Real-Time Face Detection, Identification and Emotion Analysis for Affective Human Robot Interaction

Anders Skibeli Rokkones

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2018

Real-Time Face Detection, Identification and Emotion Analysis for Affective Human Robot Interaction

Anders Skibeli Rokkones

Real-Time Face Detection, Identification and Emotion Analysis for Affective Human Robot Interaction

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

In this thesis an approach for real-time face detection, identification and emotion recognition is presented, using robust face detectors and convolutional neural networks (CNN). It is tested on a standard laptop computer without the use of Graphical Processing Units (GPUs) and is to be implemented on an autonomous robot stationed in elderly users' homes. In the real-time emotion recognition pipeline it is shown that a small CNN trained on a personalized dataset (images from a single user) is able to find faces, identify the user and classify seven different emotions in real-time, contrary to related work that processes data offline with fewer classes to predict. The use of a personalized dataset reduces training time and increases robustness since a personalized model is known to be more capable of learning person dependent features, as substantiated in the results. Offline testing produces a top recognition rate of 98% and a real-time recognition rate of 83% of seven different expressions. Performance is expected to drop when introduced to a real-time testing environment given more noise and variations. A comprehensive comparison of existing pre-trained face detectors for fast classification is also presented, showing that a Haar-cascade classifier, a Local-Binary-Pattern-cascade classifier and a Histogram-Oriented-Gradient classifier all pose as suitable options for real-time use.

In addition, offline experimentation with edge features in video combined with recurrent neural networks (RNN) and exploration of benefits and drawbacks of using sequential data for classifying facial expressions is presented. This includes the use of Kirsch masks and Local Directional Strength Pattern as features presented as LDSP-RNN, achieving an average recognition rate of 73.6% on a fairly difficult dataset. It is compared with more traditional image classifiers such as CNN and Soft Vector Machines(SVM) for classification where it is proven to outperform SVMs (72% mean accuracy) and produce a more robust result than RNN with raw grayscale images as input (72.8% mean accuracy).

Acknowledgements

I am very grateful to Md. Zia Uddin and Jim Torresen for their guidance and help during my work on this thesis, and the work they have performed prior to my thesis, giving me the opportunity to do a study within this field of healthcare and image analysis. Special thanks to my partner Mari Junker and my parents Venke Skibeli and Erik Rokkones for support combined with writing and spelling corrections. Vetle Bu Solgaard, Bjoern-Ivar Teigen and Martin Hovin deserve acknowledgments for letting me use them as test subjects along with support and encouragement; this applies to all my co-students in the Robotics Lab at the Department of Informatics. Also a great appreciation to Epigram AI for letting me use their servers for some of the experiments.

List of Abbreviations

- **CNN** - Convolutional Neural Network
- **RNN** - Recurrent Neural Network
- **SVM** - Soft Vector Machine
- **FCN** - Fully Connected Network
- **HRI** - Human Robot Interaction
- **RGB** - Red, Green and Blue
- **LDSP** - Local Directional Strength Pattern
- **LDP** - Local Directional Pattern
- **LBP** - Local Binary Pattern
- **HOG** - Histogram Oriented Gradients
- **LDRHP** - Local Directional Rank Histogram Pattern
- **LDSP-RNN** - Local Directional Strength Pattern and Recurrent Neural Network
- **RAW-IMAGE-RNN** - Raw Grayscale Image and Recurrent Neural Network
- **CCE** - Categorical Cross Entropy
- **ReLU** - Rectified Linear Units
- **LDA** - Linear Discriminant Analysis
- **GAN** - Generative Adversarial Network
- **CK+** - Cohn-Kanade Extended Public Facial Expression Dataset
- **AU** - Action Unit
- **IoU** - Intersection over Union
- **STD** - Standard Deviation
- **GPU** - Graphical Processing Unit
- **FPS** - Frames Per Second

List of Figures

List of Tables

Contents

Chapter 1

Introduction

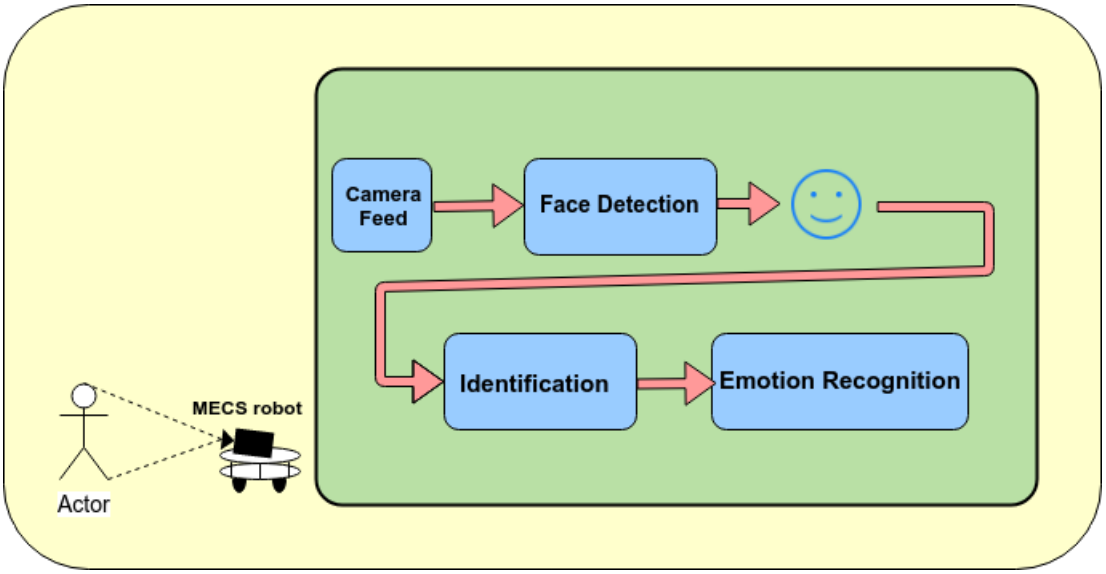


Figure 1.1: Illustration of information flow through the system.

1.1 Motivation

The welfare of seniors has always been an essential part of society. By introducing new technology and applications, the people involved will be more capable of managing their own life without too much help from the public and private health sectors.

The Multimodal Elderly Care Systems *(MECS project)* aims at making the daily life more comfortable for the elderly and at the same time assist healthcare personnel by providing additional information about the user. This thesis presents an approach for implementing an application which can analyze emotions based on facial expressions of the user. This application will be part of a mobile robot stationed in the users' home where it also will perform other tasks like tracking movement, respiration, heart rate and analyze other features associated with the user. If any anomalies are detected, a specified response will be actioned to ensure that the users' health is exposed to minimum risk, making this robot an autonomous safety alarm without the need of attachments on the users themselves.

The primary goal of the thesis is to track the facial expression in real-time, something that may be computationally hard considering the hardware limit-

ations of the mobile robot. Different types of sensors can be combined to get the optimal representation of the users' face, where depth-, thermographic- and a regular RGB-camera pose as the best options for face tracking and feature extraction. This may also introduce implications when it comes to privacy in contrast to digital information (i.e., resolution, colour, etc.). Depth- and thermal cameras eliminate the privacy issues, but on the other hand, information that might be vital to obtain a proper classification will be lost. RGB-cameras provide more detailed images for analysis and are, therefore, the selected type of sensor utilized for training and testing of the presented real-time emotion analysis system.

1.2 Research Questions

To get a better understanding of the goal of the thesis, a series of three research questions are defined and listed below. Each of the questions are constructed so that the implementation and testing of chapters 4 and 5 are focused on answering all of them in the best way possible.

1. *Is it possible to process facial expressions at a reasonable speed in real-time on a standard laptop computer, without the help of a GPU?*
2. *How well does a personalized model perform compared to a generalized model?*
3. *Can gradient oriented features in sequence combined with an RNN pose as an alternative to a standard CNN?*

Figure 1.1 illustrates the flow of information through the pipeline. Focus will be put into the emotion recognition part, considering it is the central part of the pipeline. Extensive research have been done on face detection and identification in computer vision, leaving it hard to outperform existing implementations, see sections 2.4 and 2.5. Comparing and discussing benefits and drawbacks of existing implementations within these fields will help to decide on what would improve implementation of the pipeline the most.

1.3 Contributions

This section gives an overview of different contributions related to this work.

- The first contribution is an approach for real-time emotion classification with the use of small customized convolutional neural networks trained on a personalized dataset for classifying seven different emotions which is presented in chapter 4. Included in this contribution is a comprehensive comparison of available pre-trained face detectors for real-time applicability.
- In the following chapter (chapter 5) a combination of features in sequences based on Kirsch edge detection and Local Directional Strength Pattern[1] and a recurrent neural network for emotion classification are presented. The findings state that the LDSP-RNN approach produces a more stable and faster learning progression compared with a sequence of raw image values.

1.4 Limitations

The public dataset, Cohn-Kanade+[2], used in some of the experiments has shown deficiencies in the sense of a few miss-labeled samples and quite significant variations in class representations. This may reduce the ability for a CNN to learn the proper and generalized features required for a satisfying outcome. The problems related to this dataset limits the ability to generalize the results presented in this thesis, but it presents a basis of which improvements are possible with the use of different datasets. Finding a new dataset for experiments was considered to be work beyond the time frame associated with this thesis.

1.5 Structure of the Thesis

The majority of figures and tables are made specifically for this thesis. If not, references to original figures or tables are provided in the captions of the respective figures or tables.

The thesis is structured to reflect the investigation of finding answers to the research questions presented in section 1.2.

- Chapter 2 provides an overview of previous work related to each chapter, as well as techniques to help improve different deep learning architectures.
- Chapter 3 provides details about the datasets used in the experiments.
- Chapter 4 gives an approach for solving real-time face detection identification and emotion recognition. Results are presented in section 4.2.
- Chapter 5 explores a new way of analyzing facial expressions in videos, with the use of edge features and Local Directional Strength Pattern[1].
- Chapter 6 discusses the findings from chapter 4 and chapter 5.
- Chapter 7 provides a conclusion based on results from section 4.2 and section 5.2, with answers to the research questions presented in 1.2, followed by future work in section 7.4.

Chapter 2

Background

2.1 Human Robot Interactions

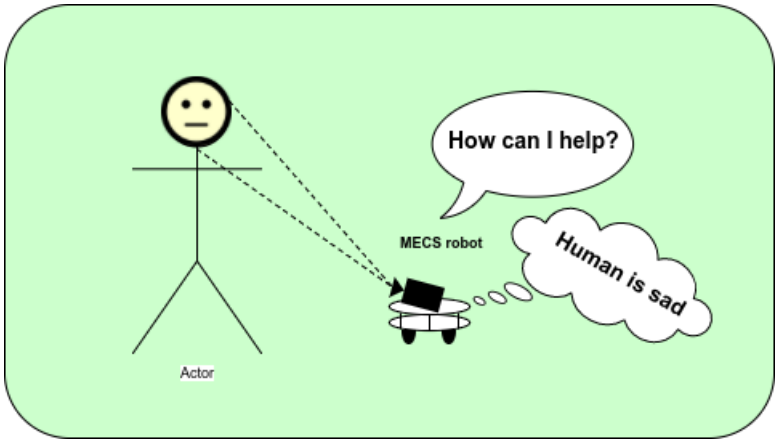


Figure 2.1: Illustration of possible use

Humans have multiple ways of displaying their feelings and they differ from person to person and culture to culture. Facial expressions have a significant place when it comes to communicating with one another and in some way remove the problem of misunderstandings when conveying a message. Digital communication is a big part of peoples lives, by the sending of e-mails and chatting on social media. In this form of communication the limited ability to put a facial expression behind the message can be troublesome. This is giving rise to problems which most likely would not be an issue if the conversation was held face to face. The introduction of emojis (small figures of faces etc.) is enabling people to put a facial expression to their message, reducing the possibility of misunderstandings.

The same principle is relevant when it comes to communication between humans and robots. With the advancements in technology, especially autonomous robots which interact with people, a good understanding between robot and human must be solved to enable further accomplishments. In the last twenty years machine learning and artificial intelligence have been the center of attention in computer science, resulting in a better environment for significant progress in the human-robot interaction domain. There are a lot of different aspects of this domain such as natural behavior, optimal path and following basic norms. For autonomous robots and other assisting applications, communication and understanding are essential factors to achieve results surpassing pre-programmed responses to different scenarios. Human faces is a very complicated part of the body, consisting of multiple tiny

muscles that together can form a wide specter of expressions. By enabling robots to use sensors like microphones and cameras, they will be given access to the domain which also humans are dependent of to interpret one another.

2.2 Feature Extraction

Features, or data descriptors, is a big part of image analysis. Different methods can be used to extract information within images beyond standard RGB-values (Red, Green, Blue) or grayscale values. In this section, an overview is given of methods commonly used in face detection and emotion classification[1, 3, 4].

Kirsch Edge Detection

Kirsch edge detection is a mask that considers edge responses in all eight directions around a single pixel. It is done by applying eight separate filters with values specifically to highlight edges oriented in the specified orientation[5]. See figure 2.2.

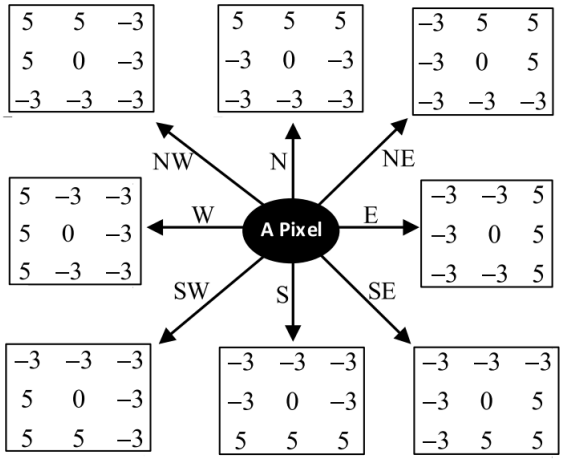


Figure 2.2: Kirsch edge masks in eight directions[1].

2.2.1 Local Binary Pattern

Local Binary Pattern (LBP) was originally designed for texture description. LBP assigns a binary label to every pixel of an image by comparing the center pixel of a 3x3 neighborhood. If the surrounding pixels are larger than the center pixel, it assigns 1, if smaller it assigns 0 [3]. The surrounding pixels, when considered in all 8 directions, produce an 8-bit representation of the area (3x3) as illustrated in figure 2.3. The operator is scanned across all pixels of the image, producing a binary representation for all the pixels converting this into a decimal representation. The result is a 1-dimensional vector or histogram of the entire image providing a texture description of the respected image.

$$LBP(x_c, y_c) = \sum_{n=0}^{7} s(i_n - i_c)2^n \tag{2.1}$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$

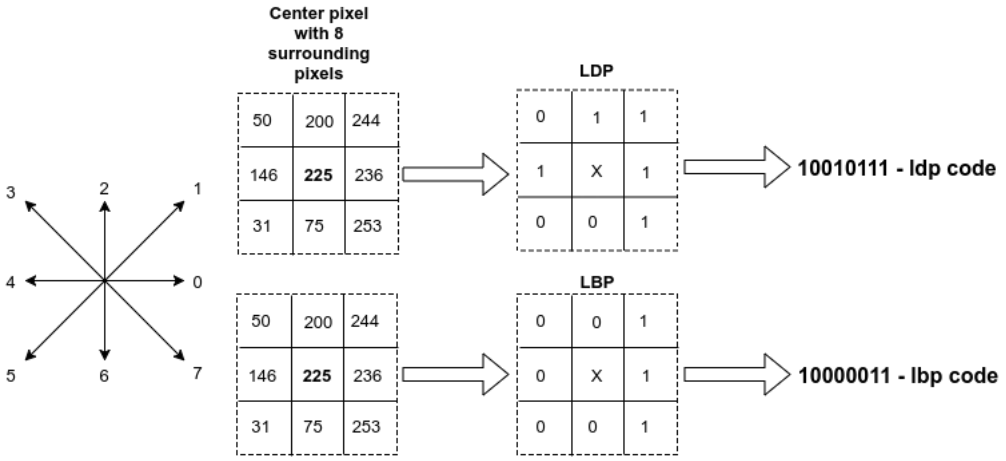


Figure 2.3: LDP vs. LBP on raw grayscale pixels.

2.2.2 Local Directional Pattern

Local Directional Pattern (LDP) compared to LBP, gives a better description of the edges. This is due to the selection of the top n absolute values provided by the Kirsch mask (see section 2.2) in all eight directions

around the center pixel contrary to LBP that only checks if the surrounding pixels are larger or smaller than the center pixel. LDP was proposed by Jabid, Kabir and Chae in 2010 as a local feature descriptor for object recognition[5]. In 2012 Jabid, Kabir and Chae proposed LDP as a feature descriptor for face recognition as well, where they described the advantages of using LDP instead of LBP. LDP provides more consistency in the presence of noise since the edge response magnitude is more stable than pixel intensities[4]. As we can see in figure 2.3, LDP gives a more detailed description of the edge response on a raw grayscale image than LBP.

$$LDP(x_c, y_c) = \sum_{n=0}^{7} s(abs(i_n))2^n \tag{2.3}$$

$$s(x) = \begin{cases} 1, & topN(x) = True \\ 0, & topN(x) = False \end{cases} \tag{2.4}$$

2.2.3 Local Directional Strength Pattern

Local Directional Strength Pattern (LDSP) is an evolved version of LDP which outputs a 6-bit representation of the surrounding 8-pixels. Instead of considering the top n absolute values of the surrounding pixels, it looks at the maximum- and minimum value. This means that it will get the strength of the edge and in which direction this edge is pointing. LDSP was first proposed in "Facial expression recognition using salient features and convolutional neural network" by Uddin, Khaksar and Torresen[1] as a feature descriptor of emotions in depth images. LDP considers only absolute values of edge strength of a pixel, and can result in the generation of equal patterns for two different types of edge pixels. LDSP can overcome this and produce more robust patterns than LDP, see figure 2.4.

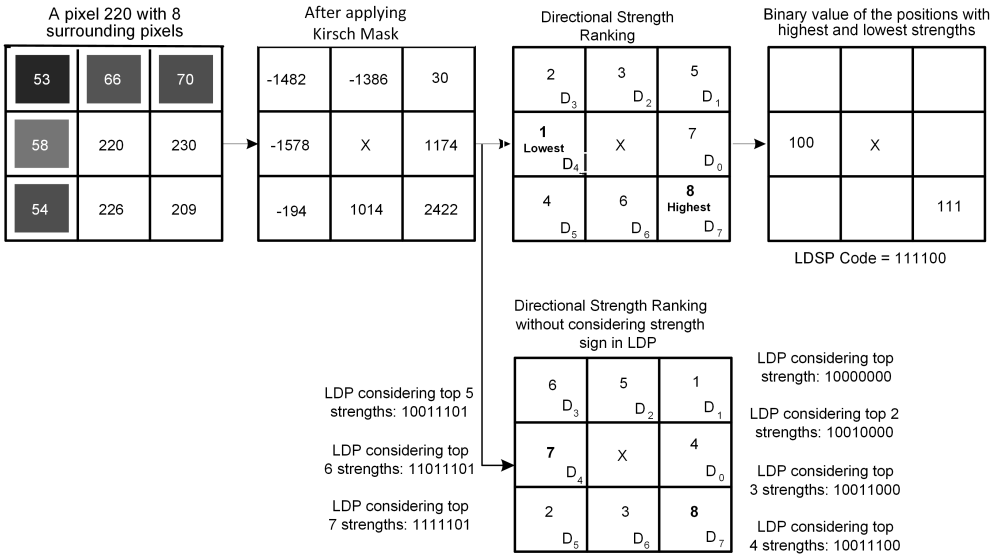
$$LDSP(x_c, y_c) = \sum_{n=0}^{7} L_n \times 2^n \tag{2.5}$$

$$L = binary(Arg(h)) \;||\; binary(Arg(l)) \tag{2.6}$$

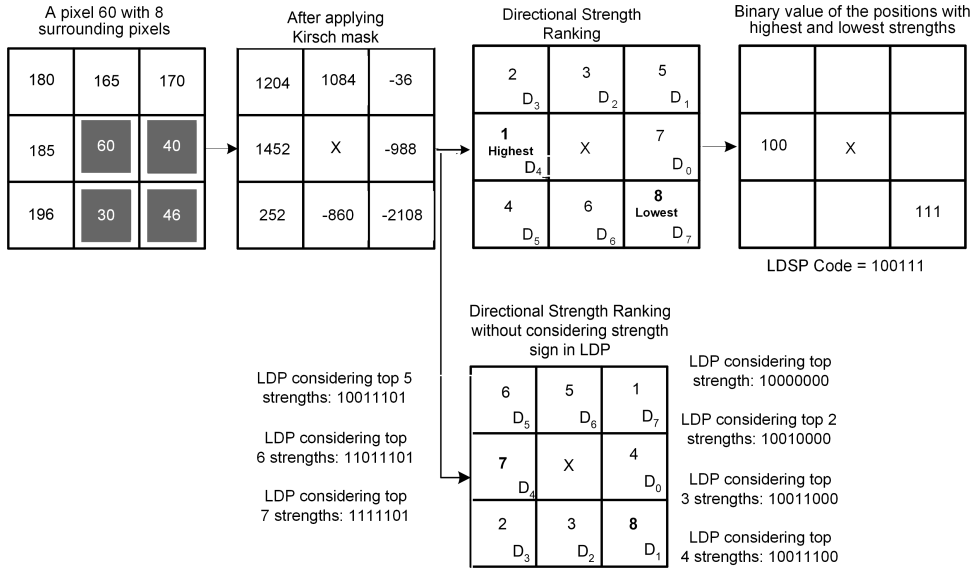
h is the highest value of the surrounding n neighborhood pixels and l is the lowest. Their location *Arg()* is then converted into a binary representation and combined to form a 6-bit equivalent of their location(Arg) around the

center pixel, where h is the three left bits and l is the three right bits, essentially making the highest neighboring pixel the most significant bits.

In figure 2.4 it is shown that LDSP is producing separate edge representations for different edges, contrary to LDP that produces the same edge representation for both edges. The figure is partially copied from Uddin, Khaksar and Torresen's paper[1].



(a) **First edge**



(b) **Second edge**

Figure 2.4: LDSP vs. LDP on Kirsch edge values[1].

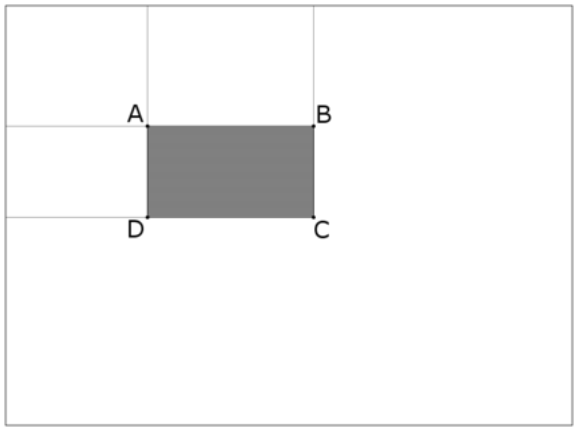


Figure 2.5: The integral image [8]

2.2.4 Haar-like Features

Haar-like features are digitally computed features used for object detection. At the beginning of digital image analysis, when considering working with only RGB-values, it was considered hard to perform complex operations. Papageorgiou published a paper discussing the possibilities working with features inspired by Haar-wavelets[6] instead of raw RGB-values [7]. Viola & Jones was inspired by this idea and proposed Haar-like features.

Haar-like features are based on dividing the image into regions and calculating the sum of the pixels within these regions. By comparing the results of neighboring regions it is possible to describe objects in the image. Haar-like features are considered fast because of the use of the *Intergral Image* which works as a table of reference points equal to the size of the image making it easy to calculate these features between two regions (see figure 2.5). Haar-like features are used to train several weak classifiers and boosted with AdaBoost to form the Haar-cascade classifier as discussed in chapter 2 section 2.4.

2.2.5 Histogram Oriented Gradients

Histogram Oriented Gradients (HOG) features is represented as a single feature vector opposed to a set of feature vectors. It does so by computing the gradients of a sub-window in the image and its orientations, dividing the directions into 9 bins from 0 to 180 degrees with 20° intervals for each bin,

[0°-20°, 20°-40°, ..., 160°-180°]. The histogram is spiked where the prominent gradients are oriented. The calculated feature vector is then fed through a classifier, usually a Soft Vector Machine(SVM) which decides if there is a face inside the sub-window or not.

Dalal and Triggs who first proposed the use of HOG-features in person detection, argued that normalized *Histogram Oriented Gradients* performed excellently compared with existing feature descriptors like Haar wavelets, and managed to prove that HOG reduced false positive rates by a magnitude relative to the best Haar-wavelet-based descriptors (i.e., Haar-like features)[9].

2.3 Machine Learning for Image Analysis

2.3.1 Deep Learning

The definition of deep learning varies somewhat between experts, but one thing is certain; it is based on the knowledge of how the human brain works with an intricate network of neurons cooperating to understand the world around it. An input vector, a hidden vector and an output vector are known to many as a Multi-layered Perceptron. By expanding the number of hidden layers the network can solve more complex problems and to some extent get a deeper understanding of the underlying structure within the dataset. This is one of the reasons why it is called deep learning.

Deep learning models have substantially improved multiple domains within computer science such as speech recognition, object detection, object classification and many other. Deep learning has made advances were problems in the artificial intelligence community have resisted the best attempts at solving them with traditional methods[10].

2.3.2 Supervised Learning

Supervised learning is a form of machine learning where the answers to each data point in the training set are known. This enables the model to get familiar with the problem in such a way that it can perform well on new unseen data. By knowing the correct answer to each input, the model can adjust itself to the point where the error between the predicted value and

actual value is minimized. This technique is widespread when it comes to classifying images. CNNs are a form of supervised learning architecture and has shown great performance in classifying images, object detection and as a feature extractor for further classification[1, 11, 12, 13, 14].

Convolutional Neural Networks for Feature Extraction

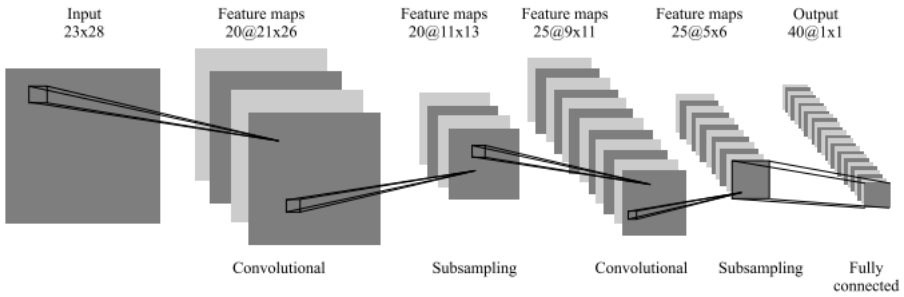


Figure 2.6: Typical CNN architecture[13].

"..manually acquiring some facial features from face images is relatively difficult, while CNN could extract effective facial features automatically." [14].

CNNs can find complex features in the raw image which manually would be very difficult. By training the CNN on multiple images, it is able to extract complicated features and use them as descriptors for further classification. In two dimensional convolution, a filter of shape (NxN) slides over the image resulting in a feature map representation of the input image. Each filter consists typically of uniform initialized values. During training of a CNN these values are changed through backpropogation[15] based on the gradient of the output error and ultimately they converge on the optimal filter values to find suitable features that represent the image. CNNs usually consist of multiple layers of convolution where each layer applies multiple numbers of filters where each filter is trained to find different features, for example edges, eyebrows, mouth, hair, color etc. The deeper the layer, the more complicated the features are[13, 14]. Between each convolution layer sub-sampling is used. This is to reduce the number of trainable parameters which easily can grow quite large. Two different methods are customarily applied, *Max Pooling and Average Pooling.*

"Pooling enhances the robustness to the variations of images. Such as rotation, noise and distortion. It also reduces the dimensions of the output and reserves the notable features."[14]

This is one of the reasons why CNNs is preferred when it comes to image processing. Figure 2.6 is a simple illustration of a convolutional neural network.

2.3.3 Classification

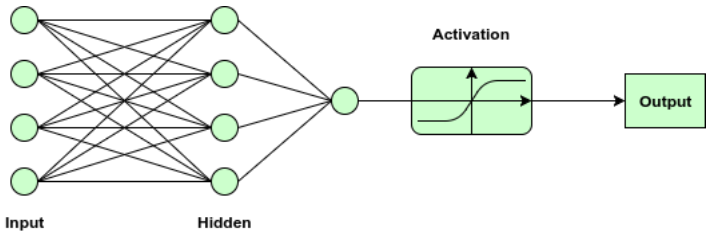


Figure 2.7: Simple illustration of a fully connected network

After good features have been extracted from the image, only a matter of classification remains. There are numerous different classification approaches such as Soft Vector Machines, K-Nearest Neighbours, decision trees, etc. However, this section focus on probably the most commonly used classifying method when it comes to deep learning. A fully connected network (FCN), also known as a multilayered perceptron, can be used for classification. The input to the FCN is a flattened vector representation of the extracted features from the images, as illustrated in figure 2.6. This vector is then fed through the FCN producing values at the output nodes. The number of output nodes is equal to the number of classes to predict, but for a binary classification problem only one node is typically used. Every node in the network is subjected to an activation function that forces a node to "fire" if presented with a particular input which is inspired by how the brain works. The Sigmoid function is an example of such a function applied for a binary classification problem on the single output node. The Rectified Linear Units Function[16] *(ReLU)* is more commonly used in the rest of the network because it avoids gradient clipping of nodes with high values. As we can see in the function below, high values for x will result in $f(x) = 1$ and the gradient is then equal

to 0 which does not contribute to the learning of the network.

$$f(x) = \frac{1}{1 - e^{(-x)}} \tag{2.7}$$

The Sigmoid function "squashes" the value of the output node to be between 0 and 1, making it a probabilistic function that states the probability of the input belonging to class A or class B.

$$class = \begin{cases} A, & \text{if } f(x) \geq 0.5 \\ B, & \text{otherwise} \end{cases} \tag{2.8}$$

Between the layers of nodes are weighted connections called "weights". These weights are multiplied with the input to produce the next layer of nodes, connecting each node in a layer to every node in the previous layer (see figure 2.7). After a forward pass through the network, the weights are adjusted with backpropagation based on the gradient of the error function (example Mean Squared Error[17]) at the output, and this process is repeated until the changes of the weights have converged and hopefully resulted in good classification accuracy.

2.3.4 Improving Convolutional Neural Networks

Through the development of deep neural networks over the recent years different techniques have been explored to stabilize weights with regularization, reduce overfitting and generalization of the network during training.

A standard convolutional neural network consists of N convolution layers followed by sub-sampling and an activation function with a fully connected network at the output. The subsequent section gives a description of techniques which have improved the performance of deep CNNs over the recent years and gives an understanding of why the architecture used in the experiments was chosen.

Batch Normalization

Batch normalization was introduced by S. Ioffe and C. Szegedy in 2015 as a normalization layer in convolutional neural networks to reduce the number of training steps by decreasing *Internal Covariate Shift*, a change in distributions of internal nodes of a deep neural network during training[18]. Batch normalization also has a positive influence on the gradient flow through the network during backpropagation,

"... by reducing the dependence of gradients on the scale of the parameters or of their initial values.."

making it possible to use a higher learning rate with a reduced risk of divergence, as described in their paper[18].

This is done on a subset of training samples called *mini-batches*. Mini-batches or batches is useful in many ways. For example, when calculating the gradient of the loss on a mini-batch, you get an estimation of the gradient over the entire training set[18]. It also gives the advantage of parallel computations compared with single data point examples. Iofee and Szegedy's work has inspired many adaptations of batch normalization, such as *Layer Normalization*[19] for use in recurrent neural networks and *Adaptive Batch Normalization*[20] to increase generalization of deep neural networks.

Mean and variance is calculated for each of the mini-batches and used to normalize the activations. To avoid limiting what a layer represents, Ioffe and Szegedy introduced two parameters for each activation $x^{(k)}$ in a layer, γ^k and β^k, to scale and shift the normalized values.

"... normalizing the inputs of a sigmoid would constrain them to the linear regime of the nonlinearity"[18].

These parameters are trained along with the weights and other trainable parameters in the network. For CNNs the parameters γ^k and β^k are learned for each feature map in a layer instead of each activation.

In algorithm 1 the procedure to calculate a batch normalization for a mini-batch is described as in Ioffe and Szegedy's paper. Epsilon(ϵ) is a small constant that avoids the scenario of dividing by zero.

In their paper they also argue that with the use of Batch Normalization the need of Dropout is reduced, which leads us to the next section.

Algorithm 1 Batch Normalization Transform[18]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$; Parameters to be trained: γ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ ▷ mini-batch mean

$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$ ▷ mini-batch variance

$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$ ▷ normalize

$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ ▷ scale and shift

Dropout

Dropout was introduced by Srivastava et al. in 2014 as a regularization technique to reduce overfitting in large deep neural networks[21]. Overfitting is the result of deep neural networks that contain many hidden layers and is capable of learning complex relationships between their inputs and outputs, learning sampling noise which is only present in the training set and not in the test set even though it is drawn from the same distribution[21]. This means the network has become too familiar with the data it is training on resulting in poor performance when introduced with new unseen data. A layer of neurons will learn what to expect as output from its previous layer making it vulnerable to small changes. To overcome this dropout was introduced.

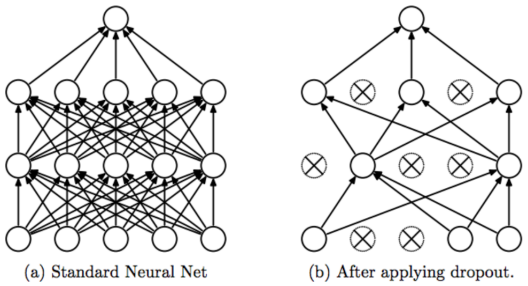


Figure 2.8: Dropout illustration from Srivastava et al. paper [21]

Dropout is a term that refers to "dropping" a unit or node either in hidden or visible layers in a neural network. Essentially thinning the network, as shown in figure 2.8[21]. The dropout rate is the probability of a neuron "dy-

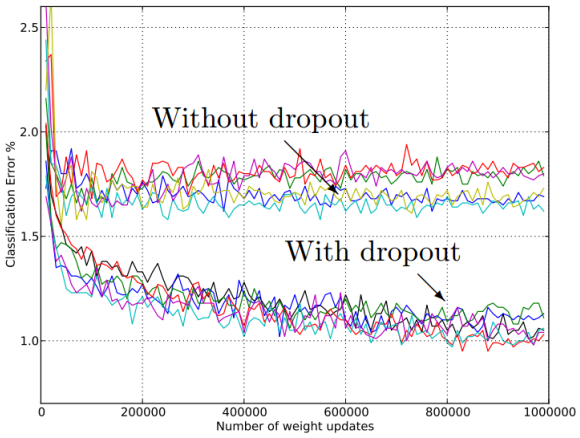


Figure 2.9: Dropout improvement plot from Srivastava et al. paper [21]

ing" in a specified layer, essentially producing a node-output equal to zero. This means that a neuron in the following layer will never learn with certainty what it will receive as input, drastically reducing overfitting. Dropout has become a standard when implementing deep neural networks, underlining the quite positive effect dropout has introduced to the deep learning domain. Figure 2.9 illustrates of how much networks without dropout is improved with the use of dropout. It is tested on different architectures where each is subjected to both cases.

Loss Functions

Categorical Cross Entropy(CCE) is one of the most commonly used loss functions for classification of multiple class labels and in many cases outperform other loss functions such as *Squared Error* [22, 23, 24].

CCE is a measurement of entropy between two probability distributions. Considering *Softmax* as the activation function on the output nodes and one-hot encoded labels, CEE would make a preferable choice as loss function.

This function describes the mean cross entropy with batch m:

$$\mathcal{L} = \frac{1}{m} \sum_{i}^{m} -y_i \times log(\hat{y}_i) \tag{2.9}$$

Where y is the class labels represented as a one-hot encoded matrix and $\hat{y}$ is the predicted, Softmax-activated, matrix.

	Softmax			**Labels**			**Hit?**	**Loss CCE**	**Class Error**
A	**0.4**	0.3	0.3	**1**	0	0	yes	**1.07**	**0.33**
	0.3	**0.5**	0.2	0	**1**	0	yes		
	0.6	0.2	0.2	0	0	**1**	no		
B	**0.8**	0.1	0.1	**1**	0	0	yes	**0.51**	**0.33**
	0.05	**0.9**	0.05	0	**1**	0	yes		
	0.6	0.1	0.3	0	0	**1**	no		

Table 2.1: Categorical Cross Entropy vs. Classification Error

Comparing CCE to a simple classification error function that checks whether the correct class is predicted or not, the benefits of using CCE as a loss function becomes apparent. In table 2.1 we can see that CCE is much more descriptive of how well the softmax predictions match the labels. Output A produces the same correct predictions as output B, but the probability of the predicted class is not as prominent as in output B. The classification error is 0.33 for each of the cases resulting in the same amount of weight adjustment during a backpropagation, contrary to CCE which will take into consideration the differences between these distributions.

Optimizer

Optimizers are used to navigate the solution space of a function with the goal of finding the global and optimal solution for that function.

In machine learning we try to minimize the loss function as much as possible without overfitting during training. This is done with weight adjustment through backpropagation where an optimizer tries to steer the gradient of the loss function in a direction towards the global optimum, meaning that it attempts to find the correct weight values between each layer in a network to produce a result closest to the correct answer. The derivative of the loss function gives the optimizer an estimation in which direction it is moving relative to the optimal solution. It can encounter false optimal solutions called *local optima* that can seem like a good solution, but in the bigger picture produce a sub-optimal result. Optimizers are prone to get stuck in local optima and different optimizers have been developed over the years trying to escape

these and reaching the global optimum as fast as possible. Techniques such as momentum have been added to increase the chance of escaping.

Adam is an optimization algorithm proposed by Kingma and Lei Ba in 2015[25]. The name *Adam* is derived from *Adaptive Moment estimation* and is based on the best qualities from two popular methods: *RMSProp*[26] and *AdaGrad*[27]. Kingma and Lei Ba concluded that Adam is a suitable method for large datasets or high dimensional parameter spaces. It is robust, easy to implement and an efficient optimization algorithm with little memory requirements[25]. Taking this into consideration Adam is a much-suited optimizer to be used in this thesis.

Data Augmentation

Figure 2.10: Random rotation of +/- 10° and horizontal flipping.

Convolutional Neural Networks are commonly known to require a substantial amount of training data. In the cases of sparse training sets with low variance, CNNs can easily overfit during training. A solution to this problem can be to generate more data either artificially or by manually labeling new data.

A second option can be to augment the existing data. The idea is to apply a random set of operations on the input images, such as rotation, horizontal/vertical flipping, adding noise and rescaling. In the perception of the network it will get new training data for every new augmentation, essentially increasing the number of training samples and reducing overfitting. Perez and Wang explored *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*[28], testing different traditional augmentation techniques as mentioned above, but also experimented with the use of Generative Adversarial Networks(GAN[29]) to generate new images. They concluded with increased classification accuracy and a reduction of overfitting with the use of traditional augmentation techniques.

In this thesis, more traditional augmentation methods are applied such as a small degree of random rotation and random horizontal flipping of the images. See figure 2.10.

2.3.5 Recurrent Neural Networks

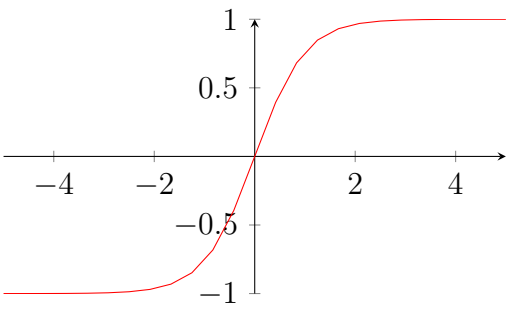
Recurrent Neural Networks (RNN) are designed to handle sequences of data making classifications based on data point occurrences over time. Examples of this can be text (sentences) or videos. By looking at how specific observations develop over time it can predict the next step in a sequence. RNNs have resulted in many impressive results such as Zach Thoutt's attempt to train an RNN on the famous book series *Game of Thrones* written by George R.R. Martin, enabling it to write the "next" book[30]. The result did not make much sense in most cases, but it is apparent that the network was able to learn essential writing techniques like quotes, paragraphs and generally proper spelling, but also relationships between the different characters in the book.

RNNs process a sample for each time step from the sequence of one-dimensional vectors. The hidden state is computed based on the previous hidden state(h_{t-1}) at time(t-1) and the current input(x_t) at time(t):

$$h_t = \sigma_h(W_i x_t + W_h h_{t-1}) \tag{2.10}$$

The input and previously hidden state is multiplied with W_i, which is the input weight matrix, and W_h which is the recurrent matrix. The sum of these two results is then fed through a hidden activation function σ_h, usually TanH-activation:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.11}$$



which compared to sigmoid-activation extends the "activation range" from -1 to 1.

$$y_t = \sigma_y(W_o h_t) \tag{2.12}$$

For every time step an output is computed (y_t), where W_o is the output matrix and σ_y the output activation function. This means that we can get the output at the time step that we prefer as illustrated in figure 2.11.

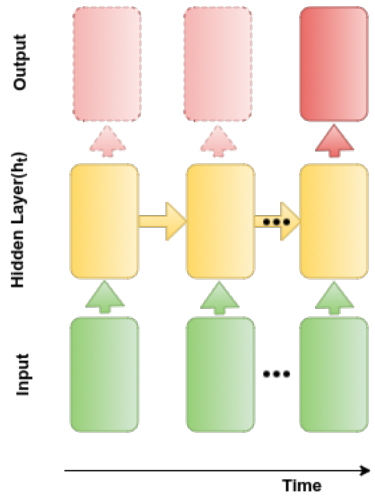


Figure 2.11: Simple RNN illustration, based on figure from[31]

".. an RNN has the ability to learn to detect an event, such as the presence of a particular expression, irrespective of the time, at which it occurs in a sequence."[31]

In Ebrahimi Kahou et al.'s paper[31] they present an approach that includes using an RNN to classify facial expressions in videos. Their work is

very much related to the proposed implementation described in chapter 5 where the classification of facial expressions is done with recurrent neural networks. Contrary to the implementation in chapter 5 they make use of a CNN to extract features, whereas the approach explained in this thesis experiments with *Local Directional Strength Patterns*[1] as features and RNN combined with FCN as classification.

In Ebrahimi Kahou et al.'s paper they use a variant of RNN called IRNN. Usually an RNN utilizes TanH as activation function, but with the problem of exploding and vanishing gradients, which is a problem when handling long sequences, an alternative was proposed were Rectified Linear Units was used instead and with a recurrent matrix that was initialized with scaled variations of the identity matrix, as described in [31, 32].

2.4 Face Detection

The first problem to be solved is to find the face in real-time, which is commonly the first stage in almost every face related computer vision system, and the aim is to get a high positive detection rate even with challenging backgrounds. In this section an overview of previous work related to face detection is presented.

Face detection can be quite tricky considering the diversity of faces in the world including local variations such as light conditions, orientation, etc. Despite this, significant progress has been made in recent years[33]. A fast algorithm for face detection is needed in the proposed implementation in this thesis due to the robots limited computational power. Viola and Jones proposed a suitable algorithm for fast face detection in "Robust real-time face detection"[8]. In their paper they introduce a model based on Haar-like features (see section 2.2.4, and figure 2.12) in a cascade classifier. If a proposed segment fails on one of the weak classifiers, it is discarded. This ensures a high classification accuracy, and even though the number of sub-windows and features are immense, the detector calculates this very quickly. Though Haar-like features are suitable for real-time face detection, it works best when it comes to well lit frontal faces as stated in Ranjan, Patel and Chellappa's paper[34]:

"It has been shown that in unconstrained face detection, features like HOG or Haar wavelets do not capture the discriminative facial information at different illumination variations or poses."

However, the cascade method can be used in combination with different features like Local Binary Patterns or automatically accumulated features from a convolutional neural net, like Li et al. did in their paper [35]. Since the hardware on the robot is limited, deep CNNs pose a threat when it comes to speed, leaving traditional methods such as cascade classifiers with Haar-features[8], local binary patterns or HOG-features as excellent options.

Rajan, Patelan, and Chellappa proposed a quite inventive approach to combine task solving in a CNN [34]. They called it *HyperFace*, and it is a CNN that predicts face detection, landmark localization, pose estimation and gender recognition. The idea is based on the knowledge of what information each layer of a CNN is able to extract. The first layers usually detect rough features like corners, edges, etc. and is suitable for localization and pose estimation. The deeper layers have more fine-tuned and specific features; these can be used for gender recognition and face detection. This work can be applied to my application where face detection, identification and emotion recognition is constrained to a single CNN, but it is likely not suitable for real-time classification as stated earlier.

The proposed emotion recognition system only needs to find the face once. Rather than scanning the whole image searching for a face for each frame captured by the camera it can look in regions close to the location of where the face was in the previous frame. This can be done with face tracking. The python library *dlib*[36] has a function called *correlation tracker* which tracks a given object in an image stream by guessing the next position of the object in the next frame based on previous frames and a confidence score. This will probably increase the speed since it only needs to scan the whole frame until it finds the correct face, optimally only once each time the user is entering the field of vision.

A similar tracking alternative is implemented by *OpenCV*[37], called *Kernelized Correlation Filters* (KCF). KCF is a tracking method that exploits circulant matrix properties to enhance processing speed. As written in OpenCV's reference guide: This tracking method is an implementation of Henriques, Caseiro, Martins, Batista *Exploiting the Circulant Structure of Tracking-by-Detection with Kernels*[38] which then is extended to KCF with color features[39].

Figure 2.12: Haar-like feature extraction[40]

When the face is extracted, it leads to the next step in the pipeline, identification. This is to ensure that only the users face is analyzed, and so the privacy of visitors and other people is respected.

2.5 Identification

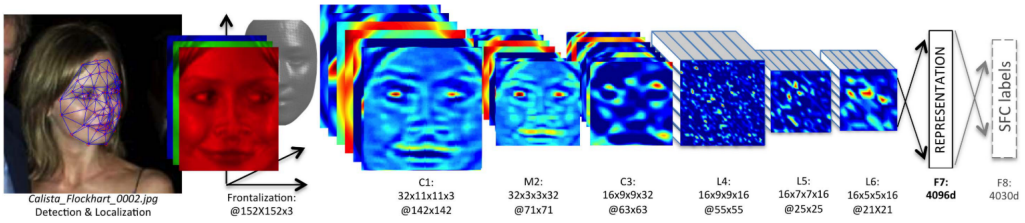


Figure 2.13: Identification CNN architecture used in DeepFace[41]

In this section, previous work related to the identification of human faces by the use of the different models available is presented. Also in this stage, speed is essential as well as accuracy. By identifying the correct face quickly the system/pipeline will be able to start analyzing the facial expression and make a classification.

Face identification is a technology with an increasing use on different devices ranging from unlocking smart-phones and payment verification to security measures when entering restricted areas. When it comes to mobile devices, the identification models are expected to be highly accurate, but also small and fast. Most CNN models that predict with high accuracy are very deep and is reliant on powerful hardware to make the calculations within a reasonable time. Chen et al. proposed efficient CNNs for accurate real-time face verification on mobile devices[42]. The number of trainable parameters in a CNN is one of the factors that consume processing time. A fairly big

architecture can easily contain multiple millions of parameters. VGG[43] is an example of this with 138 million parameters. The MobileFaceNets[42] has managed to reduce the number to less than 1 million significantly increasing the speed with no cost in accuracy.

As described earlier, CNNs are one of the best feature extractors of images. A method utilizing this is described in Guo, Chen and Li's paper where they propose a way to enhance classification rates by the use of CNN and Support Vector Machine(SVM). In their paper they use the CNN to extract features and the SVM to do the classification. They chose the SVM as an alternative to a fully connected network;

"..we chose SVM to recognize faces as its excellent performance in solving linear inseparable problem. [14].

Support Vector Machine is a machine learning classifier which uses support vectors to define a decision line that separates data based on their features. As Guo, Chen and Li says in their paper;

" For linear inseparable problem, SVM maps input in low dimensions into higher dimension feature space that makes separation easier."[14]

They proved that the combination of CNN + SVM improved the average classification rate of 0.615 % on their test data compared to a standard CNN with a fully connected classifier. This approach is a strong candidate, but since the identification task related to this system is a binary classification problem (user/not user), a simple CNN should suffice.

2.6 Emotion Recognition

Today, facial expression recognition is a very researched field in computer vision [1, 11, 44, 45]. Despite this it is proven hard to implement an automatic model to perform this task [11, 46]. To actively monitor the facial expression frame by frame and make predictions can be quite computationally difficult. The best results within facial expression classification are done with comprehensive feature extractions combined with deep neural networks or CNNs as described in Uddin, Khaksar and Torresen's paper[1].

Offline training and testing of models can give good results, but if presented with live image streams it may cause poor performance. In recent years,

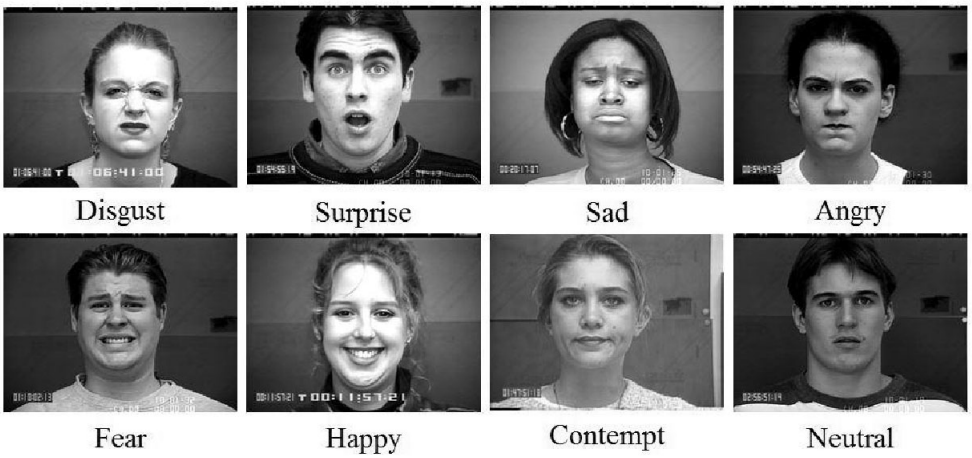


Figure 2.14: Emotion recognition in images[2]

approaches involving CNNs have shown the best advancements in the computer vision field [11, 12, 14, 1, 35, 42]. Exploiting the ability of feature extraction that CNNs possess and perhaps combining them with other features to get better classification is proven favorable in many cases[1, 11].

In Uddin, Khaksar and Torresen's paper they used an approach with Local Directional Rank Histogram Pattern (*LDRHP*) in combination with Local Directional Strength Pattern (*LDSP*) as features for classifying emotions in depth images. *LDSP* calculates relative edge response values of a pixel in different directions based on the outputs from a Kirsch edge detector since it considers edges in all eighth directions[5] (see section 2.2). This combination where *LDSP* is augmented with *LDRHP* is presented as *LDRHP* || *LDSP*, and it generates a histogram were dimensions are reduced with *kernel principal component analysis* and *general discriminant analysis* to better represent the features in the images. A CNN is then trained on these features, and the idea is that the network is more capable of learning salient features (important features) which again results in better classification. Essentially this approach is based on directions of the gradients in the images and classified with a CNN. By replacing the CNN with an RNN it is possible to answer the question about gradient based descriptors combined with an RNN.

Uddin, Khaksar and Torresen's model[1] was trained on 6 classes with depth images expressing these emotions:

1. Anger
2. Happiness
3. Sadness
4. Neutral
5. Surprise
6. Disgust

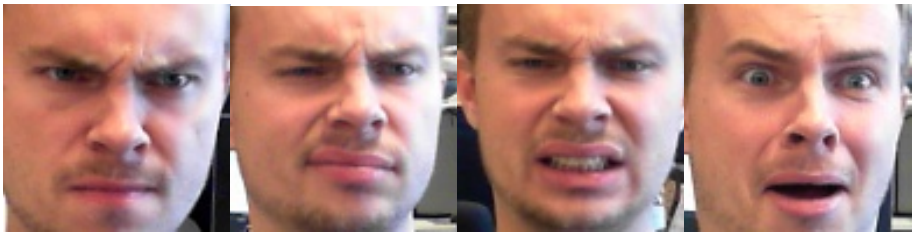
Contempt and fear is the seventh and eighth facial expressions as described in the Cohn-Kanade+ dataset documentation[2], but was not used in the approach described above.

In Alizadeh and Fazel's paper[11], they describe an approach solely based on a CNN architecture and fully connected layers for classification on an emotion recognition dataset of grayscale images from Kaggle[47]. They applied different techniques to reduce overfitting on the training set such as *dropout* (see section 2.3.4), *batch normalization* (see section 2.3.4) and *L2-regularization*, but their best results did not surpass 60% accuracy on the test set compared to Uddin, Khaksar and Torresen's approach that managed to achieve a recognition rate of 97.08%. Note that these two models were trained on different datasets where [47] may contain more variations in light, orientation etc. than [1] that trained on their own depth image dataset.

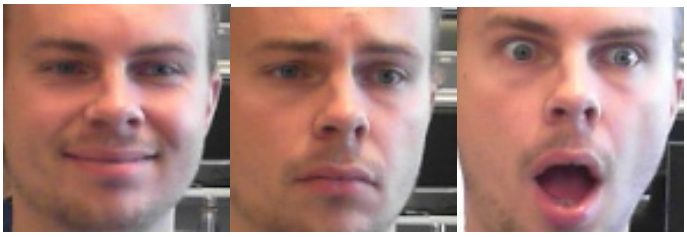
Chapter 3

Datasets

In this chapter, a brief overview of the applied datasets is presented. Comparing differences and drawbacks related to both a personalized dataset and a public dataset.



(a) Anger (b) Contempt (c) Disgust (d) Fear



(e) Happiness (f) Sadness (g) Surprise

Figure 3.1: Images of emotions in personalized dataset.

3.1 Emotion Datasets

For this thesis personalized datasets of myself, as the user, will be applied for training and testing of the real-time classification system. This is to enable exploration of the CNNs ability to enhance classification rate and compare the findings with the use of public datasets like the CK+ facial expression dataset [2], highlighting the benefits of using a personalized dataset contrary to a public dataset.

Three data sets was constructed with varying number of facial expressions, see table 3.1.

Every result is compared with the Cohn-Kanade+ dataset as benchmark [2]. The four class personalized dataset will be a bit different than the CK+ equivalent; this is due to the lack of degrees of happiness in the CK+ dataset. Instead, degrees of sadness/anger will be used as the CK+ alternative. One can argue that feature wise, happy/glad is the inverted version of angry/sad.

Cohn-Kanade+ dataset is a public dataset containing sequences of varying length with eight different expressions as listed in table 3.1. The eighth expression is *Neutral*, but has no sequence dedicated to illustrating

Two classes	Four classes	Seven classes
Glad	Sadness	Anger
Not glad	Neutral	Disgust
	Glad	Contempt
	Happy	Fear
		Happy
		Sadness
		Surprise

Table 3.1: The three different dataset with their respective classes

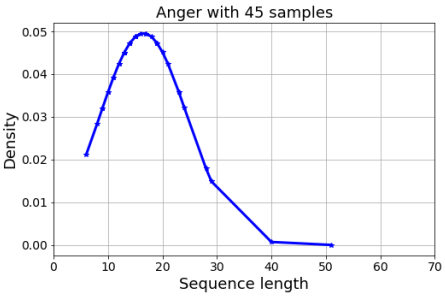
this expression. It is counted as an expression in the CK+ dataset due to every sequence containing a neutral state before entering full emotion display, meaning each participant displays an expression from *neutral* to *peak expression.*

When constructing an emotion, certain features must be present in the face called *Action Units - AU* where 30 different actions are labeled, see table 3.2 and 3.3. These action units, or muscle contractions, for categorizing facial expressions are described in *Manual For The Facial Action Coding System*[48] also known as *FACS*, written by P. Eckman and W. Friesen.

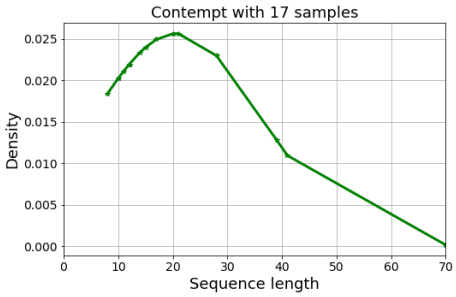
AU	Name	N	AU	Name	N	AU	Name	N
1	Inner Brow Raiser	173	13	Cheek Puller	2	25	Lips Part	287
2	Outer Brow Raiser	116	14	Dimpler	29	26	Jaw Drop	48
4	Brow Lowerer	191	15	Lip Corner Depressor	89	27	Mouth Stretch	81
5	Upper Lip Raiser	102	16	Lower Lip Depressor	24	28	Lip Suck	1
6	Cheek Raiser	122	17	Chin Raiser	196	29	Jaw Thrust	1
7	Lid Tightener	119	18	Lip Puckerer	9	31	Jaw Clencher	3
9	Nose Wrinkler	74	20	Lip Stretcher	77	34	Cheek Puff	1
10	Upper Lip Raiser	21	21	Neck Tightener	3	38	Nostril Dilator	29
11	Nasolabial Deepener	33	23	Lip Tightener	59	39	Nostril Compressor	16
12	Lip Corner Puller	111	24	Lip Pressor	57	43	Eyes Closed	9

Table 3.2: Action Units for Facial Expressions[2]

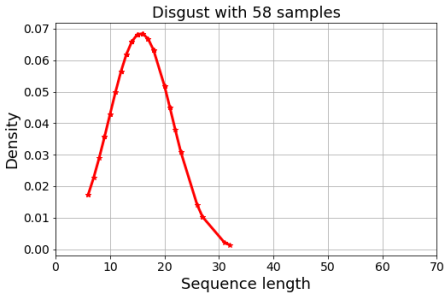
Regarding generating a **personalized dataset**, a few things have to be considered. A person dependent emotion classifier has to get familiar with the unique expression of its user, meaning that features which make a person dependent emotion unique have to be present in the images provided to the classifier, at the same time trying to include the proper action units describing that emotion in a general view. This is to enhance the ability to make a reasonable comparison of results between the Cohn-Kanade+ dataset and the personalized dataset.



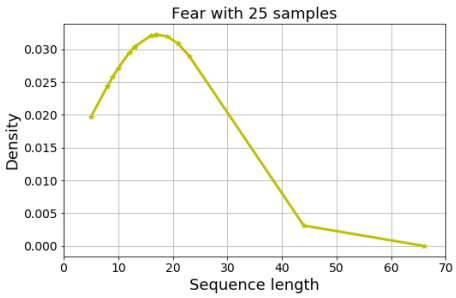
(a) Anger



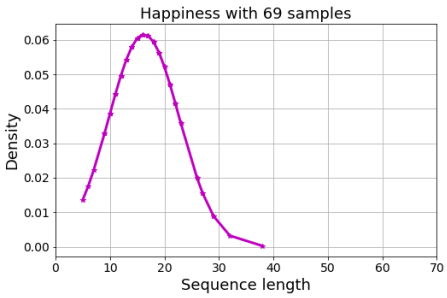
(b) Contempt



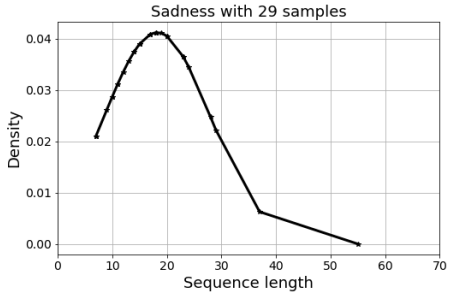
(c) Disgust



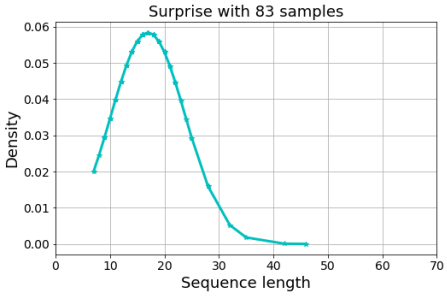
(d) Fear



(e) Happiness



(f) Sadness



(g) Surprise

Figure 3.2: Sequence length distribution in the CK+ dataset

Emotion	Criteria
Angry	AU23 and AU24 must be present in the AU combination
Disgust	Either AU9 or AU10 must be present
Fear	AU combination of AU1+2+4 must be present
Happy	AU12 must be present
Sadness	Either AU1+4+15 or 11 must be present. An exception is AU6+15
Surprise	Either AU1+2 or 5 must be present
Contempt	AU14 must be present (either unilateral or bilateral)

Table 3.3: Emotion Description with Action Units[2]

In figure 3.3 the class distribution of the whole CK+ dataset is shown, consisting of 5880 separate grayscale images divided over seven classes. As we can see, there is not an equal distribution of images between each expression. The difference between *Happy* and *Contempt* is approximately 1100 images, this means that *Happiness* is 550% greater than *Contempt* which is only 3,4% of the entire dataset and is shown in figure 3.3. This skew in the distribution may be a problem when training the model for classification. One explanation could be that there are more neutral images in the classes with higher representation. In the CK+ documentation[2] it is said the sequences vary in length from 10 to 60 frames, but in reality, it is 5 to 70 frames. By looking at the sequence length distributions for each expression in figures 3.2a-g, the classes with the lowest number of images representing each emotion contain the long sequences. By comparing figure 3.3 and figure 3.4, which is the reduced dataset with only peak emotion displays, the distribution between the classes remain the same. This indicates that neutral images in each sequence do not interrupt with the amount of actual emotion displays for each class.

The CK+ dataset has to be reduced in size since we are only interested in the peak expressions from each sequence when training a CNN on those images, making the total size of both the CK+ dataset and the personalized dataset reasonably similar.

Figure 3.5 presents the class distribution of the personalized dataset containing images of one user. As mentioned earlier, expression criteria - *Action Units* - and person independent features of the individual expression, is attempted to co-exist in every emotion display.

A personal judgment was made when deciding which emotion that contains the most subtle features. *Contempt* stood out as one of those expres-

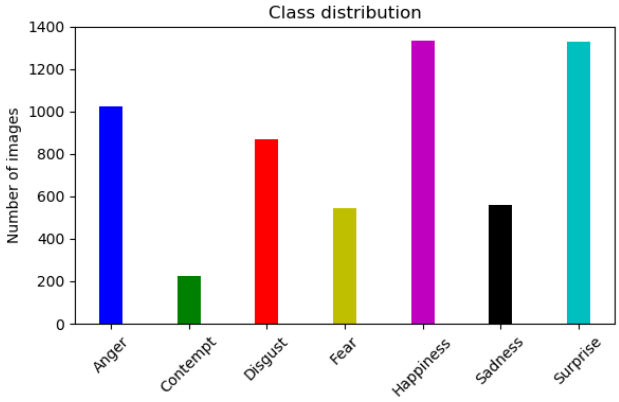


Figure 3.3: Class distribution in **full** CK+ dataset.

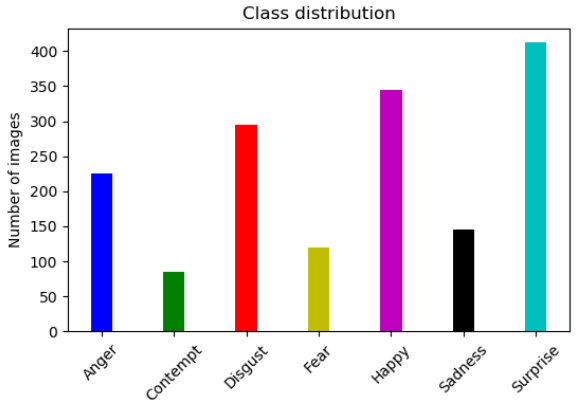


Figure 3.4: Class distribution in **reduced** CK+ dataset.

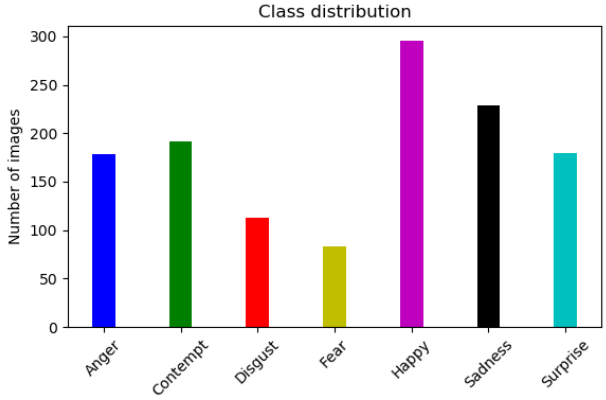


Figure 3.5: Class distribution in **personalized** dataset.

sions, including *sadness*. In light of these assumptions an increase in the number of images representing those expressions was made, while keeping almost the same ratio between the other classes. This difference might have an impact on how well the models presented can learn from them, but it will become clear it is not a significant contributor to the contrast between the results of the personalized and generalized model (i.e., model trained on a public dataset).

(a) Anger (b) Contempt (c) Disgust (d) Fear (e) Happiness

(f) Sadness (g) Surprise

Figure 3.6: Images of emotions in Cohn-Kanade+ dataset[2].

3.2 Identification Dataset

Separating two classes using deep learning is commonly considered to be a less complex task. With the need of a relatively small dataset or at least fewer data representations of each class, a reduction in the number of hidden layers in the deep learning model is possible, which again results in a decrease in training time.

The two classes represented in this dataset is *User/Not-User*, with roughly the same amount of images in each class. The *Not-User* class is a set of images of fellow students which have agreed to participate as test subjects and that their data can be used for training of an identification model.

Chapter 4

Real-Time Emotion Recognition Pipeline

In this chapter the proposed approach for constructing a functional pipeline of *Face Detection - Identification - Emotion Recognition* for real-time applicability is presented, visiting different complications and obstacles and the solutions applied to overcome these.

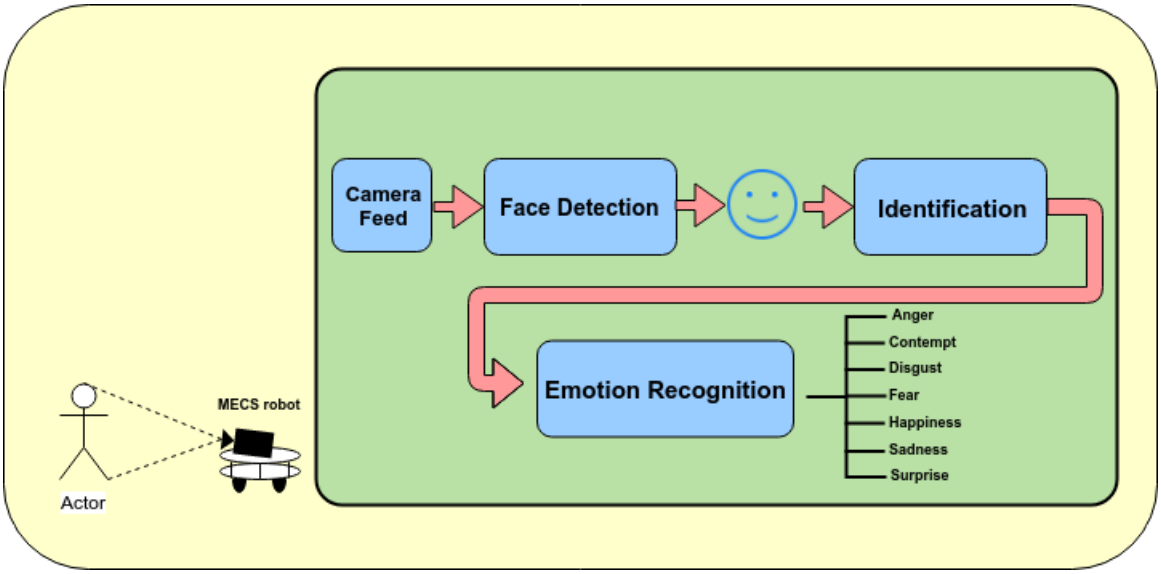


Figure 4.1: Illustration of emotion analysis in the system.

4.1 Approach

Simplifications and adjustments have been critical procedures when the goal is to build a robust and fast application for the *MECS-Robot*, enabling problems to be solved gradually and systemically, leaving a low possibility of failure.

4.1.1 Image Preprocessing

Figure 4.2: RGB image split into separate channels of red - green - blue

Red - Green - Blue Images (RGB) contains almost all the information needed to understand which actions, objects and messages occurring in the presented image, at least for the human eye. With machine learning the goal is to transfer this human knowledge over to a computer, so that it

can perform tasks that previously only humans could do. An RGB image is represented as a three-layered matrix with shape (width x height x channels), where the channels are *Red, Green and Blue* ranging in value from 0 to 255. In an image with pixel values that are all red, the channel values will thereby look like this [255, 0, 0]. [255, 255, 255] = white, [0, 0, 0] = black.

Image preprocessing is a term used for techniques that are applied before any calculations or processing is done. An example of this can be feature extraction, like Local Directional Pattern (see section 2.2), filtering e.g. Kirsch edge detection and also image value normalization.

Normalization aims at smoothing out the data, removing or muting extreme values called outliers. *Max-Min Normalization* is a popular normalization method and is used in this approach. Below is the function that calculates the normalized pixel values z_i which is being used as a normalizing function on the images before they are presented as input to the CNNs described in the identification and emotion recognition experiments, section 4.2.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

x = *(x_1, x_2, ..., x_n)* is the data points or pixels in this case. The minimum value of x is subtracted from x_i and then divided by the difference between the maximum and the minimum value of x. The result is z where all the pixel values are "squashed" between 0 and 1. OpenCV's normalize implementation[37] was used for preprocessing of the image datasets.

Neural networks modifies the weights so that the output value matches the target or class label. With seven classes, the value to be predicted is ranging from 0 to 6 where these values are arranged in a way that the correct class corresponds to its location in a list, called one-hot encoding. With the example below, we have four images with class labels 4, 2, 5 and 3 respectively. In one-hot encoding, it will be converted into a matrix where a single row represents one sample and the location where this row equals 1 corresponds to the class label of that same sample.

$$[4, 2, 5, 3] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

With the labels represented in this way, we can use *Softmax*[49] activation at the output nodes. Softmax is a logistic activation function similar to *Sigmoid*, but it can handle multiple outputs. It takes the output and rescales it between 0 and 1, turning it into a probabilistic representation where the node closest to 1 is the most likely class prediction of that input.

In equation 4.1 an example image of class 4 is passed through the deep neural network. It produces a set of example output values represented in the first vector with length equal to 7.

$$\begin{bmatrix} 1.0 \\ 2.0 \\ 0.5 \\ 3.0 \\ 5.0 \\ 0.2 \\ 0.1 \end{bmatrix} \Rightarrow S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \Rightarrow \begin{bmatrix} 0.014 \\ 0.040 \\ 0.009 \\ 0.110 \\ 0.812 \\ 0.007 \\ 0.006 \end{bmatrix} \tag{4.1}$$

y is the predicted value at each output node, called *logits*. After applying *Softmax*, the predicted class, as we can see in the far right vector, is closest to 1 in position 4. Using this new vector, we can calculate the loss by comparing it with the one-hot encoded class labels illustrated above. Since it is the correct class, the loss will be smaller compared to a different prediction. The calculated error, or loss, depends on which loss function is used. See section 2.3.4 about *Categorical Cross Entropy.*

This leads us back to the normalization part. Instead of the neural network generating weight matrices that downscale the pixel values, we apply normalization before we train the network. This will result in better adjustments of weights during training, and the need of a normalization layer in the CNN is reduced, but for the models classifying 4 or more emotions, a normalization layer has been used in the first layer. See section 2.3.4 about *Batch Normalization* and figure 4.8.

4.1.2 Face Detection

It exists many face detection algorithms in public python-libraries such as OpenCV[37] and dlib[36]. One criterion when it comes to choosing a face detector is speed, but also accuracy. Therefore a comparison between the ex-

isting methods available from these libraries is conducted. This is to ensure that the face detector used provided fast calculations and high accuracy. When measuring accuracy, false-positives are ignored and not counted although it detects a positive face. Considering this, subsequent stages in the pipeline can focus on actual faces, hence saving time and making the entire analysis faster.

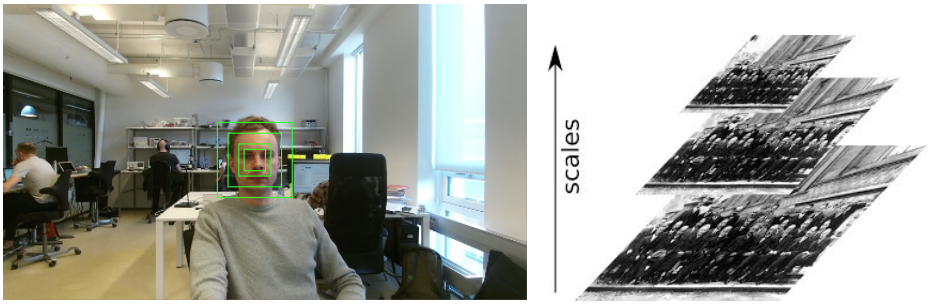
Scale Factor

Viola & Jones' face detector scans the input image at many different scales, like most face detection systems. An image can contain numerous faces located at various distances as displayed in figure 4.9a and 4.9b. The scale factor ensures that faces close- and far from the camera has a higher possibility of detection. Conventionally a scale factor of 1.25 is used to generate a pyramid of 12 images where each image is 1.25 times smaller than the previous one. Processing every layer of images in this pyramid and applying features at each scale is proven to be quite a comprehensive operation[8]. Viola & Jones's face detector scans the input image at 12 scales. Starting at the base dimensions of 24x24 to detect faces, for each iteration the sliding window increases by a factor of 1.25[8]. With a cascade classifier, eliminating regions and applying features can be done at any scale and location in a few operations, dramatically speeding up the face detection process.

scaleFactor is a parameter that has to be passed to OpenCV's *detectMultiScale*[37]. It is important to test for different scales to find the suitable value which ensures a good classification and processing speed. Low scale factors are equal to a small difference in size of the sliding window for each iteration, resulting in a slower, but more accurate classification. A high scale factor will do the opposite. It requires fewer iterations due to more rapid increase in sliding window size, but with a higher chance of incorrect classification. An illustration of how the sliding window increases are depicted in figure 4.3a, and in figure 4.3b the conventional image pyramid.

Minimum Neighbors

When using a set of features to decide if an object is present or not, it is almost unavoidable to encounter false-positive classifications, meaning that the object has been classified to something it is not. An example of this



(a) Different Image Scales (b) Image Pyramid [50]

Figure 4.3: Image Scales Vs. Image Pyramid

can be finding faces in an image with no faces. Somewhere in the fore- or background at a specific scale and location, the features describe what is known to the classifiers as a face and will mark the region accordingly. This can and will happen if we do not try to dim the sensitivity of the classifier.

(a) Example of minNeighbors=**0** [51] (b) Example of minNeighbors=**3** [51]

Figure 4.4: Different *minNeighbors* values.

minNeighbors is the second key parameter to be passed to OpenCV's *detectMultiScale*[37]. It specifies how many neighbors or overlapping bounding boxes that are required for a face to be accepted. In figure 4.4a minNeighbors is set to 0. As we can see, the number of detections is overwhelming considering that it finds faces in trees, rocks, ground, etc. because features describing these areas are similar to the features of a face. By increasing the threshold of the minimum number of neighbors, we eliminate single "weak" classifications and focus on those areas which have stronger features. In figure 4.4b minNeighbors is set to be 3, and the true faces in the image have been found and it ignores false-positives. Again, finding the right number of neighbors is also important when it comes to choosing a good face detector.

Selecting a number which is too low, will result in many false-positives. On the other hand, selecting too many and it might overlook areas where there is a face.

Face Descriptor Comparison

The pre-trained face detectors used in this comparison is:

1. Haar-cascade classifier
2. LBP-cascade classifier
3. dlib's get_frontal_face_detector - returns a face detector based on histogram of oriented gradients (HOG-features)
4. Other pre-trained face detectors available in OpenCV for better comparison, that are present in figures 4.10 and 4.11.

Above is the list containing the different face detectors used for comparison in section 4.2.1 with the aim of selecting a face detector best suited for real-time applicability in the presented emotion recognition system.

The first pre-trained detector to be tested is *Haar-cascade classifier* which is trained on Haar-like features as described in section 2.2.4 and 2.4.

The second classifier is also a cascade classifier, but it is trained on Local Binary Patterns(see section 2.2.1) instead of Haar-like features. Both cascade classifiers are tested on different scales and minimum neighbors to find the best-suited parameter settings for good face localization and classification, see section 4.1.2 and 4.1.2.

The final classifier to be tested is trained on *Histogram Oriented Gradients*. HOG-features is what dlib's default face-detector uses[36] combined with a linear classifier, a sliding window scheme and an image pyramid. The image pyramid is decided in the second parameter in dlib's face detector called *upsample*. It determines how many times the image is being enlarged to enhance smaller faces in the image. The downside with an image pyramid, as mentioned earlier, is an increase in processing time. One extra scale iteration results in a tripling of the time it takes to process the image, so zero iterations have been selected for this face detector to see whether it can compete with a cascade classifier based on LBP or Haar-like features.

4.1.3 Identification

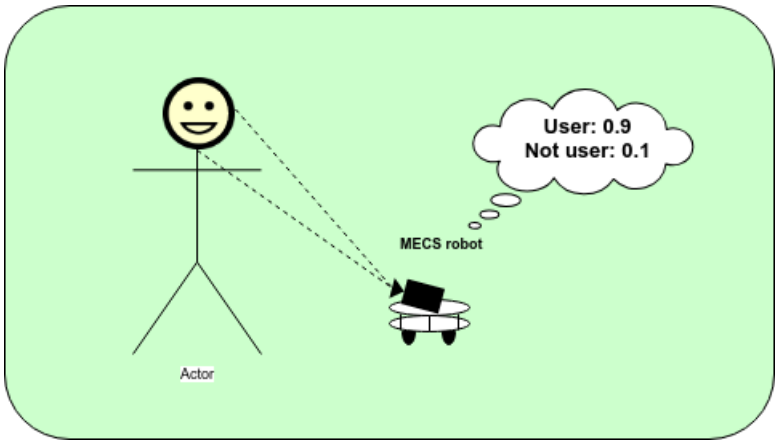


Figure 4.5: Illustration of face identification.

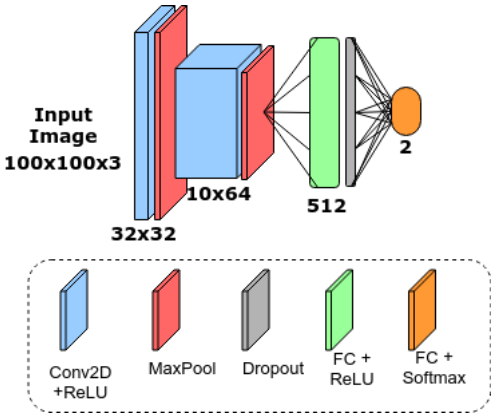


Figure 4.6: Identification model architecture.

Enabling the robot to differentiate between its user and other people present will require a classifier that can separate these two cases. A simple CNN should be able to distinguish between user and not user as discussed in chapter 2 section 2.5.

For identification a simple CNN architecture depicted in figure 4.6 is chosen and is based on techniques presented in chapter 2 section 2.3.4, feeding the input with RGB-images. A batch size of 20 images, Adam optimizer(see section 2.3.4) with learning rate of 0.001 and CCE-loss(see section 2.3.4) was selected. Softmax activation was used at the output node and ReLU activation in the rest of the network. A dropout(see section 2.3.4) probability

of 0.3 was used in the fully connected layer to increase generalization of the network. The kernel size in the convolutional layers was set to 3x3 and with the same pool size in the max-pool layers.

One of the concerns before testing the network was the lack of robustness. With two layers of convolution and a fully connected network with one hidden layer at the output, it is considered quite shallow and therefore possibly making it sensitive to changes in illumination and orientation. To further reduce this risk, training images were taken during different hours of the day on separate days with a variation in face position and orientation. To accommodate this, a simple data augmentation technique was applied such as random rotations within 10 degrees in each direction as described in section 2.3.4 about data augmentation. Horizontal flipping was not applied since it would change the location of person dependent features like wrinkles and freckles in a face essentially confusing the CNN.

Possible Problems Related to Real-Time Identification

For this robot to work correctly, it must be able to distinguish between different faces in a crowd. It is critical since it is only supposed to track the user which it has been assigned to and not violate the privacy of other people. This means that the accuracy when it comes to classifying identity must be almost perfect. Real-time tracking of faces will subject the model to extreme variations in light and orientation of the tracked face, which can result in poor classification accuracy. So, it is crucial to build and train a model that is light- and orientation invariant. Considering the need for a compact and fast model the depth will be quite shallow and may limit the networks ability to generate complex features.

4.1.4 Emotion Recognition

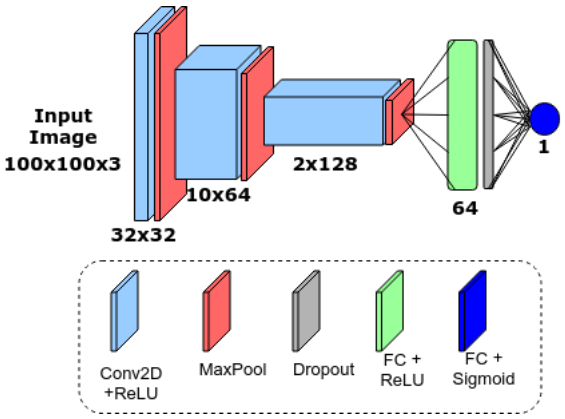


Figure 4.7: Two-class model architecture.

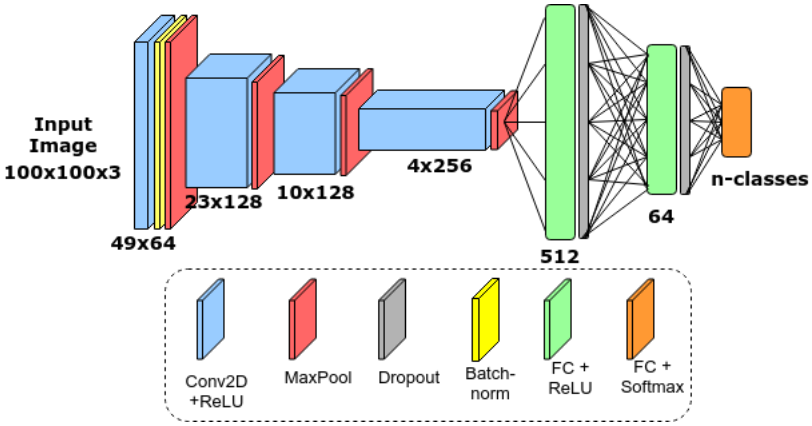


Figure 4.8: Four- & seven-class model architecture.

Convolution Neural Network architecture

The models used for training varies in size depending on the number of classes to predict, as illustrated in figure 4.7 and figure 4.8. The smaller model is used for binary classification and the bigger models for four and seven class classification problems. The depth of the model is somewhat equal to its ability to extract complex features as described in Ranjan, Patel and Chellappa's paper[34]. This is one of the reasons why a shallow model was chosen for the binary classification problem of *Glad/Not Glad*. The features

representing the difference between glad and not glad is not as complicated as the features separating the seven different classes. It is therefore assumed faster during training and real-time testing with no cost in accuracy.

Two-class Model Architecture

For training of the two-class model, which is a three-layered CNN with ReLU activation and a two-layered FCN at the output with dropout and sigmoid activation(see figure 4.7), a standard accuracy metric was used. 30 epochs over a training set of 1371 images, 60 validation images and 94 test images posed as a suitable duration of training time. The number of expressions started as a binary classification between *Glad* and *Not Glad*, which is every expression not displaying happiness.

Four- & Seven-Class Model Architecture

When expanding the two-class model to a four- and seven-class model, the number of layers in the network was increased to four convolutional layers, and batch normalization(see section 2.3.4) was added in the first convolution layer. Additionally, a three-layered FCN was added at the output with dropout between each layer and softmax activation on the output nodes to handle multiple classes. The four-class version was trained over 30 epochs contrary to the seven-class version which was trained over 60 epochs. The reason behind this was a prior knowledge that it takes longer to understand the differences between the seven classes compared to four classes.

Loss & Optimizers

Both architectures depicted in figure 4.7 and 4.8 utilizes CCE(see section 2.3.4) as loss function and Adam optimizers with a learning rate of 0.001(see section 2.3.4). The two-class model architecture produces only one output, converting the categorical cross entropy into binary cross entropy, essentially indicating that it processes vectors with length equal to 1.

4.1.5 Evaluation of Real-Time Performance

When it comes to evaluating the model in real-time, a recording of three videos of myself predicting different emotions ranging from 2 to 7 classes was used, so each model will be evaluated on their respected video to get an estimation of real-time performance. The videos are an image stream of roughly 300 - 800 frames, depending on the number of classes, containing all the emotions for which the model was trained, making it easier to estimate the accuracy. Since the system is using a tracking algorithm(see section 2.4) to increase processing speed, a sanity check has been implemented to ensure tracking of the true object. For every 100 frames processed, the face detector is re-run, leaving a couple of frames without any emotion predictions. This results in an *unknown class* during testing in real-time, but it has been excluded from the classification report and confusion matrices in section 4.3.

A pseudo-code representation of figure 4.1 is presented in algorithm 2. It includes all the stages associated with real-time emotion recognition.

Algorithm 2 Real-Time Emotion Recognition

```
initialize camera
initialize face_detector
load identification_model
load emotion_model
Face_found = False
while True do
    img ← camera.read()
    if not Face_found then
        faces ← Face_detector.findFaces(img)
        prediction ← identification.predict(faces)
        if prediction = User then
            Face_found = True
    else
        face_detector.track(face)
        expression ← emotion.predict(face)
        img.putText(expression)
    show_image(img)
```

Possible Problems Related to Real-Time Emotion Recognition

As stated earlier, facial expressions vary a lot from person to person and culture to culture. Building and training a generalized model for real-time use can be a difficult problem to solve. In the public datasets, a set of specific criteria has to be present for a facial expression to be valid [2], possibly resulting in a non-genuine expression of that person.

Considering that CNNs extract features to be classified, the network has to learn which features that best describes the majority of its inputs. Thus limiting the possibility of learning subtle person dependent features that may be a crucial factor when it comes to classifying the expression.

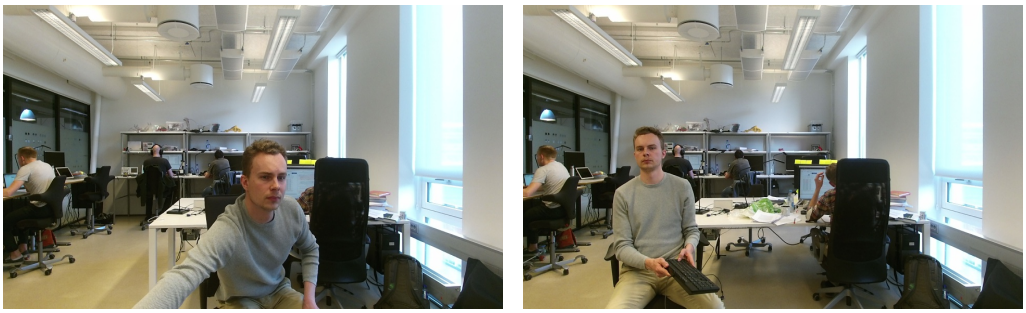
Fortunately, the robot is being designed as a personal robot. Meaning that it only needs to track and analyze its user, which in this thesis is me. The main differences between the implementation presented in chapter 4 and the previous work described in the background are a personalized approach and real-time testing on a mobile robot. Related papers also limit the number of classifications to six different expressions, whereas this thesis considers all seven classes.

Displaying true emotions for the camera can prove to be difficult. In the paper describing the CK+ benchmark dataset [2], the subjects portrayed in the images were told what emotion to display, leaving it possible to question the sincerity of that emotion. As discussed in chapter 3 the emotions displayed in the CK+ dataset are based on Action Units describing each emotion. In the personalized dataset, I express each emotion as I interpret them, this means including traits that are uniquely related to me.

The difference between these two types of datasets will be highlighted when training two identical deep CNNs which will prove a significantly better learning ability of the respected emotions in the personalized dataset compared to the public dataset. This is expected since a personalized model will always perform better, but it is interesting to see the true difference in performance proportional to the number of classes.

4.2 Experiments & Results

4.2.1 Face Detection Results



(a) sample location 1 (b) sample location 2

Figure 4.9: Face detection test images.

A set of 100 images were taken where my face is located at various places in the image. Each detector was tasked with finding my face in every frame. When calculating the performance of each detector, Intersection over Union (IoU) was used:

$$IoU = \frac{AreaOfOverlap}{AreaOfUnion} \tag{4.2}$$

This evaluation method provides an estimation of how well the true frame overlaps with the predicted frame. A $IoU > 0.5$ is considered a good match and normally counted as a hit.

Presented in figure 4.10 are the top performing face detectors given the parameters in the caption, and figure 4.11 shows what happens with accuracy if the parameters are not adjusted properly. Dlib's face detector(black dot - HOG) does not have any parameters to tune except for upsampling, but it has been left out due to the time consumption if increased from 0 iterations as mentioned earlier. That is why it is showing a static position in each plot. One can interpret this as dlib's face detector being the most stable option, but as illustrated in the figures LBP-cascade(pink square) pose as the fastest one with around 0.06 second processing time per image. By looking at the Haar-cascade option(blue triangle), it shows promising results with higher accuracy than LBP-cascade(pink square) and less processing

time than dlib's face detector(black dot) using HOG-features. These results match with the description presented in the background, chapter 2 section 2.2. The Haar-cascade classifier is fast and robust but is more likely to produce more false positives than HOG-features as shown on the accuracy axis. Haar-like features are also more complicating to calculate than LBP-features but are better suited as a face descriptor. Essentially the option lies between a *LBP-cascade classifier*, *Haar-cascade classifier* and *HOG-feature combined with a linear classifier* where the best choice is depending on what the specific requirements are. In the implementation, a decision was made to go for Haar-cascade classifier, which based on the results pose as the middle ground option that detects faces at an acceptable speed and accuracy.

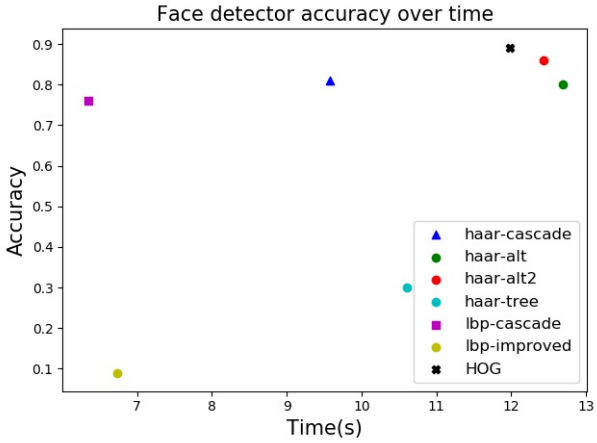


Figure 4.10: Face detector comparison with scale=1.25, neighbors=5

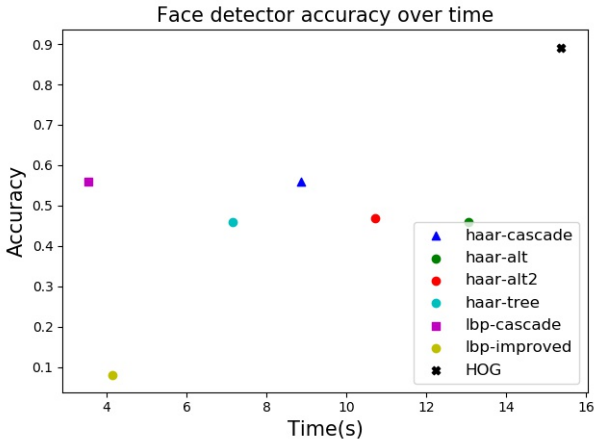
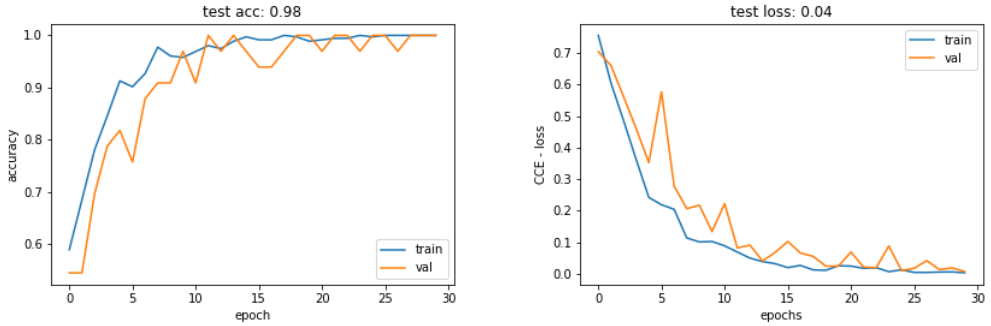


Figure 4.11: Face detector comparison with scale=1.4, neighbors=2

4.2.2 Identification Results

The identification model was trained over 30 epochs with the specification described in section 4.1.3. As we can see from the results, the model managed to get a 98% accuracy on the test set, that is one miss-classified image given the size of the test set as shown in the confusion matrix displayed in figure 4.13. It is better to classify a "user" as "other" rather than "other" as a "user", in the sense of overlooking the user instead of analyzing the face of an unknown individual. Judging by the somewhat steady progression of validation loss compared with training loss in figure 4.12, this model is a suitable choice for identifying between two individuals.



(a) Accuracy

(b) CCE loss

Figure 4.12: Identification training/validation accuracy and loss over epochs.

	precision	recall	f1-score	support
User	1.00	0.97	0.98	33
Others	0.97	1.00	0.99	33
avg / total	**0.99**	**0.98**	**0.98**	**66**

Table 4.1: Classification report on identification model

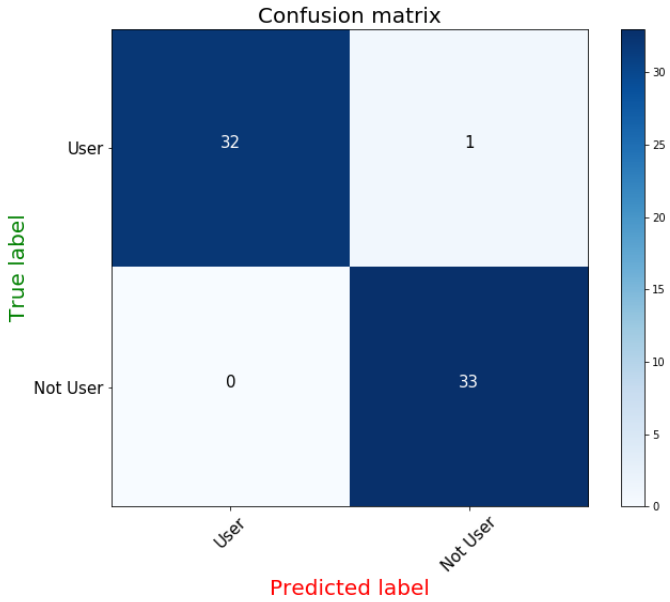


Figure 4.13: Identification results confusion matrix.

4.2.3 Emotion Recognition Results

Information Regarding Training & Testing of Emotion Classifiers

Deep neural networks are stochastic processes and can produce a different result for separate training runs. A result of random initialization of weights and other parameters during construction of a network. To get a statistically viable result, one should train each network over a series of runs, depending on how much the result is fluctuating. In these experiments, 10 runs were viewed as a suitable number for repetition of training and testing.

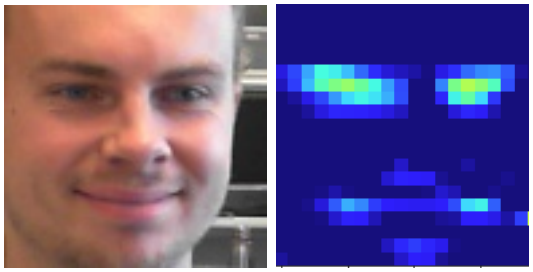
The two-class and four class models were trained for 30 epochs with a repetition of 10 runs. The seven-class model was trained for 60 epochs with a repetition of 10 runs. The tables below (4.2, 4.3 and 4.4) are from the top three performances on the test set with a comparison between public (CK+) and personalized datasets. In the appendix section 8.2 a progression of loss and accuracy during training is presented for each of the three different cases of two, four and seven classes. The real-time evaluation was only done on the models trained on the personalized dataset considering it showed the best results during offline training and testing, and it is the goal associated with this thesis.

When progressing from two to four classes the model used for two classes, depicted in figure 4.7, was switched out with a model designed to handle a bit more complex data structures which utilize batch normalization after the first layer as shown in figure 4.8.

Overall Results

In tables 4.2, 4.3, 4.4 the difference in accuracy's between the CK+ and personalized dataset is increasing when adding more classes to predict. In figure 4.15 an apparent difference is illustrated where the models' performances on two-, four-, and seven-classes is shown with average recognition rate(accuracy). For seven classes the residual of percentage points equals 13,3 on the average prediction accuracy, underlining the benefits of using a personalized dataset for complicated facial expression recognition. The average recognition rate of the top three results from the personalized model is 98.6% on two classes, 96.3% on four classes and 97.6% on seven classes. It is somewhat surprising that the model performs better on seven classes then four classes. It can be minor differences when initializing weights for training, which has given the seven class model a head start, or the case of distinguishing degrees of one emotion which is the case in the four class dataset making it a bit harder to separate the classes. The most likely reason is that the four-class model was trained for 30 epochs contrary to the seven class model which was trained for 60 epochs; thus it is not reasonable to compare those two results. As for the CK+ model results, it managed to get a top three average of 94.3% on two classes, 91,3% on four classes and 84.3% on seven classes. When increasing from four to seven classes, the CK+ model experienced a notable dip in accuracy, even more than expected. Given that the CK+ dataset is quite skewed in the class distribution, the effect of this has been prominent in the case of seven classes, allowing room for much improvement.

In figure 4.14, the activation map of layer 2 in the CNN model used for seven class emotion prediction on the personalized dataset is illustrated. It is clear that the model is capable of focusing on correct regions in the face where most of the information describing that emotion lies, such as the eyes and the angle of the mouth.



(a) Input image (b) Activations

Figure 4.14: Activation map of layer 2 in personalized model.

Top 3 runs	Accuracy
1.	0.95
2.	0.94
3.	0.94
Average	**0.943**
Standard Deviation	0.005

(a) CK+

Top 3 runs	Accuracy
1.	1.00
2.	0.98
3.	0.98
Average	**0.986**
Standard Deviation	0.009

(b) Personalized

Table 4.2: Top 3 results from both datasets with **two** classes.

Top 3 runs	Accuracy
1.	0.94
2.	0.93
3.	0.87
Average	**0.913**
Standard Deviation	0.031

(a) CK+

Top 3 runs	Accuracy
1.	0.97
2.	0.96
3.	0.96
Average	**0.963**
Standard Deviation	0.005

(b) Personalized

Table 4.3: Top 3 results from both datasets with **four** classes.

Top 3 runs	Accuracy
1.	0.88
2.	0.83
3.	0.82
Average	**0.843**
Standard Deviation	0.026

(a) CK+

Top 3 runs	Accuracy
1.	0.98
2.	0.98
3.	0.97
Average	**0.976**
Standard Deviation	0.005

(b) Personalized

Table 4.4: Top 3 results from both datasets with **seven** classes.

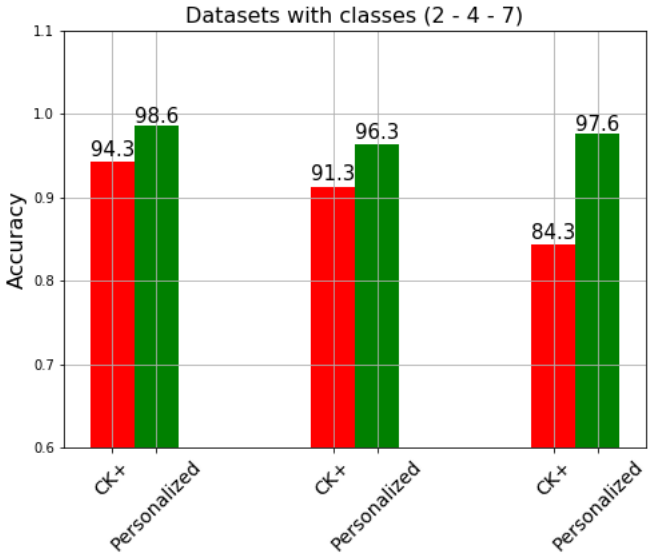
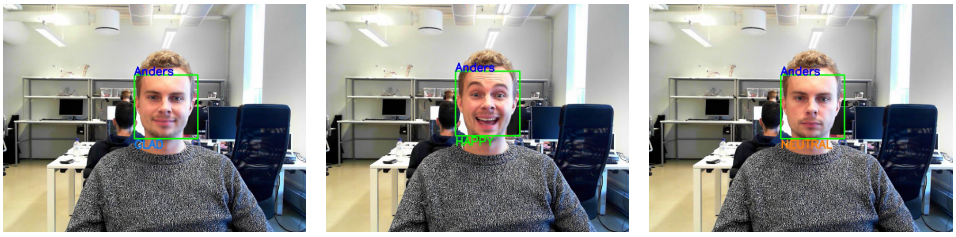


Figure 4.15: Average model performances. (Two-classes **left**), (Four-classes **center**), (Seven-classes **right**).

4.3 Real-Time Evaluation



(a) Predicted glad (b) Predicted happy (c) Predicted neutral

(d) Predicted sad (e) Predicted unknown

Figure 4.16: Predictions from the real-time evaluation of 4 classes.

This section presents the results from the real-time evaluation of a recorded 20 FPS video stream. The predictions were performed while recording and the labels were assigned afterward to preserve the notion of live predictions.

By looking at the results in the confusion matrix presented in figures 4.17, 4.18 and 4.19 real-time testing accuracy is somewhat lower than offline testing results. The conditions when the personalized training set was constructed may differ in many ways compared to conditions when testing in real-time. So it is to be expected that the real-time testing results show a less satisfying outcome than offline testing.

Given these priors, the personalized model can produce a total accuracy of approximately 93% correct classifications on two classes, 83% on four classes and 83% on seven classes. In figure 4.19 the emotion sad greatly overlaps with contempt and anger. A reason for this can be that my expression of sad can easily be interpreted as one of the two other classes which then comes to light in real time testing. Contempt and disgust also seem to be easily misclassified, given the action unit 12, *Lip Corner Puller*, is present in both of these classes in the personalized dataset.

Real-Time Evaluation of Two Classes

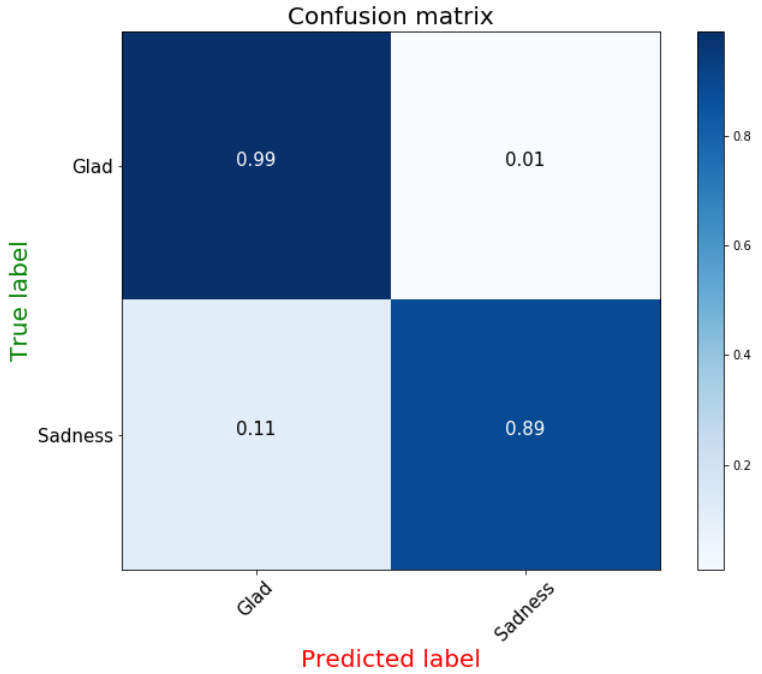


Figure 4.17: Two-class confusion matrix of real-time evaluation.

	Precision	Recall	F1-score	Support
Glad	0.85	0.99	0.92	123
Not-glad	0.99	0.89	0.94	183
avg / total	**0.9**	**0.93**	**0.93**	**306**

Table 4.5: Classification report on two classes.

Real-Time Evaluation of Four Classes

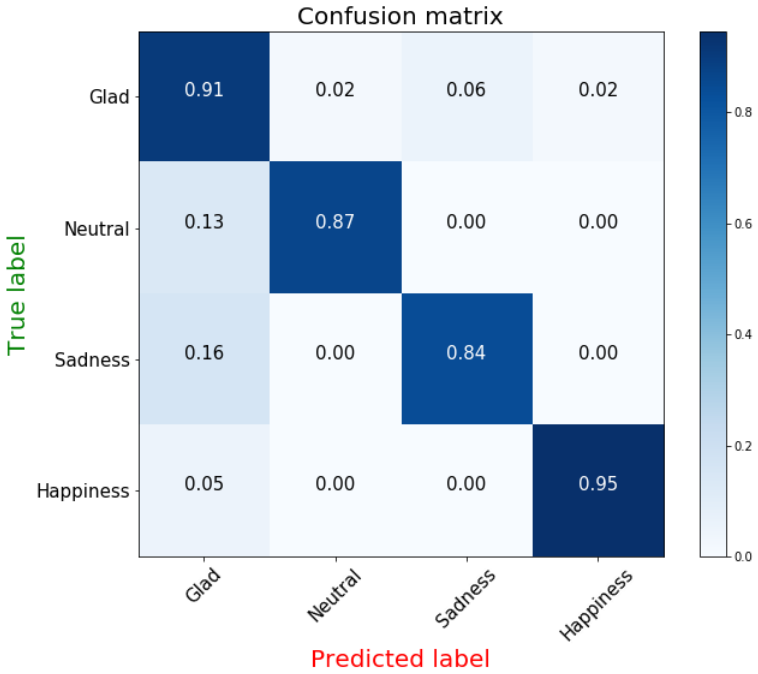


Figure 4.18: Four-class confusion matrix of real-time evaluation.

	Precision	Recall	F1-score	Support
Glad	0.83	0.88	0.85	124
Happy	0.96	0.87	0.91	52
Neutral	0.90	0.75	0.82	84
Sad	0.95	0.90	0.92	39
avg/total	**0.91**	**0.85**	**0.88**	**299**

Table 4.6: Classification report on four classes

Real-Time Evaluation of Seven Classes

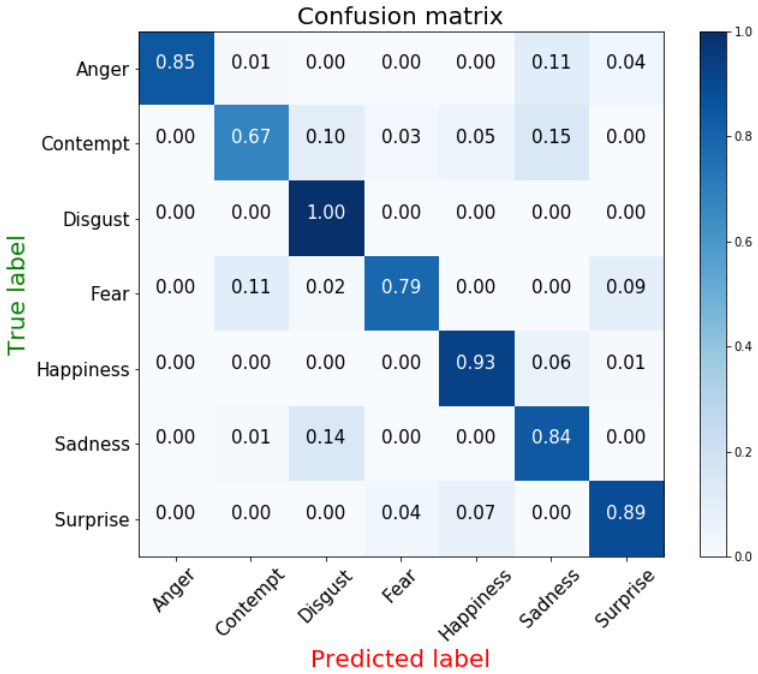


Figure 4.19: Seven-class confusion matrix of real-time evaluation.

	precision	recall	f1-score	support
Angry	1.00	0.85	0.92	208
Contempt	0.94	0.67	0.79	203
Disgust	0.66	1.00	0.80	61
Fear	0.81	0.79	0.80	56
Happy	0.86	0.93	0.89	124
Sad	0.50	0.84	0.62	70
Surprise	0.87	0.89	0.88	108
avg / total	**0.87**	**0.83**	**0.84**	**830**

Table 4.7: Classification report on seven classes

Chapter 5

Emotion Recognition Using Salient Features & Recurrent Neural Network in Video

In this chapter for research purposes, I will modify the work from Uddin, Khaksar and Torresen[1] by using a recurrent neural network instead of a convolutional neural network as described in section 2.6 and feed the network with an image stream of 5-6 grayscale images contrary to depth images. Based on the robustness of Local Direction Pattern for facial expression description presented in their paper, an adaptation of the feature extraction process is presented in this chapter. The idea is to classify the progression of the facial expression using LDSP rather than looking at the expression in a single RGB-image. The proposed implementation is named **LDSP-RNN** in this thesis. This will not be implemented on the robot itself, but trained and tested offline.

5.1 Approach

Applying LDSP on the input images causes some information to disappear due to noise and other variations in the dataset. To check whether LDSP in sequence poses as an alternative to a CNN approach and the difference in performance between LDSP-RNN and CNN(from chapter 4) pose as the best option to estimate the benefits and drawbacks. This is compared with an RNN taking raw grayscale value input images as 1-dimensional vectors and a Soft Vector Machine with both LDSP and raw grayscale images as input.

5.1.1 Making Sequence Dataset

The first stage in training and testing new models and methods is constructing and preparing the datasets. RNNs, as stated earlier, needs a sequence of data over time. For example, in this case, facial expressions evolving over a series of different images. If the input sequence is too long, the RNN will "forget" during the processing of the images, resulting in poor performance. The key is then to find a sequence length which is not too long but still maximizes the RNNs information extraction. Given that the shortest length of a sequence in the benchmark dataset[2] is 5 images, the number of input samples is then set to 5. Figure 5.1 are sample sequences of three images per sequence progressing from a small degree of that expression to a full display. The illustrated expression are disgust, surprise, and anger respectively.

The selected image sequences are then divided into three subsets; training set, validation set, and test set. Where the training set is approximately 50%, validation 25% and test set 25% of the original dataset.

The LDSP-RNN model expects a fixed length on the input sequence, and since the length of a sequence in the CK+ dataset varies between 5 and 70 images, alterations have been made. In the CK+ dataset documentation[2] it is stated that the emotions evolve from neutral to full display within each sequence. This has also been verified by looking at the images manually. To extract the most potent development of each sequence when constructing the adjusted dataset, a selection of 5 images covering the steepest "emotion-gradient" is extracted. By selecting each of these 5 positions within a sequence with varying length; first, center-left, center, center-right, last - the new sequences will have an approximately equal progression of emotion display invariant to the original sequence length. The assumption is that the

highest gradient lies in the middle of each sequence.

Figure 5.1: Emotions evolving over time.

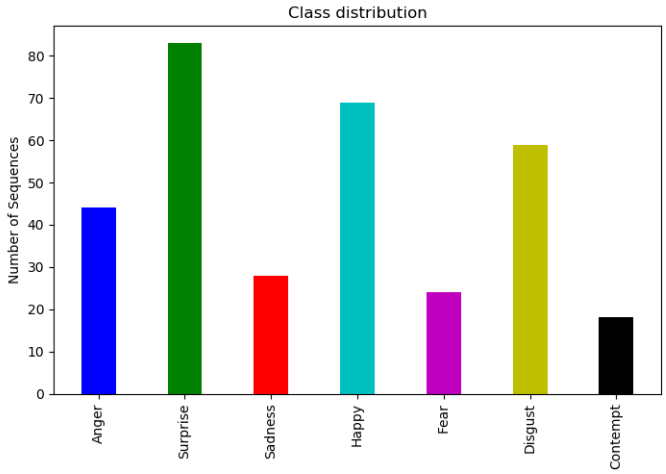


Figure 5.2: Class distribution in sequence dataset.

5.1.2 Image Preprocessing

Before introducing the image sequences to the RNN model, feature extraction is needed to enhance the classification accuracy as described in the paper[1]. The camera produces RGB images, but the benchmark dataset is grayscale contrary to [1] which is trained on depth images. This may introduce more noise compared to the previous work since the presence of edges is far greater

in raw RGB/gray images than depth images, but on the other hand, it might find variations that can lead to a better classification.

As stated earlier, in Uddin, Khaksar and Torresen's paper *Local Directional Strength Pattern* and *Local Directional Ranking Histogram Pattern* is extracted from each image in the sequence and combined, where the LDSP is augmented with LDRHP. Dimensionality reduction is applied through *Kernel Principal Component Analysis* and *Generalized Discriminant Analysis* which results in a 1-dimensional vector for each image in a sequence. So the input is a 3 dimensional matrix with shape [***number-of-images*** x ***sequence-length*** x ***features***].

Figure 5.3: Kirsch mask applied in all 8 directions on a single grayscale image.

In the implementation of LDSP-RNN, and subsequent experiments, a reduction of the feature extraction process down to *Local Directional Strength Pattern*[1] was done, enabling to experiment with the effect of solely applying a robust gradient descriptor in sequence into an RNN. LDSP is directly fed into the RNN without the conversion to a histogram representation. The extraction method is described in figure 5.4 where the binary representation is converted into decimal numbers. In figure 5.3 an image after applying Kirsch edge detection is depicted and is what the LDSP is working on.

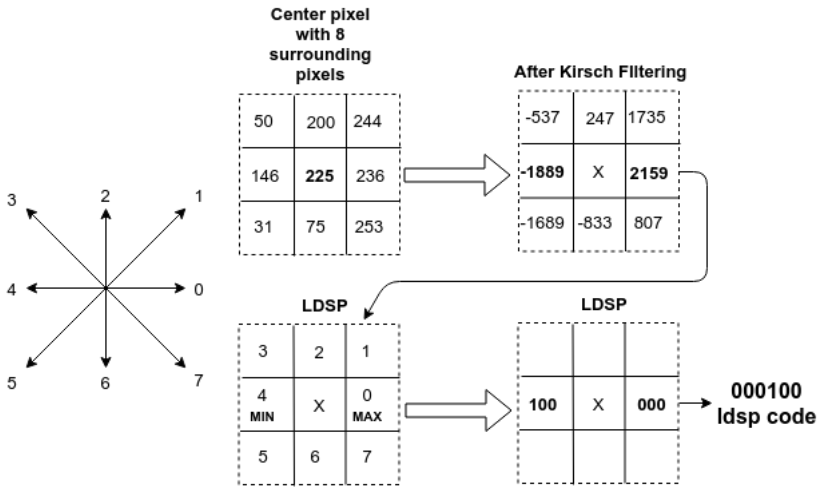
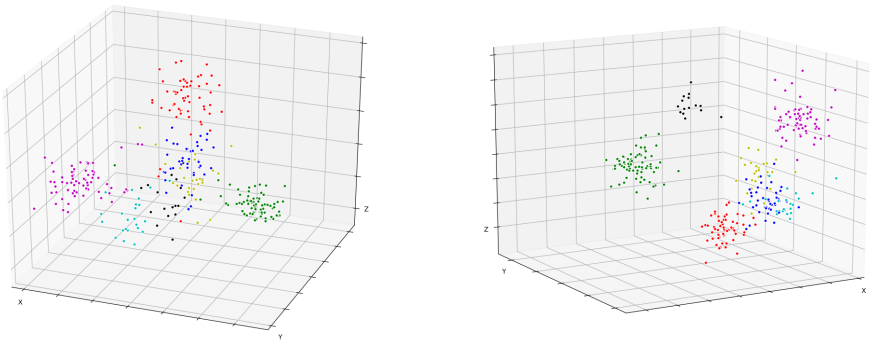


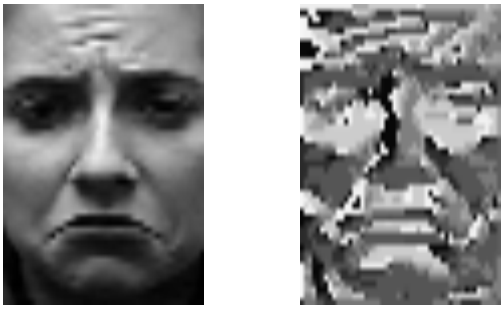
Figure 5.4: Local Directional Strength Pattern on a 3x3 pixel neighborhood in an image. The resulting LDSP code is now a representation for the center pixel. This is repeated for all pixels of an image.

5.2 Experiments & Results



(a) Grayscale images (b) LDSP-features

Figure 5.5: 3D-plot of raw pixel values and LDSP-features on grayscale images using LDA dimension reduction. Anger(blue) - Contempt(black) - Disgust(red) - Fear(cyan) - Happy(pink) - Sad(yellow) - Surprise(green)



(a) Grayscale image of sad

(b) Image after Kirsch mask and LDSP

Figure 5.6: Grayscale image before and after applying Kirsch mask and Local Directional Strength Pattern.

5.2.1 Visualization of Features

In figure 5.5 Linear Discriminant Analysis has been applied to the last frame of each sequence, reducing the dimension of the feature space down to 3 axes. It is done purely for illustration purposes, and is not applied on the data before training of the LDSP-RNN. LDA chooses axes based on their ability to separate the different classes. Compared to the 3D plot in Uddin, Khaksar and Torresen's paper[1] where they manage to separate all the 6 different facial expressions with the use of Kernel Principal Component Analysis and General Discriminant Analysis which utilizes a non-linear discriminant function, the plot in figure 5.5 does not show as good separation. Since the images used in this experiment are raw grayscale images displaying seven different emotions contrary to six class display on clear and highly detailed depth images used in Uddin, Khaksar and Torresen's paper, it is not expected to show equally good results.

Defining a decision line between the grouped classes can be a challenging problem to solve. The results in chapter 4 indicate a problem when trying to separate seven different classes from the public dataset. The variations in the CK+ dataset pose as the main obstacle when training a model, irrelevant to what type of deep learning architecture used. The results in this chapter bring light to the major faults associated with this dataset.

5.2.2 Training & Testing of Recurrent Neural Network Approach

By looking at the progression of the training in the LDSP-RNN approach, the model overfit very quickly indicating a possibility that the dataset is too small (see figure 5.7). With 325 samples in total (50% for training, 25% for testing and 25% for validation) where class representations vary greatly, none of the different approaches (listed in table 5.1) manage to achieve a classification accuracy above 81%. The RNN model contains 100 recurrent cells, meaning the data is processed 100 times per time step, see figure 2.11.

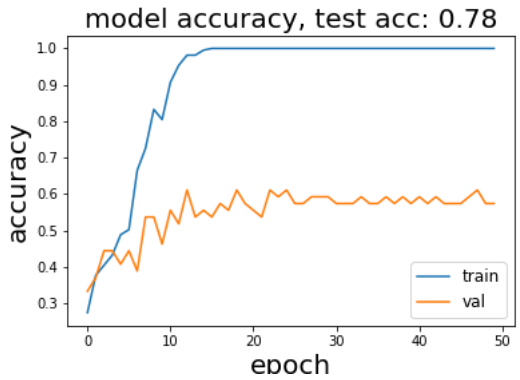


Figure 5.7: Best training run of LDSP-RNN model.

RAW-IMAGE-RNN

When applying raw grayscale images as input to the RNN, consisting of 100 recurrent cells and sigmoid activation, the performance comes short with roughly 11 percentage points on average recognition rate (72.8% contrary to 84.3%) compared to the seven class facial expression model in chapter 4 which is trained on the peak expressions from the CK+ dataset (see section 4.2).

LDSP-RNN

Introducing the RNN with LDSP feature vectors as input (**LDSP-RNN**), the average recognition rate is increased to 73.6%. The model has been trained over 10 different runs, similar to the competing approaches in this

chapter. Though the RNN with raw images as input (RAW-IMAGE-RNN) produce a higher top recognition rate (81%), the average is less than LDSP-RNN's average recognition rate, indicating that with the use of LDSP features it produces more robust results in the form of a lower standard deviation and a higher mean value (see table 5.1). Note that with so few testing images it is hard to conclude, but it indicates a possible outcome when fine-tuning hyperparameters and utilizes a better dataset for training.

Soft Vector Machine

As for comparing RNN to a standard image classifier, SVM is trained and tested on the same input data with a gamma parameter value of 0.01. For both cases of LDSP and raw grayscale images as input, it produced the same recognition rate of 72%.

5.2.3 Overview of Results

In table 5.1 the result of 10 training iterations with 50 epochs per iteration is presented. Since deep neural networks are stochastic, a different result can be produced for each of the separate training runs, meaning multiple runs has to be applied to get statistically viable results. In figure 5.8 the average recognition rate from each approach on seven classes is illustrated and showing that LDSP-RNN produces the best outcome. The confusion matrices from testing of LDSP-RNN and RAW-IMAGE-RNN are presented in figure 5.9, highlighting the emotions that both approaches struggle to classify. Contempt, anger, and sadness are the classes that show the most prominent variations between the two models. Both anger and sadness are present in the cluster of inseparable classes illustrated in figure 5.5. Contempt is most likely misclassified due to low class representation in the training set as discussed in chapter 3.

	Top	Mean	Std
RNN w/ Raw image	0.81	0.728	0.086
RNN w/ LDSP features	0.78	**0.736**	**0.023**
SVM w/ Raw image	0.72	0.72	0
SVM w/ LDSP features	0.72	0.72	0

Table 5.1: Test results from the different approaches with top, mean and standard deviation of the 10 separate runs.

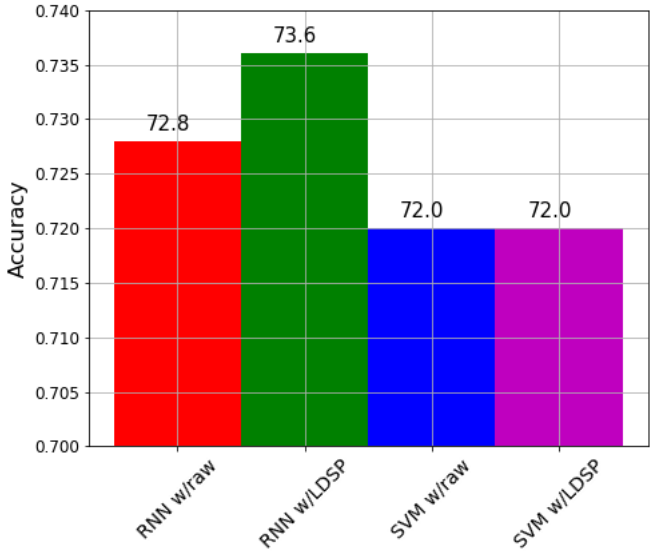
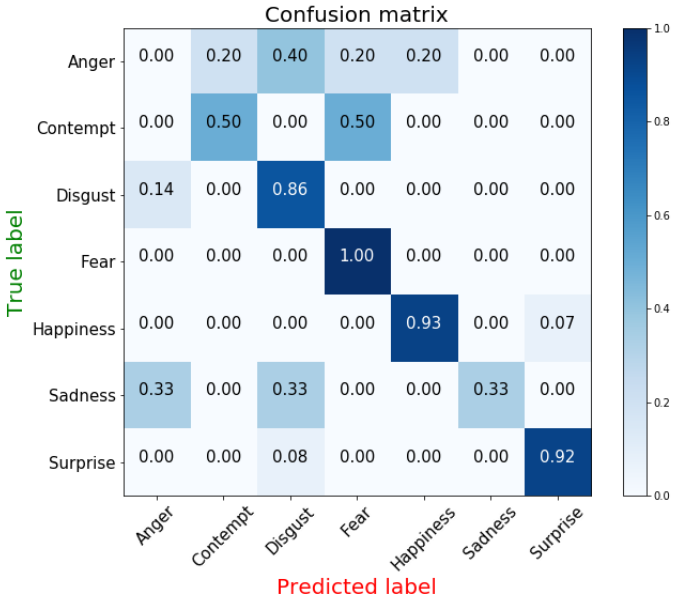
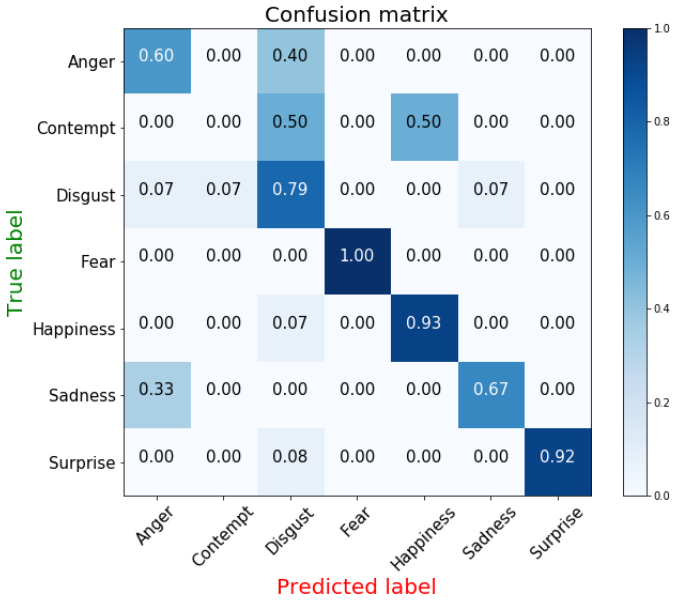


Figure 5.8: Model performances on raw image features and LDSP features.



(a) Confusion matrix LDSP-RNN, showing recall for each emotion.



(b) Confusion matrix RAW-IMAGE-RNN, showing recall for each emotion.

Figure 5.9: Confusion matrices from RNN emotion recognition on test set.

Chapter 6

Discussion

In this chapter results from both experiments, *Real-time Face Detection, Identification & Emotion Recognition* and *Emotion Recognition Using Salient Features & RNN in video* are discussed.

6.1 Real-Time Emotion Recognition Pipeline

The results have proved to be significant in the form of implementing a functional real-time emotion recognition system and establishing the benefits of using a personalized dataset contrary to a public dataset, although it is common knowledge that personalized models outperform generalized models. As discussed in chapter 3, there are some distinctions between the two dataset to be noted, such as different class distributions and RGB-images contrary to grayscale images. Despite these variations, the ten different runs testing the models reveal that the model based on a personalized dataset outperforms the model based on a public dataset in every instance, regardless of number of classes to predict. On average, the the generalized model (CK+) achieved a recognition rate of 84.3% on seven classes, whereas the personalized model got a recognition rate of 97.6% on the same amount of classes.

Since the variance in the public dataset is more significant, a deeper classification model is necessary to increase accuracy. A deeper neural network is more capable of understanding a complex structure within the data, as discussed in the background and in LeCun, Bengio and Hinton's paper[10]. It is reasonable to assume that a deeper model would process data at a slower rate since the number of calculations significantly increase with incrimination of the number of hidden layers. The Multimodal Elderly Care System is primarily developed for assisting a single user, which means it has to familiarize itself with that user. As shown in the results, a fairly shallow CNN can accomplish an accuracy up to 98% on seven different expressions with the use of techniques which is quite standard in today's image classification field, such as Batch Normalization, Dropout, etc. The personalized model can learn more complex features that describes the user's facial expression with fewer calculations, ergo faster processing.

When reviewing the results for each of the three cases (two-, four-, and seven-classes), one can see that the difference in accuracy between the CK+ trained models and the personalized models with fewer classes at the output are not as prominent as the models with a higher number of classes. Happy and Sad is quite different expressions, making the two-class problem an easier task to solve. It is when the number of classes is increased that a noticeable difference in performance develops. For seven classes the personalized models have almost ten percentage points higher accuracy than the CK+ models. Also, the variance in both loss and accuracy of the CK+ models are quite high, as can be seen in the plots in the appendix. Figures 8.3 and 8.5

clearly indicate that the models are struggling to generalize good features for classification. Since the variance in the CK+ dataset is much higher to begin with, one should train the models with a lower learning rate or the number of validation samples should be increased to reduce the variance in the validation accuracy and loss.

Validation accuracy and loss are only indications of how training progresses. Since the training accuracy and loss looked promising and test accuracy somewhat matched what is to be expected, a decision not to increase the number of validation samples was made since this would mean a reduction in training samples. At the same time minimizing the differences in the outset for both the personalized models and the generalized models, i.e. models trained on the CK+ public dataset.

One of the disadvantages of basing the models on a personalized dataset, is the accumulation process of a proper dataset containing every expression of its user. Ideally, a well-performing model trained on a public dataset pose as the best option, since it does not need to be tailored to each person and therefore only needs to be trained once. However, as shown in the thesis, small, fast CNNs trained on multiple different faces cannot produce the desired result as it can with a personalized dataset.

6.2 Emotion Recognition Using Salient Features & Recurrent Neural Network in Videos

In "Facial expression recognition using salient features and convolutional neural network"[1] the model was trained on a depth image dataset and included a small experiment on a selected number of samples from the CK+ dataset[2]. A different approach has been presented in this thesis based on their work, by essentially reducing the feature extraction part to solely Kirsch edge detection[5] and Local Directional Strength Pattern[1]. A recurrent neural network is trained with these features and called LDSP-RNN. It accomplished an average recognition rate of 73.6% on the test set. The results produced by this implementation cannot match what Uddin, Khaksar and Torresen accomplished with their approach, due to several reasons. Firstly, it is not sustainable to compare the results, since they selected 40 great samples with short sequence lengths and used a more comprehensive feature extraction method. Nonetheless, it is not accurate to conclude that

the approach presented in this thesis come short of good results. Taking into account that the RNN implementation considers all the seven different classes described in chapter 3 as well as utilizing all samples available in the CK+ dataset, it leaves the possibility of an increased recognition rate if presented with better samples, i.e. filtering out samples that would distort the network's understanding of the problem.

The results obtained show a better average recognition rate using LDSP-RNN compared to RAW-IMAGE-RNN and LDSP-RNN outperformed a Soft Vector Machine classification in both cases, as presented in table 5.1. Understanding that edge detectors are sensitive to noise in the image is a key factor when selecting dataset to be used for training. Utilizing the whole CK+ dataset for training has shown not to be ideal when applying an edge detector for feature extraction. Noise can very easily be amplified, causing worse performance during training. Filtering techniques to remove noise pose as an option for further work within this implementation. Based on the results presented in chapter 5 section 5.2, the LDSP-RNN approach has shown much potential, and with the correct parameters and training data it is most likely a suitable choice for classifying facial expressions in videos.

6.3 Ethical Dilemmas

Considering that the robot is aimed to be used in the users' homes, there are several things to take into account regarding the implementation. The welfare and interests of the users must always be respected above anything else, and it is reasonable to discuss and evaluate the possibilities of malfunction, malpractice and exploitation of both the robot's behavior and the information it can collect and process, combined with how this information is to be distributed to persons of interest, i.e. family and healthcare personnel. The worst case scenario would be illegal distribution of private and sensitive information about a reasonably vulnerable group in our population, seniors, due to their vulnerability when it comes to burglary and fraud. It is our responsibility as system designers to ensure that this does not occur.

With the rapid advancements in technology, it can be argued that safety also progresses. Such as our life expectancy with the help of modern medicine and technology, but that is not necessarily true in the case of information security. We now live in an era of information where we have reached a paradigm shift. Social media has contributed significantly to what we now

consider private information. Every internet site we visit collects information about its user and we trust online solutions with our pictures and other private files. The only thing keeping this from being distributed across the world for everyone to see is the security measures the storage providers have implemented. It is a constant battle between new security updates and malicious software trying to reach private data. The companies also have a responsibility not to misuse the information provided by the user. Privacy legislation set by the government is there to ensure that privacy is respected and not exploited in a negative way. However, there is always a risk that the rules are not obeyed or that loopholes can be found and exploited.

Despite the risk we take entrusting that our information is safe, it has come to good use. By allowing technology to make use of information that is commonly not accessible, e.g. facial expressions, we can build and develop solutions which will help us and hopefully increase our standard of living. Introducing assisting technologies into elderly peoples' homes may enable the user to live at home for a more extended period, relieving resources to be used elsewhere. By limiting the information the robot distributes to entrusted personnel, for example with encryption or a purer form of communication, it is possible to reduce the risk of sensitive information to go astray. With this in mind, the robot needs to process more of the information it collects locally. Applying the implementation presented in chapter 4, a complex mental health analysis, compared to existing facial expression recognition systems, is possible without the need of cloud-based processing.

Chapter 7

Conclusion

This chapter presents a conclusion to the work presented in this thesis, including a proposition of future work related to chapter 4 and 5.

7.1 Real-Time Emotion Recognition Pipeline

The thesis has presented an approach for real-time emotion recognition of human faces, which includes face detection and identification. By comparing public and private datasets, benefits of using a personalized dataset for classification of multiple expressions and surpassing previous number of classifications in related work have been established. The introduction described limitations which may be highlighted when using deep learning models, especially image analysis, on a mobile robot without high performance hardware. With the presented approach, a personalized real-time emotion recognition system with the ability of classifying seven different expressions is capable of running smoothly on a standard CPU, making it somewhat insensitive to different hardware specifications.

7.1.1 *Is it Possible to Process Facial Expressions at a Reasonable Speed in Real-Time on a Standard Laptop Computer, Without the Help of a GPU?*

Yes, but with some precautions. When increasing the resolution of the input image stream, processing speed is dramatically reduced. This is not an issue when it comes to facial expression recognition since the CNN takes a fixed size image as input, 100x100 pixels, but it is a problem for the face detection algorithm. The experiments are conducted on an image stream where each frame has a resolution of 640x480 pixels. If the resolution is increased to 1280x720 (HD-ready), the number of pixels to process triples in size. With the use of a GPU this resolution is not an issue, but with the assumption that all images have a resolution around 640x480, it is possible to detect faces, identify and classify seven different facial expressions at 20 FPS.

7.1.2 *How Well Does a Personalized Model Perform Compared to a Generalized Model?*

A personalized model (trained on images from one person) will always outperform a generalized model (trained on images of multiple people) in every instance of number of classes to predict (see section 4.2), but with the use of a small customized CNNs complicated facial expressions can be recognized

with fewer training iterations and faster calculations.

7.2 Emotion Recognition Using Salient Features & Recurrent Neural Network in Video

In this chapter(see chapter 5) an adaptation of the features extraction process presented in [1] combined with a recurrent neural network was proposed, and named LDSP-RNN. With a standard RNN architecture with LDSP features as input, the model managed to get an average recognition rate of 73.6% on the test set of seven classes (see section 5.2). The CNN version's top performing model trained on the CK+ dataset got 88% accuracy (see section 4.2), but with an average of around 84.3% over ten runs. Note that the LDSP-RNN was trained on the CK+ dataset and is therefore compared with the CNN model trained on the same dataset only with single image input. As a starting point, we can conclude that an LDSP-RNN model cannot compete with a CNN approach of single image *peak emotion display* classification. LDSP-RNN, on the other hand, produces an overall better recognition rate compared to a RAW-IMAGE-RNN approach, proving that LDSP features pose as a suitable option for classifying facial expressions and complies with the findings presented in Uddin, Khaksar and Torresen's paper[1].

7.2.1 *Can Oriented Gradient Features in Sequence Combined With an RNN Pose as an Alternative to Standard CNNs?*

Under the circumstances presented in this thesis it is not able to compete with the CNN approach, but with a better dataset combined with an expansion of the feature extraction process, the possibility of an equally or better performance is very much present.

7.3 Dissemination

During the work on this thesis, the Minister of Commerce (Torbjorn Roe Isaksen) and the Minister of Higher Education (Iselin Nybo) showed interest

in projects within Robotics and Artificial Intelligence due to higher priorities within these fields of inventions. A presentation and live demonstration of the introduced approach for real-time emotion recognition was held, and was broadcasted on the evening news on NRK[52], and morning news on TV2[53]. Links to both of the news stories are provided in the bibliography, see citation above.

7.4 Future Work

7.4.1 Real-Time Emotion Recognition System

As discussed in section 6.1, a generalized model would be preferable when constructing a facial expression recognition system. This is due to the comprehensive work of collecting a person dependent dataset for each user of the system. Utilizing more of the work described in "MobileFaceNets: Efficient CNNs for Accurate Real-time Face Verification on Mobile Devices" by Chen et al.[42], would enable the use of deeper CNNs with fewer parameters to update during training. It can be applied to all stages in the pipeline, given that a CNN currently is the preferred architecture for object detection and classification. This can lead to experiments using the approach presented in "HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition" by Ranjan, Patel and Chellappa[34], constraining face detection, identification and emotion recognition to a single convolutional neural network.

Due to time limitations on implementing and writing this thesis, testing of the proposed system on a moving robot in different environments has not been done, where areas of improvements are more easily detected.

7.4.2 Emotion Recognition Using Salient Features & Recurrent Neural Network in Video

During this thesis much attention has been put to the drawbacks of using the CK+ dataset[2], leaving the possibility of an improvement if a different dataset had been used. Furthermore, experimenting with a greater variety of RNN architectures and parameters can lead to new findings and hope-

fully increase the recognition rate of an LDSP-RNN on facial expressions. By expanding the feature extraction process before training an RNN, a better separation of expressions is possible. Based on the robustness of LDSP presented in this thesis and [1], one can utilize the techniques used in Uddin, Khaksar and Torresen's paper and expect better results. In section 5.2.1 LDA was applied for vizualization of features in three dimensions. In ongoing experiments LDA has been included in the feature extraction process, reducing the feature space down to 6 dimensions before applying RNN. It is showing promising results, surpassing approaches presented in chapter 5.

Bibliography

[1] Md. Zia Uddin, Weria Khaksar and Jim Torresen. "Facial expression recognition using salient features and convolutional neural network". In: *IEEE Access* 5 (2017), pp. 26146–26161. ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2777003. URL: http://ieeexplore.ieee.org/document/8119492/.

[2] P. Lucey et al. "The extended cohn-kande dataset (CK+): A complete facial expression dataset for action unit and emotionspecified expression". In: *Cvprw* July (2010), pp. 94–101. ISSN: 01628828. DOI: 10.1109/ISIEA.2010.5679500.

[3] Timo Ahonen, Abdenour Hadid and Matti Pietikäinen. "Face description with local binary patterns: Application to face recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (2006), pp. 2037–2041. ISSN: 01628828. DOI: 10.1109/TPAMI.2006.244.

[4] Taskeed Jabid, Md Hasanul Kabir and Oksam Chae. "Local directional pattern (LDP) for face recognition". In: *International Journal of Innovative Computing, Information and Control* 8.4 (2012), pp. 2423–2437. ISSN: 13494198. DOI: 10.1109/ICIP.2010.5652374.

[5] Taskeed Jabid, Md. Hasanul Kabir and Oksam Chae. "Local Directional Pattern (LDP) – A Robust Image Descriptor for Object Recognition". In: *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, Aug. 2010, pp. 482–487. ISBN: 978-1-4244-8310-5. DOI: 10.1109/AVSS.2010.17. URL: http://ieeexplore.ieee.org/document/5597095/.

[6] Kamrul Hasan Talukder and Koichi Harada. *Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image*. Tech. rep. URL: https://arxiv.org/pdf/1010.4084.pdf.

[7] Constantine P Papageorgiou, Michael Oren and Tomaso Poggio. *A General Framework for Object Detection*. Tech. rep. URL: https://pdfs.semanticscholar.org/76f5/60991d56ad689ec32f9e9d13291e0193f4cf.pdf.

[8] Paul Viola and Mj Jones. "Robust real-time face detection". In: *International journal of computer vision* 57.2 (2004), pp. 137–154. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000013087.49260.fb. URL: http://link.springer.com/article/10.1023/B:VISI.0000013087.49260.fb.

[9] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. Vol. I. IEEE, 2005, pp. 886–893. ISBN: 0769523722. DOI: 10.1109/CVPR.2005.177. URL: http://ieeexplore.ieee.org/document/1467360/.

[10] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539. URL: http://www.nature.com/articles/nature14539.

[11] Shima Alizadeh and Azar Fazel. "Convolutional Neural Networks for Facial Expression Recognition". In: *arXiv preprint arXiv:1704.06756* (2017). ISSN: 21945357. DOI: 10.1007/978-3-319-47952-1{_}6. URL: http://arxiv.org/abs/1704.06756.

[12] Ihor Paliy. "Face Detection Using Haar -like Features Cascade and Convolutional Neural Network". In: *2008 Proceedings of International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science* (2008), pp. 375–377. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5423372.

[13] Steve Lawrence et al. "Face Recognition : A Convolutional Neural Network Approach". In: *Neural Networks* 8.1 (1997), pp. 98–113. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=554195.

[14] S Guo, S Chen and Y Li. "Face recognition based on convolutional neural network & support vector machine". In: *2016 IEEE International Conference on Information and Automation, IEEE ICIA 2016* August (2017), pp. 1787–1792. DOI: 10.1109/ICInfA.2016.7832107. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015775572&doi=10.1109%2FICInfA.2016.7832107&partnerID=40&md5=2523c4287769e853e4620a26fa58f701.

[15] Yann LeCun et al. “A theoretical framework for back-propagation”. In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann. 1988, pp. 21–28.

[16] George E. Dahl, Tara N. Sainath and Geoffrey E. Hinton. “Improving deep neural networks for LVCSR using rectified linear units and dropout”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2013, pp. 8609–8613. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6639346. URL: http://ieeexplore.ieee.org/document/6639346/.

[17] Zhou Wang and A.C. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (Jan. 2009), pp. 98–117. ISSN: 1053-5888. DOI: 10.1109/MSP.2008.930649. URL: http://ieeexplore.ieee.org/document/4775883/.

[18] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (Feb. 2015). URL: http://arxiv.org/abs/1502.03167.

[19] Jimmy Lei Ba, Jamie Ryan Kiros and Geoffrey E. Hinton. “Layer Normalization”. In: (July 2016). URL: http://arxiv.org/abs/1607.06450.

[20] Yanghao Li et al. “Revisiting Batch Normalization For Practical Domain Adaptation”. In: (Mar. 2016). URL: http://arxiv.org/abs/1603.04779.

[21] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. ISSN: 15337928. DOI: 10.1214/12-AOS1000. URL: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf.

[22] Pavel Golik, Patrick Doetsch and Hermann Ney. *Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison*. Tech. rep. 2013. URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2013/i13_1756.pdf.

[23] Douglas M. Kline and Victor L. Berardi. “Revisiting squared-error and cross-entropy functions for training neural network classifiers”. In: *Neural Computing and Applications* 14.4 (Dec. 2005), pp. 310–318. ISSN: 0941-0643. DOI: 10.1007/s00521-005-0467-y. URL: http://link.springer.com/10.1007/s00521-005-0467-y.

[24] Zhilu Zhang and Mert R. Sabuncu. "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels". In: (May 2018). URL: http://arxiv.org/abs/1805.07836.

[25] Diederik P Kingma and Jimmy Lei Ba. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. Tech. rep. URL: https://arxiv.org/pdf/1412.6980.pdf.

[26] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[27] John Duchi, Elad Hazan and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159. ISSN: ISSN 1533-7928. URL: http://www.jmlr.org/papers/v12/duchi11a.html.

[28] Luis Perez and Jason Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: (Dec. 2017). URL: https://arxiv.org/abs/1712.04621.

[29] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[30] Zack Thoutt. *GOT Book 6 Generator*. URL: https://github.com/zackthoutt/got-book-6.

[31] Samira Ebrahimi Kahou et al. "Recurrent Neural Networks for Emotion Recognition in Video". In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction - ICMI '15*. New York, New York, USA: ACM Press, 2015, pp. 467–474. ISBN: 9781450339124. DOI: 10.1145/2818346.2830596. URL: http://dl.acm.org/citation.cfm?doid=2818346.2830596.

[32] Quoc V. Le, Navdeep Jaitly and Geoffrey E. Hinton. "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units". In: (Apr. 2015). URL: https://arxiv.org/abs/1504.00941.

[33] Ali Sharifara, Mohd Shafry Mohd Rahim and Yasaman Anisi. "A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection". In: *Proceedings - 2014 International Symposium on Biometrics and Security Technologies, ISBAST 2014* (2015), pp. 73–78. DOI: 10.1109/ISBAST.2014.7013097.

[34] Rajeev Ranjan, Vishal M. Patel and Rama Chellappa. "HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* XX.Xx (2017), pp. 1–16. ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2781233.

[35] Haoxiang Li et al. "A convolutional neural network cascade for face detection". In: *Cvpr* (2015), pp. 5325–5334. ISSN: 1063-6919. DOI: 10.1109/CVPR.2015.7299170.

[36] Davis. E. King. "Dlib-ml: A Machine Learning Toolkit". In: *Journal of Machine Learning Research* 10 (2009), pp. 1755–1758. ISSN: 15324435. DOI: 10.1145/1577069.1755843. URL: http://jmlr.csail.mit.edu/papers/v10/king09a.html.

[37] G Bradski. "The OpenCV Library". In: *Dr Dobbs Journal of Software Tools* 25 (2000), pp. 120–125. ISSN: 1044-789X. DOI: 10.1111/0023-8333.50.s1.10. URL: http://opencv.willowgarage.com.

[38] João F. Henriques et al. "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels". In: Springer, Berlin, Heidelberg, 2012, pp. 702–715. DOI: 10.1007/978-3-642-33765-9{_}50. URL: http://link.springer.com/10.1007/978-3-642-33765-9_50.

[39] Martin Danelljan et al. "Adaptive Color Attributes for Real-Time Visual Tracking". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[40] *haar-feature-image*. URL: http://siret.ms.mff.cuni.cz/facereco/method.

[41] Yaniv Taigman et al. "DeepFace: Closing the gap to human-level performance in face verification". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708. ISSN: 10636919. DOI: 10.1109/CVPR.2014.220.

[42] Sheng Chen et al. "MobileFaceNets: Efficient CNNs for Accurate Real-time Face Verification on Mobile Devices". In: (Apr. 2018). ISSN: 16113349. DOI: arXiv:1804.07573v2. URL: http://arxiv.org/abs/1804.07573.

[43] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICRL)* (Sept. 2015), pp. 1–14. ISSN: 09505849. DOI: 10.1016/j.infsof.2008.09.005. URL: http://arxiv.org/abs/1409.1556.

[44] Md. Uddin, J. Lee and T.-S. Kim. "An enhanced independent component-based human facial expression recognition from video". In: *IEEE Transactions on Consumer Electronics* 55.4 (2009), pp. 2216–2224. ISSN: 0098-3063. DOI: 10.1109/TCE.2009.5373791.

[45] Md Hasanul Kabir et al. "Facial Expression Recognition from Depth Video with Patterns of Oriented Motion Flow". In: *IEEE Access* 5 (2017), pp. 8880–8889. ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2704087.

[46] N. Sebe et al. "Authentic facial expression analysis". In: *Image and Vision Computing* 25.12 (2007), pp. 1856–1863. ISSN: 02628856. DOI: 10.1016/j.imavis.2005.12.021.

[47] Ian Goodfellow et al. *Challenges in Representation Learning: A report on three machine learning contests*. 2013. URL: http://arxiv.org/abs/1307.0414.

[48] Paul Ekman and Wallace V Friesen. *Manual for the facial action coding system*. Consulting Psychologists Press, 1978.

[49] M Jordan, J Kleinberg and B Schölkopf. *Pattern Recognition and Machine Learning*. Tech. rep. URL: http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf.

[50] *Scale Image*. URL: https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection#TOC-Image-Pyramid.

[51] *object detection - OpenCV detectMultiScale() minNeighbors parameter - Stack Overflow*. URL: https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter.

[52] *NRK Evening News*. 2018. URL: https://tv.nrk.no/serie/kveldsnytt/NNFA23090218/02-09-2018.

[53] *TV2 Morning News*. 2018. URL: https://youtu.be/m5yu2BimALk.

Chapter 8

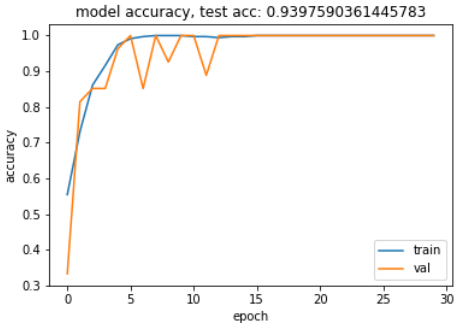
Appendix

8.1 Sensors, Software & Hardware

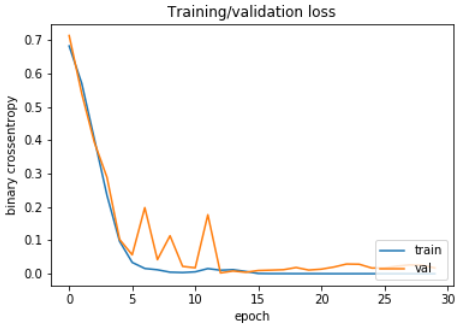
- Samsung ultrabook series 9
- Intel Core i7-3517U CPU - 1.90GHz x 4
- Microsoft 1080p HD sensor camera
- Ubuntu 16.04
- Python 3.6.6
- TensorFlow 1.8
- Keras 2.1.6
- OpenCV 3.4.2.17
- dlib 19.7.0

8.2 Emotion Recognition Training Plots

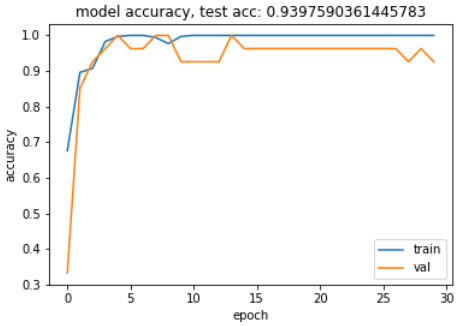
8.2.1 Two Classes



(a) Accuracy run 7/10

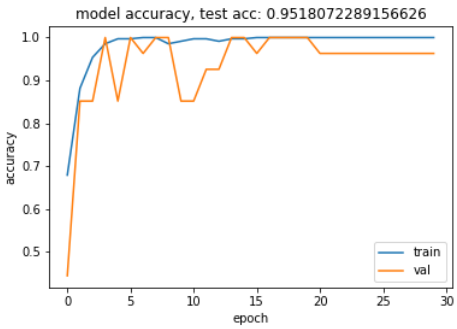


(b) Loss run 7/10

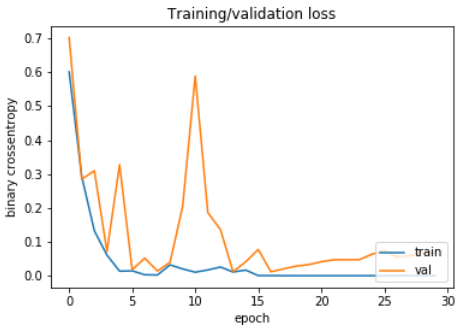


(c) Accuracy run 8/10

(d) Loss run 8/10

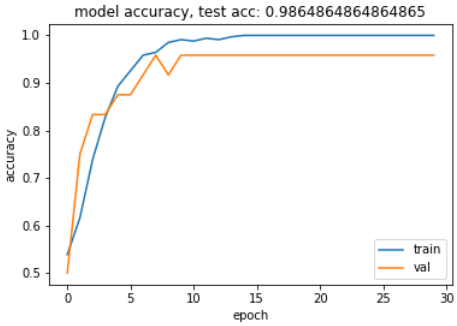


(e) Accuracy run 10/10

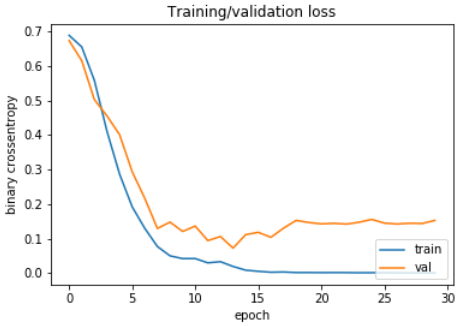


(f) Accuracy run 10/10

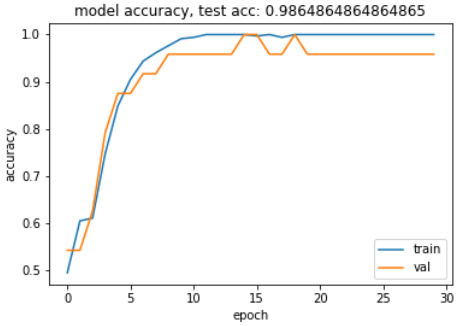
Figure 8.1: Cohn-Kanade - Two-class training and validation accuracy and categorical cross entropy loss over number of epochs



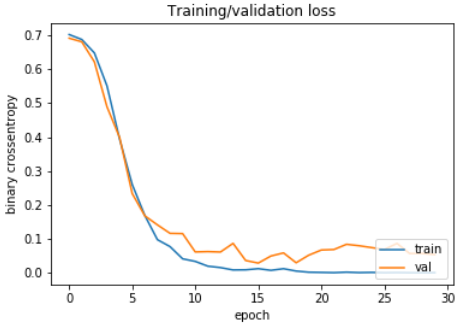
(a) Accuracy run 5/10



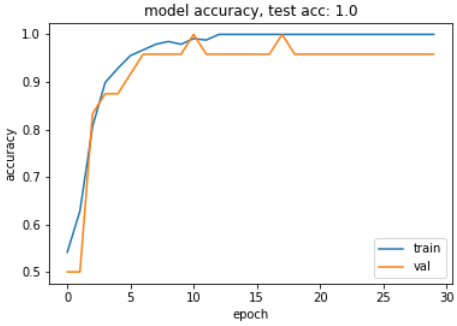
(b) Loss run 5/10



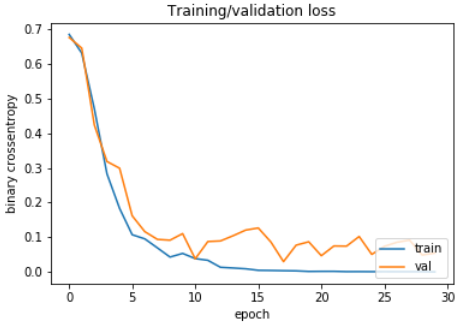
(c) Accuracy run 8/10



(d) Loss run 8/10



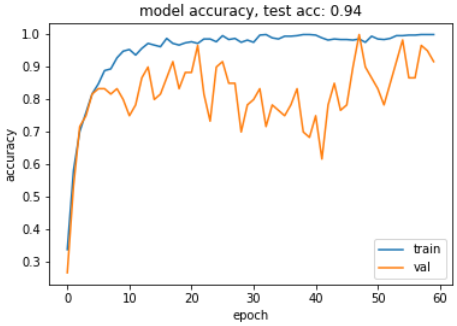
(e) Accuracy run 10/10



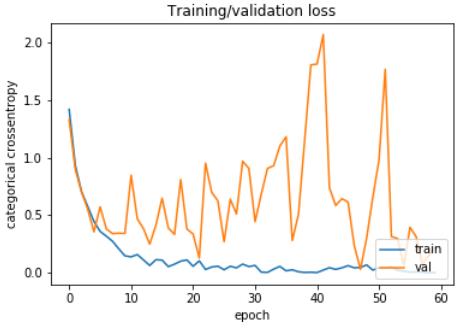
(f) Loss run 10/10

Figure 8.2: Personalized - Two-class training and validation accuracy and categorical cross entropy loss over number of epochs

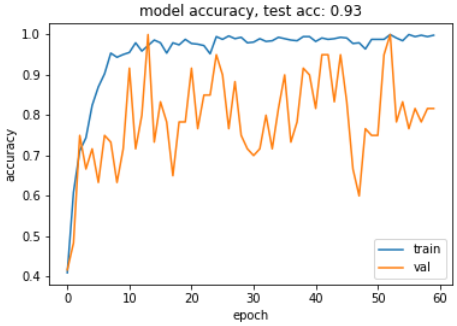
8.2.2 Four Classes



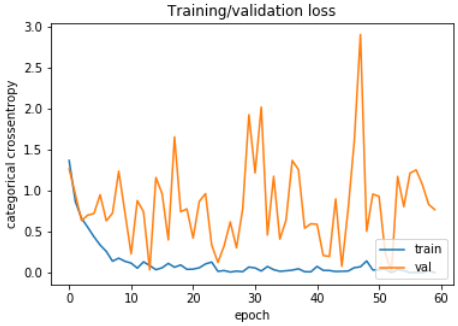
(a) Accuracy run 1/10



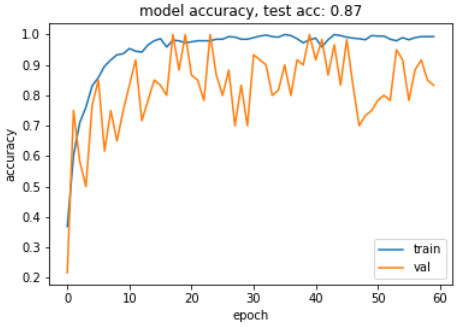
(b) Loss run 1/10



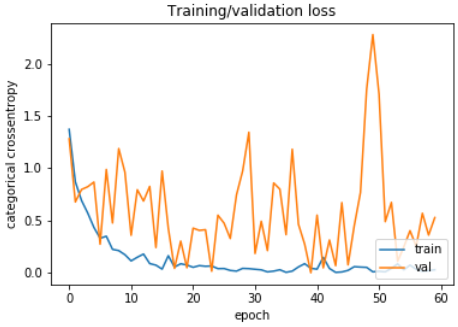
(c) Accuracy run 4/10



(d) Loss run 4/10

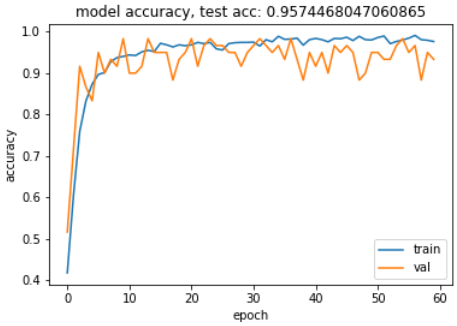


(e) Accuracy run 9/10

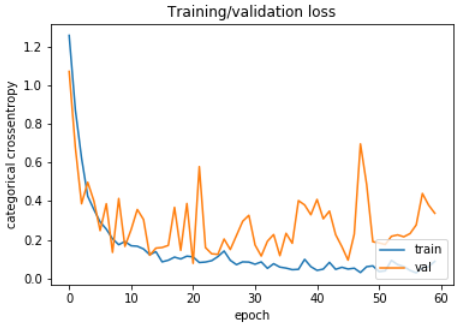


(f) Loss run 9/10

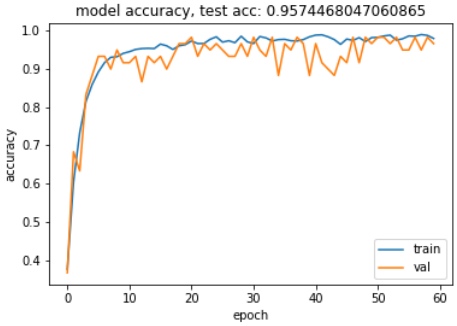
Figure 8.3: Cohn-Kanade - Four-class training and validation accuracy and categorical cross entropy loss over number of epochs



(a) Accuracy run 2/10

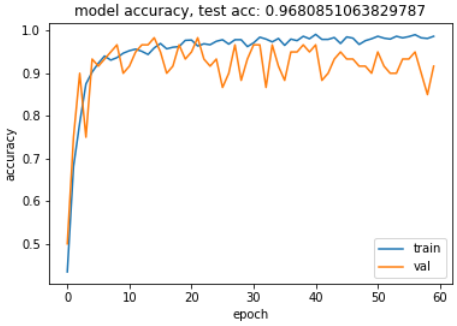


(b) Loss run 2/10

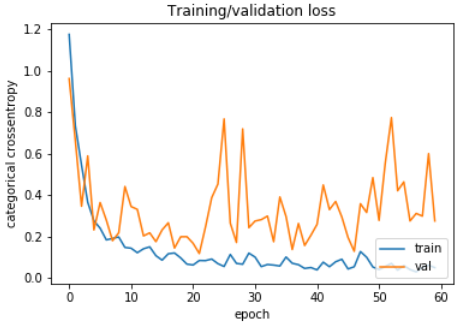


(c) Accuracy run 5/10

(d) Loss run 5/10



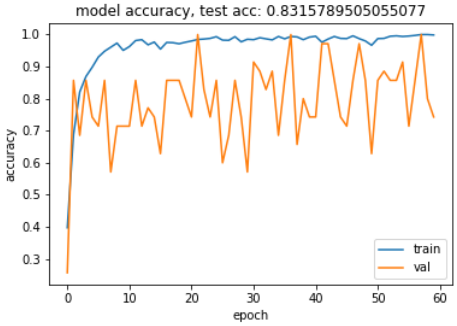
(e) Accuracy run 10/10



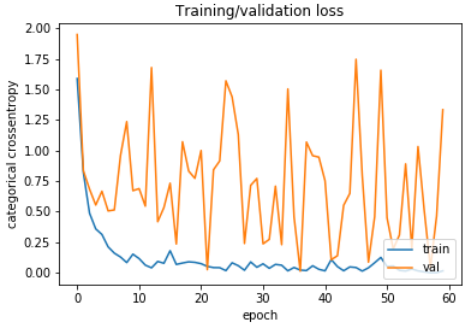
(f) Loss run 10/10

Figure 8.4: Personalized - Four-class training and validation accuracy and categorical cross entropy loss over number of epochs.

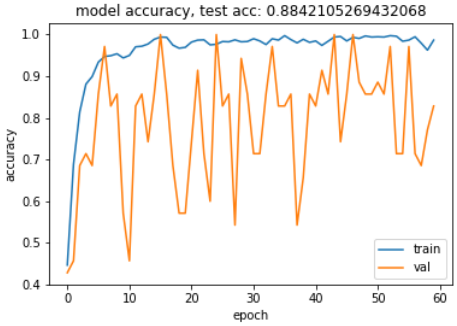
8.2.3 Seven Classes



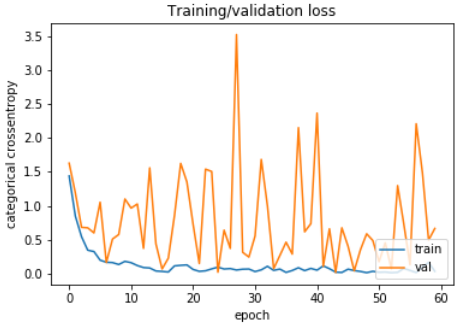
(a) Accuracy run 1/10



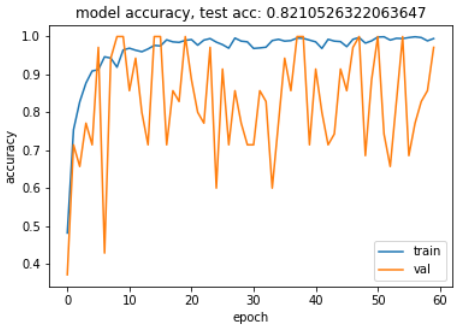
(b) Loss run 1/10



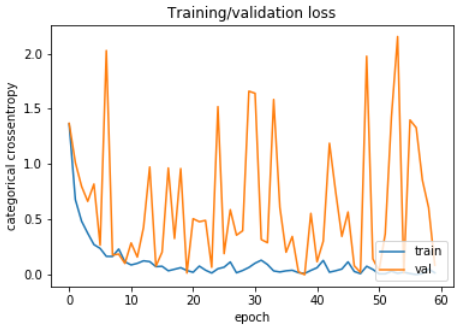
(c) Accuracy run 2/10



(d) Loss run 2/10

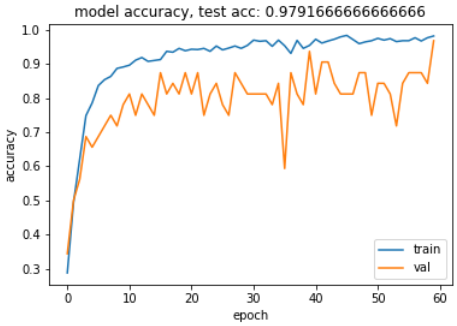


(e) Accuracy 8/10

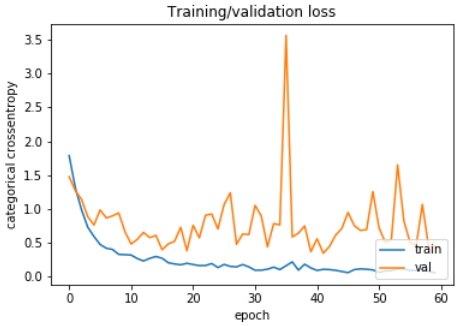


(f) Loss 8/10

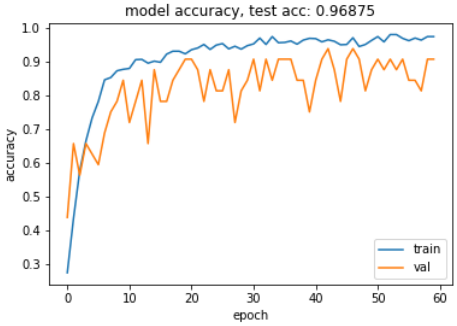
Figure 8.5: Cohn-Kanade - Seven-class training and validation accuracy and categorical cross entropy loss over number of epochs.



(a) Accuracy run 1/10

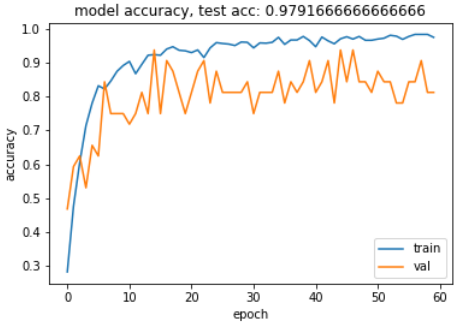


(b) Loss run 1/10



(c) Accuracy run 3/10

(d) Loss run 3/10



(e) Accuracy 5/10

(f) Loss 5/10

Figure 8.6: Personalized - Seven-class training and validation accuracy and categorical cross entropy loss over number of epochs.