DMB

DATABASE MANAGEMENT
AND
BIOMETRICS

.07426

# EXPLORING HUMAN RELATIONSHIP RECOGNITION IN EGOCENTRIC VIDEOS USING DEEP LEARNING TECHNIQUES

Bart Peeters

MASTER'S THESIS ASSIGNMENT

**Committee:**
E. Talavera-Martínez
dr.ir. D. Reidsma

December, 2022

2022DMB0010
Data Management and Biometrics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

## ABSTRACT

Automating human relationship recognition, eg., friends, strangers, colleagues, etc., has big application potential in fields such as social media analyses, intelligent business services and public security. Deep learning techniques have made this automation possible. In the work of Costa et al. [1], 3 different cues, called the face, context, and body encoding stream, were to gather information from different parts of a scene. This information was combined using an Adaptive Fusion Module. In order to access this recognition for videos instead of single images, in this work, the original architecture was extended by incorporating Convolutional LSTMs. A brand new massive-scale egocentric dataset, called Ego4d, available with video's of daily life situations was used in order to test the automated human relationship recognition model just described. This dataset has been labeled with the classes: Friend, Stranger, Service, Colleague, Parents-offs, Couple. The model achieved an accuracy of 57% and an F1-score of 0.55 on this 6 class classification problem. Comparing the model to the best performing model with only a single cue showed about an increase of 3% in accuracy and 0.04 in F1-score.

## 1 INTRODUCTION

With the rapid development of multimedia technology, the demand for automating tasks is also increasing. Automating the interpretation of social relationships will make it possible to process massive amounts of multimedia data which promote social media analyses, intelligent business services and public security. Specifically, this work will focus on human relationships in videos with the aim to be able to classify them. Scenes in these videos will contain different types of information such as information about the faces of a people inside a scene, the context surrounding those people and the body of those people. Recent work has shown that multiple cues that process this type information from different parts of a scene (in an image) can combined using an Adaptive Fusion Module [1] [2]. In this work, that architecture will be expanded with Convolutional Long short-term memories (LSTMs) in order to access video material. In order to use this architecture for the task of classification, the Ego4d dataset [3] will be used. This is A massive-scale, egocentric dataset and benchmark suite collected across 74 worldwide locations and 9 countries, with over 3,670 hours of daily-life activity video. The specific dataset used from Ego4d in this study consist of 572 video's which are each 5 minutes long. The videos consist of footage from people wearing a go pro camera in daily life situations.

From carrying out this research, insight will be gained with which the following 3 questions will be tried to be answered:

- How can a framework be build with existing deep learning techniques that can be used to recognize human relationships between the camera wearer and the person and the person they are interacting with in egocentric videos?
- How do different visual cues, e.g. face, context or body, help us to recognise the human relationship depicted in egocentric videos?
- How does combining the visual cues; face, context and body; using an adaptive fusion module, help us to recognise the human relationship depicted in egocentric videos?

As a short introduction to the method of this work, the original dataset is deemed too complex because the videos are quite long and multiple interactions can occur inside these videos, with these interaction also describing different human relationships. That is why the videos of the original dataset will first be trimmed in such a way that only interactions between the camera wearer and one other person remain. This will create more and shorter videos which are easier to label. Thereafter, frames from these videos will be selected because the architecture uses a sequence of images as an input. The frames that will be selected are the ones where the person with whom the camera wearer is interacting with is in the middle of the frame. From these frames 3 images will be created in which the face, context around the person, or body will be highlighted. These images will be used as an input for the architecture mentioned earlier and from that the videos will be classified by a certain human relationship.

In the conduction of this research, the contributions to literature listed below were made, which will be explained in more depth throughout the report.

- Creating a new dataset from the the Ego4D [3] dataset, composed of 991 video's and labelled with 6 labels namely; Friend, Strange, Service, Colleague, Parents-offs and Couple.
- Using the Ego4D dataset for the task of human relationship recognition.
- The proposed framework which extends the work proposed in [1]. Instead of using a single image input, Convolutional LSTM's will be used to combine an input sequence of images and allowing video classification.
- Ablation study assessing the robustness when only relying on one of the considered cues namely; the Face Encoding Stream, Context Encoding Stream and Body Encoding Stream.

The organization of this paper is as follows. First, in the related work section, scientific background will be given on the concepts used in this paper and the methods used in this paper will be compared to methods used in other studies. Second, the framework used to allow for human relationship recognition in video's of the Ego4d dataset will be proposed. Third, experiments will be conducted in order to test the performance of the proposed architecture and in order to be able to answer the question composed in the introduction. Lastly, the results of the experiments will be discussed, a conclusion will be drown in which the question composed in the introduction will be answered, limitations of this work will be discussed and recommendations for future work will be given.

## 2 RELATED WORKS

The related work section consist of two parts. First, the scientific background, in which the main components of the architecture used in this work will be introduced. And second, the related research, in which literature related to this research will be explored.

## 2.1 Scientific background

In this section relevant research related to this work will be explored. Firstly, an introduction to the field of Convolution Neural Networks (CNNs) and Long Short Term Memory (LSTM) will be given, which are the main components of the Architecture that will be used in this work. Thereafter, another component used inside the architecture called an attention mechanisms will be explained.

*2.1.1 Introduction to CNNs:* In the field of artificial intelligence (AI), CNN is a class of Artificial Neural Networks (ANNs). Cnns are most commonly applied on visual images. ANNs are computational processing systems inspired by the way biological neural systems operate. ANNs are mainly comprised of a high number of interconnected computational nodes (referred to as neurons), of which work entwine in a distributed fashion to collectively learn from the input in order to optimise its final output [4]. An example of an ANN is given in figure 1. The neurons, represented by circular nodes in the figure, take as an input the output of the nodes in the previous layer. The output of a neuron is the weighted sum of the inputs, so multiplying every input with a certain weight. An ANN can be trained by providing input and comparing the output of the ANN to the desired output. This comparison will be made by a so called loss function. The weights in the ANN will then be adjusted to match the desired output. This process is called backpropagation.

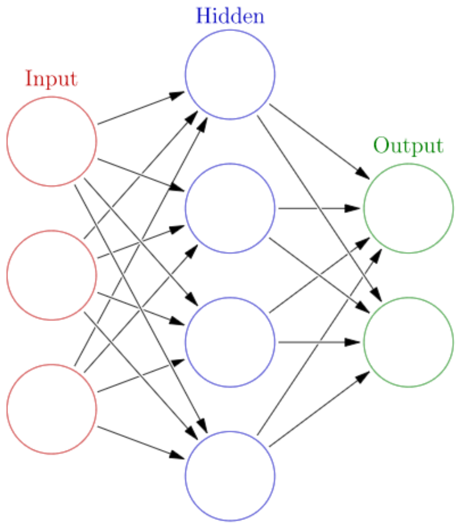


**Figure 1: An ANN is a group of interconnected nodes inspired by the simplification of neurons in the brain. Here each circular node represents an artificial neuron and the arrow he represents the connection from the output of one artificial neuron to the input of another artificial neuron.**

As CNNs are a class of ANNs, they are also based on artificial neurons that self-optimise through learning. As mentioned earlier, CNNs are primarily focused on the basis that their input consists of images. As a result, the architecture set up is optimally focused on the need to process this type of data. Consequently, the layers in a CNN consists of neurons organized in three dimensions: input spatial dimensions (height and width) and depth. The depth refers not to the total number of layers in the ANN, but the third dimension of an activation volume. Unlike standard ANNs, neurons are within specific layers only connect to a small area of the layer before it. In practice, for example, a CNN would take as input an image of 128x128x3 (height, width, depth) and give as an output a tensor with dimensions 1x1xn (n representing the number of classes). So the full input dimensionality is stored across the classes in the depth dimension.

To go from the input image to the output the image has to pas through different types of layers of the CNN. The three type of layers CNNs are comprised of are convolutional layers, pooling layers and fully-connected layers. These layers can be stacked in all kinds of ways to form a CNN architecture. An example of a CNN is given in Figure 2. The Feature extraction part of a CNN consists of one or multiple Convolutional and Pooling layers. Pooling layers are usually used after Convolutional layers, but not every Convolutional layer is followed by a Pooling layer. After the Feature Extraction part, the remaining layer is flattened to one dimension and one of multiple fully connected layers are used for Classification. Convolutional layers are used to determine the outputs of neurons that are associated with local regions of the input. This is done by computing the inner product (also known as a dot product) between the weights of the convolutional layer and the regions of the input volume. This allows the network to effectively detect and analyze patterns in the input data. A rectification linear unit (or ReLu for short) is an activation function that is applied element-wise to the output of the previous layer. This function is commonly used in neural networks and is applied to each element of the input independently. The purpose of the ReLu is to introduce non-linearity to the network, which allows it to learn more complex patterns in the data. The Sigmoid function is one example of a common activation function that can be used with a ReLu. The Pooling layer is used to downsample the spatial dimensions of the input, which reduces the number of parameters in the activation. This helps to simplify the data and reduce the computational cost of the network. By doing this, the network can focus on the most important features of the input and improve its performance. A fully connected layer then performs the same task as a standard ANN, trying to build class scores from the activations used for classification. It is also suggested that ReLu can be used between these layers to improve performance.

The field of CNNs only went booming quite recently. In this paragraph some of the classic CNN models will be stated, which are as to date still less than 10 years old. In 2012, Alex et al. [5] achieved the best classification results at the time using his Deep CNN in the ImageNet Large Scale Visual Recognition Challenge (LSVRC) with his CNN called Alexnet, garnering much attention in research and greatly advancing the development of his CNN in modern times. In 2014, VGGNets [6] won first place in the 2014 ImageNet Challenge localization track. It is based on AlexNet and improves it by sequentially replacing the large kernel-sized filters (11 and 5 in the first and second convolutional layers, respectively) with multiple 3X3 kernel-sized filters. VGGNets are a series of Convolutional Neural Network algorithms proposed by the Visual

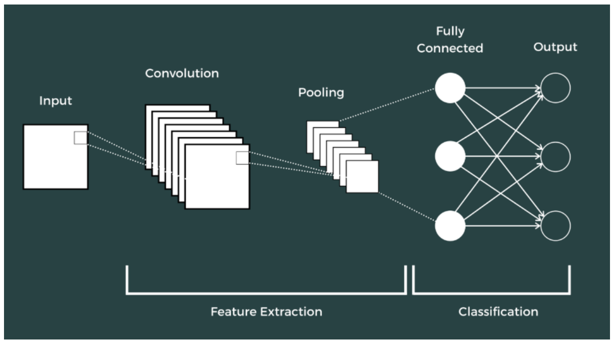


**Figure 2: Top level view of a CNN consisting of a input image, Convolution layer, Pooling layer and Fully Connected layer. Convolution and Pooling layers are used for feature extraction. Fully Connected layers are used for classification.**

Geometry Group (VGG) at the University of Oxford, including VGG-11, VGG-11-LRN, VGG-13, VGG-16, and VGG-19. GoogLeNet [7] is the winner of the ILSVRC 2014 Image Classification Algorithm. This is the first major CNN formed by stacking Inception modules, which basically is performing different type of Convolutional and Pooling operations separately on the same layer and concatenating them. For example, an input image goes to 2 different Convolutional layers, one followed by by another Convolutional layer and a Pooling layer and the other only followed by a Pooling layer. These two outputs are then concatenated. The main advantage of the Inception modules is that they heavily reduce the number of parameters. The Inception modules have 4 versions; Inception v1 [7], Inception v2 [8] [9], Inception v3 [9] and Inception v4 [10]. Lastly, He et al. [11] proposed a 34-layer residual network in 2016 called ResNet. This is the winner of the ILSVRC 2015 Image Classification and Object Detection Algorithms. One of ResNet's main contributions is a two-layer residual block built by shortcut connections. A shortcut connection is a connection in a neural network that skips one or more layers and than adds identity mapping to the output of those layers. This counteracts the effects of exploding and vanishing gradients (the weights of the neurons getting extremely big or small).

*2.1.2 Introduction to LSTMs.* Another type of ANN that will be used is the LSTM. The first thing in which an LSTM differs from a CNN is that it has feedback connections. ANNs with feedback connections are called Recurrent Neural Networks (RNNs), making an LSTM a type of RNN. The feedback connection means that the output of an RNN cell will be used as an input for the next iteration, together with the actual input to that cell. What is useful about RNNs is that they not only can process single data points such as images, but also sequences of data such as video's. The main motivation to use RNNs is because they are better at keeping memory about previous data compared to standard neural network's. A common problem with Neural Networks is that the deeper (more layers) they become, the bigger the problem of vanishing and exploding gradients. With RNNs, this becomes an even bigger problem, since gradients are computed not only in the 'depth' dimension but also in the 'time' dimension [12].

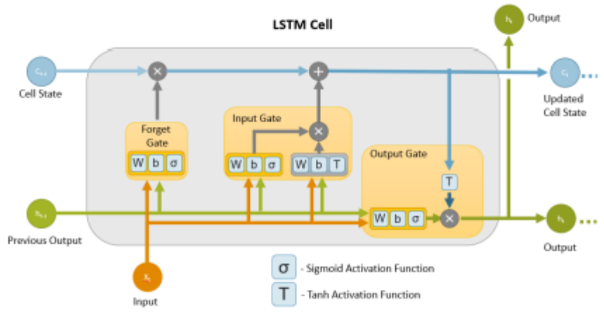


**Figure 3: The architecture of an LSTM cell [14].**

To counteract this problem, LSTMs were introduced. The structure of an LSTM is very similar to that of an RNN, the only difference being a replacement of the recurrent cell by an LSTM cell, which will remember information longer. Like RNN cells, LSTM cells also reuse the output for the next iteration, this is referred to as the hidden state and serves for the short term memory. But LSTMs they also keep an internal cell state which serves for the long-term memory. The calculation of the output/hidden state and the cell state happens inside the LSTM cell with the use of 3 gates: the Forget Gate, Input Gate and Output Gate. Figure 3 shows how they operate with the input, Hidden state/output and Cell state with different type of activation functions. In this work a Convolutional LSTM [13] will be used. The core principles are the same as with a "normal" LSTM, only the input and output/hidden layer are now two-dimensional instead of one-dimensional.

*2.1.3 Introduction to Attention mechanisms.* Another type of module that will be inside the architecture of this work is an attention module. Attention mechanism are inspired by the investigation of human vision, as humans have the ability to only focus on relevant parts of visual information. This is where their "attention" goes to. Attention mechanism in neural networks where first introduced by Bahdanau et al. [15] and have later been instrumental for research in natural language processing (NLP) such as google BERT [16]. The attention mechanism has also been adopted into computer vision, like in this work. The general form of attention mechanisms in computer vision is formulated as:

$$Attention = f(g(x), x) \tag{1}$$

Where $g(x)$ responds to the attention generation part, which is responsible for finding the relevant part of imagery. $f(g(x), x)$ means processing input x based on the attention $g(x)$, in other words getting information out of relevant regions in imagery. As a practical example what these variables and functions can be, the attention mechanism from this work will be used. In this attention module: $x$ is a feature tensor of 4 dimensions; the function $g$ consists of 2 convolutional layers and a softmax layer where a feature tensor can go through, and the function f will be a multiplication between $g(x)$ and $x$. The attention module just described. has been used before for the same purpose as in this work [1] [2] and very relatable methods have used similar attention modules for the purpose of visual sentiment recognition [17] [18].

## 2.2 Related research

Now that an understanding has been shaped on the concepts that will be used in this work, there will be continued by looking what has been done in the field of egocentric vision, social interaction analysis and social relation analysis and a comparison to this work will be made.

*2.2.1 Egocentric vision.* Egocentric vision, also called first-person camera vision, refers to all Computer Vision and Machine Learning methods for extracting semantic information from visual lifelog data. Visual Lifelogging consists of capturing images that capture a user's everyday experience by wearing a camera for an extended period of time. In this work, Egocentric vision will be combined with human relationship recognition. Other examples of usages of Egocentric vision with social interaction are: Multi-face tracking [19], where all face that appear in a video are tracked; and, Detecting Social Interaction in an Egocentric Photo-stream [20], where the aim is to automatically detect the moments a social interaction occurs from wearing a camera during the day. This work will contribute to the field of egocentric vision by exploring how human relationship recognition will perform on these type of videos.

*2.2.2 Relationship extraction from video's.* In this work, the relationship between humans will be determined through the visual data from video's. Many other studies [21] [22] [23] [24] [25] [26] try to achieve similar feats, either by constructing social structures of finding relationships between characters from video analysis. One work that is closely related to this is called "A Multimodal Approach for Multiple-Relation Extraction in Videos" by Liu et al. [27], as the same labeling has partly been used in this work and the visual modality also focuses on the face and body of a person. In the work of Liu et al., scenes from movies were used as an input and through the analysis of the visual data and the subtitles the relation between every character in the scene was determined. Furthermore, a component called SlowFast [28] is adopted in their framework to capture semantic information and action information.

The main differences between the work of Liu et al. [27] and this one are: First of all, in their work has the 3 modalities just described, while this work only works with the visual modality. Secondly, in their work multiple relationships per video were determined whilst in this work there will only be one, namely the one between the camera wearer and the person he or she is interacting with. And lastly, a different dataset was used. The reason a different dataset and only the visual modality was used in that this research focuses on egocentric vision. The reason why only part of the same labeling has been used is because not all labels were relevant for the dataset used in this work.

*2.2.3 Visual information extraction.* A video can also be seen as a sequence of images. Therefore, image based technique's can be used to extract visual feature's from the video's. Liu et al. [27] used 20 frames of the original video's. From these face and body clusters where extracted using Mask R-CNN [29] to detect characters' body and RetinaFace [30] to detect characters' face. After that, pre-trained ArcFace [31] and ResNet [11] networks were used to extract face and body features separately. These features were later concatenated and fused with the features of the other models.

Similarly, in the work by Jiyoung Lee et al. [2], a sequence of images was used as input from which the face and the context were inputted into two separate networks. The network for the face was called the face encoding stream and consisted of 5 convolutional layers and 4 max pooling layers with batch normalization and ReLu layers after each convolutional layer. The context encoding stream is similar, with 1 convoluational and 1 max pooling layer less, followed by an attention interference module. The output of the two networks was used as an input for the adaptive fusion network to combine the features of the face and context streams and give a definitive classification, in the case of this specific research emotion.

Costa et al. [1] extended the framework of Lee et al. [2] by adding adding a third network that, like the work of Liu et al. [27], takes as an input the body of the person in the image and puts it through a pre-trained ResNet [11]. The adaptive fusion network has a third stream to allows for the extra input. In this work, the framework is even further extended by adding a Convolutional LSTM to allow a sequence of images to be used from each of the egocentric videos.

## 3 PROPOSED FRAMEWORK FOR MULTI-CUE SOCIAL RELATIONSHIP RECOGNITION

In this section the methodology is proposed in order to extract a relationship between the camera wearer and the person he is interacting with, will be described. This will be done using the architecture shown in Figure 5. However, this Neural network takes specific images as an input and not video's. Therefore, first the data preparation will be described in a subset of pre-processing steps such that the data can be used by the architecture. Thereafter, the architecture itself will be described.

## 3.1 Processing of video's

In this subsection there will be described how a dataset of video's should be transformed in such a way that it can be used by the Neural Network given in Figure 5. As mentioned before, the architecture has been extended to allow a sequence of image to be used as input. These images are non-consecutive frames. Therefore, the temporal aspect of the videos is neglected. The prepossessing steps will result in 8 specifically selected frames per video.

*3.1.1 Trimming the video's to only allow interaction with one single person.* The way the architecture is set up, only one label can be given per video. As it is possible for multiple interactions to occur in one video, all video's need to be trimmed in such a way that only one interaction occurs such that it is clear what label should be given to that video. In the dataset used in this work the camera holder is also part of the interaction and but does not appear in the video. If a dataset is used were the camera holder does not interact the 2 person appearing in the video that have an interaction can be used separately in the next step.

*3.1.2 Selecting and pre-processing the frames.* After the video's have been trimmed in the way just described, 8 frames per video

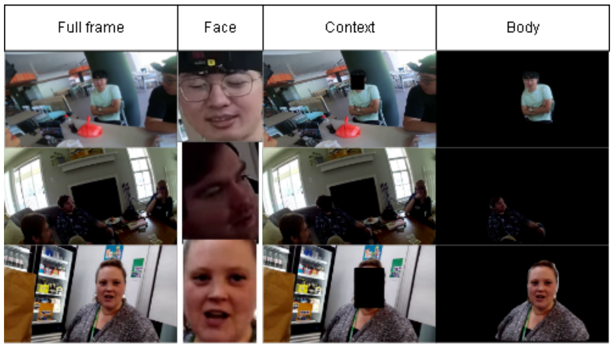


**Figure 4: Three examples of how the input images look like. The columns, from left to right, represent: the full image, the face of the main person, the context around the main person and the body of the main person.**

should be selected. If there are multiple people in this frame there should be identified which person is the one having the interaction, this will be referred to as the main person. From these frames, 3 separate images should be made. Firstly, an image of only the face needs to be extracted. Secondly, an image of the frame without the face needs to be extracted. Lastly, an image of only the body needs to be extracted. This should only be done with the main person.

In Figure 4 three examples are given of what the 3 type of images describes earlier look like and what the original frame looked like. The creation of these images was done in the following way. Firstly, it is assumed that the person the camera wearer is interacting with is in the middle of the frame most of the time during the video. Because multiple persons can appear in the video, an algorithm is used that counts how often a person appears in the video and give extra value when the person appears in the middle of the frame. The person that appears the most in the middle of the frame will be deemed the main person. From this the 8 frames in which the main person is the most in the middle of the frame will be selected to get the most context around that person as possible. With this algorithm, images of the complete frame and 8 images of the face were be saved per video. The pseudo code of the algorithm is given in Figure 14 in the Appendix.

RetinaFace [30] was used to detect the faces and ArcFace [31] was used to calculate the embedding of the face. The image of the frame without the face of the main person was extracted by again using the face of the main person but instead of extracting it, a block with black pixels will be placed on top of the face. Lastly, an image of only the body of the main person was extracted. This was done by using Mask R-CNN [29] to detect the body and filling the rest of the image with black pixels.

## 3.2 Visual network architecture

The network architecture is shown in Figure 5. This network is inspired by the works of Lee et all. [2] and Costa et all. [1]. Similar to those works, the network consists of 3 steams, namely the Face Encoding Stream, the Context Encoding Stream and the Body Encoding Stream. The 3 streams each take as an input an image. As their name suggests, the Face Encoding stream takes as input an image of a face, the Context Encoding Stream takes as input image of the context around the face and the Body Encoding Stream takes as input an image of a body and face. What is different from the works mentioned before is that a sequence of images will go through the network instead of a single image. This sequence is 8 images long and through every stream a different set of 8 images will go through it. The sequences of 8 will each be down-sampled to 1 using a Convolutional LSTM. This Convolutional LSTM is what makes this architecture diffent from the ones in the work of Lee et all. [2] and Costa et all. [1]. The features that come out of the Convolutional LSTM are fused together by a so called Adaptive Fusion Network. The structure of the 3 streams, Convolutional LSTM and the Adaptive Fusion Network will be discussed in more detail below.

*3.2.1 Face Encoding stream.* By looking the faces of two person one can often already get an indication of what type of relationship they might have. Although the face on the camera wearer is (often) not shown in the video's, also facial expression can tell a lot, for example one will look different towards a stranger than a friend. Therefore, a neural network that extract facial features is build and called the Face Encoding stream, as in the work of [1] and [2]. In the top of Figure 5 the Neural Network of the Face Encoding Stream is show. It takes as input an $3 \times 96 \times 96$ image of the face of the main person in the frame, as shown in the second column of Figure 4, which will be called $V_F$. The face encoding stream is designed to extract the facial expression features denoted as $X_F$ from the input images $V_F$,

$$X_F = \mathcal{F}(V_F; W_F) \quad (2)$$

with face stream parameters $W_F$. The Neural Network is designed using basic operations of 3D-CNNs, becasue they are suitable for spatiotemporal feature representation. It consists of 5 convolutional layers with 3x3 kernels and feature channels of sizes 32, 64, 128, 256 and again 256. Every convolutional layer is followed by a Batch Normalization and a Relu activation function. The first 4 convolutional layers are also followed by a max pooling layer with a kernal size of 2. The output of this Neural Network is a tensor of size $8 \times 256 \times 6 \times 6$. This will be used as input for the Convolutional LSTM.

*3.2.2 Context encoding stream.* Also the context inside the image is important. For example, a cash register will be a very likely indication that the person behind it has a service relation to the camera wearer. In the second row Figure 5 the Context Encoding Stream is depicted, which is the same as in the work of [2]. It takes as input an 3x480x270 image of the frame without the face of the main person, as shown in the third column of Figure 4, which will be called $V_C$. The Context encoding stream is designed to extract the context features denoted as $X_C$ from the input images $V_C$,

$$X_C = \mathcal{F}(V_C; W_C) \quad (3)$$

with context stream parameters $W_C$. The Neural Network, which is also based on 3D-CNNs, starts similar to Face Encoding Stream with 4 convolutional layer with 3x3 kernals feature channels of sizes 32, 64, 128, 256. Also every convolutional layer is followed by a Batch Normalization, a Relu activation function and a max pooling

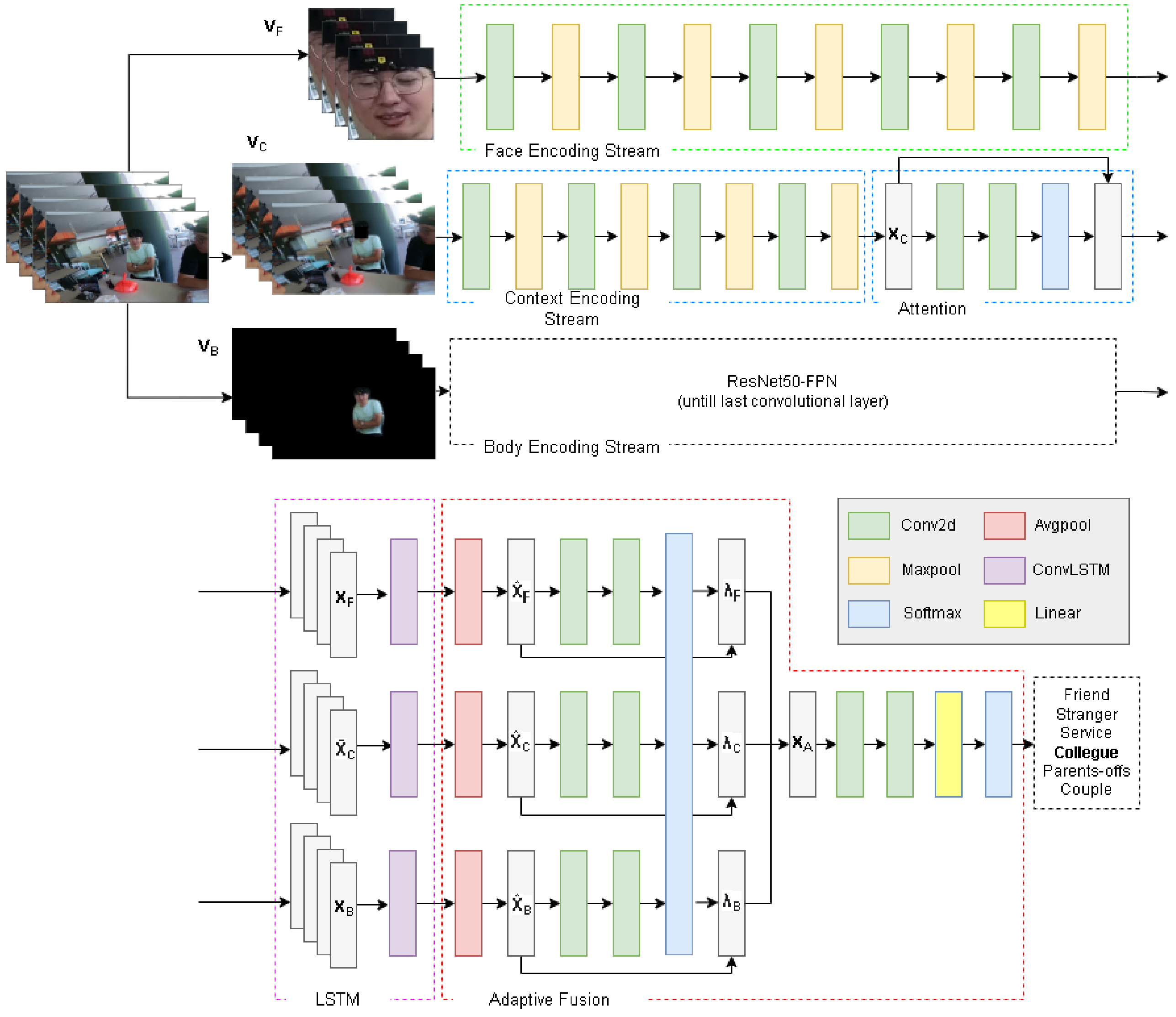
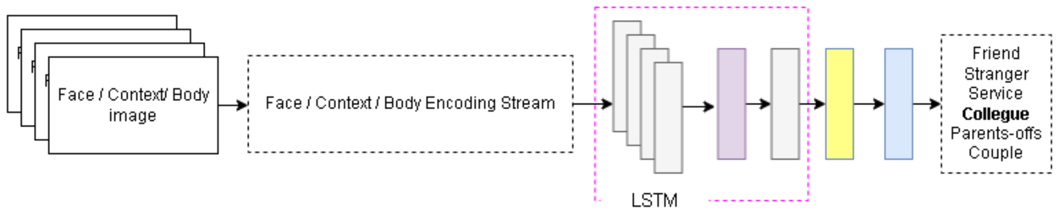


**Figure 5: The Main Architecture used to extract visual information from the images extracted from the video's, and predict the human relationship. The Face, Context and Body Encoding Stream, the Attention mechanism and Adiptive fusion are inspired by the work of [1] and [2]. The LSTM part has been added to this network in order to allow for a sequence of input images.**



Face / Context/ Body image
Face / Context / Body Encoding Stream
LSTM
Friend
Stranger
Service
Collegue
Parents-offs
Couple

**Figure 6: The Architecture used for the ablation study of the 3 different cue's. The Face/Context/Body Encoding streams and LSTMs are the same as in Figure 5. Instead of combining the information with an Adaptive Fusion module the 3 stream go to classification separately.**

layer with kernal size 2. This is followed by an Attention module to highlight small but import parts of the data, like for example the cash register. The Attention module consist of two steams: one is just the input, also know as $X_C$; and the other consist of 2 convolutional layers taking the input size from 256 to 128 and form 128 to 1. These 2 convolutional layers are also followed by a Batch Normalization and Relu activation function. After that a spatial softmax funtion is applied. This will form attention $A\epsilon\mathbb{R}^{HxW}$, where $HxW$ is the spatial resolution of $X_C$. The spatial softmax is applied as follows:

$$\hat{A}_i = \frac{exp(A_i)}{\sum_j exp(A_j)} \tag{4}$$

where $\hat{A}$ is the attention for context at each pixel i and $j\epsilon\{1,...,H \times W\}$. Since we use a 3D CNN to temporally aggregate the features, we normalize only the attention weights across the spatial dimension, not the temporal dimension. Note that attention is learned implicitly without supervision. Attention $\hat{A}$ is applied to the features $X_C$ to produce the attention enhancement feature $\bar{X}_C$ as follows.

$$\bar{X}_C = \hat{A} \cdot X_c \tag{5}$$

where $\cdot$ is an element-wise multiplication operator. This will form an output tensor of size $8 \times 256 \times 30 \times 16$. This will also be used as input for the Convolutional LSTM.

*3.2.3 Body Encoding Stream.* Lastly, also someones body language can also say a lot of the relationship towards the other person, therefore also a Body Encoding Stream will be used. As shown in Figure 5, the Body Encoding Stream is a pre-build Resnet50 module [11] that includes everything up to and including the last convolutional layer, as in the work of Costa et al. [1]. It takes as input an 3x480x270 image of the face and body of the main person in the frame, as shown in the fourth column of Figure 4, which will be called $V_B$. The Body encoding stream is designed to extract the body language features denoted as $X_B$ from the input images $V_B$,

$$X_B = \mathscr{F}(V_B; W_B) \tag{6}$$

with body stream parameters $W_B$. The output is a $8 \times 2048 \times 15 \times 9$ tensor. This will also be used as input for the Convolutional LSTM.

*3.2.4 ConvLSTM.* By using a Convolutional LSTM [13], the sequence of 8 tensors of features is downscaled to 1 tensors of features in order to let the Adaptive Fusion network be able to further process the data. In this architecture, 8 Convolutional LSTMs are put in a row, one per image of the input patch. The first LSTM takes as input the features of the 8th image in the sequence (features extracted by the stream) and zeros as initial hidden and cell state. The next LSTM takes as input the features of the 7th image in the sequence and the hidden and cell state of the previous LSTM. Continuing this untill you arrive at the last LSTM. The hidden state of this last LSTM will be passed on to the Adaptive Fusion Network. Similar as before, we have $\hat{X_F} = \mathscr{F}(X_F; W_{F2})$, $\hat{X_C} = \mathscr{F}(X_C; W_{C2})$ and $\hat{X_B} = \mathscr{F}(X_B; W_{B2})$. $F_2$, $C_2$ and $B_2$ being the network parameters of the Convolutional LSTMs handling the features coming out of the Face, Context and Body encoding stream respectively.

*3.2.5 Adaptive Fusion Network.* The features of the 3 different streams need to be combined. As simple concatenation often fails to provide optimal performance [32], an Adaptive Fusion Network will be used which has shown better performance [1][2]. The Adaptive Fusion Network in the works just mentioned will also be used in this work. This network is shown in the right bottom of Figure 5. First, the features resulting from the LSTMs will be put through an Adaptive Average Pooling layer in order to make the third and fourth dimension of the streams the same, which makes concatenation possible. The size of the third and fourth dimension is chosen to be 6 because the smaller input images (which are the faces) will have these dimension after going through the Face Encoding Stream and Convolutional LSTM. The 3 tensors that come out of the 3 different Adaptive Average Pooling layers will be saved and will still be called $\hat{X}_F$, $\hat{X}_C$ and $\hat{X}_B$. Next, the 3 tensors will each go through 2 separate convolutional layers that produce a 128 and 1 feature channel respectively, both are followed by a Batch Normalization and Relu function. The attentions are learned such that $\lambda_F = \mathscr{F}(\hat{X}_F; W_{F3})$, $\lambda_C = \mathscr{F}(\hat{X}_C; W_{C3})$ and $\lambda_B = \mathscr{F}(\hat{X}_B; W_{B3})$ with network parameters $W_F3$, $W_C3$ and $W_B3$, respectively. A softmax function will make the sum of these attentions to be 1, i.e.,

$$\lambda_F + \lambda_C + \lambda_B = 1 \tag{7}$$

The attentions will then be separated again and multiplied with the appropriate tensor that was saved earlier. The outcome of these 3 multiplication will then be concatenated, i.e.,

$$X_A = \Pi(\hat{X}_F \cdot \lambda_F, \hat{X}_C \cdot \lambda_C, \hat{X}_B \cdot \lambda_B), \tag{8}$$

where $\Pi$ is a symbol for concatenation. After that again 2 convolutional layers will produce a 128 and 1 feature channel respectively, i.e.,

$$y = \mathscr{F}(X_A; W_A) \tag{9}$$

where $W_A$ represents the remainder parameters of the adaptive fusion networks. Both convolutional layers are followed by a Batch normalization and Relu function. In order to train this network separately a fully connected layer will be put at the end in order to make an output of 6 relational categories. Lastly, a softmax will be applied.

## 4 EXPERIMENTAL SETUP

In this section experiments will be described that will validate the proposed framework and help to answer the questions stated in the introduction. Firstly, how the dataset used in this work is created from the original dataset is described. This will include how the video's are labelled and how the dataset is split. Secondly, the validation matrices used will be presented. Thirdly, how the experiments are set up and which values are chosen for certain parameters will be shown.

### 4.1 Dataset creation

**Dataset description:** As mentioned before, the dataset used is called Ego4D and is a massive-scale egocentric video dataset. It offers 3,670 hours of dailylife activity video spanning hundreds of scenarios (household, outdoor, workplace, leisure, etc.) captured by 931 unique camera wearers from 74 worldwide locations and 9 different countries [3]. In Figure 7, from the Ego4D paper, the demographics of the camera wearer are shown. This show the variety in age, gender, countries of residence and occupations of the camera wearers. It is assumed that these demographics are translated to the subset of data from Ego4D used in this work. It can be seen that there is a slight unbalance in gender with 55% male and 45% female. A large part of the ethnicity of the participant is American with 41%. The participants are more young people than old people with the age group 25-29 having 150 participants whilst all age groups above 44 having 41 participants or less. There is also unbalance in terms of occupation with (PhD) Student being the largest group.

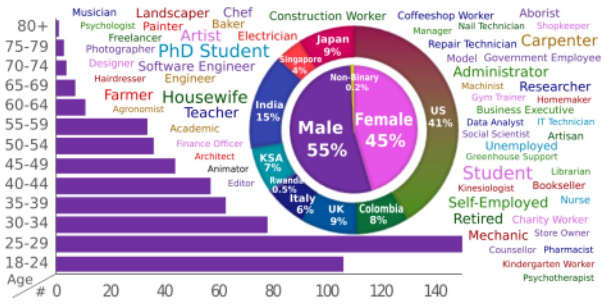


**Figure 7: Ego4D camera wearer demographics—age, gender, countries of residence, and occupations (self-reported). Font size reflects relative frequency of the occupation [3].**

**Trimming the dataset:** The original dataset consist of 572 video's which are each 5 minutes long and is called Ego4d [3]. In order to only have video's were there is only an interaction between the camera wearer and one other person, the original dataset will be split into sub-video's. These sub-video's will be were using the transcription of the video's that was given with the dataset. In the transcription the start time, end time and the person id were given. Based on that, the algorithm given in Figure 13 in the appendix was made. Using this algorithm resulted in 928 video's with only single person interactions. From these video's the input images are made as described in Section 3.1.2.

**Labeling of the video's:** As mentioned earlier, the dataset consist of 572 video's which are each 5 minutes long and is called Ego4d [3]. These original video's were trimmed into sub-video's which resulted in 928 video's. These video's were labeled into different categories with inspiration by the work of Liu et al. [27]. However, not all categories that they specified are relevant for this dataset. Therefore, the following relevant categories were used (with examples) for the labeling of the dataset: friend (Figure 8a), stranger (Figure 8b), service (Figure 8c), colleague (Figure 8d), parents-offs (Figure 8e) and couple (Figure 8f). When labeling the videos both the visual and the audio information was used. To be more explicit, in terms of visual information surroundings, facial expression and attitude were dominant factor and in terms of audio information tonality, language used (formal/informal) and the words spoken (buddy, sir, darling) were dominant factors.

**Dataset split** After labeling, the dataset was split into a train, test and validation set. Firstly, the total dataset was split into 85% train set and 15% test set. After that, this train set was split into a train and validation set with k-fold cross-validation [33] were

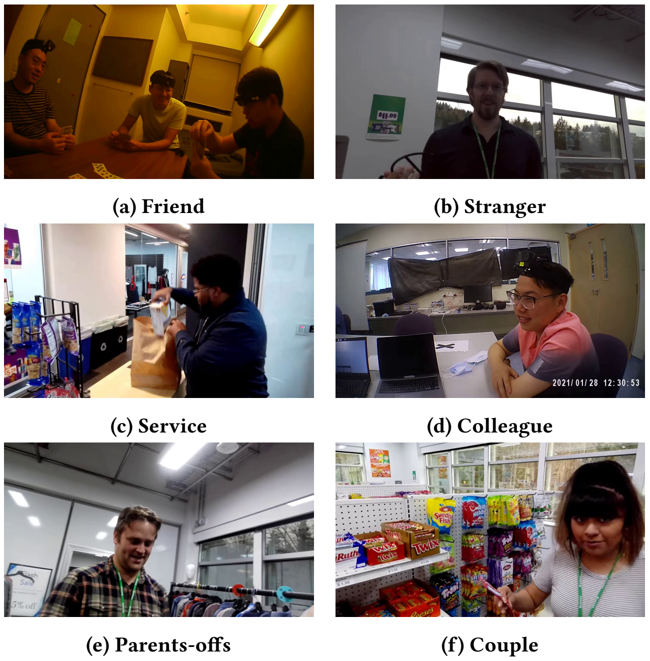


**Figure 8: Examples of the 6 different labels: Friend, Stranger, Service, Colleague, Parents-offs and Couple.**

k was set to 5. This implies that 80% of the train set is used for training and 20% for validation. Table 1 shows the total label count of every label and the counts of the train and test set. Table 2 shows the train and validation counts for every fold of the 5 fold split. As the train set of the first split was used for the second split, the total counts in Table 1 are the same as the train counts in Table 2. To accommodate for the imbalance in the amount of samples per class, weighted classes were used where the weight of every class was set to the total amount of training data divided by the number training samples in that class. This will make sure each class is trained equally.

| Label | Total | Train | Test |
|---|---|---|---|
| Friend | 457 (49%) | 380 | 77 |
| Stranger | 102 (11%) | 87 | 15 |
| Service | 132 (14%) | 107 | 25 |
| Colleague | 95 (10%) | 79 | 16 |
| Parents-offs | 46 (5%) | 37 | 9 |
| Couple | 96 (10%) | 81 | 15 |
| Total | 928 (100%) | 771 | 157 |

**Table 1: Label counts of the Train-Test split. Next to the total label count the percentage each label occupies in relation to the whole dataset is shown.**

### 4.1.1 *Baseline approaches.*

We propose to compare the robustness of are architecture against single visual cues. Figure 6 presents an illustration of these baseline experiments. The images will follow

| Fold 0 | | | |
|---|---|---|---|
| Label | Total | Train | Validation |
| Friend | 380 | 308 | 72 |
| Stranger | 87 | 71 | 16 |
| Service | 107 | 82 | 25 |
| Colleague | 79 | 64 | 15 |
| Parents-offs | 37 | 27 | 10 |
| Couple | 81 | 64 | 17 |
| Fold 1 | | | |
| Label | Total | Train | Validation |
| Friend | 380 | 300 | 80 |
| Stranger | 87 | 70 | 17 |
| Service | 107 | 86 | 21 |
| Colleague | 79 | 66 | 13 |
| Parents-offs | 37 | 26 | 11 |
| Couple | 81 | 69 | 12 |
| Fold 2 | | | |
| Label | Total | Train | Validation |
| Friend | 380 | 297 | 83 |
| Stranger | 87 | 72 | 15 |
| Service | 107 | 90 | 17 |
| Colleague | 79 | 59 | 20 |
| Parents-offs | 37 | 32 | 5 |
| Couple | 81 | 67 | 14 |
| Fold 3 | | | |
| Label | Total | Train | Validation |
| Friend | 380 | 306 | 74 |
| Stranger | 87 | 65 | 22 |
| Service | 107 | 89 | 18 |
| Colleague | 79 | 64 | 15 |
| Parents-offs | 37 | 35 | 2 |
| Couple | 81 | 58 | 23 |
| Fold 4 | | | |
| Label | Total | Train | Validation |
| Friend | 380 | 309 | 74 |
| Stranger | 87 | 65 | 22 |
| Service | 107 | 89 | 18 |
| Colleague | 79 | 63 | 16 |
| Parents-offs | 37 | 28 | 9 |
| Couple | 81 | 66 | 15 |

**Table 2: Label counts of the Train-Validation split for all 5 folds.**

the same process as in Figure 5 until the Adaptive Fusion, so they will each go through their desired encoding stream and go through a Convolutional LSTM module. Thereafter, each of the feature tensor coming out of the Convolutional LSTM will individually go through 1 linear layer and a softmax layer from which the human relationship can be categorized. Additionally, an experiment will be performed where the original unedited full frames will be in the baseline Context Encoding stream, as the Context images are most similar to the original full frames.

## 4.2 Validation

The performance metrics that will be used are balanced accuracy and F1-score, both will be calculated using the sklearn package [34]. The balanced accuracy and f1-accuracy will give an unbiased estimation of the performance of the network, which is necessary since the dataset in imbalanced. The balance accuracy, which can be defined as the average obtained on either class, is defined in equation 10.

$$Balancedaccuracy = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{FP+TN}) \tag{10}$$

Where: TP stands for true positive, an outcome where the model correctly predicts the positive class; TN stands for true negative, an outcome where the model correctly predicts the negative class; FN stands for false negative, an outcome where the model incorrectly predicts the negative class.

The F1-score is defined in Equation 11. The F1-score gives a balanced score of the precision and recall. Precision quantifies the number of correct positive predictions made out of positive predictions made by the model. In other words, precision calculates the accuracy of the True Positive and is calculated with Equation 12. The recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could be made by the model and is calculated with equation 13. In these equations TP and FN are the same as before. FP stands for false positive, an outcome where the model incorrectly predicts the positive class.

$$\begin{aligned} F1 &= \frac{2 * Precision * Recall}{Precision + Recall} \\ &= \frac{2 * TP}{2 * TP + FP + FN} \end{aligned} \tag{11}$$

Where:

$$Precision = \frac{TP}{TP+FP} \tag{12}$$

$$Recall = \frac{TP}{TP+FN} \tag{13}$$

Furthermore, the train and validation loss per epoch will be plotted in a graph. The loss function will be computed using the cross-entropy loss function. With the Train-validation set show in Table 2 the 3 baseline architectures and the main architecture will be trained and tested on performance. From this, the best performing architecture will be selected. This architecture will then be trained using the Training part of the Train-Test split shown in Table 1. After training the best performing architecture the balanced accuracy and f1-score, a confusion matrix and examples of correct and incorrect classifications will be given.

## 4.3 Implementation details

The Architecture that were described in the methodology implemented using the Pytorch library [35] from scratch, as no code was available from the works of Costa et al. [1] and Lee et al. [2]. The dataset that was used consisted of 991 video's of daily life scenario's filmed in a point of view fashion. As explained earlier, 20 frames per video were used. The 6 different labels that are used are shown in the previous subsection. The visual network consists of 3 different streams and an Adaptive Fusion Network. These 3 encoding streams were first build independently and tested on its performance on the classification problem. After that, the Adaptive Fusion Network was added and the combination of 3 streams will be tested tested on its performance. The 4 different architectures will all be trained with the 5 different folds as shown in Table 2 of the data-split section. Every fold was trained for 30 epochs. The learning rate was set to 1e-4 and the models were trained using a cross-entropy loss function and a batch size of 8. The best performing model will be trained for 50 epochs using the train-test split shown in Table 1.

## 5 RESULTS

In this section the results of the experiments conducted will be presented. Starting with comparing the different architectures and thereafter showing more in dept results of the best performing architecture.

### 5.1 Dataset exploration

In order to gain some insight in the feature that come out of the architecture, dimensionality reduction techniques were applied in order to visualize the data. Specifically, the features coming out of the last convolutional layer of the Adaptive Fusion module were analysed. Principal Component Analyses (PCA) [36] and T-distributed stochastic neighbour embedding (t-SNE) [37] were used to reduce the dimensions of the features from 214 to 2 in order to be able to visualise them in a 2d plot. This was done for all date, the training data and the test data, as shown in Table 1. The resulting plots are shown in Figure 9. The formation of clusters means that a label can be recognized by the features which makes it easier to classify them. In the training data, it can be seen that the labels form some nice clusters, especially when using t-SNE. However, in the test data these clusters are a lot less nice. Only the label Colleague has all train and test data close to one cluster. The label Friend has a pretty nice cluster and some outliers in both train and test data. The label Parents-offs has only one or a few points from the test data in its cluster from all data, and all the outliers are also from the test data. The labels Stranger, Service and Couple show some nice clusters in the train data but they are very close to each other (and the label Friend). In the test data the samples of these 3 labels are very much mixed in the same space.

Given this information, it is expected that the classier will do well on the label Colleague. It will do alright on the Label Friend with some miss classifications to every other label. The labels Stranger, Service and Couple will be miss classified a lot as each other. And the label Parents-offs will be miss classified a lot as the other labels except Colleague. It is also expected that not a lot of samples will be miss classified as Parents-offs because its cluster lays far away from the other clusters with no samples of other clusters close except for some samples of the label Friend in the PCA plot. The same can be said for the label Colleague with some samples of Friend close to it in both the PCA and t-SNE plot.

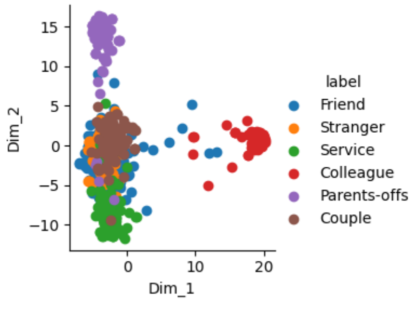
To quantify these results, the silhouette score [38] of the same features, without the dimension reduction techniques, were computed and plotted in the graphs. These graphs are shown in Figure 10. The silhouette score ranges from -1 to 1. A value of +1 indicates that a sample is far away from a neighboring cluster and close to its own cluster, a value of 0 indicates that a sample is close to a decision boundary and a value of -1 indicates that that a sample lays close to a wrong cluster. Comparing the 3 graphs it can again be seen that most of the outliers come from the test data. The Silhouette graphs of the test data confirm the statements above about the labels Colleague and Friends. But these graphs also show a Service also has some samples forming a cluster, as about half of the data has a silhouette score above 0. The labels Parents-offs and Stranger show the poorest performance in forming a cluster. The fact that the model performs very well on the training data and not so well on the testing data indicates that the model might be overfitting. The silhouette graph also indicates that the model might perform poorer on the training data of the label Friend than the rest. Lastly, a average silhouette score of 0.07 test data indicates that the model might be struggling to reach a good performance.

### 5.2 Model comparison

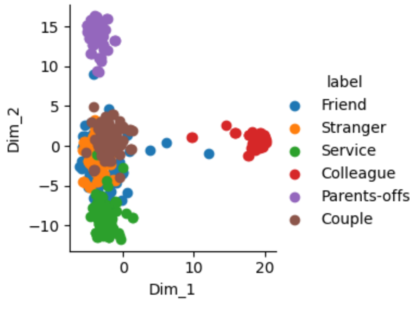
In this subsection the following 5 architectures, as described earlier, will be compared: The baseline model with the Face input images and Face Encoding stream, The baseline model with the Context input images and Context Encoding stream, The baseline model with the Body input images and Body Encoding stream, The baseline model with the Full input images and Context Encoding stream, and the Main Architecture. To avoid redundant writing, when an Encoding Stream is mentioned without the type of input image, the type will be the same as in the name of Encoding Stream.

**Train and validation loss:** In Figure 15, 16, 17, 18 and 19 the cross-entropy loss function is plotted against the number of epochs for the 5 different architecture across all fold. First taking a look at the low points of the validation loss: the Face Encoding Stream shows values between 1.43 and 1.50 across the folds, the Context Encoding Stream shows values between 1.36 and 1.43, the Body Encoding Stream shows values between 1.45 and 1.51, the Context Encoding Stream with full input images shows values between 1.37 and 1.42, and the Main Architecture shows values between 0.90 and 1.04. Then looking at the training loss, the values of the 4 Baseline Architectures go towards a value between 1.0 and 1.2 and the main architecture goes towards a value of 0.2 or lower. This could be a sign of overfitting for all architectures, but mainly in the main architecture because in that one the difference is the biggest. However, only looking at the validation loss, it can be concluded that the main architecture shows the best performance in terms of the loss function.

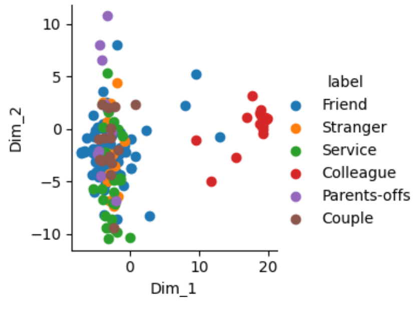
**Balanced accuracy and F1-score:** In Table 5, 6, 7, 8 and 9 the balanced accuracy and F1-score for every label in every fold is shown for all 5 Architecture is shown. Furthermore the averages and standard deviation are also shown. To make it easier to compare the models, the averages of every architecture are summarized in Table 3. Comparing these results, the first thing that is clear is that



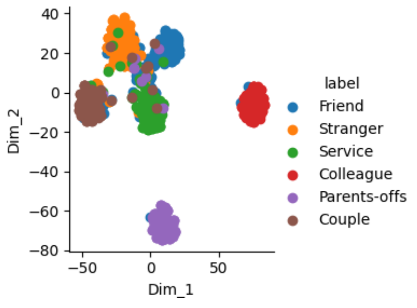
**(a) PCA all data.**



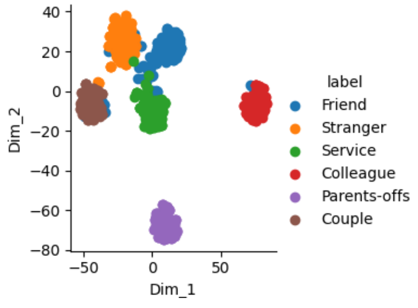
**(b) PCA train data.**



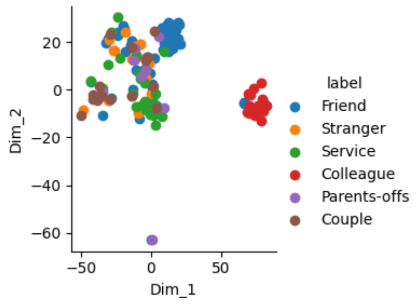
**(c) PCA test data.**



**(d) t-SNE all data.**

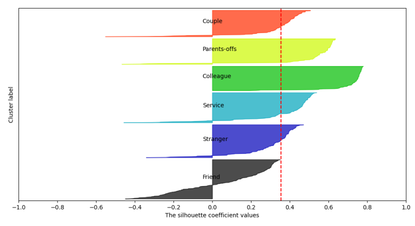


**(e) t-SNE train data.**

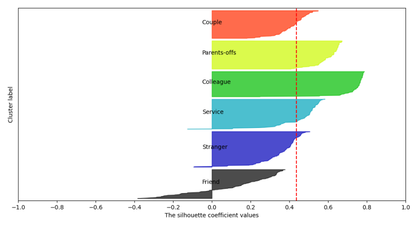


**(f) t-SNE test data.**

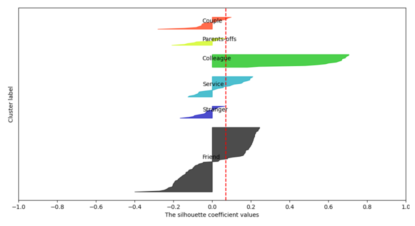
**Figure 9: Visualisation plots for the features coming out of the last Convolutional layer of the Adaptive fusion module using Principal Component Analyses (PCA) and T-distributed stochastic neighbour embedding (t-SNE). Both techniques have been applied to all date, the training data and the test data as shown in Table 1. The upper row shows the PCA results and the lower row shows the t-SNE results.**



**(a) Silhouette graph of the features of all data. Average silhouette score = 0.36.**



**(b) Silhouette graph of the features of the train data. Average silhouette score = 0.44.**



**(c) Silhouette graph of the features of the test data. Average silhouette score = 0.07.**

**Figure 10: Silhouette plots of the features coming out of the last convolutional layer of the Adaptive Fusion Module. The average silhouette score is reported under each plot and is indicated by a red dotted line. The labels of the clusters are ordered in the following way from top to bottom: Couple, Parents-offs, Colleague, Service, Stranger and Friend.**

the Body Encoding Stream architecture performs worst in terms of top-1 accuracy and F1-score on both the validation set and the test. For the other 4 architectures, the results are closer and more mixed across the different criteria. In the validation set the accuracy is almost the same with 54% for the Context Encoding Stream with both type of input images and 55% for the Face Encoding Stream and Main Architecture. In terms of F1-score, Context Encoding stream with full input images performs the best with 0.52 closely followed by the Main Architecture with 0.51. The Face and Context Encoding stream both have an F1-score of 0.50. In the test set, the Main architecture also scores highest in both accuracy (51%) and F1-score (0.50). The Face Encoding stream Architecture has an accuracy of 48% and an F1-score of 0.45. The Context Encoding stream Architecture has an accuracy of 48% and an F1-score of 0.45. The Face Encoding stream Architecture has an accuracy of 47% and an F1-score of 0.45. The Body Encoding stream Architecture has an accuracy of 46% and an F1-score of 0.43 and the Context Encoding Stream with full input images has an accuracy of 48% and an F1-score of 0.47.

**Model selection:** On the validation set, the Accuracy and F1-score of all architectures except the Body Encoding stream were very similar. On the test set, the Main Architecture outperformed the closest baseline model, which is the Context Encoding Stream

| Validation set | | | | |
|---|---|---|---|---|
| | Accuracy | std | F1-score | std |
| Face im+Face ES | 55% | 1% | 0.50 | 0.02 |
| Context im+Context ES | 54% | 3% | 0.50 | 0.05 |
| Body im+Body ES | 52% | 3% | 0.46 | 0.02 |
| Full im+Context ES | 54% | 4% | 0.52 | 0.04 |
| Main Architecture | 55% | 3% | 0.51 | 0.03 |
| Test set | | | | |
| | Accuracy | std | F1-score | std |
| Face im+Face ES | 48% | 3% | 0.45 | 0.03 |
| Context im+Context ES | 47% | 3% | 0.45 | 0.04 |
| Body im+Body ES | 46% | 2% | 0.43 | 0.02 |
| Full im+Context ES | 48% | 2% | 0.47 | 0.03 |
| Main Architecture | 51% | 3% | 0.50 | 0.01 |

**Table 3: The Balanced accuracy and F1-score for the validation and test set across the 5 different folds for the different architectures with their standard deviation. The first 4 architectures are different combinations of the Architecture shown in Figure 6. The last Architecture is the architecture shown in Figure 5. im is short for image and ES is short for Encoding Stream.**

with full input images, by 3% in accuracy and 0.03 in F1-score. Furthermore, the Main Architecture shows a much loss score on the validation set. As the Accuracy and F1-score are not that much better compared to the other architectures. This suggest that an error is made on the same amount of data but that the error are smaller. This could indicate that the Main Architecture is closer to predicting the right labels. For the these reasons, the Main Architecture will be chosen for the experiments in Section 5.3.

## 5.3 Experiments on the Main Architecture

For these experiments, the Main Architecture model was trained on the whole training set shown in Table 1 for 50 epochs. This subsection will include the performance of the Main model in terms of Balanced accuracy and F1-score, a Confusion matrix and a qualitative analyses of the frames.

**Performance in terms of Balanced accuracy and F1-score on different labels after training on the Train-Test set split compared training on the Train-Validation split:** In Table 4 the performance of the Main Architecture is shown after training it on the full dataset. Comparing these results to the average performance of the cross fold validation on the training set shown in Table 9, it can be seen that the overall performance is better with the Accuracy going from 51% to 57% and the F1-score going from 0.50 to 0.55. Interestingly, all the labels showed an increase in performance in terms of accuracy and F1-score except for the label "Parents-offs". The label "Friend" went from an accuracy of 60% to 62% and a F1-score of 0.69 to 0.72. Whilst the label "Stranger" went from an accuracy of 30% to 60% and a F1-score of 0.24 to 0.41; the label "Service: went from an accuracy of 52% to 60% and a F1-score of 0.50 to 0.52; the label "Colleague" went from an accuracy of 94% to 100% and a F1-score of 0.91 to 0.94: the label "Parents-offs" went from an accuracy of 36% to 22% and a F1-score of 0.36 to 0.31 and the label "Couple" went from an accuracy of 36% to 40% and a F1-score of 0.28 to 0.39. Comparing this to the dataset exploration earlier, most of the results are what was expected. Only the label Stranger has a much higher accuracy than expected.

| Test set | | |
|---|---|---|
| | Accuracy | F1-score |
| Friend | 62% | 0.72 |
| Stranger | 60% | 0.41 |
| Service | 60% | 0.52 |
| Colleague | 100% | 0.94 |
| Parents-offs | 22% | 0.31 |
| Couple | 40% | 0.39 |
| Average | 57% | 0.55 |
| std | 24% | 0.22 |

**Table 4: The Accuracy and F1-score for the Main Architecture trained with the whole training set.**

**Confusion matrix:** In Figure 11 the performance of the main architecture on the testing set of Table 1 is shown in the form of a confusion matrix. The first this that stands out is that the label Colleague is always predicted correctly and no other label is predicted as Colleague except for Friend which as predicted as colleague 3% of the time. The label Friend furthermore is correct 62% of the time and often miss classified as Stranger with 16% and Service with 12% and sometimes as Couple with 6%. The label Stranger is predicted 60% correct, and is predicted often as Service 20%, Couple 13% and sometimes as Friends with 7%. The label Service is predicted correct 60% and predicted most often wrong as Stranger with 20%. So the label Stranger and Service are often miss classified as each other. The label Parents-offs is only predicted correctly 22% of the time and is often miss classified as Service with 33% and Friend with 22%, whilst the labels Service and Friend are almost never miss classified as Parents-offs. Lastly, the label couple is predicted correctly 40% of time time, followed by being predicted as Friend 27%, as Service 20% and Stranger 13% of the time. Comparing this Confusion matrix to Figure 9c and 9f a lot of similarities can be seen. The labels Stranger and Service have a lot of points close together. The label Friend forms a pretty nice cluster but has a lot of outliers close to every other cluster. The label Parents-offs has samples all over the plot except for close to the label Colleague and only 2 samples of the test data are close to the cluster formed in the train data. The label Couple forms a pretty nice cluster in the plots of the train data but a lot of test samples are very close to the Stranger and Service clusters, which also comes back in the confusion matrix. Lastly, the label Colleague has a pretty nice cluster with only some samples of Friend close to it, which also is shown in the confusion matrix.

**Examples of right and wrong classifications:** In Figure 12, 3 examples are given of right classifications and 3 examples are given of wrong classification for everyone of the 6 labels. Only the first input image of the sequence (which consists of 8 images) is shown. The category Colleague had no wrong classifications,

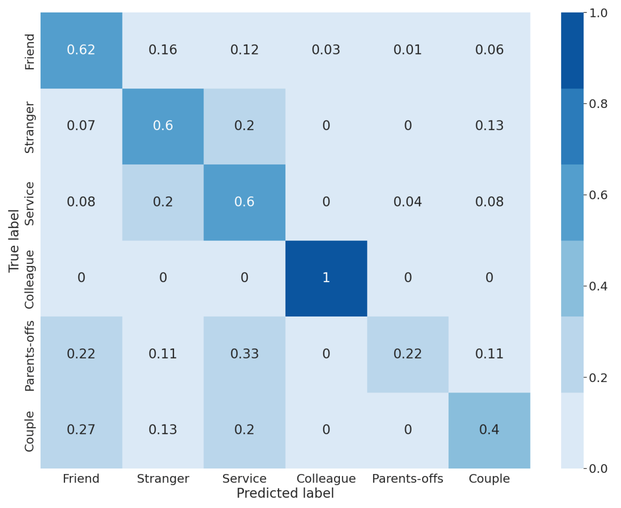


**Figure 11: Confusion matrix of the main architecture using the testing set and after training with the training set of Table 1 for 50 epochs.**

therefore only the 3 correct examples are shown. The same goes for the category Parents-offs, which only has 9 samples in the test set of which only 2 were predicted correctly.

The most interesting result from the experiments is that the category Colleague has a 100% accuracy. From labeling the videos, it was notable that a lot of the videos that were labelled with colleague had people with Asian ethnicity appearing. This unbalancedness in ethnicity could make it easier for the model to distinguish this category. Another thing that would make it easy to recognize is that the most of the scenes in the videos took place in a conference room or cafeteria and a lot of the time laptops appear inside the video. These points are also reflected in the examples of Colleagues shown in Figure 12.

For the service category, a lot of the video's were filmed in the same store. Therefore, the same personnel would appear in multiple video's and make it easier for the architecture to recognize these people. Another thing that could help recognize personnel is the employee card that they often whore with a chain around their neck.

From labeling the videos, it is clear that the first miss classification of Friend and the third miss classification of Service should be labeled as Service. But they are labeled as Couple. This shows that in some video's the person the camera wearer is talking to is not always the person he is looking towards the most. This issue was also evident when labeling the videos, in some videos the camera wearer was constantly looking around or even looking at someone else whilst talking to someone. Consequently, the assumption that the main person appears in the middle of the video most of the time does not always hold.

Furthermore, sometimes faces are recognized in other thing than an actual person. In the third right example of Friend a mirror reflection of the camera wearer is recognized as the main person, in some videos a face on a magazine is recognised as a person and in some videos a manikin is recognized as a person. As the videos these images are selected from were made from a transcript, therefore someone has to be talking in these video's. This again shows that the assumption that the main person appears in the middle of the video the majority of the time does not always hold. It is also clear that this would make it harder for the architecture to classify the video's correctly because the person on which the label is based does not even appear in the image.

What is also an issue is that in some of the frames the main person is not clearly in the frame. This is shown in the second right example of Friend, Stranger and Service and in the second wrong example of Service and Parents-offs. This will likely distort the learning of the model because the information on which it should make its prediction is not or badly supplied.

Another point, which could make it easier for the model to recognize the categories Colleague and Service, is that colleague and service are easier to recognise with only visual information. For example a service worker can stand behind a counter or have a checkout machine and a colleague can sit behind a desk or have a laptop. Whilst differentiating between Friend, Stranger, Parents-offs and Couple you have to look a lot more a facial expression and body language, which is harder to read and there is probably an overlap in how you behave towards people in these categories wheres Service and Colleague almost always behave in a professional and polite manner.

## 6 DISCUSSION

In this section, first the research question formulated in the introduction will be answered. Thereafter, some factors that have or may have impacted the results will be discussed. This will include the dataset and the selected parameters. Lastly, limitations of this work will be discussed and a small ethical discussion will be given about the consequences of this technology.

### 6.1 Addressing the proposed research questions

**How can a framework be build with existing deep learning techniques that can be used to recognize human relationships in egocentric videos?** A framework was build in python using the python library. The layers that were chosen inside the network were based on the Multi-cue adaptive emotion recognition network [1]. In addition, an Convolution LSTM was placed inside this network in order to allow for an input sequence of images. The input images extracted from the videos were transformed in a way that made sense for stream they were going to. Using these images the Network was able to predict a label which indicated the human relationship in that video.

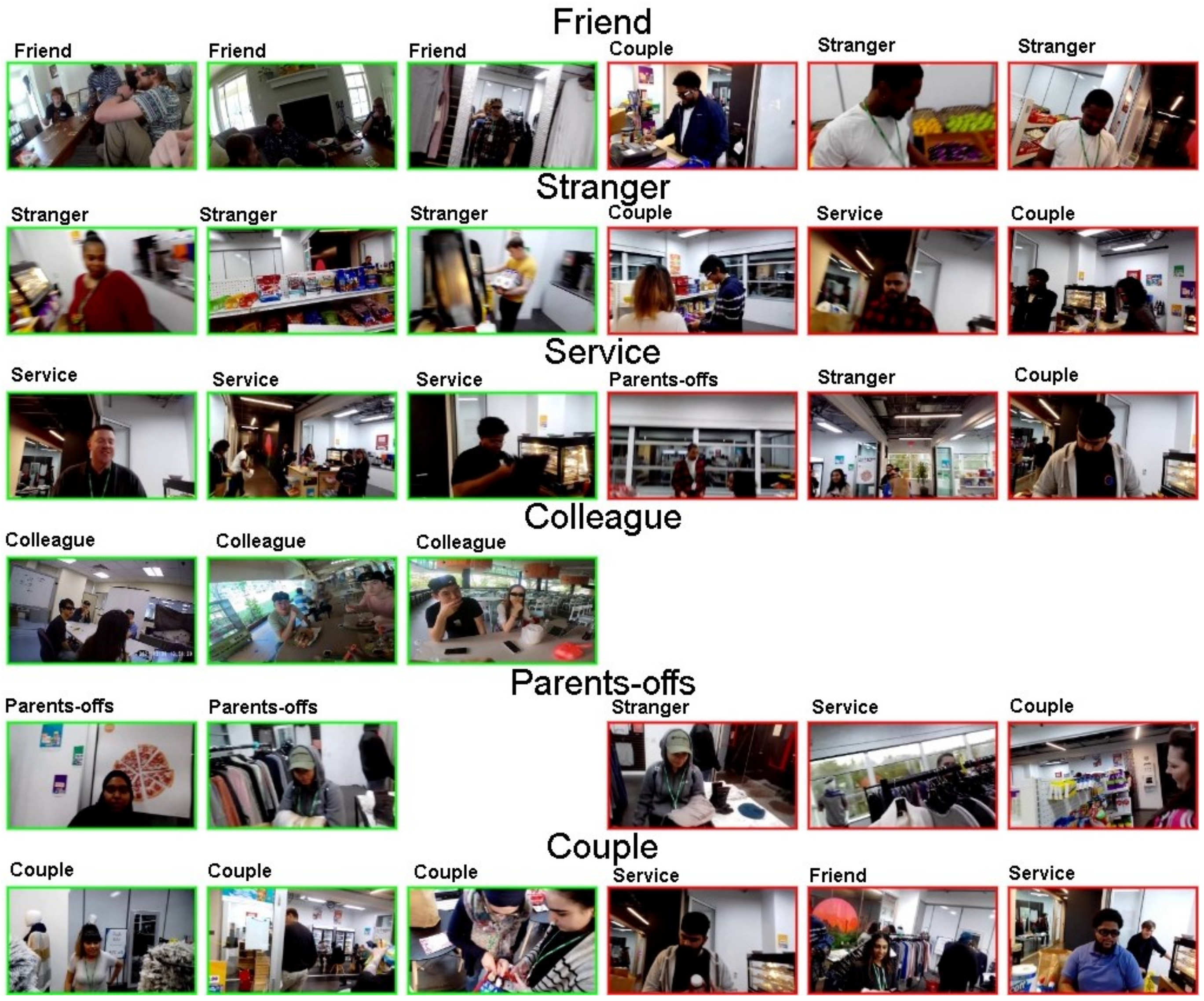


**Figure 12: Examples of the first input image of the input sequence and their predicted label. For each of the classes present in the dataset, 3 examples of right classification and 3 examples of wrong classification for every label are presented.**

**How do different visual cues, e.g. face, context or body, help us to recognise the human relationship depicted in egocentric videos?** Since this is a multi-class problem with 6 classes, the baseline would a random prediction, which would be 1 in 6 or around 17%. Compared to this, all 3 cues definitely helped to recognise the human relationship depicted in egocentric videos. However, there is a big difference in the performance between the classes. The face encoding stream shows a standard deviation of 22% between the classes for the validation set, whilst this is 13% for the context and 24% body encoding stream. This is similar to the test set with 26%, 26% and 27% respectively. In Table 5, 6 and 7 it can also be seen that the labels on which the models perform well are very similar, indicating that this has more to do with the dataset than the models. In terms of overall performance between the models, it is very close. The Face Encoding Stream performs the best with 55% accuracy and an F1-score of 0.50 on the validation set and an accuracy of 48% and F1-score of 0.45 on the test set. The Context Encoding Stream performs 1% worse in terms of accuracy and performs the same in terms of F1-score. The Body Encoding Stream has an accuracy of 52% and an F1-score of 0.46 on the validation set and an accuracy of 46% and an F1-score of 0.44 on the test set

**How does combining the visual cues; face, context and body; using an adaptive fusion module, help us to recognise the human relationship depicted in egocentric videos?** Using an adaptive fusion model to combine the cues showed a very slight increase in terms of accuracy and F1-score, with a 3% accuracy increase in the testing set and an increase of 0.05 in terms of F1-score on the testing set. However, the main difference was found in the loss, which was significantly lower than the loss of the separate cues. This tells us that the mistakes in the predictions were smaller but the eventual predicted labels stayed similar. This indicates that the main model has more potential to do better, for example when more samples of data is provided.

## 6.2 Addressing the contributions of this work

**Dataset:** The dataset used has some issues that could negatively influence the performance of the architecture and the classification results. First of all, the dataset is imbalanced as can be seen in Figure 1, meaning that not every category has the same amount of samples. Some categories do not have that many samples, for example Parents-offs only has 46 samples. This low amount of training data will make it harder to learn to predict this class well. Next, there could have been made some mistakes in the labeling of the video's, as this was done by one person and it is very tedious work. Lastly, padding was used to fill up missing pieces in the image creation. It would probably have been better if this was not needed. The fact that the models struggle to translate the data from training to testing data might indicate low generalisation in the dataset.

**Algorithm hyper parameter optimization:** Algorithm hyper parameters such as learning rate, number of epochs and batch size have a big influence on the performance of the network. However, not a great amount of effort was given towards optimizing these parameters. As discussed earlier, the batch size was limited by memory space and a higher batch size could improve performance. For the learning rate, it was set by looking at whether the initial learning of the well. This could be further optimised by for example a decaying learning rate [39].

**Selecting frames:** In the selecting frame algorithm 14, 1 in every 10 frames were retrieved. This was done to reduce processing time, not having the frames selected too close to each other and still finding (almost) enough frames have a face in them. In hindsight, this could still cause that only from a small part of the videos the frames are selected. Since the videos are filmed in 30 frames per second, in the worst case scenario all images are retrieved in only a 3 second time span. Ideally you would like to have these images more spread out throughout the video to capture more information.

## 6.3 Limitations

**Size limitations:** When putting a machine learning model on a GPU (or CPU), the size of the model of course has to fit on the GPU. Although there are a lot of factors determining the size of the model, there are 2 that do not change the general architecture but have a great effect on the size of the model and very possibly the performance of the model. The first one of these factors is the batch size. The smaller the batch size, the less memory is required. However, a smaller batch size also means that the gradient of the loss function will be less accurate. Meaning that the model will have a harder time finding the optimum for its parameters. Because there are 2 factors, the batch size was set to 8 and the maximum value was found for the other. The other factor is the length of the input sequence with the images. This one has a major effect on the size of the model. The larger this sequence is the more images have to be put on the GPU, but also the more Streams and LSTM components there are. With a batch size of 8 it was found that the maximum length of the input sequence is 8. If this number could be higher there would be more information for the model to work with, which also could lead to a better performance.

**Face detection:** Another limitation that might actually improve the results of the model but in the unwanted way is face detection. As mentioned before, some reappear in different videos. This already is the case in the original dataset, but because dataset set is trimmed into smaller videos this happens even more often. Some of the orinal videos would create 10 sub videos with sometimes the same person reappearing 4 or 5 times. These videos are likely spread over the different data split, which could cause the architectures to learn the faces instead of things like facial expression and body language. One counterargument to this not happening is that the Face Encoding Stream achieved similar performance as the Context Encoding Stream, in which the faces are removed from the images.

**Dataset size:** Another limitation of this work is the dataset size for certain categories. The reason why the label Friend probably had a relatively good performance is because a lot more examples are provided in that category with almost half of the data, giving the model much more data to learn from. For the labels Parents-offs, Couple and Stranger, it could be the case that the model was not provided enough data to distinguish them. This is also indicated by the fact that the samples of these labels are very much overlapping in the feature plots on the test data in Figure 9.

## 6.4 Possible negative consequences of this AI technology

One could ask the question whether we would want this type of technology in our lives. One of its main application is social media analysis. When this technology reaches more accurate results, social media platforms could figure out exactly which of your followers/friend on that platform are your friends, family, colleagues or people who are strangers when posting pictures or videos. It is imaginable that not everyone would like this.

## 7 CONCLUSION

In this work, an approach was represented for human relationship recognition in egocentric videos. The approach consisted of 3 cues that each took important non-verbal communication as an input. Using Convolutional LSTMs, multiple of these non-verbal communications could be provided such that multiple points in the video could be used as an input. The dataset used is a brand new one called Ego4d. which was labeled in this work with the categories: Friend, Stranger, Service, Colleague, Parents-offs and Couple, multi-class classification model with 6 classes. The proposed architecture reached an accuracy of 57% and and F1-score of 0.55. With the dataset having quite 'raw' date and given there are 6 classes to predict, these numbers are quite alright as a first step into exploring human relation recognition in these type of videos. Although there is still lots of room for improvement. Because the performance form the baseline models was very similar to that of the main model, overfitting because of a too complex model for the given data does not seem to be the main issue. As all models struggle to translate the training data to the testing data, the main issue presumably the dataset. Making the dataset more balanced and providing more samples such that the training data will generalize better towards the testing data will probably give the biggest improvement in performance.

## 8 FUTURE WORK

In this section there will be discussed what could be done in order to improve this work.

**LSTM placement:** Firstly, the LSTM placement will be discussed. The convolutional LSTM could have been placed at 3 different places. Namely: between the input images and the streams, between the streams and the Adaptive Fusion Network and behind Adaptive Fusion Network. As a design choose the convolution LSTM was put between the Streams and the Adaptive Fusion Network, because it was easier to implement and no prediction could be made which design choose would have been the best for performance. In Future work, all 3 design option could be investigated. However, the most interesting one would be the one between the input images and the streams. This is because during the implementation it was discovered that having every stream 20 times takes up a lot of RAM space on the GPU. If the performance of this design choose would be the same or even better it would definitely be a sensible choice, since you can than also test out the performance of different image and batch sizes.

**Expanding the dataset:** As already mentioned before, the dateset is very imbalanced and some of the labels have a very limited amount of samples. Expanding the dataset and making it more balanced would be very beneficial to better test the architectures described in this work. This could be done by expanding the amount of videos of the current dataset. Or, if finding or creating similar videos is difficult, data augmentation techniques could be used to increase the volume of the dataset.

**Using Audio information:** In this work only the visual side of the videos has been explored. However, in videos of course also audio information is provided. This type of information can also be very helpful in determining the human relationship between 2 people. Calling someone 'buddy' or 'mate' are strong indications of people being friends, similar to 'honey' for a couple or 'mom' or 'dad' for Parents-offs. And towards a service worker or stranger more polite words would be used like 'sir' or 'miss'. Therefore, combing the visual information with audio information could definitely be beneficial in determining the human relationship.

## REFERENCES

[1] Costa W, Macêdo D, Zanchettin C, Figueiredo LS, Teichrieb V. Multi-Cue Adaptive Emotion Recognition Network. arXiv; 2021. Available from: https://arxiv.org/abs/2111.02273. doi:10.48550/ARXIV.2111.02273.

[2] Lee J, Kim S, Kim S, Park J, Sohn K. Context-Aware Emotion Recognition Networks. arXiv; 2019. Available from: https://arxiv.org/abs/1908.05913. doi:10.48550/ARXIV.1908.05913.

[3] Grauman K, Westbury A, Byrne E, Chavis Z, Furnari A, Girdhar R, et al. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In: IEEE/CVF Computer Vision and Pattern Recognition (CVPR); 2022. .

[4] O'Shea K, Nash R. An Introduction to Convolutional Neural Networks. arXiv; 2015. Available from: https://arxiv.org/abs/1511.08458. doi:10.48550/ARXIV.1511.08458.

[5] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJ, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc.; 2012. Available from: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[6] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv; 2014. Available from: https://arxiv.org/abs/1409.1556. doi:10.48550/ARXIV.1409.1556.

[7] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al.. Going Deeper with Convolutions. arXiv; 2014. Available from: https://arxiv.org/abs/1409.4842. doi:10.48550/ARXIV.1409.4842.

[8] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv; 2015. Available from: https://arxiv.org/abs/1502.03167. doi:10.48550/ARXIV.1502.03167.

[9] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. arXiv; 2015. Available from: https://arxiv.org/abs/1512.00567. doi:10.48550/ARXIV.1512.00567.

[10] Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv; 2016. Available from: https://arxiv.org/abs/1602.07261. doi:10.48550/ARXIV.1602.07261.

[11] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. arXiv; 2015. Available from: https://arxiv.org/abs/1512.03385. doi:10.48550/ARXIV.1512.03385.

[12] Guo J. BackPropagation Through Time; 2013. .

[13] Shi X, Chen Z, Wang H, Yeung DY, Wong Wk, Woo Wc. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. arXiv; 2015. Available from: https://arxiv.org/abs/1506.04214. doi:10.48550/ARXIV.1506.04214.

[14] Vennerød CB, Kjærran A, Bugge ES. Long Short-term Memory RNN. arXiv; 2021. Available from: https://arxiv.org/abs/2105.06756. doi:10.48550/ARXIV.2105.06756.

[15] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv; 2014. Available from: https://arxiv.org/abs/1409.0473. doi:10.48550/ARXIV.1409.0473.

[16] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv; 2018. Available from: https://arxiv.org/abs/1810.04805. doi:10.48550/ARXIV.1810.04805.

[17] Yang J, She D, Lai YK, Rosin PL, Yang MH. Weakly Supervised Coupled Networks for Visual Sentiment Analysis. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018. p. 7584–7592. doi:10.1109/CVPR.2018.00791.

[18] You Q, Jin H, Luo J. Visual Sentiment Analysis by Attending on Local Image Regions. Proceedings of the AAAI Conference on Artificial Intelligence. 2017 Feb;31(1). Available from: https://ojs.aaai.org/index.php/AAAI/article/view/10501. doi:10.1609/aaai.v31i1.10501.

[19] Aghaei M, Dimiccoli M, Radeva P. Multi-face tracking by extended bag-of-tracklets in egocentric photo-streams. Computer Vision and Image Understanding. 2016;149:146–156. Special issue on Assistive Computer Vision and Robotics - "Assistive Solutions for Mobility, Communication and HMI". Available from: https://www.sciencedirect.com/science/article/pii/S1077314216000679. doi:https://doi.org/10.1016/j.cviu.2016.02.013.

[20] Aghaei M, Dimiccoli M, Radeva P. With Whom Do I Interact? Detecting Social Interactions in Egocentric Photo-streams. arXiv; 2016. Available from: https://arxiv.org/abs/1605.04129. doi:10.48550/ARXIV.1605.04129.

[21] Barr JR, Cament LA, Bowyer KW, Flynn PJ. Active Clustering with Ensembles for Social structure extraction. In: IEEE Winter Conference on Applications of Computer Vision; 2014. p. 969–976. doi:10.1109/WACV.2014.6835999.

[22] Chiu YI, Huang CR, Chung PC. Character Relationship Analysis in Movies Using Face Tracks. In: MVA; 2013. .

[23] Ding L, Yilmaz A. Inferring social relations from visual concepts. In: 2011 International Conference on Computer Vision; 2011. p. 699–706. doi:10.1109/ICCV.2011.6126306.

[24] Weng CY, Chu WT, Wu JL. RoleNet: Movie Analysis from the Perspective of Social Networks. IEEE Transactions on Multimedia. 2009;11(2):256–271. doi:10.1109/TMM.2008.2009684.

[25] Yeh MC, Tseng MC, Wu WP. Automatic Social Network Construction from Movies Using Film-Editing Cues. In: Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops. ICMEW '12. USA: IEEE Computer Society; 2012. p. 242–247. Available from: https://doi.org/10.1109/ICMEW.2012.48. doi:10.1109/ICMEW.2012.48.

[26] Zhou L, Lv J, Wu B. Social Network Construction of the Role Relation in Unstructured Data Based on Multi-view. In: 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC); 2017. p. 382–388. doi:10.1109/DSC.2017.78.

[27] Liu Z, Hou W, Zhang J, Cao C, Wu B. A multimodal approach for multiple-relation extraction in videos - multimedia tools and applications. Springer US; 2021. Available from: https://link.springer.com/article/10.1007/s11042-021-11466-y.

[28] Feichtenhofer C, Fan H, Malik J, He K. SlowFast Networks for Video Recognition. arXiv; 2018. Available from: https://arxiv.org/abs/1812.03982. doi:10.48550/ARXIV.1812.03982.

[29] He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017. p. 2980–2988. doi:10.1109/ICCV.2017.322.

[30] Deng J, Guo J, Zhou Y, Yu J, Kotsia I, Zafeiriou S. RetinaFace: Single-stage Dense Face Localisation in the Wild. arXiv; 2019. Available from: https://arxiv.org/abs/1905.00641. doi:10.48550/ARXIV.1905.00641.

[31] Deng J, Guo J, Xue N, Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019. p. 4685–4694. doi:10.1109/CVPR.2019.00482.

[32] Kosti R, Alvarez JM, Recasens A, Lapedriza A. Emotion Recognition in Context. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 1960–1968. doi:10.1109/CVPR.2017.212.

[33] Stone M. Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society Series B (Methodological). 1974;36(2):111–147. Available from: http://www.jstor.org/stable/2984809.

[34] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825–2830.

[35] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024–8035. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[36] S KPFR. LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1901;2(11):559–572. doi:10.1080/14786440109462720.

[37] van der Maaten L, Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research. 2008;9:2579–2605. Available from: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[38] Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics. 1987;20:53–65. Available from: https://www.sciencedirect.com/science/article/pii/0377042787901257. doi:https://doi.org/10.1016/0377-0427(87)90125-7.

[39] You K, Long M, Wang J, Jordan MI. How Does Learning Rate Decay Help Modern Neural Networks? arXiv; 2019. Available from: https://arxiv.org/abs/1908.01878. doi:10.48550/ARXIV.1908.01878.

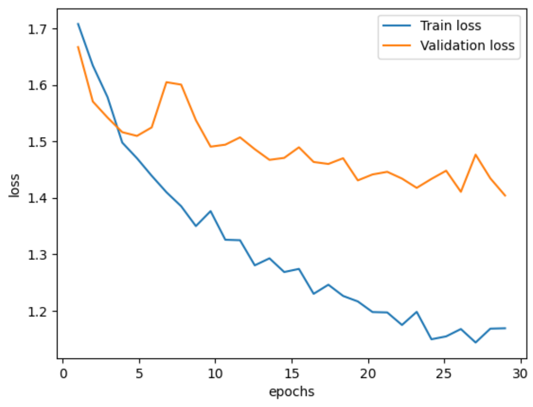
## A APPENDIX

```
For every row in transcription:
    if person_id is not -1:
        if start is 0:
            start = 1
            save start_time, end_time, person_id and length of transcription
        else if start is 1:
            if id is '0':
                id = person_id
            if person_id = 0:
                update end_time and lenth of transcription
            elif person_id is id:
                update end_time and lenth of transcription
            else:
                if video > 10 sec and both person say more than 25 characters
                    save video
                reset saved items
```

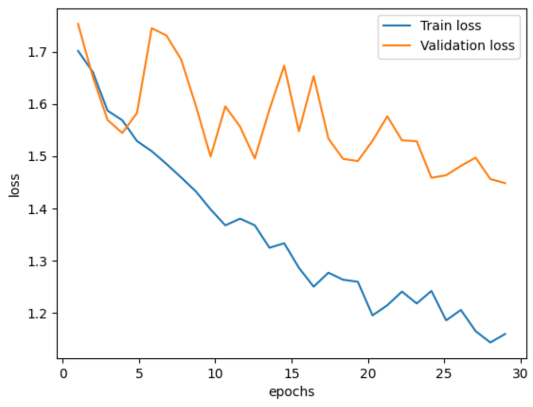
**Figure 13: Algorithm used for trimming the video's. Person-id = -1 means background noise, person-id = 0 means the camera wearer. The algorithm tries to find an interaction between the camera wearer and one other person. As soon as a third person enters the transcript, the last else statement is entered and the video is saved when the requirements are met.**

```
retrieve frame
while frame is being retrieved:
    detect faces
    if no saved faces:
        for all detected faces:
            get face area
            save face
    else:
        for all detected faces:
            get face area
            get face embedding
            for all saved faces:
                get face embedding
                score = max(cosine_similarity(embedding detected face, embedding saved face))
            if score < threshold:
                save face
            else:
                for all saved faces:
                    get embedding
                    if score == cosine_similarity(embedding detected face, embedding saved face)
                        save in list for particual face (sin(face_height/frame_height) + sin(face_width/frame_width))
                        if confidence score of new face is higher
                            replace face

for every list:
    take sum of list
    main_face is highest sum

save the index of the 20 highest items in the list of the main_face
```

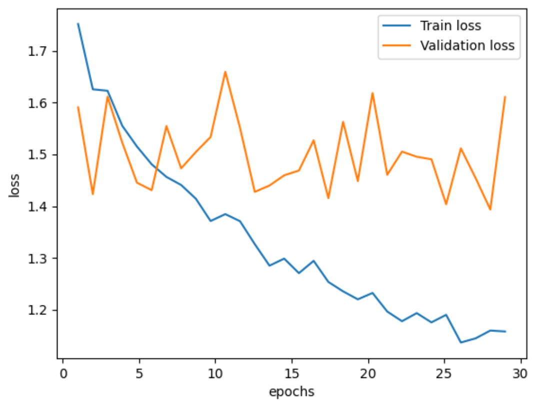
**Figure 14: Algorithm used for selecting frames. This algorithm is used to detect the main person in the video and also find the 8 most relevant frames**



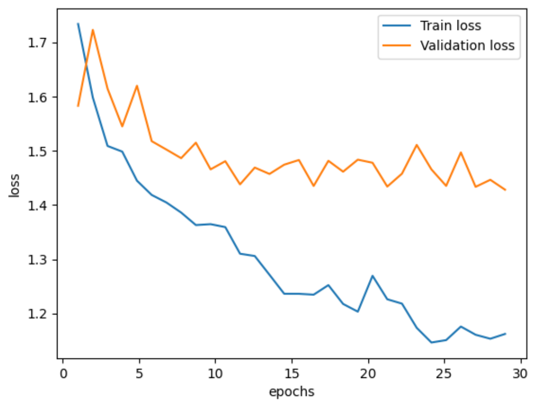
**(a) Fold 0**



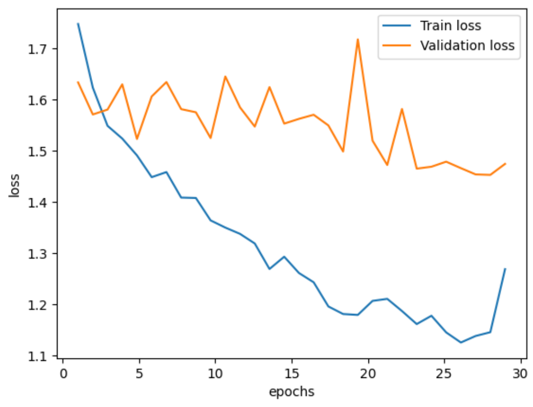
**(b) Fold 1**



**(c) Fold 2**

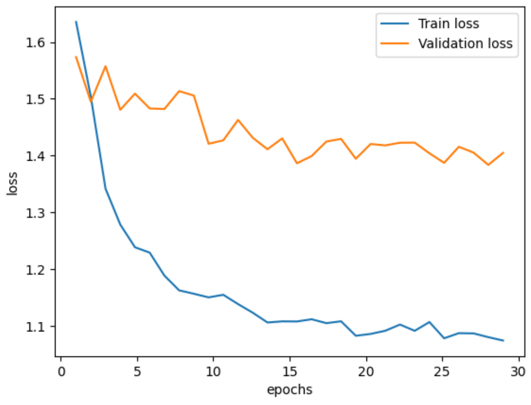


**(d) Fold 3**

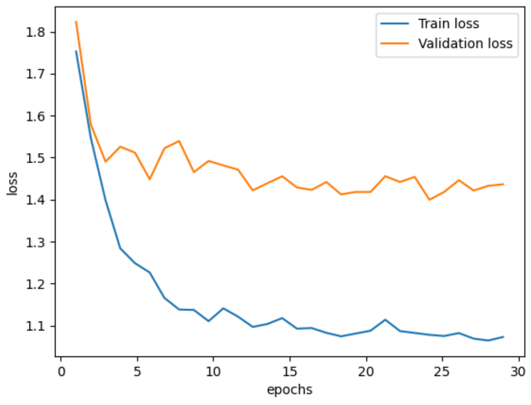


**(e) Fold 4**

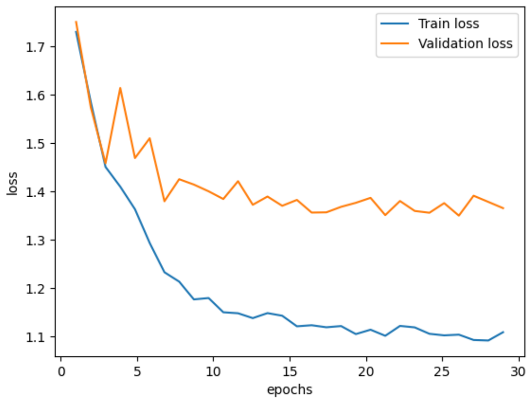
**Figure 15: The Train and Validation loss of every fold for the Face Encoding Stream with a learning rate of 1e-4**



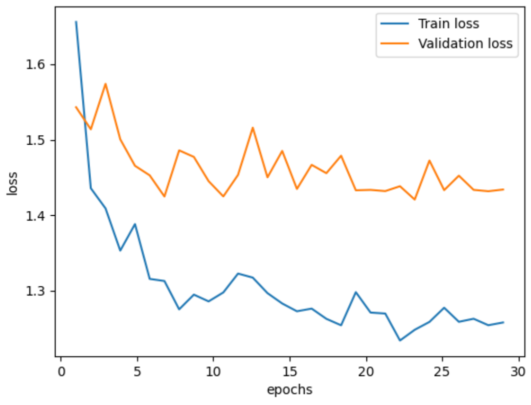
**(a) Fold 0**



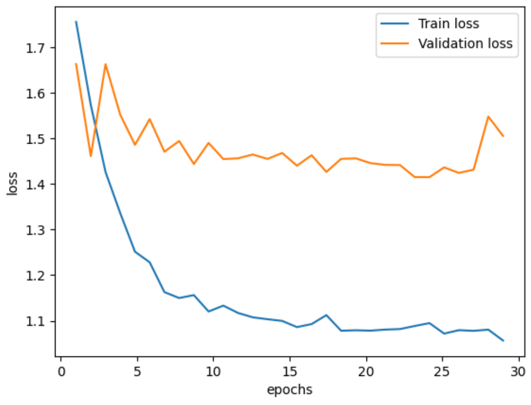
**(b) Fold 1**



**(c) Fold 2**

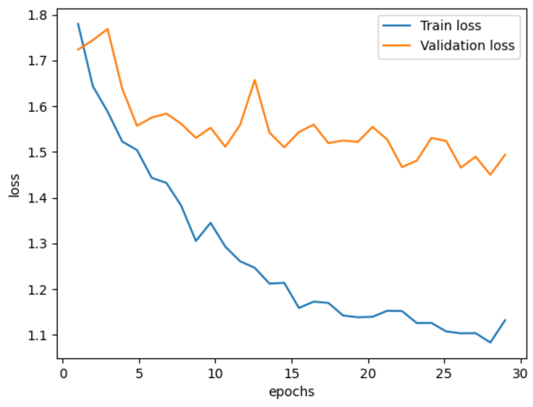


**(d) Fold 3**

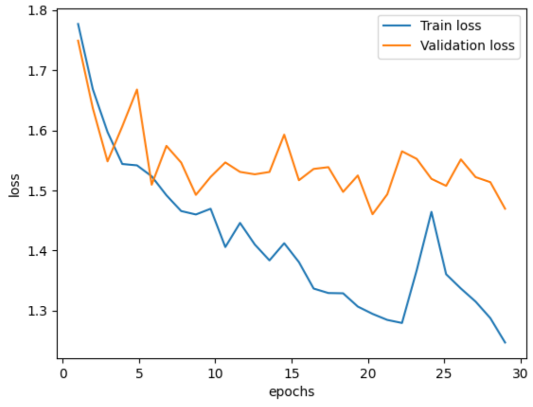


**(e) Fold 4**

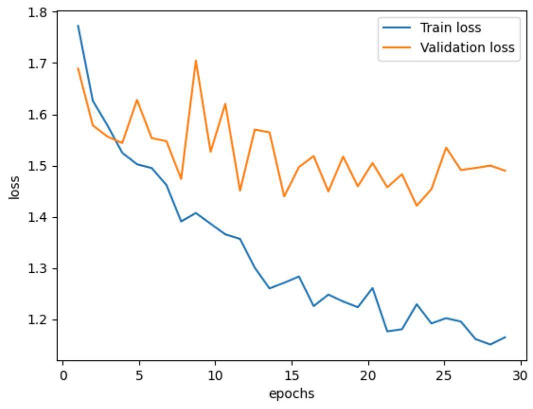
**Figure 16: The Train and Validation loss of every fold for the Context Encoding Stream with a learning rate of 1e-4**



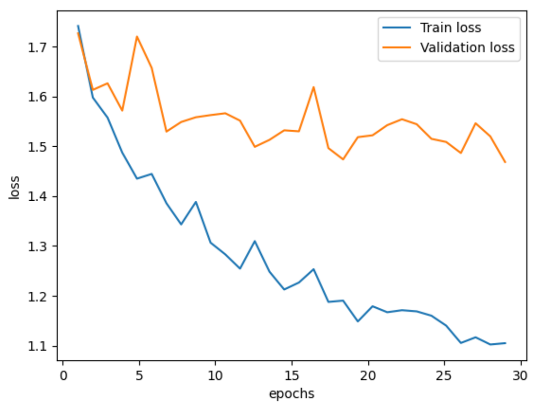
**(a) Fold 0**



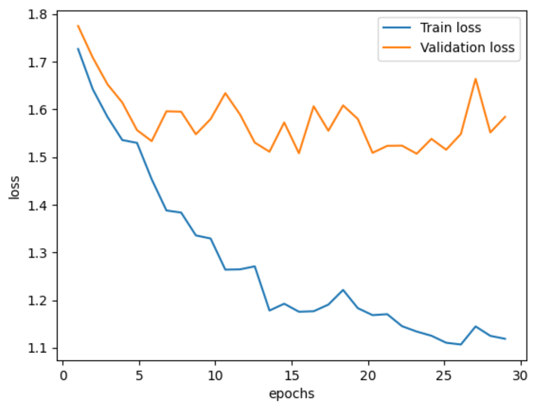
**(b) Fold 1**



**(c) Fold 2**

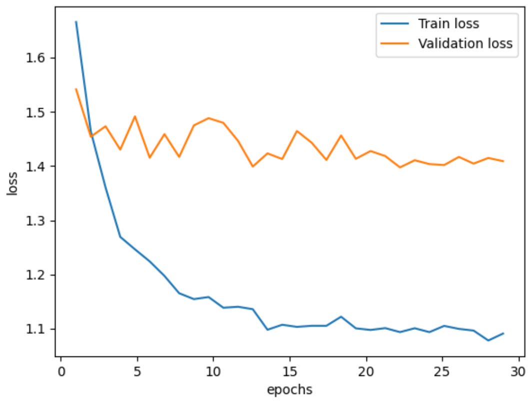


**(d) Fold 3**

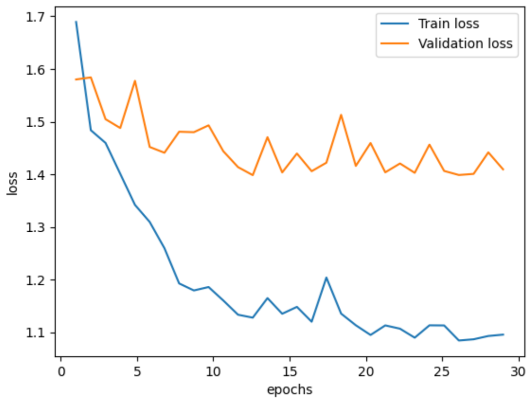


**(e) Fold 4**

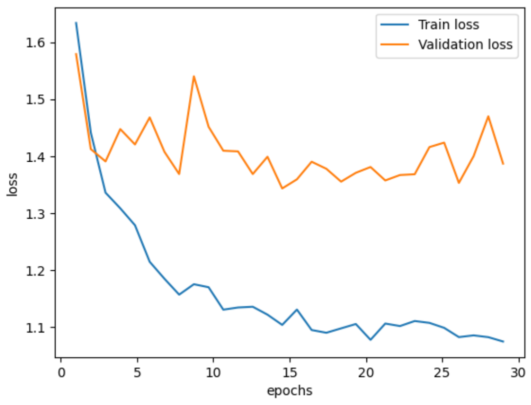
**Figure 17: The Train and Validation loss of every fold for the Body Encoding Stream with a learning rate of 1e-4**



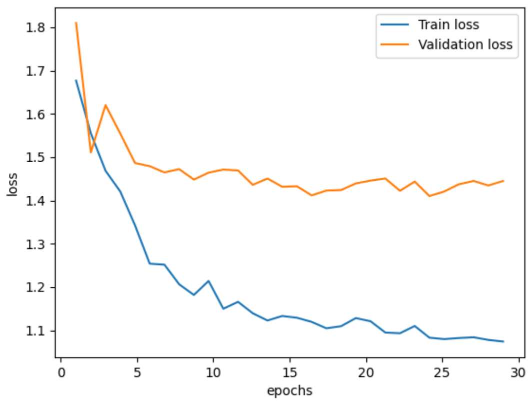
**(a) Fold 0**



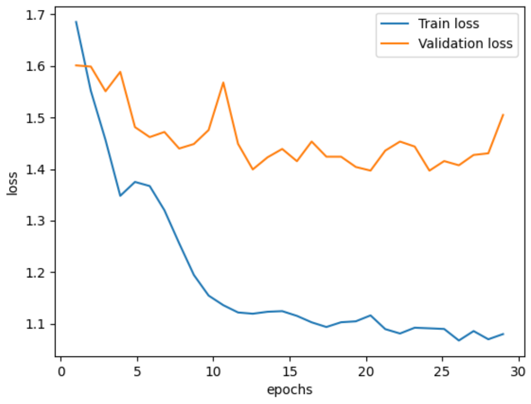
**(b) Fold 1**



**(c) Fold 2**

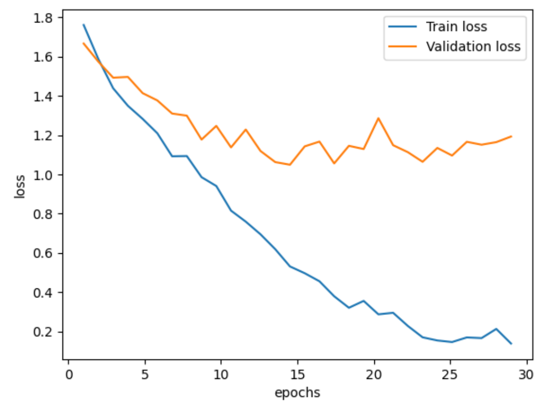


**(d) Fold 3**

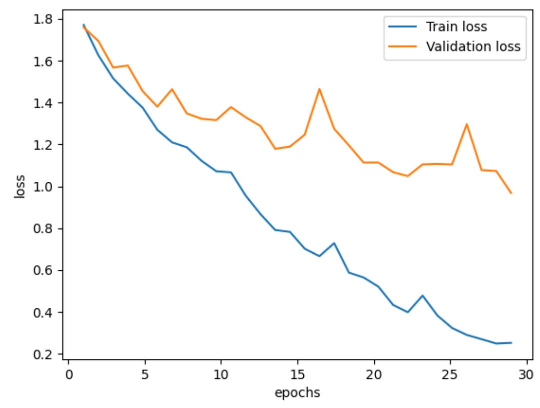


**(e) Fold 4**

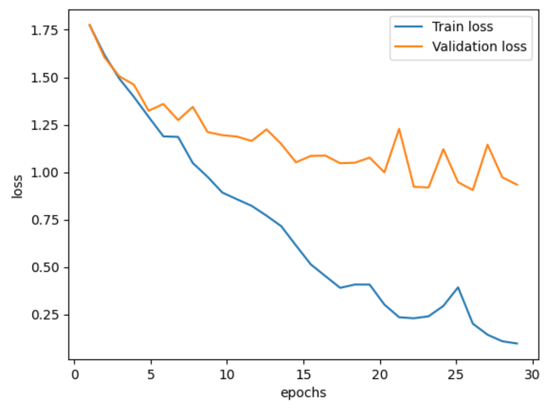
**Figure 18: The Train and Validation loss of every fold for the Context Encoding Stream with a learning rate of 1e-4 using the full frame shown in Figure 4 as input.**



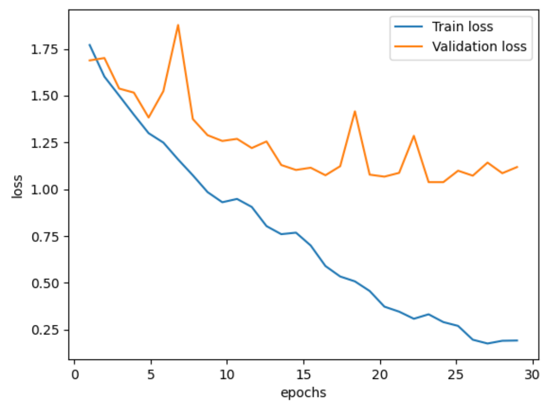
**(a) Fold 0**



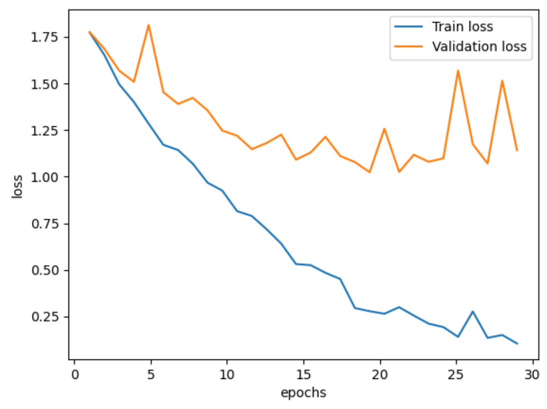
**(b) Fold 1**



**(c) Fold 2**



**(d) Fold 3**



**(e) Fold 4**

**Figure 19: The Train and Validation loss of every fold for the Main Architecture with a learning rate of 1e-4**

| Face Encoding Stream | | | | | | | |
|---|---|---|---|---|---|---|---|
| Accuracy validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 61% | 65% | 65% | 60% | 50% | 60% | 6% |
| Stranger | 37% | 23% | 66% | 63% | 47% | 47% | 18% |
| Service | 68% | 47% | 47% | 44% | 69% | 55% | 12% |
| Colleague | 100% | 92% | 90% | 93% | 100% | 95% | 5% |
| Parents-offs | 40% | 54% | 40% | 50% | 55% | 48% | 7% |
| Couple | 23% | 33% | 28% | 13% | 13% | 22% | 9% |
| Average fold | 55% | 52% | 56% | 54% | 56% | 55% | 1% |
| std | 25% | 22% | 20% | 24% | 26% | 22% | |
| F1-score validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.71 | 0.74 | 0.74 | 0.69 | 0.65 | 0.71 | 0.04 |
| Stranger | 0.32 | 0.27 | 0.37 | 0.45 | 0.31 | 0.34 | 0.07 |
| Service | 0.61 | 0.53 | 0.50 | 0.41 | 0.67 | 0.54 | 0.10 |
| Colleague | 0.83 | 0.67 | 0.92 | 0.90 | 0.82 | 0.83 | 0.10 |
| Parents-offs | 0.35 | 0.38 | 0.25 | 0.29 | 0.45 | 0.34 | 0.08 |
| Couple | 0.24 | 0.26 | 0.38 | 0.16 | 0.13 | 0.23 | 0.10 |
| Average fold | 0.51 | 0.48 | 0.53 | 0.48 | 0.51 | 0.50 | 0.02 |
| std | 0.22 | 0.19 | 0.23 | 0.25 | 0.23 | 0.21 | |
| Accuracy testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 58% | 61% | 57% | 58% | 58% | 58% | 2% |
| Stranger | 26% | 26% | 53% | 46% | 40% | 38% | 12% |
| Service | 56% | 36% | 20% | 60% | 68% | 48% | 20% |
| Colleague | 100% | 100% | 87% | 100% | 100% | 97% | 6% |
| Parents-offs | 22% | 11% | 44% | 11% | 33% | 24% | 14% |
| Couple | 20% | 40% | 6% | 13% | 20% | 20% | 13% |
| Average fold | 47% | 46% | 45% | 48% | 53% | 48% | 3% |
| std | 28% | 29% | 26% | 30% | 26% | 26% | |
| F1-score testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.73 | 0.71 | 0.67 | 0.70 | 0.72 | 0.71 | 0.02 |
| Stranger | 0.24 | 0.27 | 0.23 | 0.29 | 0.31 | 0.27 | 0.03 |
| Service | 0.44 | 0.37 | 0.24 | 0.51 | 0.55 | 0.42 | 0.12 |
| Colleague | 0.89 | 0.78 | 0.90 | 0.91 | 0.80 | 0.86 | 0.06 |
| Parents-offs | 0.24 | 0.13 | 0.38 | 0.18 | 0.35 | 0.26 | 0.11 |
| Couple | 0.15 | 0.26 | 0.10 | 0.13 | 0.19 | 0.17 | 0.06 |
| Average fold | 0.45 | 0.42 | 0.42 | 0.45 | 0.49 | 0.45 | 0.03 |
| std | 0.27 | 0.24 | 0.28 | 0.28 | 0.22 | 0.25 | |

**Table 5: The Accuracy and F1-score of every category and in every fold of the k-fold cross validation for the validation set and the testing set for the Face Encoding Stream. Furthermore, the average of the categories for every fold and the average of every label across the 5 folds with standard deviation are reported. Also, the average average across the 5 folds for the average of every fold is reported, also with standard deviation. Lastly, the standard deviation of the labels for every fold and the standard deviation the average of the labels over the folds is reported.**

| Context Encoding Stream | | | | | | | |
|---|---|---|---|---|---|---|---|
| Accuracy validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 76% | 61% | 74% | 59% | 59% | 66% | 8% |
| Stranger | 18% | 58% | 33% | 63% | 41% | 43% | 18% |
| Service | 56% | 61% | 64% | 50% | 65% | 59% | 6% |
| Colleague | 86% | 100% | 100% | 100% | 81% | 93% | 9% |
| Parents-offs | 50% | 9% | 0% | 50% | 44% | 31% | 24% |
| Couple | 29% | 33% | 28% | 0% | 60% | 30% | 21% |
| Average fold | 53% | 54% | 50% | 54% | 58% | 54% | 3% |
| std | 24% | 28% | 33% | 29% | 13% | 22% | |
| F1 validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.80 | 0.72 | 0.81 | 0.72 | 0.67 | 0.74 | 0.06 |
| Stranger | 0.25 | 0.38 | 0.27 | 0.37 | 0.39 | 0.33 | 0.07 |
| Service | 0.57 | 0.52 | 0.51 | 0.45 | 0.58 | 0.53 | 0.05 |
| Colleague | 0.87 | 0.96 | 1.00 | 0.91 | 0.84 | 0.92 | 0.07 |
| Parents-offs | 0.45 | 0.15 | 0.00 | 0.15 | 0.57 | 0.26 | 0.24 |
| Couple | 0.21 | 0.28 | 0.28 | 0.00 | 0.43 | 0.24 | 0.16 |
| Average fold | 0.53 | 0.50 | 0.48 | 0.43 | 0.58 | 0.50 | 0.05 |
| std | 0.25 | 0.27 | 0.34 | 0.31 | 0.15 | 0.25 | |
| Accuracy testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 63% | 62% | 66% | 54% | 64% | 62% | 5% |
| Stranger | 20% | 46% | 40% | 46% | 46% | 40% | 11% |
| Service | 72% | 60% | 72% | 48% | 52% | 61% | 11% |
| Colleague | 93% | 87% | 93% | 87% | 81% | 88% | 5% |
| Parents-offs | 11% | 11% | 11% | 22% | 11% | 13% | 5% |
| Couple | 33% | 26% | 6% | 0% | 20% | 17% | 14% |
| Average fold | 49% | 49% | 48% | 43% | 46% | 47% | 3% |
| std | 29% | 25% | 32% | 27% | 24% | 26% | |
| F1 testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.71 | 0.71 | 0.71 | 0.65 | 0.71 | 0.70 | 0.03 |
| Stranger | 0.22 | 0.30 | 0.27 | 0.23 | 0.36 | 0.28 | 0.06 |
| Service | 0.61 | 0.53 | 0.60 | 0.43 | 0.48 | 0.53 | 0.08 |
| Colleague | 0.91 | 0.93 | 0.94 | 0.76 | 0.90 | 0.89 | 0.07 |
| Parents-offs | 0.13 | 0.15 | 0.18 | 0.24 | 0.20 | 0.18 | 0.04 |
| Couple | 0.24 | 0.25 | 0.09 | 0.00 | 0.15 | 0.15 | 0.11 |
| Average fold | 0.47 | 0.48 | 0.47 | 0.39 | 0.47 | 0.45 | 0.04 |
| std | 0.29 | 0.27 | 0.31 | 0.26 | 0.27 | 0.28 | |

**Table 6: The Accuracy and F1-score of every category and in every fold of the k-fold cross validation for the validation set and the testing set for the Context Encoding Stream. Furthermore, the average of the categories for every fold and the average of every label across the 5 folds with standard deviation are reported. Also, the average average across the 5 folds for the average of every fold is reported, also with standard deviation. Lastly, the standard deviation of the labels for every fold and the standard deviation the average of the labels over the folds is reported.**

| Body Encoding Stream | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Accuracy validation set** | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 65% | 65% | 46% | 60% | 40% | 55% | 12% |
| Stranger | 31% | 47% | 46% | 27% | 23% | 35% | 11% |
| Service | 44% | 57% | 70% | 77% | 61% | 62% | 13% |
| Colleague | 93% | 92% | 100% | 100% | 100% | 97% | 4% |
| Parents-offs | 10% | 0% | 20% | 50% | 66% | 29% | 28% |
| Couple | 47% | 25% | 35% | 17% | 33% | 31% | 11% |
| Average fold | 48% | 48% | 53% | 55% | 54% | 52% | 3% |
| std | 26% | 29% | 26% | 28% | 26% | 24% | |
| **F1 validation set** | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.72 | 0.71 | 0.63 | 0.69 | 0.57 | 0.66 | 0.06 |
| Stranger | 0.20 | 0.37 | 0.38 | 0.32 | 0.29 | 0.31 | 0.07 |
| Service | 0.42 | 0.62 | 0.41 | 0.45 | 0.51 | 0.48 | 0.09 |
| Colleague | 0.85 | 0.75 | 0.87 | 0.83 | 0.84 | 0.83 | 0.05 |
| Parents-offs | 0.15 | 0.00 | 0.20 | 0.40 | 0.34 | 0.22 | 0.16 |
| Couple | 0.48 | 0.17 | 0.30 | 0.22 | 0.23 | 0.28 | 0.12 |
| Average fold | 0.47 | 0.44 | 0.47 | 0.49 | 0.46 | 0.46 | 0.02 |
| std | 0.25 | 0.28 | 0.22 | 0.21 | 0.21 | 0.22 | |
| **Accuracy testing set** | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 61% | 67% | 44% | 59% | 50% | 56% | 9% |
| Stranger | 33% | 26% | 40% | 6% | 20% | 25% | 13% |
| Service | 52% | 36% | 60% | 56% | 56% | 52% | 9% |
| Colleague | 93% | 100% | 100% | 100% | 100% | 99% | 3% |
| Parents-offs | 22% | 0% | 11% | 44% | 44% | 24% | 20% |
| Couple | 6% | 26% | 33% | 13% | 20% | 20% | 11% |
| Average fold | 45% | 43% | 48% | 46% | 48% | 46% | 2% |
| std | 28% | 32% | 27% | 31% | 27% | 27% | |
| **F1 testing set** | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.72 | 0.73 | 0.60 | 0.67 | 0.67 | 0.68 | 0.05 |
| Stranger | 0.29 | 0.19 | 0.29 | 0.08 | 0.18 | 0.21 | 0.09 |
| Service | 0.39 | 0.38 | 0.48 | 0.44 | 0.44 | 0.43 | 0.04 |
| Colleague | 0.94 | 0.86 | 0.82 | 0.78 | 0.86 | 0.85 | 0.06 |
| Parents-offs | 0.24 | 0.00 | 0.12 | 0.50 | 0.32 | 0.24 | 0.19 |
| Couple | 0.06 | 0.22 | 0.25 | 0.13 | 0.15 | 0.16 | 0.08 |
| Average fold | 0.44 | 0.40 | 0.43 | 0.43 | 0.44 | 0.43 | 0.02 |
| std | 0.30 | 0.30 | 0.24 | 0.26 | 0.26 | 0.26 | |

**Table 7: The Accuracy and F1-score of every category and in every fold of the k-fold cross validation for the validation set and the testing set for the Body Encoding Stream. Furthermore, the average of the categories for every fold and the average of every label across the 5 folds with standard deviation are reported. Also, the average average across the 5 folds for the average of every fold is reported, also with standard deviation. Lastly, the standard deviation of the labels for every fold and the standard deviation the average of the labels over the folds is reported.**

| Context Encoding Stream (full input image) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Accuracy validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 73% | 71% | 63% | 77% | 60% | 69% | 7% |
| Stranger | 43% | 47% | 60% | 36% | 47% | 47% | 9% |
| Service | 52% | 57% | 47% | 44% | 65% | 53% | 8% |
| Colleague | 100% | 100% | 100% | 100% | 100% | 100% | 0% |
| Parents-offs | 30% | 18% | 40% | 0% | 55% | 29% | 21% |
| Couple | 17% | 33% | 28% | 26% | 26% | 26% | 6% |
| Average fold | 53% | 54% | 56% | 47% | 59% | 54% | 4% |
| std | 27% | 26% | 23% | 33% | 22% | 25% | |
| F1 validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.82 | 0.78 | 0.77 | 0.78 | 0.70 | 0.77 | 0.04 |
| Stranger | 0.33 | 0.43 | 0.37 | 0.40 | 0.34 | 0.37 | 0.04 |
| Service | 0.52 | 0.50 | 0.40 | 0.33 | 0.61 | 0.47 | 0.11 |
| Colleague | 0.88 | 0.96 | 0.95 | 1.00 | 0.97 | 0.95 | 0.04 |
| Parents-offs | 0.35 | 0.27 | 0.21 | 0.00 | 0.56 | 0.28 | 0.20 |
| Couple | 0.17 | 0.23 | 0.40 | 0.31 | 0.25 | 0.27 | 0.09 |
| Average fold | 0.51 | 0.53 | 0.52 | 0.47 | 0.57 | 0.52 | 0.04 |
| std | 0.26 | 0.26 | 0.26 | 0.33 | 0.24 | 0.26 | |
| Accuracy testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 63% | 68% | 61% | 68% | 67% | 65% | 3% |
| Stranger | 40% | 33% | 60% | 20% | 46% | 40% | 15% |
| Service | 52% | 72% | 52% | 72% | 56% | 61% | 10% |
| Colleague | 100% | 93% | 87% | 93% | 93% | 93% | 5% |
| Parents-offs | 0% | 11% | 33% | 11% | 11% | 13% | 12% |
| Couple | 13% | 20% | 0% | 26% | 13% | 14% | 10% |
| Average fold | 45% | 50% | 49% | 48% | 48% | 48% | 2% |
| std | 33% | 30% | 27% | 31% | 29% | 29% | |
| F1 testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.73 | 0.75 | 0.70 | 0.71 | 0.74 | 0.73 | 0.02 |
| Stranger | 0.28 | 0.29 | 0.34 | 0.23 | 0.31 | 0.29 | 0.04 |
| Service | 0.47 | 0.60 | 0.51 | 0.58 | 0.56 | 0.54 | 0.05 |
| Colleague | 0.97 | 0.97 | 0.88 | 0.97 | 0.97 | 0.95 | 0.04 |
| Parents-offs | 0.00 | 0.20 | 0.30 | 0.14 | 0.18 | 0.16 | 0.11 |
| Couple | 0.11 | 0.16 | 0.00 | 0.25 | 0.11 | 0.13 | 0.09 |
| Average fold | 0.43 | 0.50 | 0.46 | 0.48 | 0.48 | 0.47 | 0.03 |
| std | 0.34 | 0.30 | 0.29 | 0.30 | 0.31 | 0.30 | |

**Table 8: The Accuracy and F1-score of every category and in every fold of the k-fold cross validation for the validation set and the testing set for the Context Encoding Stream using the full frame shown in Figure 4 as input. Furthermore, the average of the categories for every fold and the average of every label across the 5 folds with standard deviation are reported. Also, the average average across the 5 folds for the average of every fold is reported, also with standard deviation. Lastly, the standard deviation of the labels for every fold and the standard deviation the average of the labels over the folds is reported.**

| Main architecture | | | | | | | |
|---|---|---|---|---|---|---|---|
| Accuracy validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 80% | 62% | 44% | 66% | 59% | 62% | 13% |
| Stranger | 25% | 70% | 46% | 50% | 35% | 45% | 17% |
| Service | 36% | 61% | 52% | 44% | 65% | 52% | 12% |
| Colleague | 100% | 100% | 100% | 100% | 100% | 100% | 0% |
| Parents-offs | 60% | 27% | 20% | 50% | 44% | 40% | 16% |
| Couple | 29% | 8% | 35% | 34% | 33% | 28% | 11% |
| Average fold | 55% | 55% | 50% | 57% | 56% | 55% | 3% |
| std | 28% | 30% | 25% | 21% | 23% | 23% | |
| F1 validation set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.78 | 0.73 | 0.61 | 0.74 | 0.69 | 0.71 | 0.06 |
| Stranger | 0.27 | 0.47 | 0.31 | 0.42 | 0.29 | 0.35 | 0.09 |
| Service | 0.44 | 0.52 | 0.55 | 0.46 | 0.69 | 0.53 | 0.10 |
| Colleague | 0.94 | 0.96 | 0.91 | 0.94 | 0.94 | 0.94 | 0.02 |
| Parents-offs | 0.48 | 0.32 | 0.09 | 0.22 | 0.31 | 0.28 | 0.14 |
| Couple | 0.29 | 0.08 | 0.24 | 0.34 | 0.27 | 0.24 | 0.10 |
| Average fold | 0.53 | 0.51 | 0.45 | 0.52 | 0.53 | 0.51 | 0.03 |
| std | 0.25 | 0.28 | 0.27 | 0.25 | 0.26 | 0.25 | |
| Accuracy testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 66% | 62% | 38% | 66% | 66% | 60% | 12% |
| Stranger | 20% | 40% | 40% | 26% | 26% | 30% | 9% |
| Service | 48% | 56% | 60% | 44% | 52% | 52% | 6% |
| Colleague | 100% | 100% | 93% | 75% | 100% | 94% | 11% |
| Parents-offs | 44% | 33% | 55% | 33% | 22% | 37% | 13% |
| Couple | 33% | 33% | 40% | 40% | 33% | 36% | 4% |
| Average fold | 52% | 54% | 54% | 47% | 50% | 51% | 3% |
| std | 26% | 23% | 19% | 18% | 27% | 21% | |
| F1 testing set | | | | | | | |
| | fold 0 | fold 1 | fold2 | fold 3 | fold 4 | Average Label | std |
| Friend | 0.71 | 0.73 | 0.56 | 0.74 | 0.72 | 0.69 | 0.07 |
| Stranger | 0.18 | 0.30 | 0.31 | 0.18 | 0.21 | 0.24 | 0.06 |
| Service | 0.52 | 0.50 | 0.52 | 0.44 | 0.52 | 0.50 | 0.03 |
| Colleague | 0.97 | 0.94 | 0.83 | 0.86 | 0.94 | 0.91 | 0.06 |
| Parents-offs | 0.33 | 0.35 | 0.43 | 0.40 | 0.31 | 0.36 | 0.05 |
| Couple | 0.29 | 0.29 | 0.24 | 0.30 | 0.27 | 0.28 | 0.02 |
| Average fold | 0.50 | 0.52 | 0.48 | 0.49 | 0.50 | 0.50 | 0.01 |
| std | 0.27 | 0.24 | 0.19 | 0.24 | 0.26 | 0.24 | |

**Table 9: The Accuracy and F1-score of every category and in every fold of the k-fold cross validation for the validation set and the testing set for the whole architecture. Furthermore, the average of the categories for every fold and the average of every label across the 5 folds with standard deviation are reported. Also, the average average across the 5 folds for the average of every fold is reported, also with standard deviation. Lastly, the standard deviation of the labels for every fold and the standard deviation the average of the labels over the folds is reported.**