroom [70][27]. Since our problem partially overlaps with engagement estimation, we will describe with some more detail a selection of these systems. We will also describe an alternative approach using variational autoencoders that has been used to recognize emotions in movie audiences [19].

2.4.1 Measuring the engagement level of TV viewers

Hernandez et al. [32] studied the feasibility of using visual information (a video) in order to estimate the engagement level of an audience. Unlike previous approaches, the proposed technique does not require specialized equipment like eye trackers or physiological sensors.

For this purpose, 47 people were recorded in a naturalistic setting and the resulting videos were annotated manually in a 4-level scale: None, Low, Medium, and High (see Fig. 2.9 for examples). A set of hand-picked facial features was extracted via 2 techniques: a face tracking system detected face distances, angle, and head roll, while head size and position were detected with Viola-Jones object detection framework [67]. These features were also aggregated across a time window in order to capture their variation in time.

The final problem was further simplified to a binary classification problem, with None/Low and Medium/High engagement levels paired together. An SVM was then used to predict

Figure 2.9: Examples of audience members with engagement level of High (top left, bottom left) and None (top right, bottom right) [32].

the final result, obtaining up to 82.54% precision score in discriminating low and high levels of engagement.

2.4.2 Measuring the engagement level of students in the classroom

The classroom is another interesting environment for emotion recognition. Engagement has been independently linked to learning outcomes by several researchers [69][51]. Therefore, there have been several system proposals designed to estimate engagement automatically. The main purpose of these systems is to serve as guidance for teachers and educators so that they can improve their teaching skills and the learning outcome of their students.

One approach [70] to this issue considered engagement estimation as a binary classification problem where a student could be either engaged or disengaged. They ran a data collection experiment where they recorded 34 undergraduate students playing a set of games and annotated them manually on a 4-point scale (where 1 was the lowest and 4 the highest possible value of engagement). Four classifiers were trained to identify whether the given subject had a particular engagement level. While several alternative classifier architectures were tested, the authors chose an SVM applied on the Gabor features [30] of the input for the full pipeline. The four classification results are combined via either linear regression or multinomial logistic regression to obtain a final estimate.

Another approach [27] treats engagement estimation directly as a regression on the interval $[-2, 2]$, which indicates the degree to which a subject is off-task (negative values) or on-task (positive values). The authors ran a study with 52 volunteer students that were videotaped during regular university sessions. Questionnaires were used to determine individual learning prerequisites and the videos were annotated manually through reference indicators in the scale above (a student walking around with the intention to interrupt would indicate a score of -2, for example). From each video, the head pose, gaze direction, and facial expression coded in Facial Action Units (FAUs) [20] were extracted and a linear SVM was used trained for the regression task.

2.4.3 Recognizing emotions in movie audiences using variational autoencoders

Deng et al. [19] used a radically different approach from the ones above. Drawing on more recent deep learning techniques, they decided to use a modified version of Variational AutoEncoders (VAEs) [42] to learn an embedding (low-dimensional representation of data) of the audience's faces that retains important semantic information.

VAEs work by forcing a network to reconstruct input data by first compressing it by passing it through a set of encoder layers and then decompress the information through a set of decoder layers. After proper training, the output of the encoder layers of the VAE should be a representative embedding of the input.

In general, it is hard to determine semantically meaning factorizations for a given embedding. To address this issue, the authors propose a modified version of the VAE, the Factorized Variational AutoEncoder (FVAE). This architecture simultaneously learns an embedding and a factorization of the embedding. In particular, FVAEs are used to determine a factorization of facial embeddings across different possible face reactions, the number of which is determined as a hyperparameter.

For the data collection process, a movie theatre was equipped to record the audience with infra-red cameras and illuminators. Then, over 150 viewings of 9 different movies were recorded, with an audience ranging between 30 and 120 for each recording.

For each video, faces are detected using Max Margin Object Detection (MMOD) [41], while Ensemble Regression Trees (ERTs) [40] are used to detect facial landmarks. With this information, each face is then associated to a $3D$ mesh from Face Warehouse [12].

The embedding itself already contains information about the emotional state of the audience. However, interpretation of the results requires careful analysis once the FVAE is trained.

2.5 Videoconference systems

Given the restrictions on physical meetings, we needed a videoconferencing tool to run our experiment and collect data. The basic requirements were: being able to play videos trough the app and being able to record the participants while doing so. Additionally, we needed to extract an exclusive video stream for each participant from the recording. There are two approaches to this issue: either we directly record separate videos for each participant, or we record all participants in a mosaic view and then manually extract each sub-video from the composite view. The first option is more attractive, since it reduces manual processing steps and allows higher resolution for each participant. Furthermore, since each one of them would use a separate webcam, it seemed like a natural choice.

An additional feature that we were interested in is preventing the participants from hearing or seeing each other, to reduce biases. This could have been done with individual experiments, but that would be much more time consuming and, therefore, less practical. With these requirements in mind, we surveyed the available applications.

2.5.1 Google Meet

Google Meet is free, widely available, and provides a recording functionality. Videos can be streamed by screen sharing. By default, it only records the current speaker and displayed content at any given moment. However, by using the Google Meet Grid View plugin [24], it's possible to record everyone in mosaic view. Thus, this app fulfills all of our essential requirements but none of the additional ones.

2.5.2 Bb Collaborate

Blackboard Collaborate is commonly used for online classes in higher education and while it's not free, our institution provides us access to it. It can play videos using screen sharing, and it also allows recording sessions. However, these recording only include active speaker video or displayed content. Therefore, this app is unsuitable for us.

2.5.3 Zoom

Another widely known videoconferencing application, Zoom is similar to Google Meet in that it allows playing videos and recording of everyone in the session, but not recording participants individually nor preventing them from seeing each other, therefore it's a viable candidate but does not satisfy our extra requirements.

2.5.4 Jitsi

A somewhat less known alternative, Jitsi is free, open source, and offers video streaming and mosaic recording functionality. It is a viable candidate but does not satisfy our extra requirements. However, it might be possible to create a plugin or modify the source code so that it does.

Chapter 3

The ERVF Dataset

In order to train the system, a fundamental requirement is access to adequate training data. Specifically, a dataset of audience recordings during presentations or public speaking events, labeled with the emotional state and level of engagement of each individual. While we found some data sets that fit these two conditions (see section 2.3.3), they presented two main problems. First, they were not available to the general public. Second, they were recorded in person and so were a poor fit for online environments.

As a consequence, it became necessary to gather our own training data. This entailed precisely defining the variables we wanted to measure, designing an experimental setup to obtain those measurements, and processing and curating the results into a usable form. In this chapter we will explain this process and the results we obtained.

The first step was defining a quantitative measure of audience experience, which is introduced in Section 3.1. Afterwards, we performed two experiments, described in Section 3.2, and then we extracted the results, explained in Section 3.3. Finally, we will clarify some limitations of our process.

3.1 Quantifying audience experience

The study of communication and public speaking is a huge and very active academic field. As complete outsiders to this field, it was hard for us to find relevant literature on the subject of audience experience. In addition, the number of publications on audience experience is dwarfed by those on other aspects of public speaking and performances, such as speaker anxiety.

Despite these issues, we found some previous work on the subject, which is documented in Section 2.3.2. These works conceptualize audience experience using a combination of dimensions, drawing inspiration from psychological theory and qualitative interviews, and also provide experimental validation of some of these metrics.

In particular, in *Capturing the audience experience: A handbook for the theatre* [13] a five-dimension framework is proposed. These five dimensions are:

1. **Engagement and concentration:** The extent to which the performance captures and maintains the audience's attention.

2. **Learning and challenge:** The challenge can be on knowledge, expectation, or attitudes.
3. **Energy and tension:** Physiological reactions to the performance, such as excitement or anxiety.
4. **Shared experience and atmosphere:** The sense of collective experience afforded by a performance.
5. **Personal resonance and emotional connection:** The extent to which member of the audience can feel empathy or identify themselves in the performance.

We initially considered applying this framework directly. However, dimension 4 does not play a big role in the online setting, where members of the audience are typically isolated from each other. Therefore, we decided to drop it and focus on the other four dimensions.

The next step after having specified a conceptual framework was providing a concrete metric for each of the four components, a process usually called operationalization. Conveniently, the authors of the report also provide a set of guidelines and example questions to measure these components from self-report questionnaires.

To complement this guidelines, we reviewed the data-gathering process used to create the audience experience datasets found earlier, especially the work of Curtis et al. [17]. From this we incorporated their work in engagement, as well as their comprehension metric, combining it with dimension 2 of the conceptual framework.

In addition, we make use of the abundant literature on affective response [10] and use it as an operationalization of energy and tension.

Thus, our final framework is composed of the following four dimensions: affective response (Af), engagement (En), emotional connection (Ec), and learning (Le).

With respect to actually measuring these components in an experiment, the following options were available:

1. **Self-reports**: the audience fills in a questionnaire after watching the performance, with questions about their subjective experience. The main advantages of this method are that it's fast, cheap, relatively unbiased, and does not disturb the experience. The main disadvantage is that it only provides one data point for each individual and performance, and therefore has very low temporal resolution. This approach was the one followed in *Capturing the audience experience: A handbook for the theatre* [13].
2. **External rating**: an external observer evaluates each component over the duration of the performance. The main advantages are that it provides much greater temporal resolution, since we can annotate, for example, each 1-minute interval with a different score. The main disadvantage is that it requires time-consuming manual labor and is prone to bias on the part of the annotator. This approach was followed by Curtis et al. [17].
3. **Physiological measurements**: Physiological sensors are connected to the participants, which measure galvanic skin response, heart beat, and other relevant signals. This is probably the most direct way of measuring the physical and emotional state

of the audience, but it requires specialized machinery and equipment that was not available to us. Furthermore, it is intrusive and may have a direct influence on the audience experience.

3.2 Experimental design

For our experimental design we iterated through several proposals, spotting and correcting the problems we found until we converged to the final design. The basic idea always remained the same: record an audience as they react to a performance, either live or videotaped. In addition, register the audience experience using our framework and either self-report or external annotations.

In the first iteration we considered two different setups: an in-person one, on campus, with a tightly controlled environment to remove any confounding variables and validate our methodology; and a crowdsourced version, using the volunteers' webcams, to gather more data in a scalable way. However, we soon realized that the public health situation was not favorable to any in-person experiments and thus decided to perform our experiment entirely online.

For the second iteration, since we knew the experiment would be online, we determined that the performances would need to have the form of video recordings that we could stream over the Internet. Thus, we started selecting a pool of videos for the experiment, detailed below.

As for data collection, we considered a two-pronged approach. We would use external annotations for engagement, in combination with a self-report questionnaire for all of the components. This way, we would have high-resolution data for one of the components, and we would be able to perform cross-checks with the self-report data to ensure the soundness of our method. At this stage, we developed the questionnaire, which will be explained below.

Due to time constraints, we determined it would be infeasible for us to annotate the dataset. So, for the third iteration, we were forced to settle on self-reports to assess audience experience.

3.2.1 Tools used

Having decided that the setting would be online, the performances would be recorded, and the data collection method would be self-report, we needed three extra pieces to perform the experiment: a videoconferencing tool, a selection of video performances, and a self-report questionnaire.

Video selection

To make our experiment flexible in terms of time commitment, we decided to use short videos, of about 10 minutes. The videos had to be varied in content and style, and in Spanish language, since our experimental audience would likely be native Spanish speakers. Eventually, we chose ten videos, which can be found in Table 3.1. These emotional, political, comedic and divulgative videos.

Self-report questionnaire

The final questionnaire included the following questions, divided into sections for each component of the audience experience:

Title	Speaker	Link
"La soledad del adicto: Comprender al otro puede salvarle la vida"	Laura Vergara	https://www.youtube.com/watch?v=z6FoUeSohnk
"Lo imposible a veces sólo cuesta un poco más"	Eduardo Llano	https://www.youtube.com/watch?v=9KrZsJuENt0
Pablo Casado's response to Santiago Abascal at the 2020 motion of censure	Pablo Casado	https://www.youtube.com/watch?v=9Ehhf94YDG09
Intervention of Gabriel Rufián at Mariano Rajoy's investiture debate	Gabriel Rufián	https://www.rtve.es/alacarta/videos/especiales-informativos/la1-rufian-020916/3709343/
"Tengo un sueño"	Dani Rovira	https://www.youtube.com/watch?v=8JWsg4Psmt0
"¿Por qué 'runner'?"	Ana Morgade	https://www.youtube.com/watch?v=N-NdyyHc_Gk
"La selección latinoamericana de cerebros"	Juan Enríquez	https://www.youtube.com/watch?v=GglVs9scY6I
"A nadie le importa la verdad"	Rocío Vidal	https://www.youtube.com/watch?v=b_I6WmatS2o
"¿Por qué me vigilan, si no soy nadie?'	Marta Peirano	https://www.youtube.com/watch?v=NPE7i8wuupk
Spain's 2015 Royal Christmas message	Felipe VI	https://www.youtube.com/watch?v=PtVm83f2vmk

Table 3.1: Video selection.

Affective Response This component is measured using the Self Assessment Manikin [10]. For each of the three images in Figure 3.1, the participants select which cell they feel most identified with. The images represent the three factors of affective response found in the literature. Those are, in order, valence (how good you feel), arousal (how intense your feelings are), and dominance (to what extent you feel in control). The images are labeled from 1 to 9, therefore being analogous to a 9-point Likert scale, and the final score is a normalized sum of the three answers. Concretely, we add the three answers, subtract their average, and rescale so that the end result is within the interval $[0, 1]$.

Engagement This section consists of four 5-point Likert scale questions, with two labels per question, shown in Table 3.2. The participants are asked to select their agreement level with these two labels. The total engagement score is given by the normalized sum of all the answers.

1-point label	5-point label
My mind wandered	I was completely focused on what I saw
Time seemed to pass very slowly	I hardly noticed the passage of time
The video didn't catch my attention	I couldn't keep my eyes off the screen
I don't want to talk about the video	I want to share my experience watching the video

Table 3.2: Engagement section questions.

Emotional Connection This section consists of three 5-point Likert scale questions, with two labels per question, shown in Table 3.3. The participants are asked to select

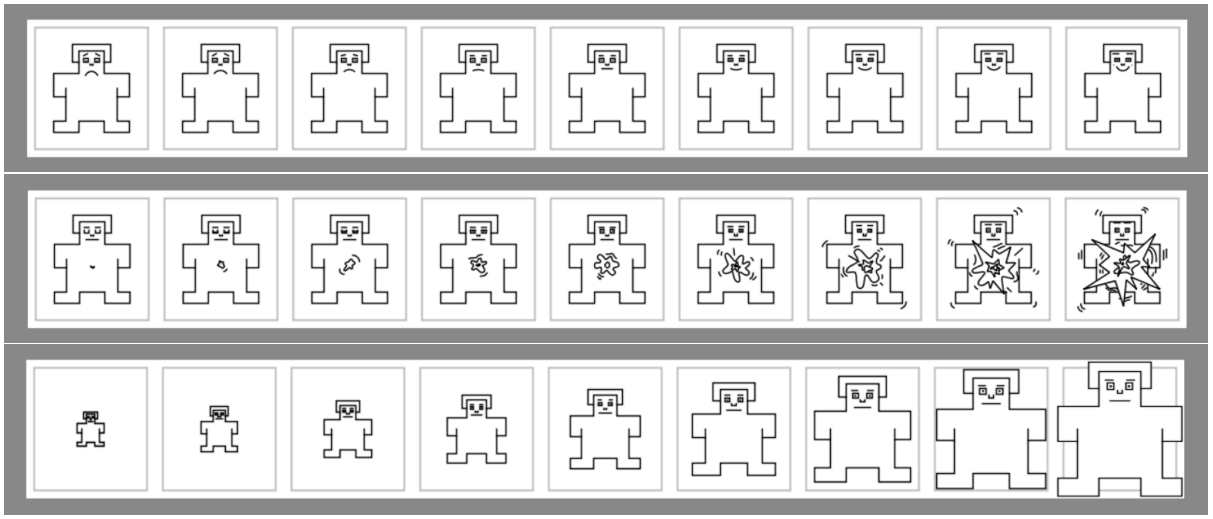

Figure 3.1: Self Assessment Manikin.

their agreement level with these two labels. As before, the score for this section is the normalized sum of all the answers.

1-point label	**5-point label**
I was not moved by the video	The video affected me emotionally
I wouldn't watch more content from the same speaker	I would like to watch more content from the same speaker
The video did not say much about my personal experiences	I felt identified at a personal level with some parts of the video

Table 3.3: Emotional connection section questions.

Comprehension and Learning This section consists of six 5-point Likert scale questions, with two labels per question, shown in Table 3.4. The participants are asked to select their agreement level with these two labels. In the first three items, the overarching question is "How understandable would you say it's been?", while for the last three items the overarching question is "How do you feel with respect to the theme of the video?". Once again, the score for this section is the normalized sum of all the answers.

1-point label	**5-point label**
I didn't understand anything	Everything was perfectly clear
It was hard to follow	At all times I knew what the speaker was talking about
I wouldn't be able to explain the content to someone else	I could explain the content to someone else without problems
There was nothing new for me	It opened my mind to new ideas or viewpoints
I knew the topic in-depth before watching the video	I did not know anything about the topic before watching the video
My ideas about the topic haven't changed	Now I have a completely different view

Table 3.4: Comprehension and learning section questions.

The form was created and administered using Google Forms.

Video conferencing tool

To perform the experiment, we would need a tool satisfying at least the following requirements:

- Ability to play videos through the app
- Ability to record the participants while doing so
- Ability to extract from the recording an exclusive video stream for each individual participant. This entails either recording each participant separately through their webcam, or recording everyone in a mosaic view and then manually demultiplexing the individual videos. The first is a more efficient and natural approach, but is not always available.

In addition, we found attractive the idea of preventing participants from seeing each other during the experiment, which would prevent them from biasing each other. While we could achieve this making multiple experiments with a single individual each, this approach is very slow.

After reviewing the most commonly used video conferencing apps, we found some that were sufficient, but none of them were ideal. More concretely, we found applications satisfying the mandatory requirements, but none of them allowed recording each participant separately, which forced us to use the manual demultiplexing method. In addition, none of them allowed us to prevent participants from seeing each other.

So, after some deliberation, we decided to create our own video conferencing tool, which would allow us to precisely control the experimental setting. This tool is able to record individual participants, and preventing them from seeing and hearing each other, in addition to fulfilling the other requirements.

The application provides two different user interfaces: one for the subjects of the experiment and one for the host. As can be seen in Figure 3.2, the host can see all the subjects in a mosaic view to the right, and a list of streams to the left. The host can create, cast, and remove streams from the list, and the streams can be captured from a webcam, screen sharing, or a YouTube video. When the host chooses to cast an stream, it will appear in the subject view. In the case of a YouTube video, the host can control the playback state using the normal controls of the embedded player. When the state changes, the change will be broadcast to all the subjects, in such a way that if the host starts, stops, or changes the video timestamp, the subjects will see the same changes. In the inferior pane there is a button to add a new stream, a button to start or stop recording, and the common chat.

The subjects only view their own image, as well as anything the host chooses to cast. They can control their camera and microphone, and write through a common chat. When the host casts a YouTube video, the subjects can't control the playback of the video in any way, but have the option to reproduce it in full screen.

When the host starts recording a separate file is created for each subject. In addition, two more files are created: one is an identification file which links the identifiers of the subjects to their respective video files, and the other one is a synchronization file. This file stores the timestamps of all recordings when the playback status of a YouTube video is modified, as well as the video timestamp and type of event that triggered the synchronization. This

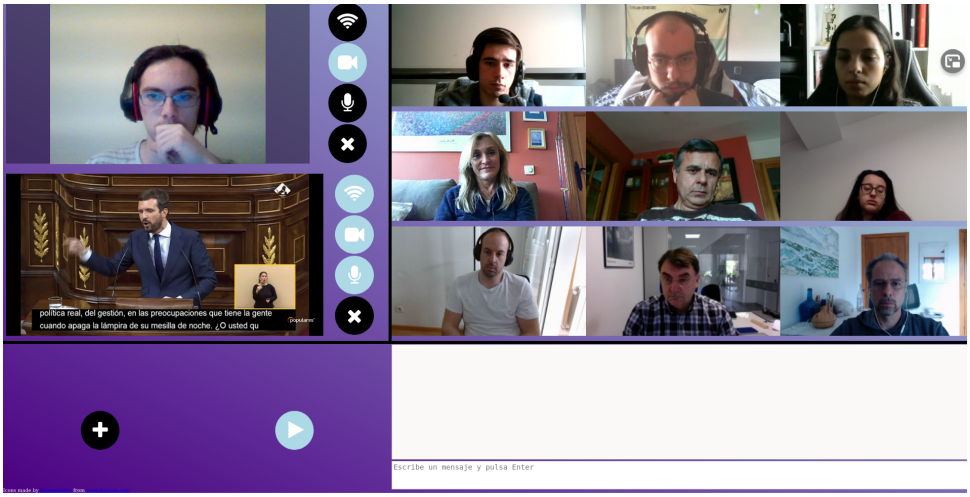

Figure 3.2: A screenshot of the host view taken during the pilot experiment.

allows the experimenters to identify which parts of the recordings correspond to a given video segment, even if there is some delay in the recordings due to network latency.

The tool was created using the WebRTC API [29], which offers the two essential functionalities needed for videoconferencing: capture and control of audiovisual streams and signaling for real time communication. We used the Pion WebRTC stack [54] for the backend, adding slight modifications to adapt it to our needs. The web interface was created using the Pion SDK and vanilla JavaScript[1].

3.2.2 Methodology

Having introduced our design process and our tools, we can now explain our complete methodology, which is introduced diagrammatically in Figure 3.3. We ran two experiments, the first of which was a pilot test. Participants were volunteers, either students from our faculty or acquaintances of the experimenters. After a suitable pool of participants was found, we emailed them asking for voluntary participation in the experiment, along with a survey to find available dates. The pilot experiment was scheduled to have a duration of two hours but, after getting feedback from the first one, we reduced the duration of the actual experiment to one hour.

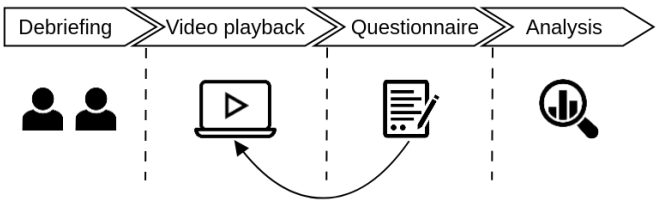

Figure 3.3: Diagram of our experimental methodology.

Once we got enough replies and found an adequate date, we asked participants to sign a consent form to take part in the experiment. We assigned numeric identifiers for each participant and decided which videos from our selection we would play during the session.

[1] All the code is available in `https://github.com/pablo-vs/rtc`, and the JavaScript RTC library which forms the bulk of the code we developed can be seen in Appendix A.

This was done randomly to reduce any biases from the ordering of the videos. A few hours before the experiment, we emailed participants their identifiers and the URL they would use to connect to the video conferencing tool.

The actual experiments were conducted as follows: after a brief presentation and solving any technical issues, we explained how the experiment would work to the participants. In the pilot we did not introduce the questionnaire at the beginning of the experiment but, after noticing that the subjects found part of the questionnaire confusing, in the actual experiment we decided to let participants openly explore the questionnaire at the beginning and answered any questions they had.

Then we played each one of the selected videos through the app, removing any other content from the screen so that the participants could only see themselves and the video. After the video finished, participants filled in the form, and then we moved on to the next video.

The form used in the experiments contained three sections: one for the beginning of the video, another for the middle, and the last one for the ending. Each section contained the full set of questions outlined above. That way, we would have three data points per subject and video, instead of one. The responses to the questionnaire were associated with the recordings using the unique identifier of each subject.

During the pilot experiment, we ran into a technical issue with the videoconferencing tool that prevented us from recording anything. Concretely, the sever we were using to host the application hit a resource limitation of which we were not aware, and dramatically reduced its performance. Thus, we decided to fall back to Google Meet and continue the experiment there. Unfortunately, due to a mistake on our part, the recording from Google Meet was unusable.

However, after experiencing this issues we were able to solve the underlying cause and use our video conferencing tool successfully in the next experiment.

The pilot experiment was conducted with 10 people and 5 videos, while the next one was conducted with 8 people and 3 videos.

3.3 Experimental results

As mentioned in the previous section, the pilot experiment did not provide any viable results. However, the second experiment was more successful. After a preliminary exploration of the data, it seems that our experiment was able to successfully capture variation in experience across subjects, videos, and parts. In addition, the four components are not completely correlated with each other. In Figure 3.4, we can see this variation across videos, as well as some common patterns: comprehension tends to be higher than the other components, while emotional connection tends to be lower.

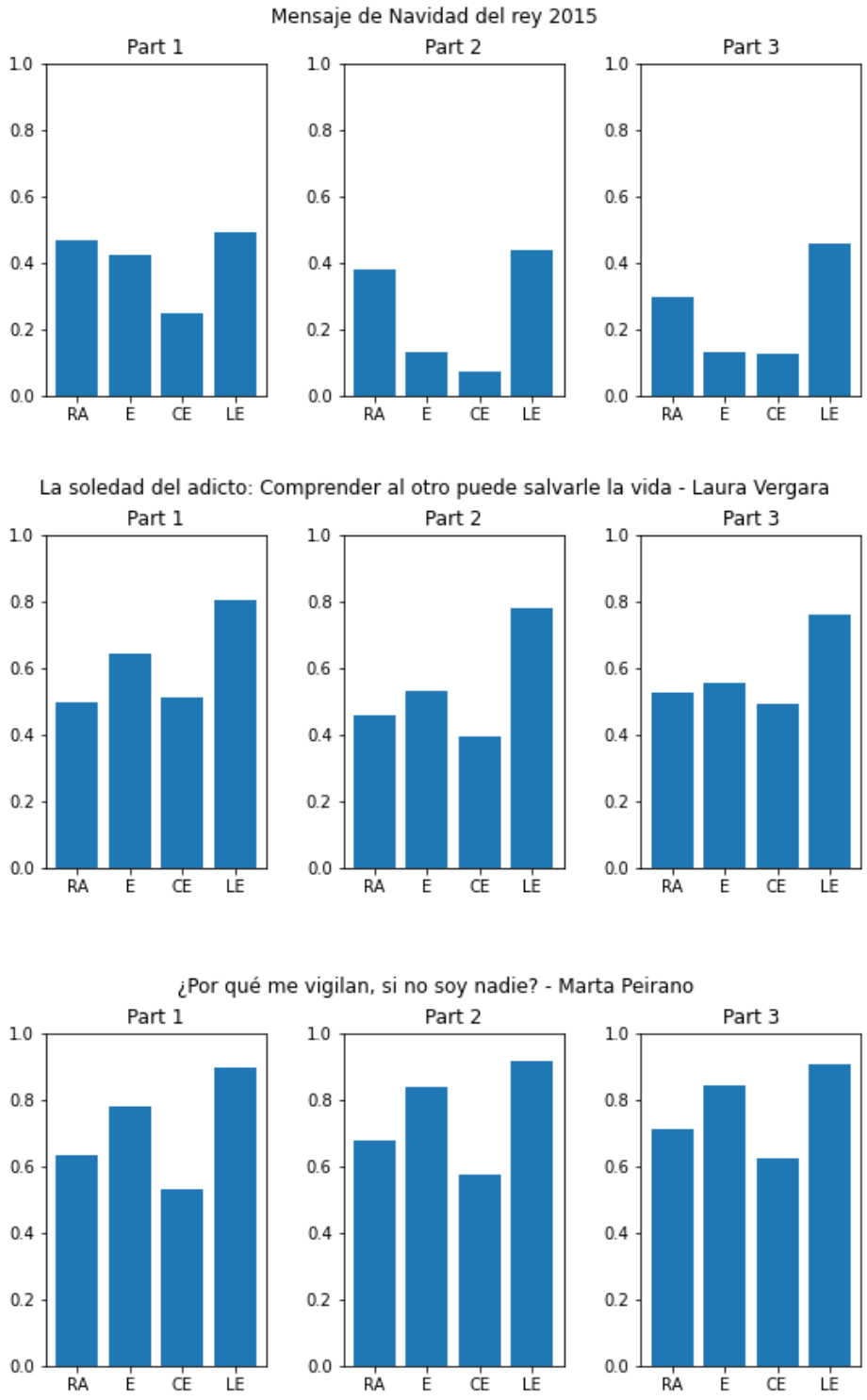

Figure 3.4: Average of the dimensional scores for all participants in three different videos.

The dataset

To create the actual dataset, we synchronized the recordings with the videos and cut them to match the duration of the performances. Then, the recordings were processed by our tracking module, producing a stream of constant resolution tracking each participant. During this step, we had to discard the three recordings of one of the participants because of an unforeseen problem: in the background there were a series of anime posters, which

our tracking module sometimes erroneously detected as people. Since these detections were very irregular the tracking was unable to pick up a stable identity for the actual subject, resulting in several tracking streams with erroneous frames.

The final step was associating each frame of the resulting streams with the four experience scores. We selected our three datapoints for each recording to lie in the first, middle, and last frame and used linear interpolation to generate consistent scores for all the other frames.

The final result of this process is the ERVF (Emotion Recognition from Video Feeds) dataset, consisting of 21 recordings for a grand total of almost 250,000 labeled frames. It will not be available to the general public due to privacy concerns.

3.4 Limitations

The greatest limitation that we've found so far is that there is a lot of redundancy due to our choice to use self-reports instead of annotation, which results in a lot of similarity between frames and labels. This presents a problem to most learning algorithms, since the nonverbal cues that they could use to evaluate different experiences are not clearly distinguishable from the rest of the recording. A consequence of this, as we'll see in Chapter 5, is that 95% of the frames in the dataset can be dropped without significantly affecting the performance of the learning algorithm.

Additionally, the low number of individuals and performances results in a low degree of diversity in the video data. All of our volunteers were Spanish college students and all our performances were short and made by professional speakers.

Chapter 4

The visual tracking module

In the context of the proposed application there is a need to perform object tracking to a certain extent. There may be, for example, several agents on a video feed that we might need to isolate and keep track of in order to apply emotion recognition to each one of them. However, there are some particular characteristics associated with the setting in which the proposed system is designed to function

1. Agents will mostly remain static and they will not cross each other very often.
2. The number of agents to track will usually be small.
3. Agents will be relatively close to the camera.
4. Most of the times only the face and the upper body will be visible.

These characteristics greatly reduce the complexity of the visual tracking problem we need to tackle. We do not need, for example, a very high tolerance to agents crossing each other or to crowding. For this reason we opted to keep implementation complexity of the tracking module to the bare minimum. The main concern we still face is that of latency. Even under the simplified scenario presented above, we still need a fast method to keep down the processing time of the whole system.

4.1 Module architecture

To adapt to the conditions presented above, we propose a 2-step tracking module that works in a frame-by-frame manner. Firstly, we apply YOLO, specifically YOLOv4, to each frame in order to track objects in each frame. Then, we apply a custom-made tracking algorithm that assigns identities to the objects detected by YOLO and keeps them updated across frames. In the following sections we will describe in detail each of the components of the module.

4.1.1 Single frame object detection with YOLOv4

YOLOv4 takes an input image and extracts a feature map by running it through a pre-trained backbone network. This feature map is refined by an intermediate network we call the neck of the system. Finally, YOLOv3 works as the head of the network, being used to detect objects in the image and predict their positions and bounding boxes. The neck and head of the network are collectively referred to as the detector. Both the backbone

and the detector make extensive use of several techniques during training and inference to optimize the accuracy of the final system. These are referred to as bag of freebies (BoF) or bag of specials (BoS) depending on whether they are used during training time or inference time respectively. The BoF and BoS methods used by the standard YOLOv4 model [9] are given in table 4.1 and will be discussed in more detail further in this section.

	Bag of Freebies	**Bag of Specials**
Backbone	CutMix	Mish activation
	Mosaic data augmentation	Cross-stage partial connections (CSP)
	Dropblock regularization	Multi-input weighted residual connections (MiWRC)
	Class label smoothing	
Detector	Complete-IoU (CIoU) loss	Mish activation
	Cross mini-batch normalization (CmBN)	Spatial pyramid pooling (SPP)
	Dropblock regularization	Spatial attention module (SAM)
	Mosaic data augmentation	Path aggregation network (PAN)
	Self-adversarial training (SAT)	Distance-IoU loss with non-max suppression (DIoU-NMS)
	Grid sensitivity elimination	
	Optimized anchors	
	Cosine annealing scheduler	
	Optimized hyperparameters	
	Random training shapes	

Table 4.1: BoF and BoS used by YOLOv4.

The backbone used by YOLOv4 is CSPDarknet53, a network based on Darknet53, the backbone network used in YOLOv3 (see Fig. 4.1), that applies a cross-stage partial network (CSPNet) strategy [68]. CSPNet proposes to split the network in blocks of computation (CSP blocks) where input data is broken into 2 parts. One of them is processed through the usual route and a transition layer while the other one is forwarded through a skip connection. The results are then concatenated, processed by another transition layer, and forwarded to the next block (see Fig. 4.2). The goal of this method is to allow backpropagation to propagate the error signal to the first layers in a more direct manner while simultaneously reducing the computational cost of processing information. This method has the additional advantage of being able to reuse features from previous layers due to the introduced skip connections.

The neck network of YOLOv4 is a modified version of the Path Aggregation Network (PANet) [47]. PANet works in the same fashion as Feature Pyramid Network (FPN), by creating a feature hierarchy to extract a rich set of features that are then propagated to the head of the system. However, low level features (those produced in the first layers), are propagated differently. In FPN, these features may have to traverse a large set of layers and, therefore, be subject to degradation along the network. PANet adds a shortcut to propagate these lower level features to the upper layers more directly. The version of PANet used in YOLOv4 further differs from the original in the way it aggregates feature

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4.1: Darknet53 architecture [56].

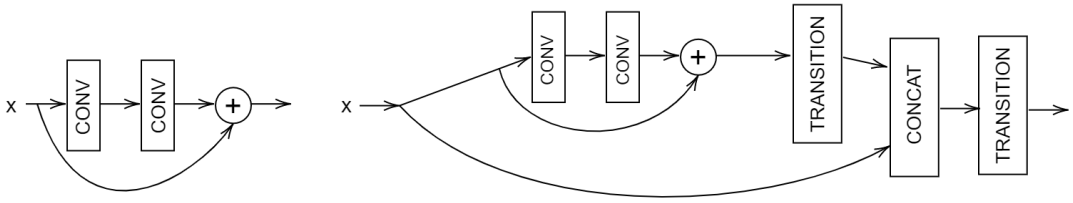

Figure 4.2: Residual block (left) and its equivalent CSP block (right).

maps. Instead of using addition as the original PANet it combines feature maps by concatenation.

Additional improvements to the backbone network

With respect to training (BoF), the backbone network makes use of data augmentation through bot CutMix and mosaic data augmentation. CutMix removes parts of the training images and fills them with information coming from other images in the dataset. Mosaic data augmentation creates new images by combining 4 images in the dataset with different arrangements and aspect ratios to create a new training image. Altogether with dropblock regularization, which masks entire blocks of feature maps sampled at random, these techniques aim to force the neural network to learn a diverse and robust set of features in order to make predictions, as opposed to relying on few but very relevant ones.

Another regularization technique used in training is class-label smoothing. Smoothing is designed to deal with noise in the dataset by considering a certain likelihood $\varepsilon > 0$ for a label to be incorrectly assigned. In this case, we consider that the true class could be any of the possible classes with equal probability. This is akin to considering that mistaken labels have a uniform distribution with respect to the possible classes. If there

are K classes and the label is a one-hot encoded vector $\mathbf{y} = (y_1, y_2, ..., y_n) \in \mathbb{R}^n$ then the smoothed label $\hat{\mathbf{y}}$ is the vector whose coordinates are given by $\hat{y}_i = (1 - \varepsilon)y_i + \frac{\varepsilon}{K}$.

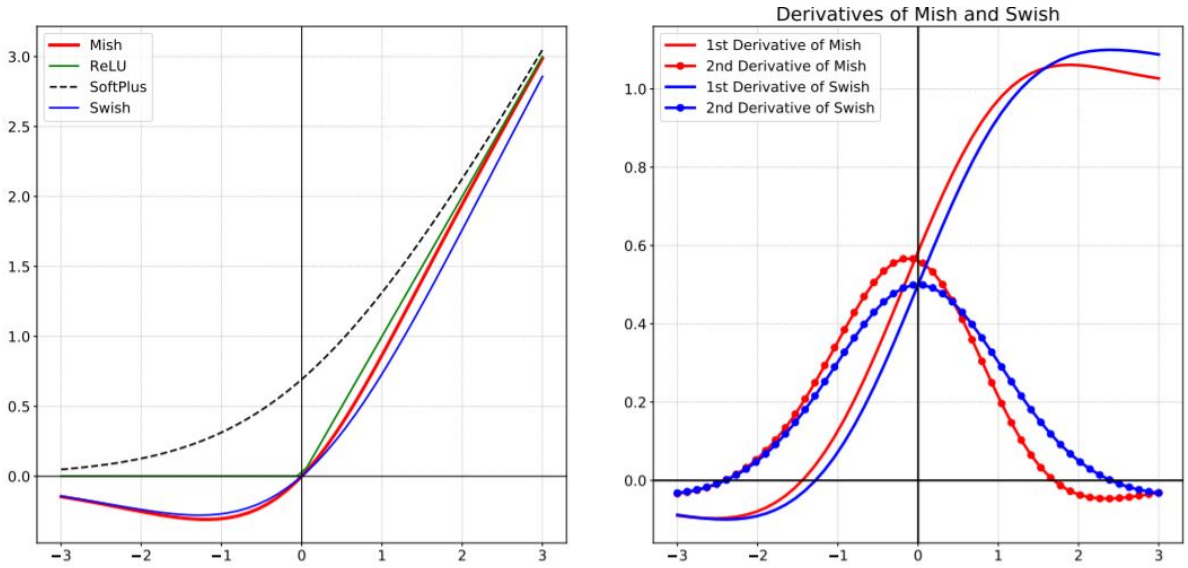

Figure 4.3: Comparison of Mish with other activation functions (left) and derivative comparison between Mish and Swish (right) [52].

When it comes to prediction (BoS) the additional techniques used by YOLOv4's backbone are the Mish activation function and MiWRC. The Mish activation function [52] is given by $f(x) = x \cdot \tanh(\text{softplus}(x)))$ where $\text{softplus}(x) = \ln(1 + e^x)$. This function is smooth (as it is $\mathcal{C}^\infty$), non-monotonic, has a lower bound, and has no upper bound; all desirable properties in an activation function. Additionally, the Mish activation is self-gated (by the tanh function), in the sense that it automatically limits how much information is passed onto the next layer, and self-regularized, in the sense that gradients are smoothed (thus being more efficient during the learning process) with respect to other similar activation functions like Swish [55] (see Fig. 4.3). MiWRC is applied in the form of residual layers that include weights for each input feature map so that the network may adjust how the information is merged.

Additional improvements to the neck network

When it comes to BoF methods, the detector uses dropblock regularization and mosaic data augmentation, already mentioned in the previous section. However, it also makes use of SAT, an additional data augmentation technique. SAT works in 2 steps:

1. First, the neural network weights are locked and the network is only allowed to modify the values of the input image in order to minimize the loss function. This results in a degraded image that works as an adversarial example to the network.

2. Then, the neural network is trained as usual using the modified image.

The detector also takes advantage of the CIoU loss function [78] given by

$$\mathcal{L}_{CIoU}(\mathcal{B}, \mathcal{B}^{gt}) = 1 - IoU + \frac{\rho^2(\mathbf{p}, \mathbf{p^{gt}})}{c^2} + \alpha V \tag{4.1}$$

where $\mathcal{B}$ denotes a bounding box, $\mathcal{B}^{gt}$ the ground truth bounding box, $\mathbf{p}$ and $\mathbf{p}^{gt}$ their respective centers, IoU is the intersection over union of both bounding boxes, $\rho(\cdot, \cdot)$

denotes the euclidean distance, c is the diagonal length of the smallest rectangle covering both $\mathcal{B}$ and $\mathcal{B}^{gt}$, and V and α are given by

$$V = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \tag{4.2}$$

$$\alpha = \begin{cases} 0 & \text{if } IoU < 0.5 \\ \frac{V}{(1-IoU)+V} & \text{otherwise} \end{cases} \tag{4.3}$$

This loss function takes into account all 3 geometric factors that differ across bounding box proposals: overlap area, distance, and aspect ratio.

Other BoF techniques used in the detector are CmBN, grid sensitivity elimination, optimized anchors, a cosine annealing scheduler, and random training shapes. Hyperparameters are also optimized. CmBN is a modification of Cross-iteration Batch Normalization where only information from mini-batches within each batch is used. Anchors are optimized by using several "base anchors" (which differ in aspect ratio) and then selecting those that best fit each object. A cosine annealing scheduler [48] is a technique used to adjust learning rate where learning rate is rapidly decreased and periodically reset to a specific value in order to speed up the learning process. Finally, input image resolution is randomized and mini-batch size is dynamically adjusted accordingly (i.e. larger mini-batch size for smaller resolution training).

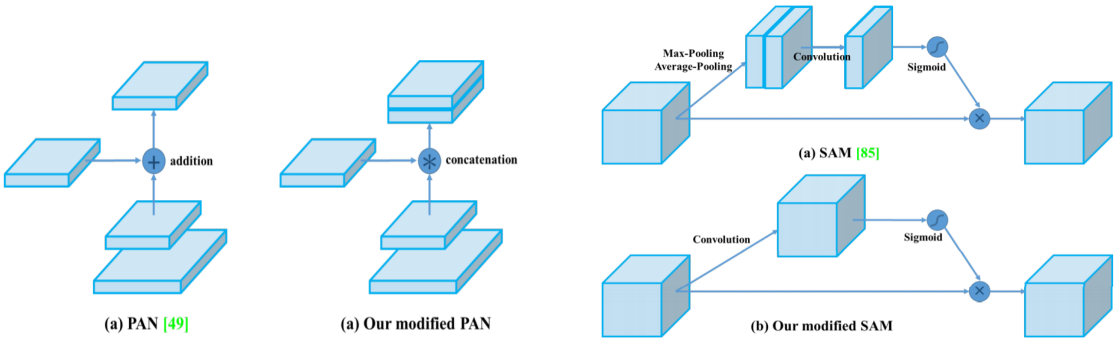

Figure 4.4: Modified PAN (left) and SAM (right) modules used in YOLOv4 [9].

For the BoS methods, the detector uses once again the Mish activation function. The detector also adds 2 modules to the network architecture: the SPP module and the SAM module. The SPP module [31] aggregates spatial information coming from the convolutional layers to obtain fixed-size feature maps without needing to warp or crop input images (thus preventing loss of accuracy coming from geometric deformation of the input). The SAM module [72][9] in YOLOv4 uses a convolutional layer followed by a sigmoid activation to extract an attention mask that determines where the relevant information is located. This is thus a different version of the original proposal which additionally used both max and average pooling to aggregate information prior to the convolution (see Fig. 4.4). Finally, the detector makes use of DIoU-NMS [77] as a criterion to suppress superfluous bounding boxes. A bounding box $\mathcal{B}_i$ will be removed if there is

a bounding box with higher prediction score $\mathcal{M}$ such that $IoU - \mathcal{R}_{DIoU}(\mathcal{M}, \mathcal{B}_i) \geq \varepsilon$ for a certain threshold ε. The term $\mathcal{R}_{DIoU}(\mathcal{M}, \mathcal{B}_i)$ is given by

$$\mathcal{R}_{DIoU}(\mathcal{M}, \mathcal{B}_i) = \frac{\rho^2(\mathbf{p}^{\mathcal{M}}, \mathbf{p}^{\mathcal{B}_i})}{c^2} \tag{4.4}$$

which matches the third term in equation (4.1).

4.1.2 The tracking algorithm

Given the context described at the beginning of the chapter, it is clear that the tracking algorithm does not need to be excessively complicated. For this, we designed a custom tracking algorithm to identify agents across frames and to keep track of them. The general idea is to match existing identities with identified objects in a "most likely" fashion. For this purpose, we keep track of the movement of objects in adjacent frames and use it to predict an "expected next position" for each of the identified agents. Each identified agents is then matched with the identity whose "expected next position" is closest to the detected one. This approach is illustrated in Figure 4.5. The main caveats of this approach are how to deal with missidentifications, agents moving off-screen, or new agents moving in from outside the scene.

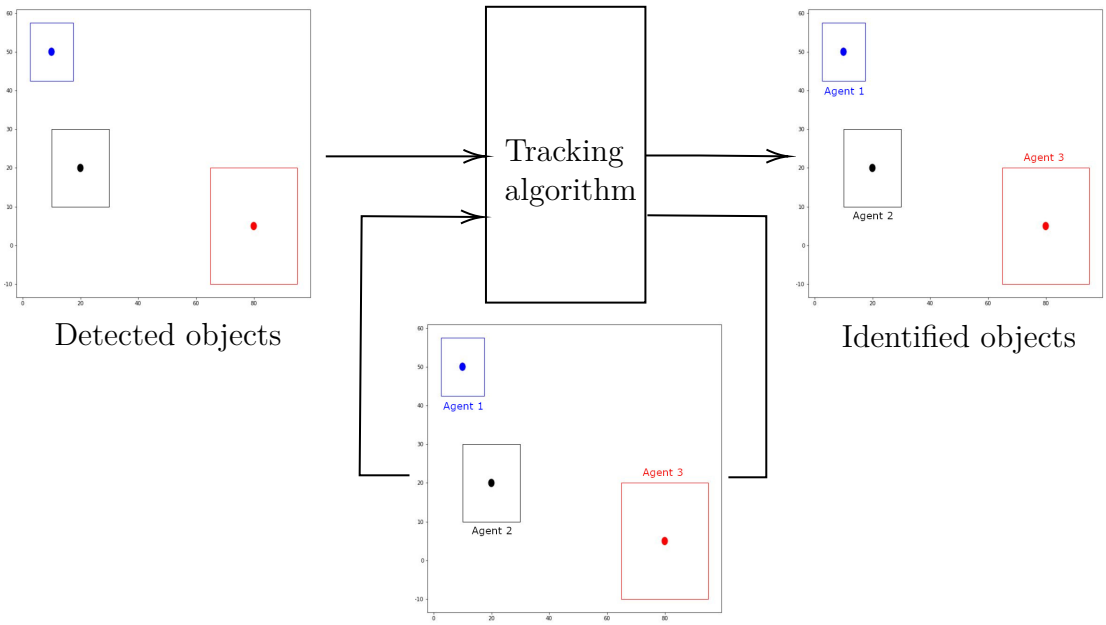

Figure 4.5: General structure of the tracking algorithm.

Identifying detected objects

To specify how the algorithm works exactly, let $\mathbf{O}_1, ..., \mathbf{O}_n \in \mathbb{R}^4$ be the bounding boxes of n detected objects at time t such that $\mathbf{O}_i = (\mathbf{p}_i, w_i, h_i) \in \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}$ where $\mathbf{p}_i$ specifies the coordinates of the object center and w_i, h_i specify, respectively, the width and height of the bounding box. Lets also consider a set of m bounding boxes for the expected identifications $\mathbf{P}_1, ..., \mathbf{P}_m \in \mathbb{R}^4$ at time t in the same fashion. Each one of the identifications is canonically associated with some $i \in \{1, ..., m\}$ and we say that $\mathbf{P}_i$ specifies the expected bounding box for agent i.

The dissimilarity between a detected object with bounding box $\mathbf{O}_i$ and an expected identification with bounding box $\mathbf{P}_j$ is measured as

$$m(\mathbf{O}_i, \mathbf{P}_j) = ||c^{\mathbf{O}_i} - c^{\mathbf{P}_j}|| + \log(1 + |h^{\mathbf{O}_i} - h^{\mathbf{P}_j}|) + \log(1 + |w^{\mathbf{O}_i} - w^{\mathbf{P}_j}|) \tag{4.5}$$

where the $\mathbf{O}_i$ and $\mathbf{P}_j$ superscripts are used to disambiguate. The function above determines a metric in $\mathbb{R}^4$. It is easily observed that $m(\mathbf{O}_i, \mathbf{P}_j) \geq 0$ and that $m(\mathbf{O}_i, \mathbf{P}_j) = 0$ if and only if $\mathbf{O}_i = \mathbf{P}_j$. Symmetry is obvious and the triangular inequality can be deduced from the fact that each of the terms verifies it. It is also interesting to note that the first term will generally be much more relevant than the other two, which are only important (in practice) when the center of the detected object and the center of the expected detection are very close.

Our algorithm matches each detected object, represented by its bounding box, $\mathbf{O}_i$ to an expected detection, represented by $\mathbf{P}_j$, by using Gale-Shapley's (GS) algorithm. For this purpose, preference lists are created for each detected object and for each expected detection. A detected object $\mathbf{O}_i$ will prefer being matched with $\mathbf{P}_j$ over $\mathbf{P}_k$ if $m(\mathbf{O}_i, \mathbf{P}_j) < m(\mathbf{O}_i, \mathbf{P}_k)$. The preference list is built analogously for each expected detection $\mathbf{P}_i$.

We decided to use GS instead of the more common Hungarian algorithm for 3 reasons. First, the GS algorithm returns matchings that are close to the optimal solution found via the Hungarian algorithm [46]. Second, we feel like the objective to be optimized is more natural in the context of matching detections and identities. In GS, we aim to match identities and detections in pairs such that any detection $\mathbf{O}_i$ (respectively, identity $\mathbf{E}_i$) of a given pair can't be matched to an identity $\mathbf{E}_j$ (detection $\mathbf{O}_j$) of another one such that identity $\mathbf{E}_j$ (detection $\mathbf{O}_j$) is a better match for detection $\mathbf{O}_i$ (identity $\mathbf{E}_i$) than its current identity (detection) and vice versa. On the other hand, the Hungarian algorithm aims to minimize the sum of the distances of each pair. Finally, the GS algorithm is more efficient than the Hungarian algorithm, running in $\mathcal{O}(nm)$ where n is the number of detections and m is the number of identities.

Calculating expected identifications

Expected identifications represent the position and bounding box at which we expect an object to appear in the next frame. They correspond to objects that have already been identified and should therefore appear in the following frames. The position is a prediction made based on the history of positions and bounding boxes for that specific identity (see Fig. 4.6).

Suppose that, for agent j, we have a history of detections $\mathbf{D}_1, ..., \mathbf{D}_t \in \mathbb{R}^4$ for frames $1, ..., t$ $t \in \mathbb{N}$, in chronological order. Then, at frame t, we define the variation in the position of agent j as $\Delta\mathbf{D} = \mathbf{D}_t - \mathbf{D}_{t-1} = (\Delta\mathbf{p}, \Delta h, \Delta w)$ where $\Delta\mathbf{p}$, Δh and Δw are, respectively, the variations in center position, height and width of the bounding box. With this information, the expected identification $\mathbf{P}_j$ for agent j at frame $t+1$ will be $\mathbf{P}_j(t+1) = \mathbf{D}_t + \Delta\mathbf{D}$. If $t \leq 1$ then $\Delta\mathbf{D} = 0$.

We observe that the process above is similar to the idea of a Kalman filter following a constant velocity model. The main difference is that this process is not probabilistic and does not take uncertainty or errors in measurements into account.

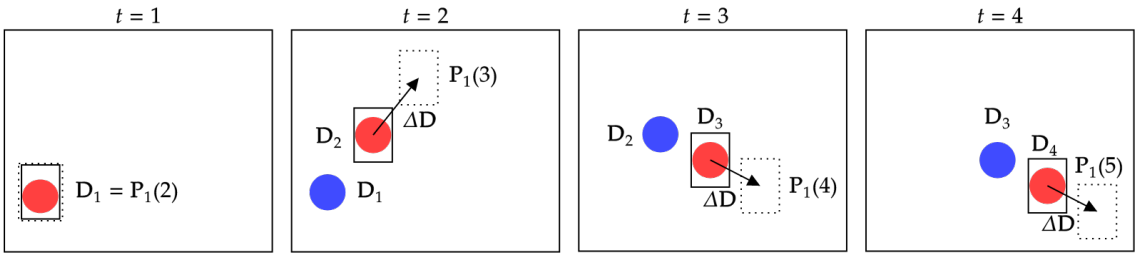

Figure 4.6: The expected position inference process illustrated. In red, the detections of the current frame. In blue, the detections in the last frame. As a dotted line, the expected bounding box and in a continuous line the bounding box of current detections.

Handling misidentifications, misdetections, and other problems in practice

Misdetections consist either of instances where the system detects objects that are not really in the frame or instances where it fails to identify objects that are actually in it (see Fig. 4.7). Misidentifications, on the other hand, represent failure to associate an object with its actual identity (see Fig. 4.8). Misdetections and misidentifications can potentially have a large impact on the performance of the system as a whole. For example, if the emotion recognition module takes into account the whole history of frames for a specific agent, then a misidentification could make subsequent predictions unusable.

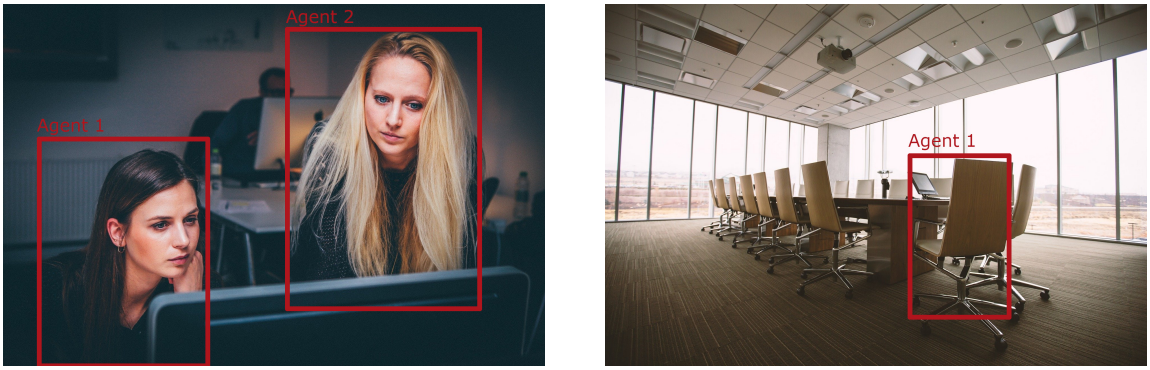

Figure 4.7: Two examples of misdetections.

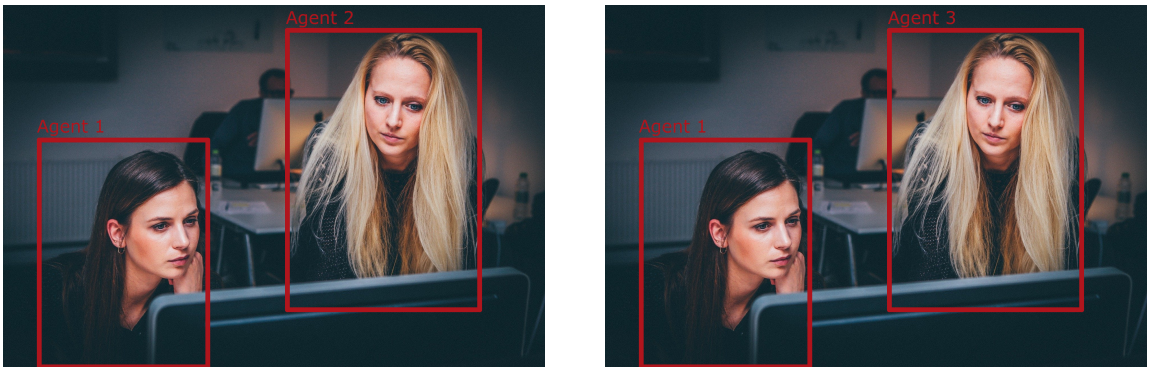

Figure 4.8: A misidentification in 2 successive frames. The system fails to properly recognize agent 2 in the second frame.

In practice, where ground truth is not available, it is virtually impossible to determine whether we are facing a misidentification or a misdetection. However, we can still mitigate

the effect of these errors and make the system more robust with respect to them. If the number of objects detected n and the amount of expected objects m do not match, then we are in one of the following scenarios

1. There has been at least a misidentification or misdetection.
2. An agent has walked out of the field of view of the camera.
3. A new agent has walked into the field of view of the camera.

Since we do not have a means to discern which one of the scenarios above best describes the situations that may appear in real-time, we need to propose a uniform handling strategy. For that reason, if there is a mismatch in the number of objects detected and the expected amount of objects (that is, $n \neq m$), we proceed as follows

1. If $n > m$ (more detected agents than expected detections) we assign the detected objects to the most suitable identity (as explained in the previous section) until there are no unassigned identities and we create new identities for the remaining $n - m$ agents.
2. If $n < m$ (less detected agents than expected detections) we first assign each of the detected objects to the most suitable identity. Then, we use a tolerance timer per identity τ_j, $j = 1, ..., m$, to decide whether to delete each of the excess identities in order to provide robustness to misdetection. If $\tau_j = 0$ and identity j is unused then we delete it. Otherwise we decrease τ_j.

Algorithm 3: Object tracking algorithm

Data: Object detections $\mathbf{O}_1, .., \mathbf{O}_n$ and expected detections $\mathbf{P}_1, ..., \mathbf{P}_m$
Result: A stable matching pairing each detected agent to an identity given by

$$M = \{(O_i, j_i) \mid i = 1, ..., n,\ j \in \{1, ..., \max(n, m)\},\ j_l \neq j_m \ \text{ if } l \neq m\}$$

Build preference lists for each detected object
Build preference lists for each expected detection
Match each agent to an identity that optimizes the metric $m(\cdot, \cdot)$ using the GS algorithm
if $n > m$ **then**
  Create new identities for extra objects.
end
if $m > n$ **then**
  for *each unused identity* j **do**
    if $\tau_j = 0$ **then**
      Delete identity j
    else
      $\tau_j = \tau_j - 1$
    end
  end
end
Return matching M

The time and space complexities for Gale-Shapley are $\mathcal{O}(nm)$, which is also the complexity of the full tracking algorithm.

4.2 Implementation

In order to minimize the amount of time necessary to implement this part of our system, we decided to use a readily available implementation of YOLOv4, present in the darknet library. We used Python 3.7.3 to build the tracking algorithm proposed in section 4.1.2 and to implement Gale-Shapley. We also make use of numpy for efficient array operations and opencv to interact with webcams and to work with images and video.

Our implementation of Gale-Shapley is completely general and works for arbitrary capacities of individual proposants and proposees.

```
def propose(proposant, propose, cap_proposes, pref_proposes,
↪   m_proposants, m_proposes):
    """
    Returns True if the proposal is successful and false otherwise.

    :proposant      int
    :propose        int
    :cap_proposes   list(int)
    :pref_proposes  list(int)
    :m_proposants   list(set())
    :m_proposes     list(set())
    """
    success = False
    # If there is place we simply accept
    if(cap_proposes[propose] > len(m_proposes[propose])):
        m_proposes[propose].add(proposant)
        success = True
    else: # If there is no place there are two options
        i = len(pref_proposes)-1 #last element in priority
        cont = True
        while cont:
            if pref_proposes[i] in m_proposes[propose]:
                # 1. Candidate better than worst match -> Accept
                idx_worst = pref_proposes[i]
                m_proposants[idx_worst].remove(propose)
                m_proposes[propose].remove(idx_worst)
                m_proposes[propose].add(proposant)
                success = True
                cont = False
            elif pref_proposes[i] == proposant:
                # 2. Candidate worse than worst match ->  Reject
                cont = False
            else:
                i -= 1

    return success
```

```
def gale_shapley(n_proposants, n_proposes, cap_proposants, cap_proposes,
↪  pref_proposants,
                 pref_proposes):
    """
    Receives a map of string to int that indicates, for each proposant
↪   or proposee
    the index at its preference matrix. Assumes values are indexed in
↪   the preference
    matrices.

    :proposants          int
    :proposes            int
    :cap_proposants      list(int)
    :cap_proposes        list(int)
    :pref_proposants     list(list(int))
    :pref_proposes       lsit(list(int)).
    """
    m_proposants = [set() for x in range(n_proposants)]
    m_proposes = [set() for x in range(n_proposes)]
    cont,current = (True,0)
    curr_prop = [0 for x in range(n_proposants)]

    if (n_proposants > 0 and n_proposes > 0):
        while cont:
            while (len(m_proposants[current]) ==
            ↪  cap_proposants[current]):
                current = (current+1) % n_proposants

            # Everyone must propose once
            i, proposal = curr_prop[current], True # Proposes until
            ↪  accepted
            while (proposal and i < len(pref_proposants[current])):
                proposee_idx = pref_proposants[current][i]

                if(propose(current, proposee_idx, cap_proposes,
                ↪  pref_proposes[proposee_idx],
                        m_proposants, m_proposes)):
                    m_proposants[current].add(proposee_idx) # Update
                    ↪  proposant
                    if(len(m_proposants[current]) ==
                    ↪  cap_proposants[current]):
                        proposal = False # We end the proposal process
                i = i+1
            # At least a proposant is free
            cont = reduce(lambda x,y: (x or len(y[0]) < y[1]),
            ↪  zip(m_proposants,cap_proposants), False)
            # At least a proposee is free
```

```
            cont = cont and reduce(lambda x,y: (x or len(y[0]) < y[1]),
            ↪  zip(m_proposes, cap_proposes), False)

    return (m_proposants, m_proposes)
```

To create the tracking function, we also need an implementation for the metric. This is done by using numpy's built-in functions and via slicing, as the bounding boxes are represented by 4-dimensional arrays (x, y, w, h).

```
def metric(B1, B2):
    """ Given two bounding boxes, returns the value of the metric. """
    return (np.linalg.norm(B1[0:2] - B2[0:2]) +
            np.log(1+abs(B1[2]-B2[2])) +
            np.log(1+abs(B1[3]-B2[3])))
```

Finally, the tracking function itself transforms the input into data structures suitable to apply Gale-Shapley and calls the matching algorithm. Matches are transformed from $list(set())$ to $list(int)$ and the tolerance and identity updates explained in algorithm 3 are applied. The parameter A indicates the last assigned agent number to guarantee uniqueness while the tol parameter is used to adjust the maximum tolerance value.

```
def naive_track(assig, expected, centers_t, tolerance, tol=2, A=0):
    """
    Tracks the given agents: associates each point
    of centers_t with an agent identity given the previous
    assignment, the expected detections, and the current tolerance
    value for each agent.

    The return values are a map from identity to point, which stores
    the last known position of each agent; and a list
    of strings, where the string in index i is the identity of the
    ith point in centers_t.

    :assig      dict(str->point)
    :expected   dict(str->point)
    :centers_t  list(point)
    :tolerance  dict(str->int)
    :tol        int
    :A          int
    :return     (dict(str->point),list(str))
    """
    # Heavily affected by granularity in time discretization
    n,m = len(expected), len(centers_t)
    previous_centers = list(expected.items()) #[(agent,val) for
    ↪  agent,val in expected.items()]
    # Create "distance" matrix (len(expected) x len(centers_t))
    d_matrix = np.array([[metric(bb_prev,bb_det) for bb_det in centers_t]
    ↪  for _,bb_prev in expected.items()])
    prefs_prev = np.array([sorted(list(range(m)), key=lambda
    ↪  j:d_matrix[i][j]) for i in range(n)]) # sort by distance (rows)
```

```
    prefs_det = np.array([sorted(list(range(n)), key=lambda
    ↪  i:d_matrix[i][j]) for j in range(m)]) # sort by distance
    ↪  (columns)
    # Call GS algorithm (returns list of size 1 sets)
    match_prev, match_det = gale_shapley(n, m, [1 for _ in range(n)], [1
    ↪  for _ in range(m)], prefs_prev, prefs_det)
    match_prev = [next(iter(x)) if len(x) != 0 else None for x in
    ↪  match_prev]
    match_det = [next(iter(x)) if len(x) != 0 else None for x in
    ↪  match_det]
    # Tracking will work properly if the agents' range of movement is
    ↪  smaller than
    # 1/(2*tol) times their separation
    assignment_dict = dict()
    assignment_list = []
    for i in range(m):
        if match_det[i] is None: # New agents
            identity = gen_identity(A)
            assignment_dict[identity] = centers_t[i]
            assignment_list.append(identity)
            A += 1 # upper bound for last assigned agent number
        else:   # Existing agent
            identity = previous_centers[match_det[i]][0]
            assignment_dict[identity] = centers_t[i]
            assignment_list.append(identity)
        tolerance[identity] = tol

    for i in range(n):
        if match_prev[i] is None: # Misdetection or agent left
            identity = previous_centers[i][0]
            if tolerance[identity] > 0: # Add to match with last
            ↪  position
                assignment_dict[identity] = assig[identity]
                assignment_list.append(identity) # These will appear
                ↪  after all detections
                tolerance[identity] -= 1
            else:
                del tolerance[identity]
                print("Deleting agent",identity)
                print("Current assignment is", assignment_dict)

    return assignment_dict, assignment_list, A
```

Chapter 5

The emotion recognition module

The last building block of our system is the emotion recognition module. This module is in charge of inferring the dimensional scores explained in Chapter 3 by using the tracking feeds extracted by the tracking module (see Chapter 4). At each point in time t, the emotion recognition module processes the identified frames generated by the tracking module and outputs the dimensional scores for each agent (see Fig. 5.1). We treat emotion recognition as a regression problem, where we want to estimate each one of the dimensional scores (Af, En, Le, Ec) in the range $[0, 1]$ for a given input video. We decided to further simplify this by considering that a pointwise estimation is good enough. That is, we calculate the scores only for the frames generated at time t without using information from previous frames.

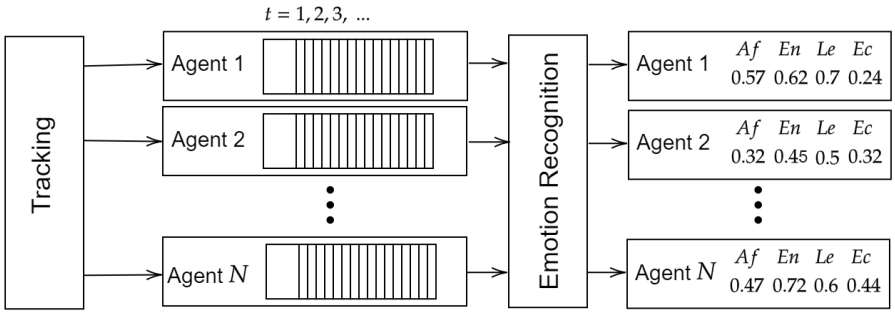

Figure 5.1: The emotion recognition pipeline illustrated.

Tracking feeds provide the system with continuous information for each one of the agents. However, as we will discuss further in this chapter, making use of the temporal information available added an additional layer of complexity to the module design that did not make sense for a proof-of-concept. As we have done in Chapter 3 with the tracking module, we opted for a simple implementation that allowed us to obtain results quickly.

In the following sections the emotion recognition module will be presented in detail. First, we will introduce the system architecture. Then, we will detail its implementation. Finally, we will present the results obtained on the ERVF dataset and the system limitations.

5.1 Architecture

The architecture is composed of two parts. First, a convolutional neural network is used to extract convolutional features from the input images. Then, four regressors (one for each dimension) determine the dimensional scores associated with the extracted features. We propose the usage of MobileNetV3 [35] pretrained on the ImageNet [18] dataset for feature extraction and FC layers with sigmoid activation to perform the regression (see Fig. 5.2).

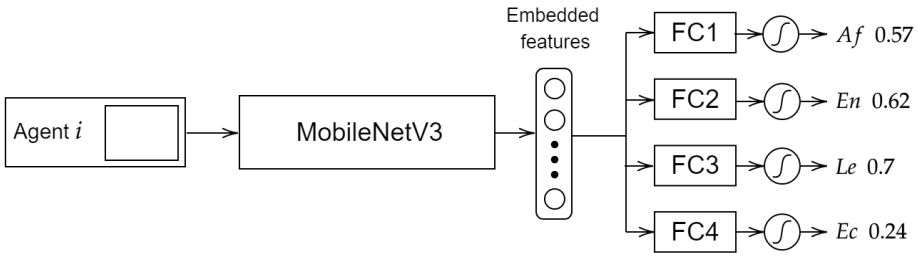

Figure 5.2: The emotion recognition module architecture.

MobileNetV3 is a very lightweight CNN and is capable of producing semantically meaningful $1,000$-dimensional embeddings for the regressors to work with. Our regressors work by linearly combining the embedded features and clipping the result to the $[0, 1]$ range via the sigmoid activation. For the network training we decided to use the Sum of Absolute Errors (SAE) across each of the dimensions

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{4} |y_i - \hat{y}_i| \tag{5.1}$$

where $\hat{\mathbf{y}}$ denotes the ground truth scores and $\mathbf{y}$ denotes the predicted scores.

We chose this loss mainly because the dimensional scores are in $[0, 1]$. Working with such low values, other common loss functions like MSE artificially reduced the loss value. In particular we wanted to avoid squaring the error value in order to keep it from vanishing prematurely. This is also our reasoning to take the sum instead of the mean value. We must also note that the loss function above has the disadvantage of $\mathcal{L}(\hat{\mathbf{y}}, \cdot)$ not being differentiable at every point, potentially leading to a slower convergence for gradient-based methods.

5.1.1 MobileNetV3

MobileNets are a set of CNN models designed to optimize performance while keeping reasonably high accuracy scores. They are meant to work on systems with lower amounts of resources like smartphones or even embedded systems [15]. MobileNetV3 is the latest iteration of the original MobileNet architecture. The original MobileNet reduced the cost of running a CNN by using depthwise separable convolutions [34]. Depthwise separable convolutions refers to the set of convolutions that can be expressed as a concatenation of a depthwise convolution and a pointwise (1×1) convolution. Depthwise convolutions with kernel differ from traditional convolutions in the fact that they use a single kernel per channel and only convolve each one of them across their respective channel. Furthermore,

these kernels must have a single channel, as opposed to the same number as the input image. Depthwise convolutions also have a fixed number of filters that matches the input number of dimensions. The pointwise convolution that follows is the one that determines the number of output channels (see Fig. 5.3).

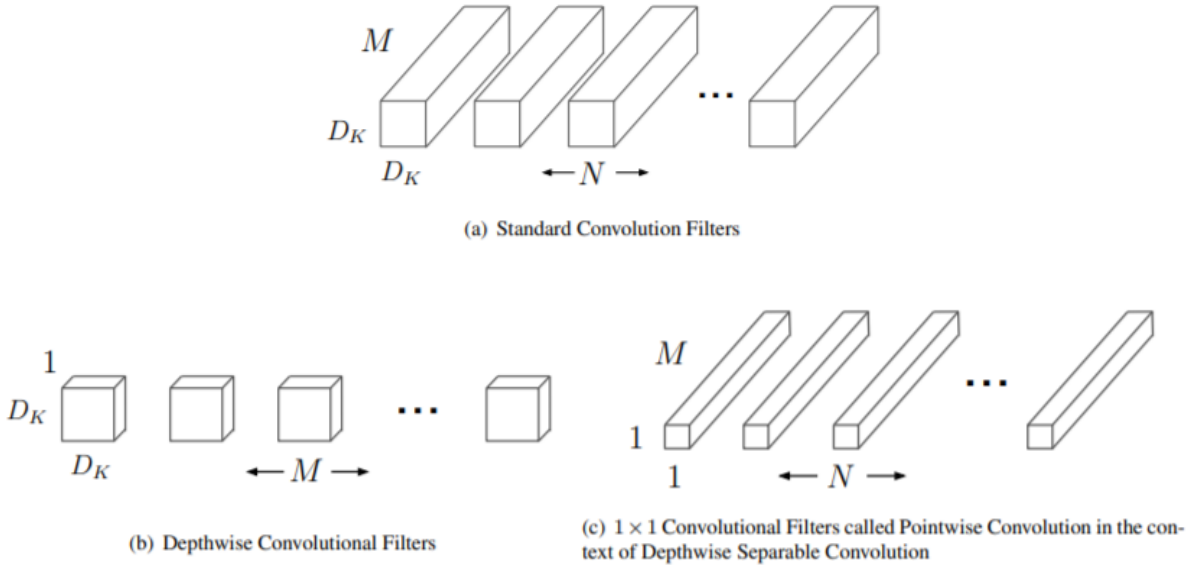

Figure 5.3: Traditional convolution (a) and its separation as depthwise convolution (b) followed by a pointwise convolution (c) [34].

In the second iteration, MobileNetV2, the authors proposed two main improvements [60]: the usage of linear bottlenecks in convolutions and inverted residual blocks. The term linear bottleneck refers to the usage of layers with small output dimension and without a non-linear activation after the convolution (see Fig. 5.4). This reduces the destruction of information about the region in which the data lies caused by the application of non-linear functions such as ReLU. The inverted residual blocks work use linear bottlenecks and a convolution in order to first expand then contract the data that flows through the network. The residual skip connection is then done between the low-dimensional bottlenecked representations as opposed to the high-dimensional ones in a typical residual block (see Fig. 5.4). In MobileNetV3, the inverted residual blocks are provided with a Squeeze-and-Excitate [36] mechanism (see Fig. 5.5), which works as an attention mechanism across channels, allowing the network to detect the most informative channels. In Squeeze-and-Excitate, each channel is pooled into a single value via global average pooling and two FC layers with ReLU and sigmoid activation provide weight values for each of the original channels.

The final building block for MobileNetV3 is the architectural search process that serves to optimize the architecture. The full description of the architectural search is out of the scope of this document, but we will give a brief overview. The search process works in two steps. First, the MnasNet-A1 [63] platform-aware Neural Architecture Search (NAS) is used to find an architecture that balances a high accuracy with a low latency (the ideal proposal being a Pareto optimal one). Then, the NetAdapt [75] algorithm is applied on the architecture generated by platform-aware NAS. This algorithm iteratively improves an architecture by trying to apply a set of proposed changes that reduce latency by a certain amount (in the case of MobileNetV3 either reducing the size of an expansion layer

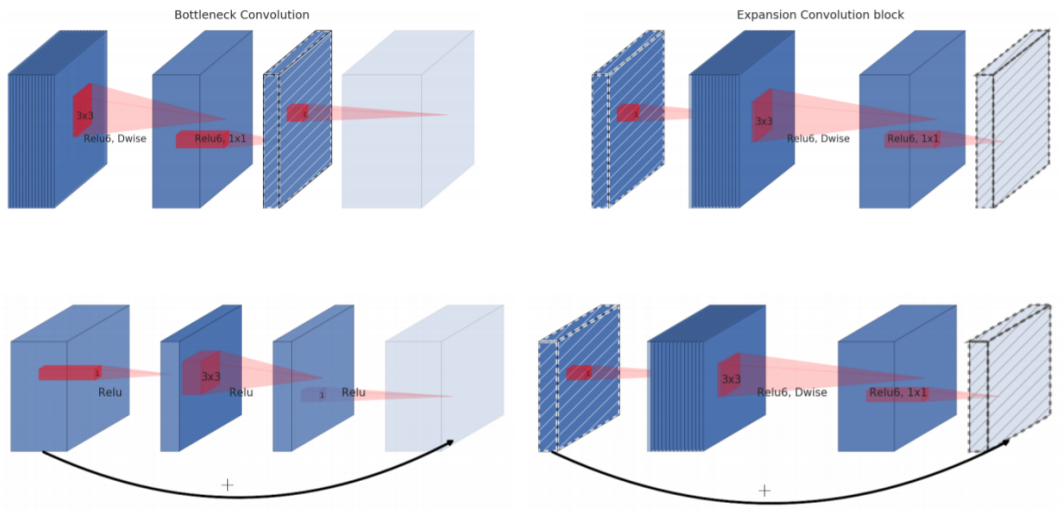

Figure 5.4: Architecture of a bottleneck convolution block (top left) and an expansion convolution block (top right) and comparison between a residual block (bottom left) and an inverted residual block (bottom right) [60]. Diagonally hatched layers do not contain non-linearities.

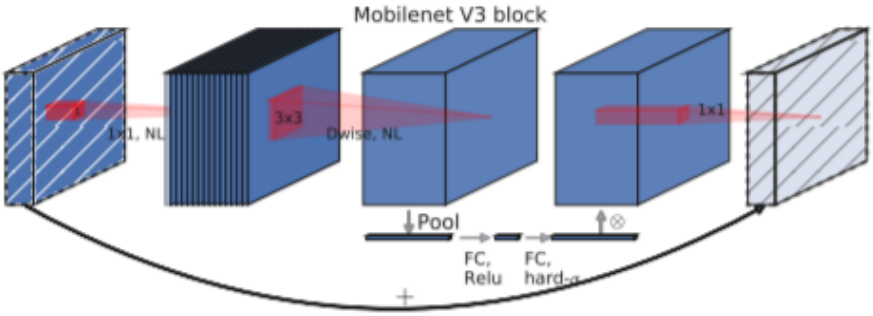

Figure 5.5: Architecture of a MobileNetV3 block, which includes the inverted residual structure and Squeeze-and-Excite [36][35].

or reducing the bottleneck size for all blocks with a certain size) and selecting the best architecture according to some metric.

Additionally, the authors apply other optimizations on the network. First, some layers that were found to be expensive are manually modified. For example, the initial number of filters is halved from 32 to 16, and the last 1×1 convolution in the bottleneck layer is moved past the global average pooling. The Squeeze-and-Excite module used in MobileNetV3 only has a size of $\frac{1}{4}$ of the number of channels in the expansion layer. Finally, a new version of the swish non-linearity that is faster to compute is used: h-swish. At the same, time h-swish depends on h-sigmoid non-linearity, an activation similar to the sigmoid that is faster to compute. These functions are given by

$$\text{h-sigmoid}(x) = \frac{\text{ReLU6}(x+3)}{6} \quad \text{h-swish}(x) = x\frac{\text{ReLU6}(x+3)}{6} = x \cdot \text{h-sigmoid}(x) \quad (5.2)$$

where $\text{swish}(x) = x\sigma(x)$, $\sigma(x)$ is the sigmoid function, and $\text{RELU6}(x) = \min(\max(0, x), 6)$. The $h - swish$ function is not used across the whole network, but only on the second

half. The reason is that the function is still more expensive than, for example, *ReLU*, and the accuracy improvement of using it is still obtained even when restricting its use to deeper layers in the network.

5.1.2 Regressors

Each regressor is composed of an FC layer with a sigmoid activation. In particular, our regressor has 1-dimensional output and $\mathbf{x} \in \mathbb{R}^n$ (the embedded features) as input. Therefore, we can describe each regressor R as $\mathrm{R}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$ where σ denotes the sigmoid function, $\mathbf{w}$ is a set of weights, and b is the bias term. Collectively, $\mathbf{w}$ and b are the trainable parameters of our regressor. The purpose of the sigmoid function $\sigma : \mathbb{R} \to (0, 1)$ is to clip the output to the adequate range. In the case of our embedding, the input size is $n = 1,000$.

An observation one may make is that, while the range of our dimensional scores is $[0, 1]$, the sigmoid function only outputs values in the open interval $(0, 1)$. However, since dimensional scores of 0 and 1 are rare and the absolute error can be made arbitrarily small by adjusting the trainable parameters we do not consider this to be an issue.

5.2 Implementation

The implementation of the emotion recognition module makes use of the following libraries:

- Numpy 1.20.3.
- PyTorch 1.8.
- OpenCV (cv2) 4.5.2.
- Torchvision 0.9.1.
- PIL 8.2.0.
- Matplotlib 3.4.2.

Additionally, it is highly advisable to install compatible verions of CUDA and cuDNN to make use of hardware acceleration during training and testing. For the remainder of this section we will consider the following imports

```
import torch
import torchvision.models as models
import torchvision.transforms as transforms
import numpy as np
import cv2
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error,
↪  mean_absolute_percentage_error, r2_score
from PIL import Image
```

We will also consider the global variables that indicate the paths to the model weights, dataset, video data subdirectory, and score data subdirectory. Respectively, these will be `MODEL_DIR`, `DATA_DIR`, `VIDEO_DIR`, and `SCORE_DIR`. A file in the dataset directory

indicates all of the video filenames and their associated score filenames. To read them, we use the following function

```
def read_filenames():
    video_files = []
    score_files = []
    with open(DATA_DIR + 'files.txt') as f:
        for line in f:
            vf,sf = line.strip('\n').split(',')
            video_files.append(vf)
            score_files.append(sf)
    return video_files,score_files
```

Specifically, the one-liner `video_files,score_files = read_filenames()` reads all of the filenames into the specified variables.

5.2.1 Network architecture

With the above specification, the architecture itself has a very straightforward implementation.

```
class Net(torch.nn.Module):

    def __init__(self):
        """Network initialization."""
        super(Net,self).__init__()
        self.backbone = models.mobilenet_v3_small(pretrained=True)
        self.classifiers = [torch.nn.Linear(1000,1) for _ in range(4)]

    def forward(self,x):
        """Network forward pass."""
        x = self.backbone(x)
        sigmoid = torch.nn.Sigmoid()
        x1,x2,x3,x4 = [sigmoid(f(x)) for f in self.classifiers]
        x = torch.cat((x1,x2,x3,x4),dim=1)
        return x
```

5.2.2 Dataset loading

Since the dataset is too large to keep in memory, we must use a data generator. Pytorch provides the tools to implement an efficient generator that uses multiple cores to speed up the data generation process. First, we consider the following function, which allows us to generate labels and scores for each frame given a video filename and its associated score filename.

```
def generate_video_labels(vid_filename,score_filename, drop=20):
    """Given a video filename and its associated score filename,
    returns the frame ids and scores downsampled by the drop factor."""
    cap = cv2.VideoCapture(DATA_DIR+VIDEO_DIR+vid_filename)
    length = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))-1
    frame_labels = [vid_filename+'_frame_'+str(i+1) for i in
    ↪  range(0,length,drop)]
```

```
    frame_scores = np.load(DATA_DIR+SCORE_DIR+score_filename)[:-1:drop,:]
    return frame_labels,frame_scores
```

Using the previous function repeatedly, we can generate the frame labels and scores for the entire dataset, reducing the framerate as we see fit via the `drop` parameter below.

```
def generate_dataset_info(filename, drop=20):
    """Creates identifiers and scores for from the information file
    downsapling videos by the drop factor."""
    video_files,score_files = read_filenames(filename)
    # Generate identifiers for each frame in the video files
    frame_ids,frame_scores = [],[]
    for v_fname,s_fname in zip(video_files,score_files):
        vf_id,vf_scores = generate_video_labels(v_fname,s_fname, drop)
        frame_ids += vf_id
        frame_scores += list(vf_scores)
        print("Video:",v_fname," - ",(len(vf_id),len(vf_scores)))
    return frame_ids, frame_scores
```

We partition the data into train, validation, and test datasets using the function above. A set of agents were selected to be used for training only, validation only, and testing only. The files for each dataset are specified in `files_train.txt`, `files_validation.txt`, and `files_test.txt`.

The dataset loader is inspired by a parallel data loader proposed at Shervine Amidi's blog [3]. It is, however, adapted to our use case and to load image data. The data generator is a class that receives a dictionary and a list in its constructor: the list indicates the identifiers of the data to be loaded and the dictionary maps each identifier to its associated label. In our case, the first dictionary indicates the frame identifier and the second one the dimensional scores associated with the specific frame. The `__getitem__()` function is then used to generate a single datapoint from a given index. Our dataset loader also applies standard preprocessing (images are resized normalized with ImageNet's mean and standard deviation values for each channel) to each frame to allow MobileNet to infer adequately.

```
class Dataset(torch.utils.data.Dataset):
  def __init__(self, list_IDs, labels):
        """Initializes the Dataset object with the list of frame
        ↪  identifiers
        and the label dictionary."""
        self.labels = labels
        self.list_IDs = list_IDs
        # Normalization params for torchvision models
        mean = [0.485, 0.456, 0.406]
        std = [0.229, 0.224, 0.225]
        # Preprocessing layer
        self.preprocess = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
```

```
            transforms.ToTensor(),
            transforms.Normalize(mean=mean, std=std)])

    def __len__(self):
        """Returns the total number of frames in the Dataset."""
        return len(self.list_IDs)

    def __getitem__(self, index):
        """Generates a processed frame with its associated scores."""
        # Select sample
        ID = self.list_IDs[index]
        # Extract video name from ID
        v_name,frame = ID.split('.')
        v_name += '.webm'
        frame = int(frame.split('_')[-1])
        # Load frame and label
        cap = cv2.VideoCapture(DATA_DIR+VIDEO_DIR+v_name)
        cap.set(1,frame)
        _,frame = cap.read()
        cap.release()
        frame=Image.fromarray(frame,'RGB')
        label = self.labels[ID]
        return self.preprocess(frame), torch.tensor(label).float()
```

The generators are then built with the following code.

```
# Creating dataset info (train,test,validation)
train_ids, train_scores = generate_dataset_info('files_train.txt',
↪  drop=20)
val_ids, val_scores = generate_dataset_info('files_validation.txt',
↪  drop=20)
test_ids, test_scores = generate_dataset_info('files_test.txt',drop=20)

# Partition initialization
partition = dict()
labels = dict()
partition['train'] = train_ids
partition['validation'] = val_ids
partition['test'] = test_ids
for x,y in zip(train_ids,train_scores):
    labels[x] = y
for x,y in zip(val_ids,val_scores):
    labels[x] = y
for x,y in zip(test_ids,test_scores):
    labels[x] = y

params = {'batch_size': 256,
          'shuffle': True,
          'pin_memory': True,
```

```
          'num_workers': 0}

train_dataset = Dataset(partition['train'], labels)
val_dataset = Dataset(val_ids,labels)
test_dataset = Dataset(test_ids,labels)

train_generator = torch.utils.data.DataLoader(train_dataset, **params)
val_generator = torch.utils.data.DataLoader(val_dataset, **params)
test_generator = torch.utils.data.DataLoader(test_dataset, **params)
```

5.2.3 Training, validation, and testing

To train a pytorch network it is necessary to program the training loop explicitly. We trained our system using the Adam optimization method using the following code.

```
# CUDA for PyTorch
use_cuda = torch.cuda.is_available()
device = torch.device("cuda:0" if use_cuda else "cpu")
torch.backends.cudnn.benchmark = True
print("USE_CUDA:",use_cuda)

# Training parameters
max_epochs = 20
model = Net()
if use_cuda:
    model = model.cuda()
criterion = torch.nn.L1Loss(reduce="sum")
optimizer = torch.optim.Adam(model.parameters(),lr=0.0005)
N_AVG = 4

train_history,val_history = [],[]
# Training and validating the network
for epoch in range(max_epochs):
    ############################## Training ##############################
    model.train()
    total_loss,running_loss,it = 0.0,0.0,1
    for local_batch, local_labels in train_generator:
        # Transfer to GPU and model computations
        local_batch, local_labels = local_batch.to(device),
        ↪  local_labels.to(device)
        print('Training batch %i/%i' % (it, len(train_generator)))
        optimizer.zero_grad()
        out = model(local_batch)
        loss = criterion(out, local_labels)
        loss.backward()
        optimizer.step()
        running_loss += float(loss)
        total_loss += float(loss)
        it += 1
        if it % N_AVG == 0:
```

```
            print('<<Training>> [%d, %5d] loss: %.3f' % (epoch + 1, it +
            ↪  1, running_loss / N_AVG))
            running_loss = 0.0
    print("<<Training>> Accumulated loss: %.3f Average loss: %.3f" %
    ↪  (total_loss, total_loss/len(train_generator)))
    train_history.append(total_loss/len(train_generator))
    ################################## Validation ####################
    model.eval()
    total_loss,running_loss,it = 0.0,0.0,0
    it = 1
    with torch.set_grad_enabled(False):
        for local_batch, local_labels in val_generator:
            # Transfer to GPU and model computations
            local_batch, local_labels = local_batch.to(device),
            ↪  local_labels.to(device)
            print('Validation batch %i/%i' % (it, len(val_generator)))
            out = model(local_batch)
            loss = criterion(out, local_labels)
            running_loss += float(loss)
            total_loss += float(loss)
            it += 1
            if it % N_AVG == 0:
                print('<<Validation>> [%d, %5d] loss: %.3f' % (epoch + 1,
                ↪  it + 1, running_loss / N_AVG))
                running_loss = 0.0
    print("<<Validation>> Accumulated loss: %.3f Average loss: %.3f" %
    ↪  (total_loss, total_loss/len(val_generator)))
    val_history.append(total_loss/len(val_generator))
print('Training loss (epochwise):', train_history)
print('Validation loss (epochwise):', val_history)
```

Note that the code above reports statistics every `N_AVG` batches. For each epoch, the network is trained on the training set and subsequently validated on the validation set. We monitored the training for 18 epochs. After 8 epochs, we found that validation and training losses started to diverge (see Fig. 5.6). Therefore, the network was retrained from scratch for 8 epochs on the combined train and validation datasets.

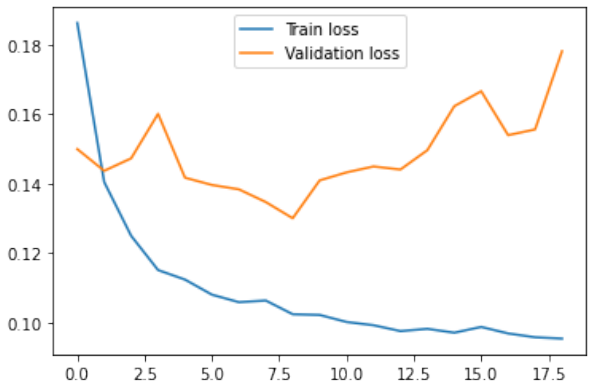

Figure 5.6: Evolution of training and validation losses.

The 3 videos corresponding to a randomly chosen agent were held out for testing purposes. We ran our model on the testing set and calculated the average inference time, of about 37 *ms* or around 27 FPS.

```
def test_model(model, generator, use_cuda=False):
    """Returns the results of the model on the given generator for each
    ↪  dimension as a dictionary,
    as well as the predictions, ground truth and average inference
↪   time."""
    model.eval()
    predicted,labels,inf_times = [],[],[]
    it = 1
    with torch.set_grad_enabled(False):
        for local_batch, local_labels in generator:
            # Transfer to GPU and model computations
            local_batch, local_labels = local_batch.to(device),
            ↪  local_labels.to(device)
            print('Testing batch %i/%i' % (it, len(generator)))
            if use_cuda:
                start = torch.cuda.Event(enable_timing=True)
                end = torch.cuda.Event(enable_timing=True)
                # Record inference time
                start.record()
                out = model(local_batch)
                end.record()
                # Sync
                torch.cuda.synchronize()
                inf_times.append(start.elapsed_time(end))
            else:
                # Record inference time
                start = time.time()
                out = model(local_batch)
                end = time.time()
                inf_times.append(end-start)
            predicted.append(out)
            labels.append(local_labels)
            it += 1
    pred,true=[],[]
    pred_np,true_np = [],[]
    if use_cuda:
        pred_np = [x.cpu().numpy() for x in predicted][:-1]
        true_np = [x.cpu().numpy() for x in labels][:-1]
    else:
        pred_np = predicted[:-1]
        true_np = labels[:-1]
    pred = np.stack(pred_np, axis=0).reshape(-1, 4)
    true = np.stack(true_np, axis=0).reshape(-1, 4)
    cols = ['Af','En','Ec','Le']
```

```
scores_by_col = dict()
for col in [0,1,2,3]:
    scores_by_col[cols[col]] = {"MSE":
    ↪  mean_squared_error(true[:,col], pred[:,col]),
    "MAE": mean_absolute_error(true[:,col], pred[:,col]),
    "MAPE": mean_absolute_percentage_error(true[:,col], pred[:,col]),
    "R2":r2_score(true[:,col], pred[:,col])}
return pred,true,scores_by_col,sum(inf_times)/len(inf_times)
```

5.3 Results and limitations

There are several metrics that measure how effectively a regression system is. To evaluate our system, we opted to use the following 4 for each dimension of the audience experience: Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R^2 score. If $Y = \{y_1, y_2, ..., y_n\} \subset \mathbb{R}$ is the set of predictions and $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\} \subset \mathbb{R}$ is the set of ground truth values, the metrics above are calculated as follows

$$\text{MSE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \qquad \text{MAE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{\max(\hat{y}_i, \varepsilon)}\right| \qquad R^2(\hat{Y}, Y) = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of $\hat{Y}$. Ideally, we wish to minimize MSE, MAE, and MAPE and maximize R^2. We must make a couple observations about our metrics

- MSE is prone to vanishing when working with very small diferences (because of the squared values in the sum). It is also expresses dissimilarity in terms of the absolute difference between values and its interpretation must be made carefully.
- MAE also expresses dissimilarity in terms of the absolute difference and its interpretation must also be careful.
- MAPE score may report high values due to low ground truth scores.
- R^2 may report low values due to ground truth values being too close to the mean.

Since our dimensional scores are in the $[0, 1]$ range, we must be particularly careful when assessing MAPE and R^2. To evaluate our problem, we take particular interest at having low MAE values, which we believe best reflects our model's performance.

Table 5.1: Results on the test dataset.

	MSE	**MAE**	**MAPE**	**R^2**
Affective Response	0,0861	0,2557	1,2240	-0,1142
Engagement	0,1556	0,3093	14,0547	-0,3560
Emotional Connection	0,0757	0,2324	0,7135	-0,4725
Learning	0,1775	0,2906	6.3201	-0,4725
Combined	0,1099	0,2642	1,5040	-0,2330

We tested our model on 3 unseen videos of a randomly chosen agent that the network had no previous information on. The results for each dimension and their combined values are shown in Table 5.1. As we can see, MSE has relatively low values while MAPE is very large. R^2 being negative indicates that a constant prediction matching the mean of the test dataset would perform better than our system. While the results in Table 5.1 are not conclusive, attending to MAE, the system seems to be able to discriminate between extreme cases (e.g: very low or very high Engagement). Furthermore, the poor R^2 score may be influenced by the linear inference used to determine the scores and the MAPE values may be influenced by the scale of the residues.

The results may also be influenced by some of the limitations of our system and our dataset:

- The dataset is composed of highly correlated data since all of the videos correspond to 7 participants. A larger-scale experiment would undoubtedly provide us with a richer and more varied dataset and, presumably, a more robust system.
- The dataset is relatively small once the downsampling has taken place. To a certain extent, it is possible that data augmentation could be used to paliate this issue.
- The proposed model is very simple. The main assumption that may not hold is that we can express each of the dimensional scores as the sigmoid of a linear function of the convolutional features. Additional layers in the regressors may allow them to provide a better estimation, although the system may be more prone to overfitting.
- The proposed model does not take into account temporal dependencies. We expect the dimensional scores of a certain individual to evolve "continuously" in time. That is, we expect that person to have similar dimensional scores at close points in time. An RNN model may be able to solve this problem, for example, by sharing the previous dimensional scores with the regressors.

Besides test metrics and results, two important lines of work to improve this system are that of explainability and fairness. Explainability is helpful in understanding exactly how to solve existing issues in the system and is key in ensuring that the system is really working as expected. Fairness studies are necessary in order to detect potential bias and noise in artificial intelligence systems, and poor fairness results may be the direct cause of performance issues. Addressing the above limitations should be the next step in the path to building a robust and accurate system for automatic audience experience measurement.

Chapter 6

Conclusions

In this document we have presented our 3 main contributions: a comprehensive framework to quantify audience experience, the Emotion Recognition from Video Feeds (ERVF) dataset to train machine learning systems using the previous framework, and a proof-of-concept machine learning system[1] that estimates audience experience from video in real-time. We have also been able to improve the matching step commonly used in tracking-by-detection systems, reducing the complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.

While there are proposals on how to measure the audience experience, as explained in Chapter 2, currently there is no standard methodology for doing so. Our theater-based framework is meant to be a first step in solving the issue. Similarly, using this framework to train ML systems requires the availability of suitable data. The experiment described in Chapter 3 allows for the construction of suitable datasets at any desired scale. The ERVF dataset provides a starting point in this regard. Finally, the object tracking and emotion recognition systems, described in Chapter 4 and Chapter 5 respectively, serve as a baseline in the design of systems for automatic audience experience estimation from video.

Additionally, we have found that most of the widely available video conferencing software only allows for very limited control. To address this issue, we have proposed our own video conferencing tool, whose code is presented in Appendix A, tailored specifically for our needs.

Our work can be applied and have an impact in a broad set of fields. In education, having a system to automatically assess the state of students allows teachers to adjust their discourse on the fly. In comedy, it allows the comedian to assess the quality of their act. More generally, in any event where public speaking is involved, it allows the speaker to receive immediate feedback and adjust accordingly.

In order for this impact to be meaningful, some of the major limitations must be addressed. Some of the dimensions of our framework are still very abstract and hard to measure. Furthermore, measuring the dimensional scores of the framework experimentally is an intrusive process, in the sense that filling a survey interrupts the audience's experience. The information obtained as described in Chapter 3 is also highly redundant,

[1]The code of the full ML system is available in our GitHub repository `https://github.com/pablo-vs/emotion-recognition`

both due to the inference process and due to the audience experience generally being continuous (similar for close points in time). This redundancy hurts the performance of ML systems.

When it comes to our ML system, there is also room for improvement. The tracking module is sensitive to occlusion and to the detector failing for a period of time. If several frames are lost, then the identity will probably be lost too. The detector in the object tracking module is also sensitive to background objects resembling humans. The emotion detection system, on the other hand, is very simple and does not take into account temporal dependencies.

There are, therefore, several lines of work that mark the next steps in the development of a robust automatic audience experience estimator. Firstly, our framework's dimensions must be studied more in-depth in order to facilitate measurement. The combined metric S must be validated, since a more carefully chosen combination of dimensions may better estimate audience experience. A larger and more varied dataset must also be created through a large-scale experiment. Tracking system improvements should be focused on reducing the sensitivity of the detector and the number of identity switches while minimizing the impact on latency. Finally, improvements of the emotion detection system should focus on incorporating information from previous predictions.

Appendices

Appendix A

Code

```
/*
 *
 *  CONFIGURATION
 *
 */

// Config variables: change them to point to your own servers
const SIGNALING_SERVER_URL = '';
const RECORDING_SERVER_URL = '';
const IDEN_SERVER_URL = '';
const STUN_SERVER_URL = '';
const TURN_SERVER_URL = '';
const TURN_SERVER_USERNAME = '';
const TURN_SERVER_CREDENTIAL = '';

// WebRTC config
// If you are testing on localhost, you can just use PC_CONFIG = {}
const PC_CONFIG = {
  iceServers: [
    {
      urls: 'turn:' + TURN_SERVER_URL + '?transport=tcp',
      username: TURN_SERVER_USERNAME,
      credential: TURN_SERVER_CREDENTIAL
    },
    {
      urls: 'stun:' + STUN_SERVER_URL + '?transport=tcp',
    },
  ],
};
//console.log(PC_CONFIG);
//const PC_CONFIG = {};


/*
```

```
 *
 *  IDENTIFICATION
 *
 */

// Find identifiers in URL params
const queryString = window.location.search;
const urlParams = new URLSearchParams(queryString);

const roomId = urlParams.get('room_id');
const subId = urlParams.get('subject');
const hostId = urlParams.get('host');

function uuidv4() {
  return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g,
   ↪  function(c) {
    var r = Math.random() * 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);
    return v.toString(16);
  });
}


let UNIQUE_ID;


/*
 *
 *  NETWORKING
 *
 */

// Sends the unique id to the AVP client for stream-id association
async function sendIden() {
        data = {
                subjectId: UNIQUE_ID,
                streamId: localStreamId,
                ssid: roomId + "_sub"
        }
        const params = {
                body: JSON.stringify(data),
                method: "POST"
        };
        res = await fetch(IDEN_SERVER_URL + '/iden', params)
        return true
}

// Asks the AVP client to start recording
async function sendStartRecording() {
```

```
        const data = {
                type: "start",
                ssid: roomId + "_sub"
        };
        const params = {
                body: JSON.stringify(data),
                method: "POST"
        };
        res = await fetch(RECORDING_SERVER_URL + '/record', params)
        res = await res.json()
        console.log(res)
        return res.ok
}

// Asks the AVP client to stop recording
async function sendStopRecording() {
        data = {
                type: "stop",
                ssid: roomId + "_sub"
        }
        const params = {
                body: JSON.stringify(data),
                method: "POST"
        };
        res = await fetch(RECORDING_SERVER_URL + '/record', params)
        res = await res.json()
        console.log(res)
        return res.ok
}

// Asks the AVP client to register the recording timestamp of a
// video timestamp
async function sendRecordTs(ts) {
        data = {
                ts: ts,
                ssid: roomId + "_sub"
        }
        const params = {
                body: JSON.stringify(data),
                method: "POST"
        };
        res = await fetch(RECORDING_SERVER_URL + '/ts', params)
        res = await res.json()
        console.log(res)
        return res.ok
}

// SFU signals and client
```

```
const signalHost = new Signal.IonSFUJSONRPCSignal(
  SIGNALING_SERVER_URL
);

const clientHost = new IonSDK.Client(signalHost, PC_CONFIG);
signalHost.onopen = () => clientHost.join(roomId + "_host");


const signalSub = new Signal.IonSFUJSONRPCSignal(
  SIGNALING_SERVER_URL
);

const clientSub = new IonSDK.Client(signalSub, PC_CONFIG);
signalSub.onopen = () => clientSub.join(roomId + "_sub");


// Waits until client is connected
async function waitForClient() {

  const poll = resolve => {
    if(clientHost.transports) resolve();
    else setTimeout(_ => poll(resolve), 400);
  }

  return new Promise(poll);
}


// Control channel: allows sending video playback commands and other
// control communication between host and subs
let control;

async function createControlDataChannel() {
  await waitForClient();
  control = await clientHost.createDataChannel("control");
  console.log("Control data channel created");
}

// For subs, sets up the control logic
function setupControlResponse() {
  control.onmessage = (msg) => {
    [video, ev, ts] = msg.data.split(" ");
    console.log(video + " " + ev);
    console.log(msg.data);
    switch(ev) {
    case "-4":
    case "-3":
```

```
      if (!(video in videos)) {
        getVideo(video, parseInt(ev)+2, ts);
      }
      break;
    case "-2":
    case "-1":
      getVideo(video, ev, ts);
      break;
    case "0":
      videos[video].stopVideo();
      el = document.getElementById("player="+video).parentElement;
      el.parentElement.removeChild(el);
    case "1":
      videos[video].seekTo(ts);
      videos[video].playVideo();
      break;
    case "2":
      videos[video].pauseVideo();
      videos[video].seekTo(ts);
      break;
    }
  }
}


/*
 *
 *  STREAMS
 *
 */

defaultConstraints = {
  resolution: "qvga",
  audio: true
};

audioConstraints = true;

let localStreams = {};
let localStreamId;

let subscribers = {};


const getLocalStream = async (constraints = null) => {

  if (constraints === null)
    constraints = defaultConstraints;
```

```
  gum = await IonSDK.LocalStream.getUserMedia(constraints).catch(
    (error) => {
      alert("Could not access local stream: " + error);
    }
  );

  if(!gum)
    return null;

  localStreams[gum.id] = gum;
  localStreamId = gum.id;

  return gum.id;
}

const getDisplayStream = async (constraints = null) => {
  gum = await IonSDK.LocalStream.getDisplayMedia({video: true, audio:
  ↪  true}).catch(
    (error) => {
      alert("Could not access screen: " + error);
    }
  )

  console.log("Got display stream " + gum.id);

  if(!gum)
    return null;

  localStreams[gum.id] = gum;
  console.log("Got display stream " + gum.id);
  return gum.id;
};


function setupClientHost() {
  clientHost.ontrack = (track, stream) => {
    console.log("got track", track.id, "for stream", stream.id);
    stream.onremovetrack = () => {
      console.log("Track ended");
      removeRemoteStreamElement(stream.id);
    }
    strElem = getRemoteStreamElement(stream.id);
    if (strElem.srcObject === null) {
      strElem.srcObject = stream;
    } else {
      strElem.srcObject.addTrack(track);
    }
```

```
  };
}

function addLocalStream(id = null, host = false) {
  getLocalStreamElement(id, host).srcObject = localStreams[id];
  //getLocalStreamElement(id).muted = true;
  console.log("Local video track added");
}

function removeLocalStream(id) {
  localStreams[id].getTracks().forEach((t) => t.stop());
  delete localStreams[id]
  removeLocalStreamElement(id);
}

function setupClientSub() {
  clientSub.ontrack = (track, stream) => {
    console.log("got track", track.id, "for stream", stream.id);
    stream.onremovetrack = () => {
      console.log("Track ended");
      removeRemoteStreamElement(stream.id);
    }
    strElem = getRemoteStreamElement(stream.id);
    if (!(stream.id in subscribers)) {
      console.log("New subscriber");
      subscribers[stream.id] = null;
      for (id in videos) {
        if (videos[id].getIframe().parentElement
              .children[1].children[0].on) {
          code = videos[id].getPlayerState() == 1 ? " -4 " : " -3 "
          control.send(id + code + videos[id].getCurrentTime());
          console.log(id + code + videos[id].getCurrentTime());
        }
      }
    }
    if (strElem.srcObject === null) {
      strElem.srcObject = stream;
    } else {
      strElem.srcObject.addTrack(track);
    }
  };
}



/*
 *
 *  UI
```

```
 *
 */

const remoteStreamContainer = document.getElementById("remote-streams");
let localStreamContainer;

function setLocalStreamContainer(host = false) {
        if (host)
                localStreamContainer =
                ↪  document.getElementById("local-streams");
        else
                localStreamContainer = remoteStreamContainer;
}


function getRemoteStreamElement(id, host = false) {
  let elem = document.getElementById("remote-stream-"+id)
  if (elem === null) {
    let div = document.createElement("div");
        if (host)
      div.classList.add("video");
        else
          div.classList.add("stream-container");

    let newstr = document.createElement("video");
    newstr.autoplay = true;
    newstr.id = "remote-stream-"+id;
    newstr.playsinline = true;
        newstr.muted = false;
    div.appendChild(newstr);
    remoteStreamContainer.appendChild(div);
    elem = newstr;
  }
  return elem
}

function removeRemoteStreamElement(id) {
  remoteStreamContainer.removeChild(
          getRemoteStreamElement(id).parentElement);
}

function getLocalStreamElement(id, host = false) {
  let elem = document.getElementById("local-stream-"+id)
  if (elem === null) {
    let div = document.createElement("div");
        if (host)
      div.classList.add("local","stream-container");
        else
```

```
      div.classList.add("stream-container");

    let newstr = document.createElement("video");
    newstr.classList.add("local","video");
    newstr.autoplay = true;
    newstr.muted = true;
    newstr.id = "local-stream-"+id;
    newstr.playsinline = true;

        if (host) {
      let contDiv = createStreamControls(newstr, id);
      div.appendChild(newstr);
      div.appendChild(contDiv);
        } else {
      div.appendChild(newstr);
        }

    localStreamContainer.appendChild(div);
    elem = newstr;
  }
  return elem
}

function createStreamControls(strElem, id) {
  let cont = document.createElement("div");
  cont.classList.add("stream-controls");

  cont.innerHTML = `
      <div class="publish button">
        <div class="icon">
          <i class="fa fa-wifi" aria-hidden="true"></i>
        </div
      </div>`;

  cont.innerHTML += `
      <div class="camera button">
        <div class="icon">
          <i class="fa fa-video-camera" aria-hidden="true"></i>
        </div
      </div>`;

  cont.innerHTML += `
      <div class="mute button">
        <div class="icon">
          <i class="fa fa-microphone" aria-hidden="true"></i>
        </div
      </div>`;
```

```
  cont.innerHTML += `
      <div class="remove button">
        <div class="icon">
          <i class="fa fa-times" aria-hidden="true"></i>
        </div
      </div>`;



  let publish = cont.children[0];
  publish.on = false;
  publish.onclick = () => publishStream(strElem, publish);

  let disable = cont.children[1];
  disable.on = false;
  disable.onclick = () => disableStream(strElem, disable);

  let mute = cont.children[2];
  mute.on = false;
  mute.onclick = () => muteStream(strElem, mute);

  let remove = cont.children[3];
  remove.onclick = () => {
    if (strElem.srcObject != undefined) {
      strElem.srcObject.unpublish();
      removeLocalStream(id);
    } else {
      videos[id].stopVideo();
      delete videos[id]
      localStreamContainer.removeChild(
              document.getElementById("player="+id).parentElement);
    }
  }

  return cont;
}

function publishStream(strElem, publish) {
  if(strElem.srcObject == undefined) {
    publishVideo(strElem.id, publish);
    return;
  }
  if (publish.on) {
    strElem.srcObject.unpublish();
    publish.on = false;
    publish.style.backgroundColor = "black";
  } else {
    clientHost.publish(strElem.srcObject);
```

```
    publish.on = true;
    publish.style.backgroundColor = "lightblue";
  }
}

function publishVideo(id, publish) {
  id = id.split("=")[1];
  if (publish.on) {
    console.log("unpublish "+id);
    control.send(id + " 0");
    publish.on = false;
    publish.style.backgroundColor = "black";
  } else {
    console.log("publish "+id + " " + videos[id].getCurrentTime());
    if(videos[id].getPlayerState() == 1) {
      control.send(id + " -2 " + videos[id].getCurrentTime());
    } else {
      control.send(id + " -1");
    }
    publish.on = true;
    publish.style.backgroundColor = "lightblue";
  }
}

function disableStream(strElem, disable) {
  if (disable.on) {
    strElem.srcObject.getVideoTracks()[0].enabled = true;
    strElem.srcObject.unmute('video');
    disable.on = false;
    disable.style.backgroundColor = "lightblue";
  } else {
    strElem.srcObject.getVideoTracks()[0].enabled = false;
    strElem.srcObject.mute('video');
    disable.on = true;
    disable.style.backgroundColor = "black";
  }
}

function muteStream(strElem, mute) {
  if (mute.on) {
    strElem.srcObject.unmute('audio');
    mute.on = false;
    mute.style.backgroundColor = "lightblue";
  } else {
    strElem.srcObject.mute('audio');
    mute.on = true;
    mute.style.backgroundColor = "black";
  }
```

```
}

function removeLocalStreamElement(id) {
  localStreamContainer.removeChild(
          getLocalStreamElement(id).parentElement);
}


let videos = {};


function getVideoSub(id, playState, ts) {
  let div = document.createElement("div");
  div.classList.add("stream-container");

  let newstr = document.createElement("div");
  newstr.id = "player="+id;
  newstr.classList.add("local","video");

  let overlay = document.createElement("div");
  overlay.classList.add("player-overlay");

  let overlay2 = document.createElement("div");
  overlay.classList.add("player-overlay");
  overlay2.onclick = () => {videos[id].playVideo();
  ↪  videos[id].stopVideo();};

  let fs = document.createElement("div");
  fs.classList.add("fullscreen-button");

  let icon = document.createElement("i");
  icon.classList.add("fa", "fa-square-o");
  icon.ariaHidden = true;

  div.resizeObs = new ResizeObserver((entries) => {
    videos[id].setSize(entries[0].contentRect.width,
    ↪  entries[0].contentRect.height);
  });

  div.resizeObs.observe(div);

  fs.appendChild(icon);
  overlay.appendChild(fs);
  overlay.appendChild(overlay2);
  div.appendChild(overlay);

  fs.onclick = () => {
    if (!document.fullscreenElement) {
```

```
      div.requestFullscreen();
    } else {
      document.exitFullscreen();
    }
  }

  div.appendChild(newstr);
  remoteStreamContainer.appendChild(div);

  function onPlayerReady(event) {
        event.target.setPlaybackQuality('hd720');
    event.target.seekTo(ts);
    event.target.playVideo();
    if (playState != -2)
          event.target.stopVideo();
  }

  player = new YT.Player('player='+id, {
    style: "height: auto; width: 100%;",
    videoId: id,
    playerVars: { 'controls': 0 },
    disablekb: 1,
    events: {
      'onReady': onPlayerReady,
      //'onStateChange': onPlayerStateChange
    }
  });

  videos[id] = player;
}


async function getVideoHost() {
  videoId = document.getElementById("video-id").value;

  let div = document.createElement("div");
  div.classList.add("local","stream-container");

  let newstr = document.createElement("div");
  newstr.id = "player="+videoId;
  newstr.classList.add("local","video");

  let contDiv = createStreamControls(newstr, videoId);
  div.appendChild(newstr);
  div.appendChild(contDiv);

  localStreamContainer.appendChild(div);
```

```
  function onPlayerReady(event) {
    //event.target.playVideo();
  }

  function onPlayerStateChange(event) {
    if (contDiv.children[0].on)
      control.send(videoId + " " + event.data + " " +
      ↪  videos[videoId].getCurrentTime());
          if (recordButton.on)
            sendRecordTs(videoId + " " + event.data + " " +
            ↪  videos[videoId].getCurrentTime());
  }

  player = new YT.Player('player='+videoId, {
    style: "height: auto; width: 80%;",
    videoId: videoId,
    //playerVars: { 'autoplay': 1, 'controls': 0 },
    events: {
      'onReady': onPlayerReady,
      'onStateChange': onPlayerStateChange
    }
  });

  videos[videoId] = player;
}



/*
 *
 * CHAT
 *
 */

let chat;

async function startChat(host = false) {
  await waitForClient();
  chat = clientHost.createDataChannel("chat");
  chat.onmessage = (msg) => {
    chatElement.value += msg.data;
  }
  messageElement.onkeyup = (ev) => {
    if(ev.keyCode == 13) {
      chat.send((host ? "Host:\n" : UNIQUE_ID + ':\n') +
      ↪  messageElement.value);
      chatElement.value += "Tú:\n";
      chatElement.value += messageElement.value;
```

```
      messageElement.value = "";
    }
  }
}
```

```
var tag = document.createElement('script');

tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName('script')[0];
firstScriptTag.parentNode.insertBefore(tag, firstScriptTag);

// 3. This function creates an <iframe> (and YouTube player)
//    after the API code downloads.
let youtubeReady = false;
var player;
function onYouTubeIframeAPIReady() {
  youtubeReady = true;
}
```

Appendix B

Paper and submission confirmation

In this appendix we include the full paper sent to the 29th International Conference on Computers and Education (ICCE 2021). Additionally, we also include the confirmation email for the submission (see Fig. B.1).

Dear authors,

We received your submission to ICCE2021 (29th International
Conference on Computers in Education):

Authors : Pablo Villalobos, Eduardo Rivero, Meriem El Yamri, Alejandro Romero and Borja Manero
Title : Tackling the design and evaluation of a theater-based intelligent system to monitor audience experience in virtual public speaking settings
Number : 94
Track : C1: Artificial Intelligence in Education/Intelligent Tutoring System and Adaptive Learning

Figure B.1: ICCE 2021 submission confirmation for our paper.

Rodrigo, M. M. T. et al. (Eds.) (2021). Proceedings of the 29th International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Tackling the design and evaluation of a theater-based intelligent system to monitor audience experience in virtual public speaking settings

Pablo VILLALOBOS SANCHEZ*, Eduardo RIVERO RODRIGUEZ, Meriem EL YAMRI, Alejandro ROMERO HERNANDEZ & Borja MANERO IGLESIAS
Complutense University of Madrid, Spain
*pavill01@ucm.es

Abstract: COVID-19 has brought about a sharp increase in the use of videoconferencing tools. Particularly in education, this creates additional difficulties for teachers to monitor their students' experience, which is essential to get the proper classroom management. Researchers have designed tools and frameworks to aid teachers in this endeavor for on-site settings, but they focus on measuring only the students' engagement and are not suitable for virtual environments.

In this paper, we present our system's architecture, which addresses those issues in two ways. First, we have adapted a theater-based framework (not only based on engagement) for measuring audience experience. Secondly, we have designed a proof-of-concept computer vision system and a companion video conferencing tool that automatically measures audience experience in the classroom and presents near real-time feedback. We also describe the experiment we carried out to obtain a dataset to test our system and present the results. Although the predictive accuracy of our proof-of-concept system is limited, it opens several directions for future research.

Keywords: computer vision, machine learning, sentiment analysis

1. Introduction

The relationship between students' classroom behavior and their academic outcomes is not new (McKinney, Mason, Perkerson, & Clifford, 1975). A recent meta-analysis (Lei, Cui, & Zhou, 2018) analyzed 69 independent studies on the topic and established a positive correlation between students' engagement in the classroom and their academic achievement and learning. However, determining whether a student is engaged or not is not a trivial task.

Teachers need to develop the necessary skills to monitor students' engagement levels if they want to improve their teaching outcomes (Pianta & Hamre, 2009). Until now, teachers were trained to do so in a very specific context: traditional in-person classrooms.

However, due to COVID-19, many learning contexts have changed. Conferences, classes, and other events where public speaking is essential have moved to virtual environments. In a video conference, only head and shoulders are visible, which makes it challenging to evaluate an audience's experience. Most teachers are unfamiliar with the “language” of the camera. Therefore, when students are on screen their cues (gaze, body language, voice…) may be misinterpreted. It is much easier for a teacher to identify when an in-person audience is getting bored than when a virtual one is.

Audience feedback is essential so that teachers can keep students interested and facilitate learning. Thus, an accurate assessment of the audience experience in online environments is vital to achieve good classroom management.

However, quantifying the audience’s experience is not an easy task. Researchers have used several tools such as physiological signals (eye gaze, electrocardiogram, or galvanic skin response among others), self-reports, or observer evaluations to tackle this challenge (Goldberg et al., 2019). Unfortunately, all these methods pose certain issues that must be resolved in order to create usable tools. Measuring physiological signals requires technical equipment and expertise, which is usually not

available in a classroom setting. Self-reports have minimal temporal resolution, meaning that they provide information related to very few time points in the whole class timeline. They are also necessarily subjective and only provide a rough summary of the whole experience. Likewise, observer evaluations require trained observers and are time-consuming.

Recent advances in artificial intelligence and computer vision make computers an ideal tool to model audience experience. Some computer vision systems have already been used to monitor students' engagement (Whitehill, Serpell, Lin, Foster, & Movellan, 2014; Goldberg et al., 2019), but they rely mostly on handcrafted features and are not suitable for virtual settings. Furthermore, these systems often do not fully make use of all the advances in machine learning and are somewhat expensive computationally, requiring specific and powerful hardware to run real-time or near real-time. Moreover, every technical solution we found in the literature measures only one specific dimension of audience experience: engagement.

Engagement is a construct that includes observable behaviors, internal cognition, and emotions such as excitement, boredom, curiosity, or anger. Outside of the field of education, it is considered one component of audience experience in human-computer interaction, games, artistic performances, and public speaking presentations (Curtis, Jones, & Campbell, 2015; Webster & Ho, 1997; Sun, Li, Tian, Fan, & Wang, 2019; Shernoff, Csikszentmihalyi, Schneider & Shernoff, 2003). Since audience experience goes beyond engagement, it seems necessary to incorporate other factors to achieve a more accurate model.

Theater is a discipline that has something to say about audiences. Drawing inspiration from the theory of psychological flow, a report commissioned by different theater associations (Independent Theatre Council, The Society of London Theatre, Theatrical Management Association, & The New Economics Foundation. 2005) identified five major components of audience experience in the theater: engagement, learning, energy, shared atmosphere, and emotional connection. These are defined as follows:

1. **Engagement:** The extent to which the performance captures and maintains the audience's attention.
2. **Learning:** The extent to which the performance opens new possibilities or challenges previous beliefs.
3. **Energy:** Physiological reactions to the performance, interpreted as indicators of the underlying emotional response.
4. **Shared atmosphere:** The sense of collective experience afforded by the performance.
5. **Emotional connection:** The feelings of empathy or identification with part of the performance.

Taking that into account, in this paper, we propose a system capable of providing real-time feedback by monitoring and analyzing the audience of online meetings. Our system has 3 differentiating factors:

- **Theater-inspired:** It monitors audience experience dynamically across four of the components identified above, extending the unidimensional engagement-based frameworks that are currently popular.
- **Video-based:** Our proposal is specifically designed to operate with videoconferencing tools and online learning environments.
- **Near real-time:** The presenters receive the analysis of the audience in near real-time. This allows them to react to changing conditions and improve their performance with minimum delay.

In addition, we show a proof of concept for our system, which we were able to test in a small scale experiment, with mixed results. While the tracking system seems to generally work well, the emotion recognition system produces results with a high margin of error. The software implementation is available for anyone who wishes to experiment and build upon our prototype.

This paper is structured as follows. First, we introduce the conditions that make virtual settings different and explain our framework for quantifying audience experience. Then, we give a general overview of the architecture and a more detailed technical explanation. After that, we describe the experiment we performed and the results that were obtained. Finally, we highlight the conclusions and limitations of our system and propose some future lines of research.

2. Architecture

2.1 The context of virtual settings

Meetings in virtual settings (provided by software like Google Meet or Zoom) differ in many aspects from in-person meetings:

1. Only the head and shoulders of each participant are visible, thus the teacher cannot observe all the non-verbal communication.
2. Each participant may have different video stream quality (image resolution, lighting, proximity to the camera, etc.) depending on the hardware they use to join the meeting.
3. Each video stream may vary or disconnect over time due to connection issues or deliberate actions by the participants.
4. Some of the cues that indicate disengaged students in on-site classrooms do not exist in the virtual environment (e.g., a participant walking around with the intention of interrupting).

Because of the differences between the context of theater performances and online presentations, we decided to adapt the framework proposed by the report on audience experience in the theater (Independent Theatre Council, The Society of London Theatre, Theatrical Management Association, & The New Economics Foundation. 2005) by making the following changes:

- Since the shared atmosphere dimension seems less applicable to learning environments, especially online, we focus on the other four components.
- We use affective response (Bradley & Lang, 1994) to measure energy.
- We incorporate the research by Curtis, Jones, and Campbell on audience engagement (Curtis, Jones, & Campbell, 2015) and comprehension (Curtis, Jones, & Campbell, 2016), which is related to the learning dimension.

Thus, our adapted framework is composed of the following dimensions: affective response (Af), engagement (En), emotional connection (Ec), and learning (Le).

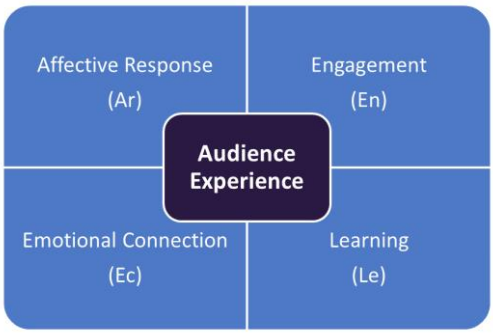

Figure 1. The four components of our framework to measure audience experience.

There are also some technical limitations that come with the use of the most popular software used for video conferencing, besides the intrinsic limitations of the virtual setting mentioned above. One of the biggest challenges we faced when setting up a data pipeline for this framework is that the most widespread applications did not have a simple way of treating the video feeds in a meeting room as separate input data streams. To overcome said challenge, we developed an ad hoc video conferencing application based on the WebRTC API (WebRTC, 2011) (see Fig. 2). The application allows the host to record individual participant video feeds, which are treated as separate streams.

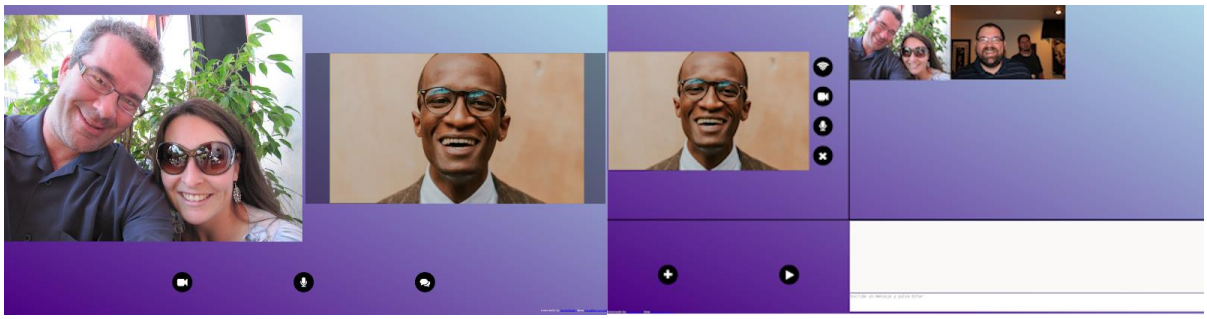

Figure 2. Respectively, participant view (left) and host view (right) side by side.

2.2 Architectural overview

The general architecture (see Fig. 3) is composed of a central server, a host for the video conference (the presenter's device), and multiple clients (the participants' devices). The flow of information goes from the client to the server, and then from the server to the host. The server runs the machine learning pipeline, which is the heaviest step in terms of computation time.

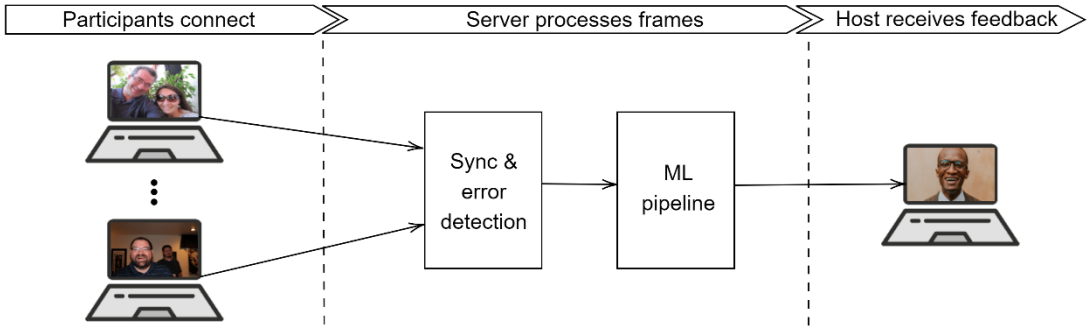

Figure 3. Overview of the system architecture (OT – Object Tracking, ER – Emotion Recognition, Avg - Average).

In our setting, each client has an associated video stream, displaying one or more participants. These streams are sent over the Internet to the server. Thus, the data received is an unsynchronized set of unreliable streams, which is then sampled by the server at regular intervals, producing synchronized batches of frames (one for each stream), and substituting missing values if necessary. These batches are the inputs to the machine learning pipeline. For each participant recognized by the first module and each dimension of audience experience, the machine learning (ML) pipeline outputs a metric value. These metrics are then pooled and can be displayed to the host as feedback individually (Af, En, Le, Ec), or aggregated as a global compound score (S). If we detect N participants, these metrics can be expressed as:

$$Af = \frac{1}{N}\sum_{i=1}^{N} Af_i \qquad En = \frac{1}{N}\sum_{i=1}^{N} En_i \qquad Le = \frac{1}{N}\sum_{i=1}^{N} Le_i \qquad Ec = \frac{1}{N}\sum_{i=1}^{N} Ec_i \tag{1}$$

$$S = \frac{Af + En + Co + Ec}{4} \tag{2}$$

First, the ML pipeline performs object detection and tracking in each of the frames, associating a persistent identity with each participant in the batch and keeping track in future frames (see Fig. 4). This is necessary to incorporate information from previous batches. At this point we have a set of identities and associated streams, all of which are synchronized. Since the next step is computationally intensive, these streams are downsampled.

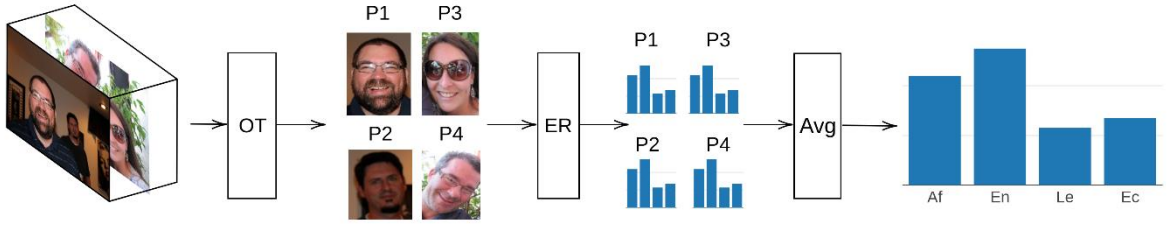

Figure 4. Overview of the system architecture (OT – Object Tracking, ER – Emotion Recognition, Avg - Average).

Then, the streams associated with each identity enter the emotion recognition module, which takes a batch of frames and outputs the value of the 4 metrics for each frame in the batch. These metrics represent the scoring determined by the emotion recognition module for each of the 4 dimensions of audience experience: affective response, engagement, emotional connection, and learning. Finally, the last step is a pooling step that returns the value of these metrics for the whole batch.

2.2.1 Architectural implementation

The object tracking module works in 2 steps. First, we apply standard YOLOv4 (Bochkovskiy, Wang, & Liao, 2020) to detect participants in the frame. We have created a custom algorithm that is then used to track specific participants. This algorithm works by keeping track of the positions of the object centers in successive frames as well as the bounding box sizes. Through this history, we can predict the next object's bounding box center and size. Let $\boldsymbol{c}_n$ and $\boldsymbol{bb}_n$ be, respectively, the last observed bounding box center and size (width and height) of a particular object. Then, the variation in center position and bounding box size Δc and Δb are defined, respectively, as

$$\Delta \boldsymbol{c} = \boldsymbol{c}_n - \boldsymbol{c}_{n-1} \qquad \Delta \boldsymbol{bb} = \boldsymbol{bb}_n - \boldsymbol{bb}_{n-1} \tag{3}$$

The values obtained by applying (3) can then be used to predict the expected center position $\boldsymbol{c}_{n+1}$ and bounding box size $\boldsymbol{bb}_{n+1}$

$$\boldsymbol{c}_{n+1} = \boldsymbol{c}_n + \Delta \boldsymbol{c} \qquad \boldsymbol{bb}_{n+1} = \boldsymbol{bb}_n + \Delta \boldsymbol{bb} \tag{4}$$

For a single observation, the variations obtained in equation (3) are zero. This process is essentially a Kalman filter (Kalman, 1960). With the predicted values and with each bounding box $\boldsymbol{bb}$ being represented as a pair of height and width (h, w) in pixels, we match existing identities to the observed participants with the most similar bounding box characteristics, minimizing the metric

$$\mathrm{m}(\boldsymbol{c}, \boldsymbol{bb}) = \left|\left|\boldsymbol{c} - \boldsymbol{c}_{n+1}\right|\right| + f(h, h_{n+1}) + f(w, w_{n+1}) \tag{5}$$

where $f(x, y) = \log(1 + |x - y|)$. Most of the time, the first term in equation (5) will be considerably larger than the other two, as the width and height are only relevant when there are various participants with very similar centers. To obtain the matching efficiently, many tracking systems (Bewley, Ge, Ott, Ramos, & Upcroft, 2016; Arun Kumar, Laxmanan, Ram Kumar, Srinidh, & Ramanathan, 2021; Luo, Xing, Milan, Zhang, Liu, & Kim, 2021) propose the usage of the Hungarian algorithm, which runs in time complexity $\mathcal{O}(n^3)$ (Kuhn, 1955; Munkres, 1957). We, on the other hand, have found an alternative approach that remains largely unexplored in the literature and is only implemented by a few select systems (Godbehere & Goldberg, 2014; Oh et al, 2020). If we formulate the matching problem above as a stable matching problem (SMP) by converting the distance matrix determined by metric $m(\cdot,\cdot)$ into preference lists for the previous detections and the detected object, we can use the Gale-Shapley (GS) algorithm (Gale & Shapley, 1962) to perform the matching. This algorithm has a lower time complexity than the Hungarian algorithm, being capable of running in $\mathcal{O}(n^2)$.

The last issue to solve in the tracking module is how to deal with situations where the number of expected detections (which is the same as the number of identities in the last frame) and the number

of detections does not match. If there are more detections than expected, new identities are created for those that remain unmatched after running the GS algorithm. Additionally, we consider a tolerance value τ_k for each existing identity which indicates how many frames the system will tolerate not finding a suitable match for said agent. If there are more identities than detections, the tolerance of unmatched identities is decreased and identities with 0 tolerance are deleted. If there is a sufficiently high frame rate with respect to the speed at which the participants move, the detection accuracy is high.

The emotion recognition module also works in 2 steps. First, we use a convolutional neural network (CNN) (LeCun, Haffner, Bottou, & Bengio, 1999) to obtain a low-dimensional embedding of the input frames. We favor architectures pre-trained on ImageNet (Deng et al, 2010), as transfer learning is standard to improve performance in computer vision models (Weiss, Khoshgoftaar, & Wang, 2016; Oquab, Bottou, Laptev, & Sivic, 2014; Hussain, Bird, & Faria, 2019). In this paper, we propose the use of MobileNetV3 (Howard et al., 2019), but there are other architectures (Zoph, Vasudevan, Shlens, & Le, 2017; Simonyan & Zisserman, 2015; He, Zhang, Ren, & Sun, 2016) that can fulfill this purpose equally well. This embedding is then passed onto 4 fully connected (FC) layers with linear activation, each trained to predict a particular dimension-specific score (Af, En, Le, Ec). Alternatively, it should also be possible to use support vector machines (SVMs) (Cortes & Vapnik, 1995).

Finally, the pooling module extracts the global scores for each dimension by averaging across the number of agents as indicated by (1). It is also possible to then calculate the summary metric as shown in (2). This module must adapt to changes in the number of participants in successive predictions.

3. Experiment, dataset, and results

3.1.1 Experimental design and participants

While there are some audiovisual datasets on audience experience and affective response (Curtis et al., 2015; Soleymani et al., 2012), they are not available to the general public and do not contain data in online settings. For this reason, we ran a data collection experiment and created a dataset of online audience response.

A total of 8 last year Spanish college students (6 males and 2 females) volunteered to take part in the study. All of them were around 20 years old. The experiment was conducted as follows: the volunteers connected to our custom video conferencing tool with their personal computers and webcams. We streamed three short (10min) video performances while we recorded the participants with their webcams. After watching each performance, the participants filled out a short questionnaire.

The video performances were selected from a pool of YouTube videos of varied content, including TED talks, monologues, and political speeches. The questionnaire was divided into three sections, measuring the first, second, and third parts of each video, respectively. Each section included 16 likert-scale questions to separately measure the four different components of audience experience.

Even though we had few participants and few video performances, a preliminary analysis of the questionnaire responses showed that our methodology was able to capture some variation in the four dimensions, both across videos and throughout each video.

3.1.2 Dataset

After the data collection phase, the recordings were cleaned up and score labels were generated for each frame in the following manner: First, each recording is synchronized with the performance using timestamps generated by our custom tool. Then the recordings are cut to the length of the performance, and are processed by the tracking pipeline, which produces a set of fixed-size video files tracking each of the subjects. Finally, frame-level labels are generated by linear interpolation from the three samples given by the questionnaire results for each video.

The resulting dataset consists of 224,206 individually annotated frames, forming 24 videos (8 participants watching 3 performances). After an initial exploration and some training runs, we determined that the data had a high degree of redundancy, so we applied a 20x downsampling, dropping 19 out of every 20 frames.

3.1.3 Experimental results

For the final training, we used an Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0005, L1 loss function, and batch size of 256. The model was trained for 2 epochs using 72% of the data, while the remaining data, recordings belonging to 2 participants, were used for validation (14%) and testing (14%). The training was performed using a single Nvidia RTX 3070 GPU, running for about 20 minutes, and the metrics of the learning system on the test dataset are shown in Table 1. We have tracked Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and R^2 score in order to assess the performance of the model on the test dataset.

Table 1. *Testing Metrics*

	MSE	MAE	MAPE	R^2
Aff. Resp.	0,0861	0,2557	1,2240	-0,1142
Engagement	0,1556	0,3093	14,0547	-0,3560
Em. Con.	0,0757	0,2324	0,7135	-0,4725
Learning	0,1775	0,2906	6.3201	-0,4725
Combined	0,1099	0,2642	1,5040	-0,2330

As we can see, the MSE is relatively low, while the MAPE is very large. However, since the error for each frame is between 0 and 1, this is to be expected and it is mostly an artifact of working with small numbers. The R^2 is negative, which implies that our model performs worse than a constant prediction that matches the mean of the test dataset. However, we must interpret this result in the light of the data generation process: since for each video the scores were interpolated from three points, the variance in the dataset is very low, which might explain such a low R2 score.

All in all, it seems the MAE is the metric that best reflects the actual performance of the system. What this metric is telling us is that our system is making average errors of around 0.3. This is not very precise but may allow the speaker to rule out extreme situations (very low or very high engagement, for example).

4. Conclusions and Future Work

In this paper we proposed and evaluated a general framework that can be used to design intelligent systems to automatically evaluate audience experience in virtual settings. The framework is based on how the theater world evaluates their audiences. It goes beyond the one-dimensional (engagement-based) current trend and specifies four dimensions: affective response (Af), engagement (En), emotional connection (Ec), and learning (Le). Besides, we specified a particular implementation using YOLOv4, a custom tracking algorithm, and a combination of fine-tuned MobileNetV3 image embeddings and FC layers to predict audience experience scores. We also described the experiment we carried out to obtain a dataset and test our system and presented the final results.

On one hand, the proposed 4-dimensional framework captures aspects of the audience experience that were not considered in the one-dimensional measurements. An audience may be highly engaged but fall short in terms of learning. In the same manner, an audience might not be fully engaged but still have a high degree of affective response. The ability to capture these subtleties makes the proposed framework a better choice than engagement-based ones to see the full picture when it comes to measuring audience experience.

At the same time, while existing architectures were designed to be used in-person, the proposed architecture is designed to fit the characteristics of virtual settings. Even if we were to adapt previous systems to work coupled with video conference software, we would still need to adapt the cues they use to determine the degree of engagement so that they would be fully functional in these environments. To mitigate the limitations of popular video conferencing software, we created an ad-hoc conference tool.

We are aware that the final results are not conclusive, but the proposed architecture and ML pipeline are just meant as a proof of concept to test the viability of using an intelligent system to monitor audience experience. While the generic building blocks that constitute the object tracking (YOLOv4) and emotion recognition (MobileNetV3 and FC layers) systems are reliable and well-proven in their respective tasks, only the tracking module performed as expected. We believe that the main obstacle to the emotion recognition module is the insufficient amount of data and its redundancy. Therefore, the starting point for future research should be a larger scale experiment to expand the dataset.

On the other hand, there are some limitations to the proposed framework and system that we also plan to address in future research:

- We selected 4 metrics to evaluate audience experience based on the literature and previous research, but there might be other dimensions that result in better accuracy.
- Misidentifications and missing frames in the data collection process pose a problem for the proper functioning of the ML model.
- We do not have information about how well the framework will perform with a large number of participants.
- We do not fully comprehend the relationship between input image features and neuron activations in the FC layers.

Finally, another important line of research would be the design of an emotion recognition architecture that considers temporal dependencies. It is to be expected that, if any of the dimensions of the audience experience is positive (or negative) at a particular moment in time, it will be similar in both the preceding and successive moments. Both the proposed framework and the implementation of object tracking have the tools necessary to tackle this problem (since they keep track of specific identities over time). However, the emotion recognition module does not consider past predictions. Likely, modifying the architecture to incorporate such predictions would result in a more robust and accurate model.

Acknowledgements

We would like to thank all the volunteers who helped us build our dataset and, therefore, enabled us to work on the predictive system. This project has been founded by the Ministry of Science, Innovation and Universities of Spain (Didascalias, RTI2018-096401-A-I00).

References

Whitehill, J., Serpell, Z., Lin, Y. C., Foster, A., & Movellan, J. R. (2014). The faces of engagement: Automatic recognition of student engagement from facial expressions. IEEE Transactions on Affective Computing, 5(1). https://doi.org/10.1109/TAFFC.2014.2316163

Sun, W., Li, Y., Tian, F., Fan, X., & Wang, H. (2019). How Presenters Perceive and React to Audience Flow Prediction In-situ: An explorative study of live online lectures. Proceedings of the ACM on Human-Computer Interaction, 3(CSCW). https://doi.org/10.1145/3359264

Goldberg, P., Sümer, Ö., Stürmer, K., Wagner, W., Göllner, R., Gerjets, P., … Trautwein, U. (2019). Attentive or Not? Toward a Machine Learning Approach to Assessing Students' Visible Engagement in Classroom Instruction. Educational Psychology Review. https://doi.org/10.1007/s10648-019-09514-z

Curtis, K., Jones, G. J. F., & Campbell, N. (2015). Effects of good speaking techniques on audience engagement. ICMI 2015 - Proceedings of the 2015 ACM International Conference on Multimodal Interaction. https://doi.org/10.1145/2818346.2820766

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. ArXiv. Retrieved from http://arxiv.org/abs/2004.10934

Independent Theatre Council, The Society of London Theatre, Theatrical Management Association, & The New Economics Foundation. (2005). Capturing the audience experience: A handbook for the theatre. https://itc-arts-s3.studiocoucou.com/uploads/helpsheet_attachment/file/23/Theatre_handbook.pdf

WebRTC. (2011). [Software]. https://webrtc.org

Shernoff, D. J., Csikszentmihalyi, M., Schneider, B., & Shernoff, E. S. (2003, June). Student engagement in high school classrooms from the perspective of flow theory. School Psychology Quarterly, Vol. 18, pp. 158–176. https://doi.org/10.1521/scpq.18.2.158.21860

Curtis, K., Jones, G. J. F., & Campbell, N. (2016). Speaker impact on audience comprehension for academic presentations. ICMI 2016 - Proceedings of the 18th ACM International Conference on Multimodal Interaction. https://doi.org/10.1145/2993148.2993194

Webster, J., & Ho, H. (1997). Audience engagement in multimedia presentations. ACM SIGMIS Database: The DATABASE for Advances in Information Systems, 28(2), 63–77. https://doi.org/10.1145/264701.264706

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. Retrieved from http://www.robots.ox.ac.uk/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem, 770–778. https://doi.org/10.1109/CVPR.2016.90

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). Learning Transferable Architectures for Scalable Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 8697–8710. Retrieved from http://arxiv.org/abs/1707.07012

Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). A survey of transfer learning. Journal of Big Data, 3(1), 9. https://doi.org/10.1186/s40537-016-0043-6

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1717–1724. https://doi.org/10.1109/CVPR.2014.222

Hussain, M., Bird, J. J., & Faria, D. R. (2019). A study on CNN transfer learning for image classification. Advances in Intelligent Systems and Computing, 840, 191–202. https://doi.org/10.1007/978-3-319-97982-3_16

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2010, March 1). ImageNet: A large-scale hierarchical image database. 248–255. https://doi.org/10.1109/cvpr.2009.5206848

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1681, 319–345. https://doi.org/10.1007/3-540-46805-6_19

Cortes, Corinna (AT&TBellLabs., Hohndel, NJ07733, U., & Vladimir, Vapnik (AT&TBellLabs., Hohndel, NJ07733, U. (1995). Support-Vector Networks. Machine Learning, 297(20), 273–297.

Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. Journal of Behavior Therapy and Experimental Psychiatry, 25(1), 49–59. https://doi.org/10.1016/0005-7916(94)90063-9

McKinney, J. D., Mason, J., Perkerson, K., & Clifford, M. (1975). Relationship between classroom behavior and academic achievement. Journal of Educational Psychology, 67(2), 198–203. https://doi.org/10.1037/h0077012

Lei, H., Cui, Y., & Zhou, W. (2018). Relationships between student engagement and academic achievement: A meta-analysis. Social Behavior and Personality, 46(3), 517–528. https://doi.org/10.2224/sbp.7054

Pianta, R. C., & Hamre, B. K. (2009). Conceptualization, Measurement, and Improvement of Classroom Processes: Standardized Observation Can Leverage Capacity. Educational Researcher, 38(2), 109–119. https://doi.org/10.3102/0013189X09332374

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Journal of Fluids Engineering, Transactions of the ASME, 82(1), 35–45. https://doi.org/10.1115/1.3662552

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1–2), 83–97. https://doi.org/10.1002/nav.3800020109

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. Proceedings - International Conference on Image Processing, ICIP, 2016-August, 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003

Arun Kumar, N. P., Laxmanan, R., Ram Kumar, S., Srinidh, V., & Ramanathan, R. (2021). Performance Study of Multi-target Tracking Using Kalman Filter and Hungarian Algorithm. Communications in Computer and Information Science, 1364, 213–227. https://doi.org/10.1007/978-981-16-0422-5_15

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics, 5(1), 32–38. https://doi.org/10.1137/0105003

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2021). Multiple object tracking: A literature review. Artificial Intelligence, 293. https://doi.org/10.1016/j.artint.2020.103448

Godbehere, A. B., & Goldberg, K. (2014). Algorithms for visual tracking of visitors under variable-lighting conditions for a responsive audio art installation. Controls and Art: Inquiries at the Intersection of the Subjective and the Objective, 181–204. https://doi.org/10.1007/978-3-319-03904-6_8

Oh, A. R., Lee, J., Lee, J. S., Moon, S. W., Nam, D. W., & Yoo, W. (2020). Multi-object tracking system using dissimilar apparatus in video sequence. International Conference on ICT Convergence, 2020-October, 1528–1530. https://doi.org/10.1109/ICTC49870.2020.9289264

Gale, D., & Shapley, L. S. (1962). College Admissions and the Stability of Marriage. The American Mathematical Monthly, 69(1), 9. https://doi.org/10.2307/2312726

Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., … Adam, H. (2019). Searching for MobileNetV3. ArXiv.

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

Glossary

Af Affective rersponse. 26

BoF Bag of Freebies. 36

BoS Bag of Specials. 36

CIoU Complete-IoU. 36

CmBN Cross mini-batchnormalization. 36

CNN Convolutional Neural Network. 7

CSP Cross-Stage Partial connections. 36

DIoU-NMS Distance-IoU loss with NMS. 36

Ec Emotional Connection. 26

En Engagement. 26

ERT Ensemble Regression Tree. 23

ERVF Emotion Recognition from Video Feeds. 34

FAU Facial Action Unit. 22

FC Fully Connected. 9

FPN Feature Pyramid Network. 37

FPS Frames Per Second. 15

FVAE Factorized VAE. 23

GS Gale-Shapley. 41

IoU Intersection over Union. 10

Le Learning. 26

LSTM Long Short-Term Memory. 15

MAE Mean Absolute Error. 60

MAPE Mean Absolute Percentage Error. 60

MiWRC Multi-input Weighted Residual Connections. 36

ML Machine Learning. 1

MLP MultiLayer Perceptron. 18

MOT Multiple Object Tracking. 7

MSE Mean Squared Error. 15

NAS Network Architecture Search. 51

NMS Non-Max Suppression. 13

PAN Path Aggregation Network. 36

PSO Particle Swarm Optimization. 14

R-CNN Regions with CNN features. 8

ReLU Rectified Linear Unit. 9

RNN Recurrent Neural Network. 15

RoI Region of Interest. 10

ROLO Recurrent YOLO. 15

RPN Region Proposal Network. 11

SAE Sum of Absolute Errors. 50

SAM Spatial Attention Module. 39

SAT Self-Adversarial Training. 36

SORT Simple Online and Realtime Tracking. 15

SPP Spatial Pyramid Pooling. 39

SVM Support Vector Machine. 8, 17

VAE Variational AutoEncoder. 23

YOLO You Only Look Once. 12

Bibliography

[1] Kelson R.T. Aires, Andre M. Santana, and Adelardo A.D. Medeiros. "Optical flow using color information: Preliminary results". In: *Proceedings of the ACM Symposium on Applied Computing* (2008), pp. 1607–1611. DOI: 10.1145/1363686.1364064.

[2] Ahmad Ali et al. "Visual object tracking—classical and contemporary approaches". In: *Frontiers of Computer Science* 10.1 (2016), pp. 167–188. ISSN: 20952236. DOI: 10.1007/s11704-015-4246-3.

[3] Amidi, Afshine and Amidi, Shervine. *A detailed example of how to generate your data in parallel with PyTorch*. URL: https://stanford.edu/~shervine/blog/pytorch-how-to-generate-data-parallel#.

[4] Ligia Batrinca et al. *Cicero - Towards a multimodal virtual audience platform for public speaking training*. Tech. rep. 2013, pp. 116–128. DOI: 10.1007/978-3-642-40415-3_10. URL: http://www.toastmasters.org/tips.asp.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[6] S. S. Beauchemin and J. L. Barron. "The Computation of Optical Flow". In: *ACM Computing Surveys (CSUR)* 27.3 (1995), pp. 433–466. ISSN: 15577341. DOI: 10.1145/212094.212141.

[7] Alex Bewley et al. "Simple online and realtime tracking". In: *Proceedings - International Conference on Image Processing, ICIP* 2016-Augus (2016), pp. 3464–3468. ISSN: 15224880. DOI: 10.1109/ICIP.2016.7533003. arXiv: 1602.00763.

[8] Erik Blasch et al. "Overview of contextual tracking approaches in information fusion". In: *Geospatial InfoFusion III* 8747 (2013), 87470B. ISSN: 0277786X. DOI: 10.1117/12.2016312.

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[10] Margaret M. Bradley and Peter J. Lang. "Measuring emotion: The self-assessment manikin and the semantic differential". In: *Journal of Behavior Therapy and Experimental Psychiatry* 25.1 (Mar. 1994), pp. 49–59. ISSN: 00057916. DOI: 10.1016/0005-7916(94)90063-9.

[11] Roberto Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. 2009, pp. 1–338. ISBN: 9780470517062. DOI: 10.1002/9780470744055.

[12] Chen Cao et al. “FaceWarehouse: A 3D facial expression database for visual computing”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.3 (2014), pp. 413–425. ISSN: 10772626. DOI: 10.1109/TVCG.2013.249.

[13] *Capturing the audience experience: A handbook for the theatre*. Tech. rep. URL: https://itc-arts-s3.studiocoucou.com/uploads/helpsheet_attachment/file/23/Theatre_handbook.pdf.

[14] Yizong Cheng. “Mean Shift, Mode Seeking, and Clustering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 790–799. ISSN: 01628828. DOI: 10.1109/34.400568.

[15] Yu Chen Chiu et al. “Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems”. In: *2020 International Conference on System Science and Engineering, ICSSE 2020* (2020). DOI: 10.1109/ICSSE50014.2020.9219319.

[16] Mathieu Chollet et al. *An Interactive Virtual Audience Platform for Public Speaking Training (Demonstration)*. Tech. rep. URL: www.ifaamas.org.

[17] Keith Curtis, Nick Campbell, and Gareth J.F. Jones. “Development of an annotated multimodal dataset for the investigation of classification and summarisation of presentations using high-level paralinguistic features”. In: *LREC 2018 - 11th International Conference on Language Resources and Evaluation*. 2019. ISBN: 9791095546009.

[18] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: Institute of Electrical and Electronics Engineers (IEEE), Mar. 2010, pp. 248–255. DOI: 10.1109/cvpr.2009.5206848.

[19] Zhiwei Deng et al. “Factorized variational autoencoders for modeling audience reactions to movies”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 6014–6023. DOI: 10.1109/CVPR.2017.637.

[20] P. Ekman and W. V. Friesen. “Facial action coding system: A technique for the measurement of facial movement”. In: *Journal of Personality and Social Psychology* 17.2 (1971), pp. 124–129. ISSN: 00223514.

[21] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000022288.19776.77. URL: https://doi.org/10.1023/B:VISI.0000022288.19776.77.

[22] D. Forsyth. “Object Detection with Discriminatively Trained Part-Based Models”. In: *Computer* 47.02 (Feb. 2014), pp. 6–7. ISSN: 1558-0814. DOI: 10.1109/MC.2014.42.

[23] Michelle Fung et al. “ROC speak: Semi-Automated personalized feedback on nonverbal behavior from recorded videos”. In: *UbiComp 2015 - Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. Association for Computing Machinery, Inc, Sept. 2015, pp. 1167–1178. ISBN: 9781450335744. DOI: 10.1145/2750858.2804265.

[24] Chris Gamble. *Google Meet Grid View*. May 29, 2021. URL: https://chrome.google.com/webstore/detail/google-meet-grid-view/kklailfgofogmmdlhgmjgenehkjoioip.

[25] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[26] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[27] Patricia Goldberg et al. "Attentive or Not? Toward a Machine Learning Approach to Assessing Students' Visible Engagement in Classroom Instruction". In: *Educational Psychology Review* 33.1 (2021), pp. 27–49. ISSN: 1573336X. DOI: 10.1007/s10648-019-09514-z.

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

[29] Google. *WebRTC API*. May 29, 2021. URL: https://webrtc.org/.

[30] M. Haghighat and S. Zonouz M. Abdel-Mottaleb. "CloudID: Trustworthy cloud-based and cross-enterprise biometric identification". In: *Expert Systems with Applications* 42.21 (2015), pp. 7905–7916.

[31] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8691 LNCS.PART 3 (2014), pp. 346–361. ISSN: 16113349. DOI: 10.1007/978-3-319-10578-9_23. arXiv: 1406.4729.

[32] Javier Hernandez et al. "Measuring the engagement level of TV viewers". In: *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2013* (2013). DOI: 10.1109/FG.2013.6553742.

[33] Sepp Hochreiter and J Urgen Schmidhuber. "Long Shortterm Memory". In: *Neural Computation* 9.8 (1997), p. 17351780. URL: http://www7.informatik.tu-muenchen.de/$%5Csim$hochreit%0Ahttp://www.idsia.ch/$%5Csim$juergen.

[34] Andrew G. Howard et al. "MobileNets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv* (2017). ISSN: 23318422. arXiv: 1704.04861.

[35] Andrew Howard et al. "Searching for MobileNetV3". In: *arXiv* (2019). ISSN: 23318422.

[36] Jie Hu et al. "Squeeze-and-Excitation Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.8 (2020), pp. 2011–2023. ISSN: 19393539. DOI: 10.1109/TPAMI.2019.2913372. arXiv: 1709.01507.

[37] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. ISSN: 0003-4851. DOI: 10.1214/aoms/1177703732.

[38] Siddiqui J.R and Khatibi S. "Visual Tracking Using Particle Swarm Optimization". In: (2014), pp. 279–292. DOI: 10.5121/csit.2014.4126. arXiv: 1401.4648.

[39] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of Fluids Engineering, Transactions of the ASME* 82.1 (1960), pp. 35–45. ISSN: 1528901X. DOI: 10.1115/1.3662552.

[40] Vahid Kazemi and Josephine Sullivan. "One millisecond face alignment with an ensemble of regression trees". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 1867–1874. ISSN: 10636919. DOI: 10.1109/CVPR.2014.241.

[41] Davis E. King. "Max-Margin Object Detection". In: (2015). arXiv: 1502.00046. URL: http://arxiv.org/abs/1502.00046.

[42] Diederik P. Kingma and Max Welling. "An introduction to variational autoencoders". In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392. ISSN: 19358245. DOI: 10.1561/2200000056. arXiv: 1906.02691.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[44] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. ISSN: 00281441. DOI: 10.1002/nav.3800020109.

[45] Bogdan Kwolek. "Multi-object tracking using particle swarm optimization on target interactions". In: *Advances in Heuristic Signal Processing and Applications* 9783642378 (2013), pp. 63–78. DOI: 10.1007/978-3-642-37880-5_4.

[46] Duc Duong Lam et al. "Weighted Stable Matching Algorithm as an Approximated Method for Assignment Problems". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12034 LNAI (2020), pp. 174–185. ISSN: 16113349. DOI: 10.1007/978-3-030-42058-1_15.

[47] Shu Liu et al. "Path Aggregation Network for Instance Segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 8759–8768. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00913. arXiv: 1803.01534.

[48] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic gradient descent with warm restarts". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2017). arXiv: 1608.03983.

[49] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2.

[50] Wenhan Luo et al. "Multiple object tracking: A literature review". In: *Artificial Intelligence* 293 (2021). ISSN: 00043702. DOI: 10.1016/j.artint.2020.103448. arXiv: 1409.7618.

[51] Florence Martin and Doris U. Bolliger. "Engagement matters: Student perceptions on the importance of engagement strategies in the online learning environment". In: *Online Learning Journal* (2018). ISSN: 24725730. DOI: 10.24059/olj.v22i1.1092.

[52] Diganta Misra. "Mish: A self regularized non-monotonic neural activation function". In: *arXiv* (2019). ISSN: 23318422.

[53] Guanghan Ning et al. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *Proceedings - IEEE International Symposium on Circuits and Systems* (2017). ISSN: 02714310. DOI: 10.1109/ISCAS.2017.8050867. arXiv: 1607.05781.

[54] *Pion*. May 29, 2021. URL: https://pion.ly/.

[55] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. "Swish: a Self-Gated Activation Function". In: *arXiv* (2017), pp. 1–12. ISSN: 23318422. arXiv: 1710.05941.

[56] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[57] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[58] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015, pp. 91–99. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[59] Niels Ole Salscheider. "Object Tracking by Detection with Visual and Motion Cues". In: (2021). arXiv: 2101.07549. URL: http://arxiv.org/abs/2101.07549.

[60] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 4510–4520. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00474. arXiv: 1801.04381.

[61] Mohammad Soleymani et al. "A multimodal database for affect recognition and implicit tagging". In: *IEEE Transactions on Affective Computing* 3.1 (2012). ISSN: 19493045. DOI: 10.1109/T-AFFC.2011.25.

[62] Wei Sun et al. "How Presenters Perceive and React to Audience Flow Prediction In-situ". In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW (Nov. 2019), pp. 1–19. ISSN: 2573-0142. DOI: 10.1145/3359264. URL: https://dl.acm.org/doi/10.1145/3359264.

[63] Mingxing Tan et al. "MnasNet: Platform-aware neural architecture search for mobile". In: *arXiv* (2018). ISSN: 23318422.

[64] Jasper Uijlings et al. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104 (2013), pp. 154–171. DOI: 10.1007/s11263-013-0620-5.

[65] Pablo Villalobos Sánchez et al. "Tackling the design and evaluation of a theater-based intelligent system to monitor audience experience in virtual public speaking settings". In: (2021).

[66] C. Vinola and K. Vimaladevi. "A survey on human emotion recognition approaches, databases and applications". In: *Electronic Letters on Computer Vision and Image Analysis* 14.2 (2015), pp. 24–44. ISSN: 15775097. DOI: 10.5565/rev/elcvia.795.

[67] Paul Viola and Michael Jones. "Robust Real-time Object Detection". In: *International Journal of Computer Vision*. 2001.

[68] Chien Yao Wang et al. "CSPNet: A new backbone that can enhance learning capability of CNN". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2020-June (2020), pp. 1571–1580. ISSN: 21607516. DOI: 10.1109/CVPRW50498.2020.00203. arXiv: 1911.11929.

[69] Jane Webster and Hayes Ho. "Audience Engagement in Multimedia Presentations". In: *Data Base for Advances in Information Systems* 28.2 (Apr. 1997), pp. 63–77. ISSN: 00950033. DOI: 10.1145/264701.264706. URL: https://dl.acm.org/doi/10.1145/264701.264706.

[70] Jacob Whitehill et al. "The faces of engagement: Automatic recognition of student engagement from facial expressions". In: *IEEE Transactions on Affective Computing* 5.1 (2014). ISSN: 19493045. DOI: 10.1109/TAFFC.2014.2316163.

[71] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric". In: *Proceedings - International Conference on Image Processing, ICIP* 2017-September (2018), pp. 3645–3649. ISSN: 15224880. DOI: 10.1109/ICIP.2017.8296962. arXiv: 1703.07402.

[72] Sanghyun Woo et al. "CBAM: Convolutional block attention module". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11211 LNCS (2018), pp. 3–19. ISSN: 16113349. DOI: 10.1007/978-3-030-01234-2_1. arXiv: 1807.06521.

[73] Xiang Xiang. "A brief review on visual tracking methods". In: *Proceedings - 2011 3rd Chinese Conference on Intelligent Visual Surveillance, IVS 2011* (2011), pp. 41–44. DOI: 10.1109/IVSurv.2011.6157020.

[74] Hao Yang, Yingqing Huang, and Zhihong Xie. "Improved correlation filter tracking with enhanced features and adaptive kalman filter". In: *Sensors (Switzerland)* 19.7 (2019). ISSN: 14248220. DOI: 10.3390/s19071625.

[75] Tien Ju Yang et al. "NetAdapt: Platform-aware neural network adaptation for mobile applications". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11214 LNCS (2018), pp. 289–304. ISSN: 16113349. DOI: 10.1007/978-3-030-01249-6_18. arXiv: 1804.03230.

[76] Xiaoqin Zhang et al. "Multi-object tracking via species based particle swarm optimization". In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009* (2009), pp. 1105–1112. DOI: 10.1109/ICCVW.2009.5457581.

[77] Zhaohui Zheng et al. "Distance-IoU loss: Faster and better learning for bounding box regression". In: *arXiv* (2019). ISSN: 23318422. DOI: 10.1609/aaai.v34i07.6999. arXiv: 1911.08287.

[78] Zhaohui Zheng et al. "Enhancing geometric factors in model learning and inference for object detection and instance segmentation". In: *arXiv* (2020). ISSN: 23318422. arXiv: 2005.03572.